Универзитет у Београду
Факултет организационих наука

OOP4FUN: OBJECT ORIENTED PROGRAMMING FOR FUN

Приручник за наставнике средњих школа

Уредници

Michal Varga, Josef Rak, Душан Савић, Zlatko Stapić

Аутори

Peter Sedláček, Nika Kvaššayová, Jozef Kostolný, Michal Mrena, Patrik Rusnák, Peter Sobe, Илија Антовић, Милош Милић, Татјана Стојановић, Нина Турајлић, Davor Fodrek, Lidija Kozina, Marko Mijač, Dijana Plantak Vukovac, Antonela Čižmešija, Dijana Peras, Goran Hajdin, Lea Masnec

Београд, септембар 2024.

Наслов
OOP4FUN: OBJECT ORIENTED PROGRAMMING FOR FUN - Приручник за наставнике средњих школа

Уредници
Michal Varga, Josef Rak, Душан Савић, Zlatko Stapić

Аутори
Peter Sedláček, Nika Kvaššayová, Jozef Kostolný, Michal Mrena, Patrik Rusnák, Peter Sobe, Илија Антовић, Милош Милић, Татјана Стојановић, Нина Турајлић, Davor Fodrek, Lidija Kozina, Marko Mijač, Dijana Plantak Vukovac, Antonela Čižmešija, Dijana Peras, Goran Hajdin, Lea Masnec

Издавач
Универзитет у Београду
Факултет организационих наука
Јове Илића 154, 11000 Београд

За издавача
Проф. др Марко Михић

Преводиоци
Душан Савић, Илија Антовић, Милош Милић, Татјана Стојановић, Нина Турајлић, Снежана Богдановић, Зоран Вучетић, Драган Грујовић и Ана Стаменић

Графичко обликовање и прелом текста
Нина Турајлић и Душан Савић

Штампарија
VIBBACO DOO

Тираж
100 примерака

Место и година издања
Београд, септембар 2024. године

ISBN
ISBN 978-86-7680-478-5 (штампано издање)

Овај приручник је објављен на чешком, словачком, немачком, хрватском и српском језику.

Одрицање од одговорности:

Финансирано средствима Европске уније.

Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the Erasmus+ Programme of the European Union

САДРЖАЈ

Co-funded by the Erasmus+ Programme of the European Union

Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the Erasmus+ Programme of the European Union

ПРЕГЛЕД ТАБЕЛА

Co-funded by the Erasmus+ Programme of the European Union

ПРЕДГОВОР

У последњих неколико година примећен је значајан пад броја ученика који се занимају за технички оријентисане студијске програме (СТЕМ[1]) у средњим школама у свету. Овај тренд указује на хитну потребу мотивисања ученика и показивања да СТЕМ, а посебно програмирање, није толико тешко као што се можда чини. Кључ за уверавање средњошколаца у ту чињеницу лежи у рукама наставника, који их воде кроз технички оријентисане предмете, а нарочито информатику.

Због тога смо 2021. године одлучили да оснујемо међународни конзорцијум пет факултета и пет средњих школа, предвођених Факултетом за менаџмент и информатику у Жилини, Република Словачка. Наш циљ био је да се развије и прошири приступ учењу објектно-оријентисаног програмирања на забаван начин, изворно осмишљен за наставнике средњих школа у Републици Словачкој.

Наш пројекат „Објектно-оријентисано програмирање на забаван начин“ (OOP4FUN – *Object Oriented Programming for Fun*) осмишљен је са јасним фокусом на наставнике средњих школа, који заправо представљају нашу примарну циљну групу. Кроз разговоре са професорима факултета укљученим у пројекат, откривено је да постоји значајан јаз између стеченог и очекиваног предзнања студената прве године, када је реч о разумевању основних принципа програмирања, нарочито у области објектно-оријентисаног програмирања (ООП). Свесни значаја тих темељних знања, одлучили смо да осигурамо да ће ученици средњих школа стећи свеобухватно разумевање ООП-а.

Како бисмо предупредили наведене проблеме, припремили смо материјале за темељно савладавање основних концепата ООП-а, који су посебно прилагођену наставницима средњих школа. Развили смо детаљне курикулуме и пропратне материјале, укључујући приручнике и *online* садржаје. Јединствена мисија нашег међународног тима (који укључује истраживаче и стручњаке са различитих факултета из Словачке, Чешке, Хрватске, Србије и Немачке) била је усмерена на дефинисање наставних планова, посебно у контексту учења и подучавања ООП-а кроз развој рачунарских игара.

У овој публикацији налазе се материјали за припрему наставних часова у оквиру различитих предмета везаних за информатику и програмирање. Ови материјали се могу користити у целости или само као инспирација за креирање сопствених наставних материјала. Припремљени примери се фокусирају на рад у *Greenfoot* развојном окружењу.

Надамо се да ће ова публикација буде користан приручник свим наставницима који предају објектно-оријентисано програмирање у средњим школама.

Ауторски тим

[1] Наука (*Science*), технологија (*Technology*), инжењерство (*Engineering*) и математика (*Mathematics*)

Co-funded by the Erasmus+ Programme of the European Union

ДОДАТНИ ЕЛЕКТРОНСКИ ИЗВОРИ

Наставни план и програм, на енглеском језику, који се прати у овом приручнику и који се односи на објектно-оријентисано програмирање, може се наћи на адреси **https://oop4fun.eu/**.

Репозиторијум са изворним програмским кôдом игре `TowerDefense`, на који се позивамо у овом приручнику, али и други пројекти који се налазе у енглеској верзији наставног плана и програма, доступни су на адреси: **https://gitlab.kicon.fri.uniza.sk/oop4fun**.

Електронски материјали на енглеском језику, који прате овај приручник, могу се преузети са *Moodle* платформе на адреси: **https://oop4fun.fon.bg.ac.rs**.

Енглеска верзија наставног плана и програма

Репозиторијум изворног програмског кôда

Електронски материјали на *Moodle* платформи

Co-funded by the Erasmus+ Programme of the European Union

УВОДНЕ ИНФОРМАЦИЈЕ

Циљ овог приручника је да представи курикулум курса за објектно-оријентисано програмирање (ООП), са пратећим материјалима, који треба да помогне наставницима у припреми наставе за подучавање ученика у средњим школама.

Основни концепти ООП су објашњени коришћењем програмског језика *Java*, док је *Greenfoot* коришћен као развојно окружење. *Java* је програмски језик који је тренутно веома популаран и који се широко користи у пракси. *Greenfoot* представља развојно окружење које је засновано на оквирима (енгл. *frame-based source code editor*), које подржава програмски језик *Stride*, а пре свега је намењено развоју *2D* рачунарских (компјутерских) игара. *Greenfoot* је развојно окружење, које користи и графичку и текстуални нотацију, што омогућава једноставан развој апликација које садрже графичке елементе (објекте) који могу међусобно комуницирати. Ученици ће одмах започети са практичним радом на развоју једне компјутерске игре, кроз коју ће се обрадити и концепти ООП-а. Коришћење овог приручника наставницима пружа могућност да ученицима прикажу концепте ООП-а на забаван и креативан начин.

Ученици ће научити основе ООП кроз игру, у тиму или самостално. Научиће како да међусобно сарађују и како да осмисле и пројектују своју игру (тј. апликацију). У оквиру овог курса, концепти ООП-а (као што су енкапсулација, наслеђивање или асоцијације) се објашњавају постепено (лагано) кроз развој *2D* компјутерске игре корак по корак, при чему се уведени концепти примењују на једноставан и интуитиван начин. Процес развоја компјутерске игре заснива се на тимском раду и практично примењује знања и вештине из других области информатике и сродних предмета. Пројектовање компјутерске игре може бити креативно, тако да ученици могу индивидуално проширити постојећу игру на свој начин. Развојем компјутерских игара, односно раду на реални апликацијама, ученици ће кроз практичне примере усвојити и применити стечена знања.

Курс је усмерен на увођење иновативног приступа подучавању ученика концептима ООП парадигме. ООП је данас доминантна парадигма када је реч о развоју апликација, због чега је пожељно да ученици поседују ова знања и вештине. У оквиру курса представљено је развојно окружење које користи различите облике уређивања изворног кôда (уређивање засновано на принципу оквира, али и писањем изворног кôда), што омогућава подучавање ученика са различитим нивоима програмерског предзнања. Својом једноставношћу и јасноћом, овај алат подржава брзо и интуитивно разумевање наставних тема, што позитивно утиче на ученике и њихову мотивацију.

Кроз програмирање интерактивних игара у *Greenfoot* развојном окружењу, ученици ће стећи знања и вештине које ће им омогућити да:

- идентификују проблем,
- идентификују објекте који су неопходни за решавање датог проблема,
- идентификују класе, њихове атрибуте и методе,
- идентификују и правилно користе везе између објеката (асоцијације и наслеђивање),
- дефинишу алгоритам за решавање проблема,
- употребе одговарајуће управљачке структуре (гранање, петље) за имплементацију дефинисаног алгоритма,
- ефикасно отклоне грешке у изворном кôду и
- развију једноставну апликацију са графичким интерфејсом у *Greenfoot* окружењу.

Образовни исходи се могу сумирати на следећи начин:

- разумевање основних принципа објектно-оријентисаног програмирања,
- разумевање основних принципа развоја алгоритама,
- разумевање синтаксе програмског језика *Java*,
- способност анализе тока извршавања програма на основу изворног кôда и
- способност развоја сопствених програма применом ООП-а.

Co-funded by the Erasmus+ Programme of the European Union

Савремен приступ осмишљавању предавања је, посебно за основно и средње образовање, дефинисање и дељење наставних сценарија (енгл. *teaching scenario*). Наставни сценарији се перципирају као савремени педагошки приступ који омогућава индивидуализацију наставног процеса узимајући у обзир различите потребе ученика. Настава заснована на наставним сценаријима је фокусирана на релевантна знања и вештине ученика. Пажљиво планирање наставних сценарија може отклонити могуће замке и недостатке који би могли утицати на наставни процес.

У контексту образовања, наставни сценарији представљају детаљан опис начина извођења наставе. Ови сценарији се често користе у обуци наставника за симулацију наставних ситуација из стварног света и стога се сматрају најбољим алатом за представљање наших иновативних идеја поучавања и учења.

Како би се обезбедио структуриран приступ дефинисању наставних сценарија, дефинисали смо следећи шаблон (Табела 1) који би требало да садржи конкретне податке везане за одређени наставни сценарио. Табела ниже садржи кратко објашњење како дефинисати сваки елемент наставног сценарија.

Сходно томе, алат за креирање наставних сценарија, доступан је на следећој веб адреси: **https://learning-design.eu**.

Пример наставног сценарија доступан је на адреси: **https://learning-design.eu/en/preview/bd6a3cf90d09a6b6495ecf82/details**, док су наставни сценарији који су изложени у овом приручнику доступни на адреси: **https://learning-design.eu/en/planning/1790**. Сваки наставни сценарио прати и скуп смерница за припрему сваке од наставних активности, али и линк ка решењу (енгл. *commit*) у репозиторијуму изворног програмског кôда, тамо где је то потребно.

Co-funded by the Erasmus+ Programme of the European Union

Табела 1. Предлог шаблона за документовање наставног сценарија

Назив	Одаберите описан назив који привлачи пажњу.
Образовни циљеви и исходи учења	Јасно наведите очекиване исходе учења. Шта би ученици требало да знају, разумеју или буду у стању да практично примене након савладавања сценарија?
Циљна група	Наведите циљну групу (и разред) којој је сценарио намењен, као и очекивано предзнање.
Временски план	Процените време које ће бити потребно да би се реализовао сценарио, укључујући и специфичне временске оквире за сваку од активности. На пример, уводна реч наставника (5 минута), самостално истраживање од стране ученика (10 минута), тимски рад на програмирању решења (20 минута), презентовање/дискусија (10 минута).
Материјали и ресурси	Наведите материјале, ресурсе и алате који ће бити потребни и наставницима и ученицима. То може да укључује уџбенике, материјале на Интернету, мултимедијалне ресурсе, софтвер итд.
Опис	• Представите наставни сценарио, објасните његову сврху и значај. • Дајте преглед основних активности у које ће ученици бити укључени како би достигли образовне циљеве. Укључите и детаље као што су дискусије, практичан рад, тимски рад, такмичења итд. • Прецизирајте начин на који ће ученици бити организовани, тј. да ли ће радити самостално или у тимовима. Колико ће тимови имати чланова? • Изложите пројекте/проблеме/задатке на којима ће ученици радити. Препоручује се примена приступа заснованих на проблему или приступа заснованих на пројекту. Поред тога, они би требало да одражавају ситуације из свакодневног живота. • Објасните начин на који ће се ученицима (тимовима) додељивати пројекти/проблеми/задаци. • Ако је предвиђен тимски рад, пружите детаљније информације о начину сарадње. • Наведите појединости свих активности у којима ученици треба да учествују. • У случају да се примењује нпр. приступ обрнуте учионице (енгл. *flipped classroom*), прецизирајте који део тематске целине ученици треба да самостално истраже.
Вредновање	Наведите детаљне информације о начину вредновања залагања и знања ученика: • Ко ће вршити вредновање: (1) наставници, (2) сам ученик (самовредновање), (3) други ученици (међусобно вредновање)? • Који критеријуми ће се користити за вредновање? • Колико често ће се вршити вредновање? • итд.

Co-funded by the Erasmus+ Programme of the European Union

Дељење решења	Објасните како ће ученици поделити своја решења са наставницима и другим ученицима. На пример, ученици (сви или само један део ученика) могу представити своја решења/резултате пред наставником и осталим ученицима, након чега може уследити поређење и дискусија.

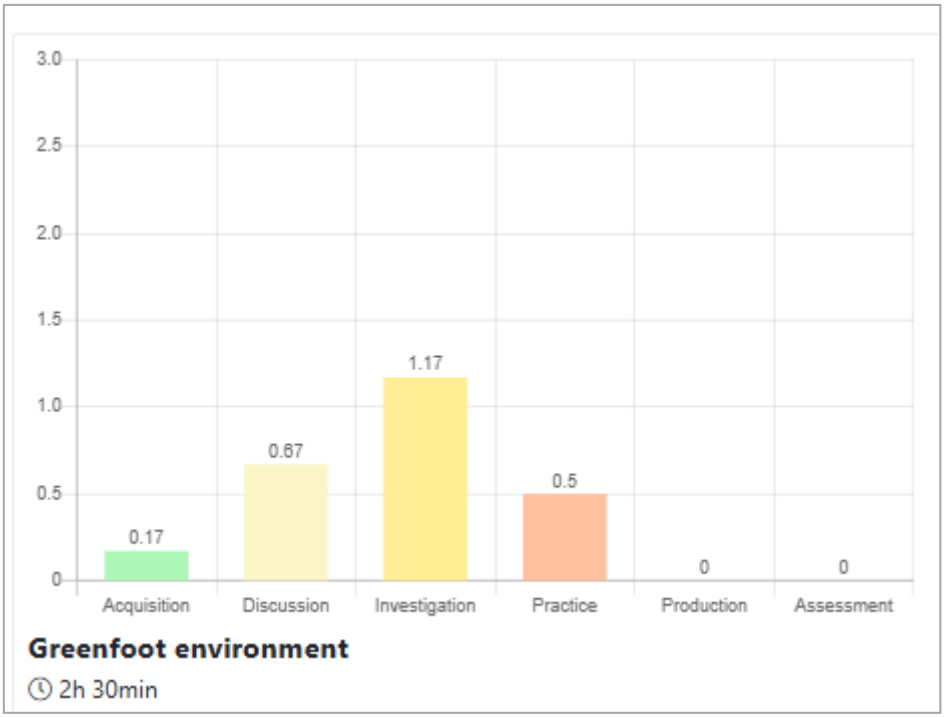
Co-funded by the Erasmus+ Programme of the European Union

1. УПОЗНАВАЊЕ СА *GREENFOOT* ОКРУЖЕЊЕМ

Greenfoot представља образовни алат (тј. развојно окружење са интегрисаним уређивачем/едитором програмском кôда и *2D* визуелизацијом) за развој компјутерских игара и симулација коришћењем програмског језика *Java.* Поред тога што је визуелан, *Greenfoot* је и интерактиван. *Greenfoot* омогућава писање програма у стандардном текстуалном формату (тј. коришћењем програмског језика *Java*), док са друге стране обезбеђује визуелизацију кôда и његовог извршавања.

1.1. Структура наставних активности и задаци

На следећој слици је дат приказ расподеле ангажовања ученика према типу активности, на основу искуства наставника средњих школа у Словачкој Републици, а који је генерисан у алату за креирање наставних сценарија.



Слика 1. Предложена расподела ангажовања ученика према типу активности за тематску целину „Упознавање са Greenfoot окружењем“[2]

Задатак 1.1. Креирање *Greenfoot* пројекта

Креирајте нов пројекат, доделите му погодан назив (нпр. `TowerDefense`, тј. одбрана куле) и сачувајте га на одговарајућој локацији.

[Commit: 9046f5353d857dcc112abd92d7b7170abcc64a80]

[2] Извештај је доступан на: https://learning-design.eu/en/preview/452257b563cbf14b6f06acfd/analysisworkload

Co-funded by the Erasmus+ Programme of the European Union

1.2. Наставни сценарији

Припремљен је **један** наставни сценарио за тематску целину „Упознавање са *Greenfoot* окружењем“.

1.2.1. СЦЕНАРИО: **Креативно упознавање са развојем рачунарских игара**

Табела 2. Креативно упознавање са развојем рачунарских игара

Назив	Креативно упознавање са развојем рачунарских игара
Образовни циљеви и исходи учења	Током ове лекције, поред тога што су успешно инсталирали *Greenfoot* и уверили се у његове могућности кроз примере пројеката, ученици су такође имали прилику да, кроз игру, испробају развојно окружење. Овај увод треба да постави тон за истраживање развоја игара, подстакне креативност, тимски дух и страственији приступ програмирању у *Greenfoot* окружењу.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **90** минута. 1. Увод (5 минута) 2. Брзи изазов (10 минута) 3. Играње игара са наставником (30 минута) 4. Формирање тимова и додела пројектног задатака (5 минута) 5. Тимски рад и програмирање (30 минута) 6. Међусобно вредновање и повратне информације (10 минута) 7. Домаћи задатак (30 минута) 8. Вредновање (30 минута)
Материјали и ресурси	Веб страница *Greenfoot* окружења и упутство за његово преузимање. Примери које је припремио наставник. Ресурси на Интернету за проналажење додатних примера.
Опис	Кроз овај наставни сценарио ученици ће завирити у свет *Greenfoot* окружења путем игре, истраживања и тимског рада. Након што наставник представи тематску јединицу и дефинише циљеве лекције, почиње брзи изазов. Ученици имају задатак (замишљен у виду такмичења) да пронађу неопходна упутства, а затим преузму и на својим рачунарима инсталирају *Greenfoot* (окружење које им је у датом тренутку још увек непознато). Три најбржа ученика добијају одређену награду као подстицај (значке, бодови, поени, слаткиши итд.). Следеће изненађење представља чињеница да ће током наредних 30 минута играти игре са наставником. Ово је блок који води наставник, а који је усмерен на отварање, компајлирање и покретање пар једноставних показних примера (пројекти почетног или средњег нивоа сложености). Ученици ће се тако упознати са основним елементима *Greenfoot* развојног окружења и основним поступцима рада са пројектним датотекама и ресурсима.

Co-funded by the Erasmus+ Programme of the European Union

	Ученици ће, потом, бити распоређени у тимове (3-4 ученика по тиму) и биће им додељен једноставан задатак (**Задатак 1.1.**). Од тимова ће бити затражено да измене „нешто“ (одређени програмски кôд) у одабраном показном примеру, са циљем да учине игру забавнијом или да у њу уведу неки фактор изненађења. Ученици заједнички раде на изменама. Уколико промене програмски кôд тако да не могу више самостално да га исправе (тј. до тачке да није више могуће покренути апликацију), могу замолити наставника за помоћ или кренути испочетка (поновним преузимањем почетне верзије кôда). Ово ће им помоћи да схвате зашто је пожељно користити неки систем за управљање верзијама кôда (нпр. *git*). Наставник бира тим (или два тима) који треба да представи своје решење осталим ученицима, како би га они вредновали и дали повратне информације. Наставник разговара са ученицима о резултатима. За домаћи задатак, сваки од ученика треба да потражи примере *Greenfoot* пројеката, а затим да одабере онај који му се највише допада и представи га остатку групе, тако што ће са њима поделити адресу пројекта, разлоге због којих га је одабрао и два до три снимка екрана развојног окружења и тока игре. Како би се увели елементи игре у наставу (енгл. *gamification*) и подстакла мотивација (путем надметања), ученици треба да гласањем изаберу три најбоља пројекта (при чему ученик не може гласати за сопствени пројекат). На крају се проглашавају победници и додељују им се одређене награде као подстицаји (значке, бодови, поени, слаткиши итд.).
Вредновање	Активности у оквиру ове лекције ће омогућити наставницима да дају ученицима повратне информације на основу формативног вредновања њиховог учешћа у спроведеним дискусијама, али и њиховог рада у окружењу обрнуте учионице (енгл. *flipped classroom*), као и рада у тиму. Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. Међусобно вредновање се може спровести *online*, као део домаћег задатка. Кроз ову активност ученици ће се подсетити важних аспеката лекције и биће подстакнути да инсталирају *Greenfoot* окружење. Разматрањем различитих примера, ученици ће лакше усвојити изложено градиво, што доприноси и повећању општег достигнућа постављених образовних циљева и исхода учења целе групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

1.2.1.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Увод (5 минута)

Циљ: Упознавање са програмским језицима (текстуалним и визуелним), интегрисаним развојним окружењима (енгл. *integrated development environment - IDE*) и изворним кôдом (енгл. *source code*).

Концепти за дискусију: програмски језици, интегрисана развојна окружења и изворни кôд.

Активности: Наставник даје кратак увод у лекцију и уводи концепте као што су:

- програмски језик
- интегрисано развојно окружење и
- изворни кôд.

Наставник показује примере изворног кôда у различитим програмским језицима, као што су *Java*, *C* и *Python*, али и примере програма у визуелном језику *Scratch* или неком другом визуелном језику.

Наставник не залази „у дубину“, већ настоји да објасни ове концепте путем једноставног примера, нпр. покушава да упореди савладавање програмског језика са учењем матерњег или страног језика.

- Природни језици имају своју граматику и правопис. Исто важи и за програмске језике. Наиме као што се, када пишемо на нашем матерњем језику, придржавамо одређених граматичких и правописних правила, то исто је неопходно учинити и приликом писања програма у неком програмском језику.
- Када желимо да напишемо причу на нашем матерњем језику, можемо да користимо свеску и оловку као помоћне „алатке“. Аналогно томе, када пишемо програм у неком програмском језику, наша „алатка“ је интегрисано развојно окружење;
- Резултат процеса писања може бити прича написана на папиру, док резултат процеса програмирања представља програм који смо развили и који називамо изворни кôд.

Наставник може осмислити и сопствени уводни пример.

2. Брзи изазов (10 минута)

Циљ: Упознавање са *Greenfoot* окружењем и начином његовог инсталирања.

Концепти за дискусију: *Greenfoot* окружење, упутство за инсталирање.

Активности: Наставник ученицима даје задатак да открију шта је *Greenfoot*. Наставник тражи од ученика да пронађу упутство за његово инсталирање. Ученици раде на задатку, након чега им наставник демонстрира како да пронађу, преузму и покрену инсталацију *Greenfoot* окружења. Упутства за преузимање и инсталирање *Greenfoot* окружења су доступна на *Moodle* платформи.

3. Играње игара са наставником (30 минута)

Циљ: Покретање различитих *Greenfoot* пројеката.

Концепти за дискусију: *Greenfoot* пројекти: пројекти на вебу и самостални (енгл. *standalone*) пројекти.

Активности: Наставник тражи од ученика да на Интернету пронађу примере пројекта развијених у *Greenfoot* окружењу. Док ученици раде на задатку, наставник надгледа њихов рад.

Наставник показује ученицима како да пронађу примере готових пројеката развијених у *Greenfoot* окружењу. На пример, у *Google* претраживачу пројекти се могу наћи навођењем кључних речи као што су „*Greenfoot Java project example*“ или слично.

Co-funded by the Erasmus+ Programme of the European Union

Наставник објашњава да постоје две врсте *Greenfoot* пројеката:

- они који се могу покренути у веб прегледачу (енгл. *web browser*) и
- они који се могу преузети са Интернета, а затим отворити и покренути у самом *Greenfoot* окружењу.

Наставник представља различите врсте пројеката:

- пројекте који се могу покренути директно у веб прегледачу и
- пројекте који се, након преузимања, могу покренути у самом *Greenfoot* окружењу.

Наставник може да преузме неколико примера пројеката или да приступи пројектима који се налазе на одређеним адресама.

4. Формирање тимова и додела пројектног задатака (5 минута)

Циљ: Упознавање са учењем заснованом на пројектима путем једноставног пројектног задатка.

Концепти за дискусију: /.

Активности: Наставник формира тимове, одређује пројекат на којем ученици треба да раде и припрема пројектни задатак. Примери пројектних задатака се могу пронаћи на *Moodle* платформи. Наставник бира један репрезентативан пројекат (који не треба да буде претерано сложен) и за дати пројекат дефинише одговарајући пројектни задатак (тј. проблем) који ученици треба да реше.

5. Тимски рад и програмирање (30 минута)

Циљ: Овладавање начином креирања *Greenfoot* пројекта. Ученици раде на задатку који им је додељен.

Концепти за дискусију: креирање *Greenfoot* пројеката.

Активности: Наставник покреће *Greenfoot* окружење и показује ученицима како да креирају пројекат. Наставник задаје ученицима задатак:

> **Задатак 1.1. (Креирање *Greenfoot* пројекта):** Креирајте нов пројекат, доделите му погодан назив (нпр. `TowerDefense`, тј. одбрана куле) и сачувајте га на одговарајућој локацији.

Ученици раде на задатку и решавају проблем.

Наставник наглашава да се пројекат (који је идентичан пројекту који су они управо креирали) може:

- преузети са *git*[3]-а

  [Commit: `9046f5353d857dcc112abd92d7b7170abcc64a80`]

- преузети, у ZIP формату, са следеће адресе: **https://oop4fun.fon.bg.ac.rs/**

Наставник такође наглашава да ученици, током данашње лекције, неће даље радити на пројекту који су управо креирали (мада ће рад на њему наставити током наредних лекција), већ ће радити само на постојећим пројектима.

[3] Неопходно је да *git* буде инсталиран на рачунару

Co-funded by the Erasmus+ Programme of the European Union

6. Међусобно вредновање и повратне информације (10 минута)

Циљ: Дискусија о предложеним решењима.

Концепти за дискусију: /.

Активности: Наставник надгледа рад ученика и, уколико је потребно, даје им смернице (инструкције). Када ученици заврше са радом, наставник бира тим који ће представити своје решење. Наставник дискутује са ученицима о предложеном решењу.

7. Домаћи задатак (30 минута)

Циљ: Проучавање *Greenfoot* пројеката.

Концепти за дискусију: /.

Активности: Наставник дефинише задатак који ученици треба да реализују у једном од постојећих пројеката.

8. Вредновање (30 минута)

Циљ: Укључивање ученика у процес вредновања решења.

Концепти за дискусију: /.

Активности: Ученици представљају своје пројекте. Наставник тражи од свих ученика да учествују у вредновању представљених пројеката, нпр. тако што са њима дели линк ка *Google* упитнику коју треба да попуне. Ученици треба да, путем бодовања, изаберу три најбоља тима.

Након тога наставник даје осврт на представљене пројекте. Наставник дели своје утиске о раду ученика и указује на то да ли је задовољан, да ли су испунили или надмашили његова очекивања.

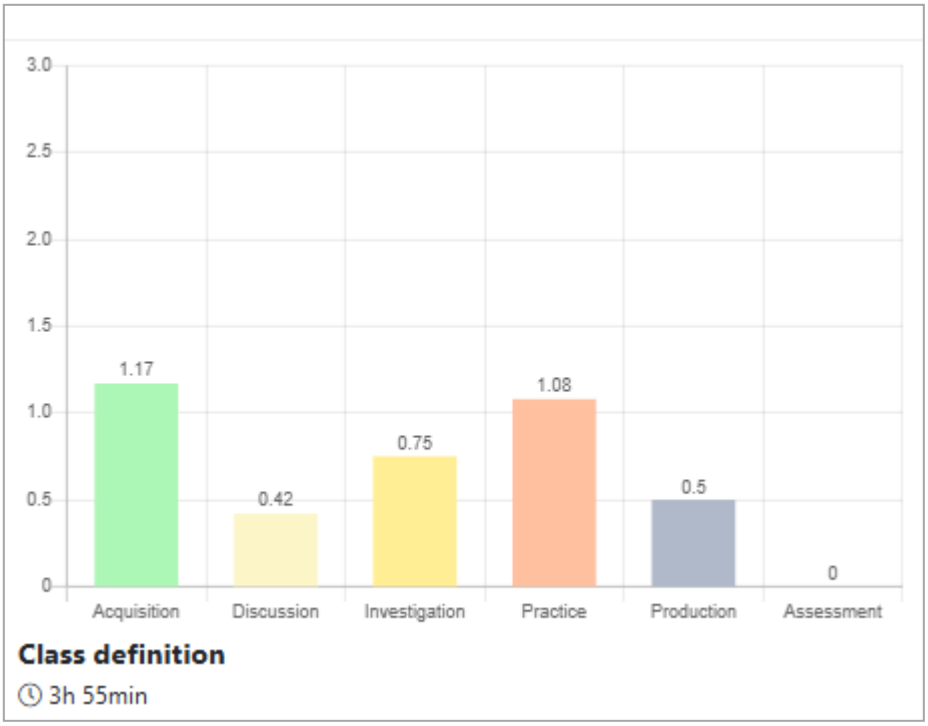
Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the Erasmus+ Programme of the European Union

2. КЛАСЕ И ОБЈЕКТИ

2.1. Структура наставних активности и задаци

На следећој слици је дат приказ расподеле ангажовања ученика према типу активности, на основу искуства наставника средњих школа у Словачкој Републици, а који је генерисан у алату за креирање наставних сценарија.



Слика 2. Предложена расподела ангажовања ученика према типу активности за тематску целину „Класе и објекти“

Задатак 2.1. Идентификовање објеката

Идентификујте објекте у вашем окружењу и наведите њихове карактеристике, као и активности које могу да изврше. Да ли је могуће идентификовати објекте који немају карактеристике? Да ли је могуће идентификовати објекте који немају понашање? Да ли је могуће идентификовати нематеријалне објекте (тј. објекте које не можете физички додирнути)?

Задатак 2.2. Припрема света

Креирајте свој свет (`MyWorld`) димензије **12 x 24** ћелија. Величина сваке појединачне ћелије треба да буде **50** пиксела.

[Commit: a593cd4a92d0fa0db78275614c3e41a2e96b4e57]

Задатак 2.3. Припрема слике за свет

Пронађите или креирајте одговарајућу слику за позадину света. Можете користити неку од понуђених слика или одабрати неку другу слику. За позадину можете користити једну велику слику, која ће прекрити читаву површину света, или једну мању слику, која ће се понављати.

[Commit: 1184980643db082cfdd6bde9984bceaddf010d49]

Co-funded by the Erasmus+ Programme of the European Union

Задатак 2.4. Дефинисање класе непријатеља

Уведите новог протагонисту – непријатеља. Непријатељ ће напредовати према сфери (`Orb`) како би је најпре оштетио, а на крају и потпуно уклонио. Дефинишите нову класу, доделите јој погодан назив (нпр. `Enemy`) и придружите јој одговарајућу слику.

[Commit: `4981400623729c3d112b54454b6e6151e18426bf`]

Задатак 2.5. Креирање инстанце класе непријатеља

Креирајте инстанцу класе `Enemy`. Прегледајте интерно стање креиране инстанце. Креирајте још једну инстанцу класе `Enemy` и поставите је на неку другу позицију. Упоредите интерна стања те две инстанце.

Задатак 2.6. Позивање метода

Позовите одговарајућу методу над инстанцом класе `Enemy`, како би се она преместила на позицију `[12,6]` и била окренута надоле. Какав ће утицај то имати на интерно стање дате инстанце?

Co-funded by the Erasmus+ Programme of the European Union

2.2. Наставни сценарији

Припремљена су **два** наставна сценарија за тематску целину „Класе и објекти“.

2.2.1. СЦЕНАРИО: **Упознавање са класама и објектима кроз развој игара у *Greenfoot* окружењу**

Табела 3. Упознавање са класама и објектима кроз развој игара у Greenfoot окружењу

Назив	Упознавање са класама и објектима кроз развој игара у *Greenfoot* окружењу
Образовни циљеви и исходи учења	Након ове лекције, ученици би требало да разумеју основне концепте класе и објекта. Разумевање ових концепата се продубљује кроз развој игре у *Greenfoot* окружењу, чиме се подстиче креативност, тимски дух и страственији приступ програмирању.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **100** минута. 1. Објекат (10 минута) 2. Идентификација објеката и њихових атрибута (15 минута) 3. Класа и инстанца (објекат) (15 минута) 4. Сналажење у *Greenfoot* окружењу (10 минута) 5. Конструктор (10 минута) 6. Задатак **2.2.** (15 минута) 7. Постављање слике (10 минута) 8. Задатак **2.3.** (15 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Кроз овај наставни сценарио ученици ће се упознати са принципима програмирања који се односе на класе и објекте, а из перспективе развоја игара у *Greenfoot* окружењу. Лекција почиње десетоминутним уводом наставника, који уводи ученике у свет објектно-оријентисаног програмирања тако што им приближава концепт објекта и његових карактеристика коришћењем примера из свакодневног живота (**Задатак 2.1.**). Потом започиње брзи изазов од 10 минута, током којег ученици треба да, у задатом текстуалном опису, препознају објекте и њихове карактеристике. Након тога следи петоминутни блок, током којег наставник, заједно са ученицима, разматра решење датог задатка. Током наредних 15 минута наставник објашњава разлику између класе и објекта и уводи, на највишем нивоу апстракције, концепт наслеђивања.

Co-funded by the Erasmus+ Programme of the European Union

	Наставник затим покреће *Greenfoot* окружење и током наредних 10 минута упознаје ученике са основим класама: `World` (свет), `Actor` (протагониста) и `MyWorld` (свој свет). Наставник представља и објашњава изворни кôд ових класа, који се аутоматски генерише када се креира пројекат. Током наредног десетоминутног блока, наставник задаје **Задатак 2.2.**, а затим показује ученицима како да припреме свет за апликацију коју треба да развију. Током наредних 25 минута, наставник најпре објашњава како се подешава слика за одређену класу, а затим заједно са ученицима решава **Задатак 2.3.**
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

2.2.1.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Објекат (10 минута)

Циљ: Увођење концепта објекта кроз примере из свакодневног живота**.**

Концепти за дискусију: објекти и атрибути.

Активности: Наставник уводи појам **објекта**. Наставник треба да приближи овај концепт ученицима коришћењем примера из свакодневног живота. На пример, наставник може да пита ученике како се зову, колико су високи, када су рођени, која им је боја очију итд. Наставник поставља оваква питања како би навео ученике да размисле о томе како се разликују једни од других, припремајући их на тај начин за наредно питање: „По чему се ученици међусобно разликују?“.

Наставник закључује да свака особа (наставник, ученици) има одређене карактеристике по којима се разликује од других и наглашава да је свако од нас заправо представља „један“ објекат. Сходно томе, објекти се међусобно разликују по вредностима својих карактеристика (које се у свету програмирања називају атрибути).

Наставник са ученицима разговара о задатку:

> **Задатак 2.1. (Идентификовање објеката):** Идентификујте објекте у вашем окружењу и наведите њихове карактеристике, као и активности које могу да изврше. Да ли је могуће идентификовати објекте који немају карактеристике? Да ли је могуће идентификовати објекте који немају понашање? Да ли је могуће идентификовати нематеријалне објекте (тј. објекте које не можете физички додирнути)?

2. Идентификација објеката и њихових атрибута (15 минута)

Циљ: Идентификовање објеката и њихових атрибута**.**

Концепти за дискусију: објекти и њихови атрибути.

Активности: Наставник тражи од ученика да, у датом тексту, идентификују објекте и њихове атрибуте.

3. Класа и инстанца (објекат) (15 минута)

Циљ: Стицање знања о класама и инстанцама тј. објектима. Разликовање појмова класе и објекта.

Концепти за дискусију: класа, објекат (инстанца).

Активности: Наставник објашњава разлику између концепта класе и концепта инстанце класе (тј. објекта) коришћењем примера из свакодневног живота, и уводи, на највишем нивоу апстракције, концепт наслеђивања. Наставник поставља питања ученицима, како би их навео да увиде разлику између класе и објекта. Наставник усмерава дискусију о идентификованим објектима и њиховој класификацији.

4. Сналажење у *Greenfoot* окружењу (10 минута)

Циљ: Представљање класе `World` (свет) и креирање њене инстанце.

Концепти за дискусију: инстанца света тј. класе `World`.

Активности: Наставник покреће *Greenfoot* окружење и креира једноставан пројекат. Наставник кроз овај пројекат објашњава ученицима како да уведу разматране концепте. Наставник указује на то да сваки пројекат, који је креиран у *Greenfoot* окружењу, садржи три класе: `World` (свет), `Actor` (протагониста) и `MyWorld` (мој свет). Наставник затим представља класу `MyWorld` (мој свет) и разјашњава њену улогу.

Co-funded by the Erasmus+ Programme of the European Union

Наставник истиче то да позадина (енгл. *background*) било које *Greenfoot* апликације заправо представља матрицу која се састоји од појединачних ћелија (енгл. *cells*). Након тога, наставник демонстрира ученицима како се може дефинисати величина позадине (тј. димензија матрице), као и величина сваке појединачне ћелије матрице. Наставник објашњава да се сваки објекат који се приказује на екрану („сцени“) налази у једној од ћелија или у више ћелија, у зависности од димензија (ширине и висине) појединачних ћелија и самог објекта.

Наставник показује ученицима изворни кôд сваке од класа.

5. Конструктор (10 минута)

Циљ: Увођење концепта конструктора.

Концепти за дискусију: конструктор.

Активности: Наставник приказује изворни кôд и уводи концепт **конструктора**.

6. Задатак **2.2.** (15 минута)

Циљ: Учешће ученика у раду на пројектном задатку.

Концепти за дискусију: свет у *Greenfoot* окружењу.

Активности: Наставник задаје ученицима задатак:

> **Задатак 2.2. (Припрема света):** Креирајте свој свет (`MyWorld`) димензије **12 x 24** ћелија. Величина сваке појединачне ћелије треба да буде **50** пиксела.

Док ученици решавају задатак наставник прати њихов рад, а затим бира једног ученика који треба да представи и објасни своје решење.

Опис решења:

> Преправите изворни кôд класе `MyWorld` (двоструким кликом на њу) како бисте дефинисали свет димензије **12 x 24** ћелија. Величина сваке појединачне ћелије треба да буде **50** пиксела.

[Commit: a593cd4a92d0fa0db78275614c3e41a2e96b4e57]

7. Постављање слике (10 минута)

Циљ: Постављање слике света у *Greenfoot* пројекту.

Концепти за дискусију: слика за класу `World` (свет).

Активности: Наставник указује на то да позадина *Greenfoot* пројекта (тј. света) може бити и слика. Наставник објашњава ученицима да се као позадина може користити било једна слика која ће прекрити читаву површину света, било слика која одговара димензијама ћелије.

Наставник показује ученицима како да подесе слику за класу `MyWorld`. Ученици прате упутства наставника и раде заједно са њим, корак по корак.

8. Задатак **2.3.** (15 минута)

Циљ: Савладавање начина подешавања позадине света у *Greenfoot* пројекту.

Концепти за дискусију: позадина класе `World` (свет).

Активности: Наставник задаје ученицима задатак:

> **Задатак 2.3. (Припрема слике за свет):** Пронађите или креирајте одговарајућу слику за позадину света. Можете користити неку од понуђених слика или одабрати неку другу слику. За позадину можете користити једну велику слику, која ће прекрити читаву површину света, или једну мању слику, која ће се понављати.

Co-funded by the Erasmus+ Programme of the European Union

Наставник ученицима појашњава шта се од њих очекује. Наставник може да покаже ученицима довршене, унапред припремљене, примере. Наставник затим ученицима даје адресу или *git* наредбу путем које треба да преузму пројекат у оквиру којег треба да раде на задатку. Ученици овај прелиминаран пројекат могу да преузму и из *git* репозиторијума или са *Moodle* платформе.

Ученици решавају задатак самостално или у групама, а наставник прати рад ученика. Након тога, наставник решава задатак, корак по корак, док ученици прате његова упутства.

Опис решења:

- Пронађите или креирајте одговарајућу слику за позадину света. Можете користити неку од понуђених слика (десним кликом на класу `MyWorld` и избором опције `Set image...` у њеном контекстном менију) или одабрати неку другу слику (копирајте жељену слику у поддиректоријум `images` који се налази у оквиру директоријума вашег пројекта, а затим је изаберите на претходно описан начин).
- Као позадину можете користити једну већу слику која ће прекрити читаву површину света (израчунајте потребну величину слике на основу димензија света) или једну мању слику која ће се понављати (употребите квадратну слику величине једне ћелије).

[Commit: `1184980643db082cfdd6bde9984bceaddf010d49`]

Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the Erasmus+ Programme of the European Union

2.2.2. СЦЕНАРИО: **Дефинисање класа и креирање објеката кроз развој игара у *Greenfoot* окружењу**

Табела 4. Дефинисање класа и креирање објеката кроз развој игара у Greenfoot окружењу

Назив	Дефинисање класа и креирање објеката кроз развој игара у *Greenfoot* окружењу
Образовни циљеви и исходи учења	Након ове лекције, ученици би требало да разумеју основне концепте класе и објекта, атрибуте и методе.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **130** минута. 1. Основни појмови (25 минута) 2. Задатак **2.4.** (10 минута) 3. Задатак **2.5.** (30 минута) 4. Интерфејс (5 минута) 5. Методе (15 минута) 6. Задатак **2.6.** (30 минута) 7. Обнављање теоријских концепата (15 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Кроз овај наставни сценарио ученици постепено, путем неколико структурираних задатака, продубљују знање о концептима објектно-оријентисаног програмирања. Ученици ће најпре посветити 10 минута дефинисању класе **Enemy** (**Задатак 2.4.**). Следећих 30 минута ће ученици посветити креирању објекта, тј. инстанце класе **Enemy** (**Задатак 2.5.**), примењујући стечено знање о креирању и иницијализацији објеката. Током наредног петоминутног блока разматра се концепт интерфејса класе, са нагласком на дефинисању операција које објекат може да изврши. Током следећих 15 минута уводи се концепт методе и ученици уче о томе како објекти међусобно комуницирају путем позива одговарајућих метода. Наредни блок од 30 минута је посвећен практичној примени, те ученици решавају **Задатак 2.6.** како би утврдили знање о понашању објеката. У последњем блоку од 15 минута даје се теоријски осврт, односно рекапитулација кључних концепата (као што су класе, објекти, инстанце, интерна стања, интерфејси и методе), чиме се осигурава темељно савладавање постављених образовних циљева.
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе.

Co-funded by the Erasmus+ Programme of the European Union

	На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

2.2.2.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Основни појмови (25 минута)

Циљ: Утврђивање знања о класама, објектима и атрибутима.

Концепти за дискусију: класе, објекти и атрибути.

Активности: Током увода у лекцију наставник се, заједно са ученицима, осврће на претходно обрађене концепте. Наставник, кроз дискусију, појашњава појмове класе, објекта и атрибута.

Наставник задаје ученицима писани задатак да, у задатом тексту, идентификују класе, објекте и атрибуте.

Наставник бира ученика који треба да изложи своје решење. Остали ученици учествују у овој активности тако што изражавају своја мишљења.

Наставник постепено објашњава како се дефинише класа у *Greenfoot* окружењу. Ученици прате упутства наставника и раде заједно са њим, корак по корак.

2. Задатак **2.4.** (10 минута)

Циљ: Учешће ученика у раду на пројектном задатку.

Концепти за дискусију: дефинисање класе.

Активности: Наставник задаје ученицима задатак:

> **Задатак 2.4. (Дефинисање класе непријатеља):** Уведите новог протагонисту – непријатеља. Непријатељ ће напредовати према сфери (`Orb`) како би је најпре оштетио, а на крају и потпуно уклонио. Дефинишите нову класу, доделите јој погодан назив (нпр. `Enemy`) и придружите јој одговарајућу слику.

Наставник ученицима појашњава шта се од њих очекује. Наставник може да преузме унапред припремљен пример довршеног пројекта и покаже ученицима шта се од њих очекује. Наставник затим ученицима даје адресу или *git* наредбу путем које треба да преузму пројекат у оквиру којег треба да раде на задатку. Ученици овај прелиминаран пројекат могу да преузму и из *git* репозиторијума или са *Moodle* платформе.

Ученици решавају задатак самостално или у групама, а наставник прати рад ученика. Након тога, наставник решава задатак, корак по корак, док ученици прате његова упутства.

Опис решења:

> Дефинишите нову класу, као изведену класу класе `Actor` (десним кликом на класу `Actor` и избором опције `New subclass...` у њеном контекстном менију). Доделите јој погодан назив (нпр. `Enemy`) и придружите јој одговарајућу слику.
>
> **[Commit: 4981400623729c3d112b54454b6e6151e18426bf]**

Наставник треба да упути ученике у то да постоје одређене конвенције којих се треба придржавати када је реч о додељивању назива класама (нпр. да је пожељно да назив класе почне великим словом), али и да се осврне на целокупну конвенцију у погледу именовања у програмском језику *Java*.

Co-funded by the Erasmus+ Programme of the European Union

3. Задатак **2.5.** (30 минута)

Циљ: Разумевање концепта стања објекта. Овладавање начином креирања инстанце класе у *Greenfoot* окружењу.

Концепти за дискусију: стање објекта, инстанца.

Активности: Наставник објашњава концепт **стања објекта** путем једноставног примера. На пример, наставник стоји на одређеној удаљености од улаза у учионицу. Ова раздаљина одређује његов тренутни положај у учионици, с тим да се корачањем унапред, или уназад, сама раздаљина мења. Сходно томе, положај наставника у односу на улаз би се могао исказати путем променљиве чија ће се вредност мењати током времена. Дакле, положај наставника се дефинише растојањем од улаза, чиме се илуструје како стање неког објекта (у овом случају наставника) карактерише вредност његовог атрибута (растојање) у датом тренутку. Односно, стање неког објекта одређено је вредностима свих његових атрибута у датом тренутку.

Наставник излаже задатак:

> **Задатак 2.5. (Креирање инстанце класе непријатеља)**: Креирајте инстанцу класе `Enemy`. Прегледајте интерно стање креиране инстанце. Креирајте још једну инстанцу класе `Enemy` и поставите је на неку другу позицију. Упоредите интерна стања те две инстанце.

Наставник отвара последњу верзију пројекта, а ученици прате упутства наставника и раде заједно са њим, корак по корак.

Опис решења:

> Наставник креира инстанцу класе `Enemy` (десним кликом на класу `Enemy` и избором опције `new Enemy()` у њеном контекстном менију), а затим је поставља на сцену левим кликом на жељену позицију. Наставник затим демонстрира како се може прегледати интерно стање креиране инстанце (десним кликом на дату инстанцу на сцени и избором опције `Inspect` у њеном контекстном менију).
>
> Наставник тражи од ученика да креирају још једну инстанцу класе, али да је поставе на неку другу позицију, како би упоредили интерна стања те две инстанце.

4. Интерфејс (5 минута)

Циљ: Увођење концепта интерфејса као скупа активности које одређени објекат може извршити.

Концепти за дискусију: интерфејс.

Активности: Наставник уводи концепт **интерфејса** путем једноставних примера. На пример, уколико се посматра нека особа и активности које она обавља током дана (као што су буђење, доручковање, одлазак на посао итд.), без залажења у појединости (како ће се пробудити, где доручкује и шта једе за доручак и како иде на посао), скуп датих активности се може поистоветити са интерфејсом. Путем интерфејса се прописује које операције објекти одређене класе могу извршавати (односно методе које се могу над њим позивати), али се не прецизира начин на који се оне извршавају (тј. не прописује се на који начин ће неко бити пробуђен – будилник или телефонски позив, шта ће јести за доручак и да ли ће ићи на посао пешке, јавним превозом или колима).

Co-funded by the Erasmus+ Programme of the European Union

5. Методе (15 минута)

Циљ: Увођење концепта методе.

Концепти за дискусију: методе.

Активности: Наставник уводи концепт **методе** и објашњава да се путем методе заправо може приступити атрибутима или мењати стање објекта.

Пример 1: Ако се посматра класа **Особа** и њен атрибут **старост**, може се уочити да се вредност овог атрибута конзистентно повећава за један, сваке године, на исти дан. Са друге стране, атрибути као што су **висина** и **тежина** ће се чешће мењати јер деца расту и тежина им варира.

Пример 2: Ако се посматра кретање неке особе од тачке **А** до тачке **Б**, и оно опише путем корака – нпр. корак напред, окрет за 45 степени налево, 8 корака напред, окрет за 30 степени надесно и још 5 корака напред – ови појединачни кораци, тј. радње, се могу објединити у тзв. методу.

Наставник показује ученицима како да у *Greenfoot* окружењу пронађу методе одређене класе, које могу позвати над неким објектом дате класе.

Наставник ученицима показује методе које су дефинисане у класи `Actor`. Наставник затим демонстрира како се позивају методе над неким објектом.

6. Задатак **2.6.** (30 минута)

Циљ: Овладавање начином позивања метода над објектом.

Концепти за дискусију: позивање метода.

Активности: Наставник излаже задатак:

> **Задатак 2.6. (Позивање метода)**: Позовите одговарајућу методу над инстанцом класе `Enemy`, како би се она преместила на позицију `[12,6]` и била окренута надоле. Какав ће утицај то имати на интерно стање дате инстанце?

Наставник отвара последњу верзију пројекта, а ученици прате упутства наставника и раде заједно са њим, корак по корак.

Опис решења:

> Наставник позива одговарајућу методу над инстанцом класе `Enemy` (десним кликом на дати објекат на сцени и избором опције *`inherited from Actor`* у њеном контекстном менију, а затим избором методе `setLocation(int x, int y)`). Наставник објашњава ученицима шта ће се догодити након позива методе и како ће се променити интерно стање дате инстанце.

Наставник показује ученицима како се дефинишу методе. Наставник дефинише методу `setPosition(int x, int y)` која омогућава постављање инстанце класе `Actor` на одређену позицију (задату путем њених координата). Наставник указује на то да је ова метода заправо еквивалента методи `setLocation(int x, int y)`, те да је пре дефинисање неке нове методе пожељно проверити да ли је таква метода већ дефинисана, како би се избегло дуплирање. Наставник наглашава да назив методе треба да јасно одражава њену сврху и понашање, тј. да се оно може одмах разумети на основу назива. Наставник такође истиче да би назив методе требало да буде концизан и даје примере добро дефинисаних, лоше дефинисаних и неправилно дефинисаних метода. Методе које корисник може да изврши (тј. позове) над неким објектом видљиве су када се на дати објекат кликне десним тастером миша.

Наставник објашњава ученицима како да дефинишу методу. Ученици прате упутства наставника и раде заједно са њим, корак по корак.

Co-funded by the Erasmus+ Programme of the European Union

Наставник тражи од ученика да дефинишу методу која ће обезбедити да се `Actor` спусти (смањивањем вредности `y` координате), као и методу која ће обезбедити да се `Actor` попне (повећањем вредности `y` координате). Наставник прати рад ученика и, уколико је потребно, усмерава сваког појединачно.

7. Обнављање теоријских концепата (15 минута)

Циљ: Резимирање целокупне области.

Концепти за дискусију: класа, објекат, инстанца, интерно стање, интерфејс, метода.

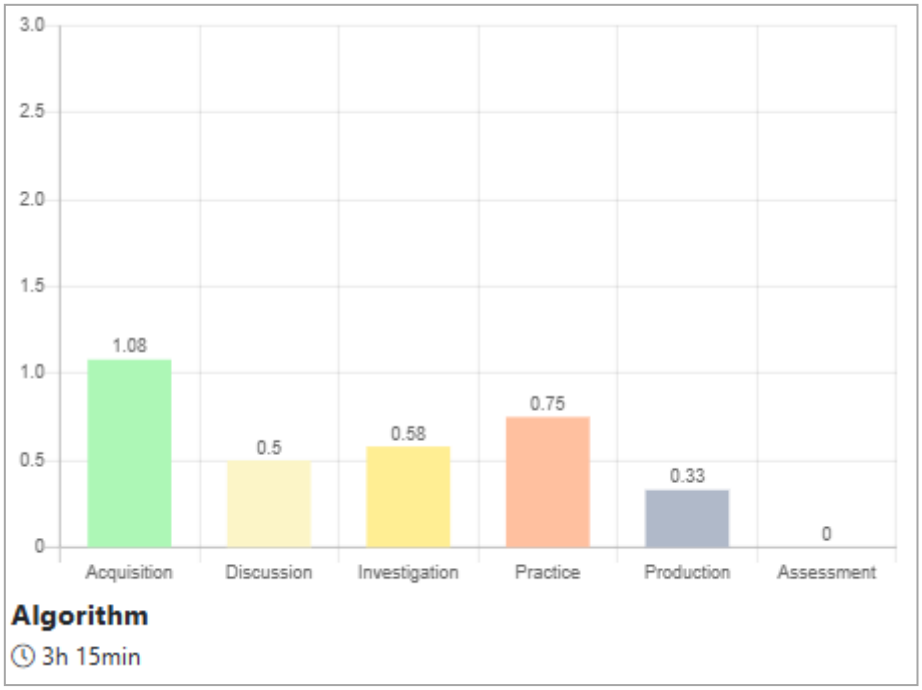
Активности: Наставник даје резиме лекције и осврће се, заједно са ученицима, на све претходно обрађене концепте.

Co-funded by the Erasmus+ Programme of the European Union

3. АЛГОРИТМИ

3.1. Структура наставних активности и задаци

На следећој слици је дат приказ расподеле ангажовања ученика према типу активности, на основу искуства наставника средњих школа у Словачкој Републици, а који је генерисан у алату за креирање наставних сценарија.



Слика 3. Предложена расподела ангажовања ученика према типу активности за тематску целину „Алгоритми“

Задатак 3.1. Дефинисање једноставног алгоритма

Напишите на папиру поступак помоћу којег ћете описати како пешак прелази улицу.

Задатак 3.2. Дефинисање општег алгоритма

Дефинишите општи алгоритам за припрему топлог напитка. Размислите о томе какви треба да буду улазни подаци да би дати алгоритам био општи.

Задатак 3.3. Позивање метода

Потребно је померити инстанцу `Enemy` класе за два корака унапред, када се позове `act()` метода. Затим креирајте додатне инстанце класе `Enemy` и позовите ову методу над сваком од инстанци. Да ли сте очекивали такав ефекат?

[Commit: `7ba327ebeba6a13be68d9d21cc7e74b0da376132`]

Задатак 3.4. Документовање метода

Додајте коментар за документовање методе `act()`.

[Commit: `68b1c82c7df2c7826f2d3f78373498569adab7e9`]

Co-funded by the Erasmus+ Programme of the European Union

Задатак 3.5. Документовање класа

Измените коментар за документовање `Enemy` класе. Додајте верзију класе и аутора. До којих промена је дошло у генерисаној *HTML* страници?

[Commit: `1a7a9f83c5271a7c0dfa46ce3b1ee65682b0c5e5`]

Задатак 3.6. Преглед документације

Истражите документацију класа `Actor` и `World`.

Задатак 3.7. Проучавање контрола *Greenfoot* окружења

Испробајте дугмад у главном прозору *Greenfoot* окружења. Креирајте више инстанци класе `Enemy`. Кликните на дугме `Act` - шта се догађа? Кликните на дугме `Run` - шта се догађа? Након што први пут кликните на дугме `Run`, кликните на дугме `Pause` - шта се догађа? Какав ефекат има померање клизача `Speed` на позивање `act()` методе, након клика на дугме `Run`? Шта се догађа када кликнете на дугме `Reset`?

Co-funded by the Erasmus+ Programme of the European Union

3.2. Наставни сценарији

Припремљена су **два** наставна сценарија за тематску целину „Алгоритми“.

3.2.1. СЦЕНАРИО: **Увод у алгоритме и алгоритамски начин размишљања**

Табела 5. Увод у алгоритме и алгоритамски начин размишљања

Назив	Увод у алгоритме и алгоритамски начин размишљања
Образовни циљеви и исходи учења	Након ове лекције, ученици би требало да су суштински разумели концепт алгоритма, усвојили алгоритамски начин размишљања, овладали вештинама пројектовања и имплементације основних алгоритама, и требало би да умеју да примене алгоритамске концепте за успешно решавање различитих проблема.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **90** минута. 1. Увод у једноставне алгоритме, алгоритам као низ корака (15 минута) 2. Задатак **3.1.** (20 минута) 3. Алгоритам и његове особине (15 минута) 4. Задатак **3.2.** (25 минута) 5. Развој алгоритама (15 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Кроз овај наставни сценарио ученици ће ући у свет алгоритама и алгоритамског начина размишљања. Лекција почиње уводом од 15 минута, чији је циљ упознавање ученика са основим аспектима алгоритама, са нагласком на њихов значај када је реч о решавању проблема. Након увода, ученици ће током 20 минута решавати **Задатак 3.1.** у оквиру којег ће се од њих очекивати да дефинишу једноставан алгоритам за решавање једног конкретног проблема. Ова активност омогућава ученицима да практично примене уведене концепте и усавршe своје алгоритамске вештине. Затим следи блок од 15 минута посвећен дискусији о алгоритмима и њиховим особинама. Теме које се обрађују укључују исправност и ефикасност, уз истицање значаја јасних и прецизних упутстава приликом пројектовања алгоритма. Како би продубили разумевање, ученици ће провести наредних 25 минута развијајући општи алгоритам за донекле сложенији проблем (**Задатак 3.2.**). Овај задатак треба да подстакне ученике на апстрактније и критичко размишљање, као и на примену алгоритамских принципа приликом решавања проблема из свакодневног живота.

Co-funded by the Erasmus+ Programme of the European Union

	У последњем блоку од 15 минута, ученици ће се усредсредити на развој алгоритама, анализу и унапређење својих алгоритама. Циљ је да се препознају могућа побољшања, оптимизује ефикасност и осигура стабилност алгоритма. Током целе лекције, ученици ће радити самостално или у мањим групама, како би се подстакла сарадња и заједничко учење. Активним учешћем у дефинисању и анализи алгоритама, ученици развијају критичко мишљење и способност алгоритамског решавања проблема. До краја лекције, ученици ће стећи дубље разумевање концепта алгоритма и усвојити алгоритамски начин размишљања, што ће их опремити вештинама неопходним за решавање сложених проблема на систематичан и ефикасан начин.
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

3.2.1.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Увод у једноставне алгоритме, алгоритам као низ корака (15 минута)

Циљ: Увођење концепта алгоритма**.**

Концепти за дискусију: основе алгоритама.

Активности: Наставник уводи концепт **алгоритма** путем примера из свакодневног живота. На пример, наставник може да пита ученике о њиховим јутарњим навикама, тј. шта раде од тренутка када се пробуде, па до тренутка када дођу у школу. Затим може да их пита да ли знају како се припрема пица или топли сендвич, односно да ли знају рецепт за припрему неког јела или колача.

Наставник повезује процес припреме јела или колача са развојем програма и наглашава да баш као што постоје рецепти за припрему јела, тако постоје и „рецепти“ за развој програма који се називају алгоритми.

Наставник закључује да алгоритам представља скуп корака који прописује, тј. дефинише, како се програм извршава.

Наставник даље објашњава да кораци не морају нужно да се извршавају узастопно, тј. један за другим, већ да извршавање одређених корака може зависити и од испуњености неког услова. Наставник тражи од ученика да наведу један такав пример (на пример, уколико дефинишемо алгоритам за припрему топлог сендвича, а немамо један од састојака, као што је шунка, али имамо сличан састојак, онда можемо или да га искористимо као замену или можемо да одемо у продавницу и купимо састојак који нам недостаје).

Наставник објашњава ученицима да се одређени кораци у алгоритму могу поновити и неколико пута. Наставник тражи од ученика да дају примере алгоритама у којима се одређени кораци понављају више пута.

2. Задатак **3.1.** (20 минута)

Циљ: Дефинисање једноставног алгоритма**.**

Концепти за дискусију: алгоритам.

Активности: Наставник задаје ученицима задатак:

> **Задатак 3.1. (Дефинисање једноставног алгоритма):** Напишите на папиру поступак помоћу којег ћете описати како пешак прелази улицу.

Наставник не појашњава додатно задатак, али прати начин на који ученици размишљају и како решавају задатак. Уколико неки ученик постави питање, које је од значаја за опис упутства за прелазак улице, наставник га похваљује и истиче зашто је дата информација значајна.

Након одређеног времена, наставник пита ученике да ли су обратили пажњу на то где пешак прелази улицу, тј. да ли прелази улицу на месту које је обележено као пешачки прелаз или не. Такође, наставник пита ученике да ли су обратили пажњу на то да ли постоји семафор на прелазу.

На крају, наставник бира неколико ученика који треба да прочитају своја упутства за прелазак улице.

3. Алгоритам и његове особине (15 минута)

Циљ: Упознавање са особинама алгоритама.

Концепти за дискусију: особине алгоритама.

Активности: Наставник упознаје ученике са особинама алгоритама. Наставник објашњава да се алгоритми могу и графички приказати, и показује неколико примера.

Co-funded by the Erasmus+ Programme of the European Union

4. Задатак **3.2.** (25 минута)

Циљ: Дефинисање општег алгоритма.

Концепти за дискусију: дефинисање алгоритама.

Активности: Наставник задаје ученицима задатак:

Задатак 3.2. (Дефинисање општег алгоритма): Дефинишите општи алгоритам за припрему топлог напитка. Размислите о томе какви треба да буду улазни подаци да би дати алгоритам био општи.

5. Развој алгоритама (15 минута)

Циљ: Овладавање начином развоја алгоритама.

Концепти за дискусију: алгоритам.

Активности: Наставник објашњава ученицима да, и у математици, постоје одређени алгоритми који се користе за решавање проблема. Наставник пита ученике да ли могу дати неки такав пример.

Пример који наставник представља ученицима се односи на израчунавање вредности сложенијег аритметичког израза, са неколико математичких операција, при чему је неопходно водити рачуна о њиховом приоритету.

Наставник такође наводи и друге примере, као што је склапање комада намештаја, који је испоручен са упутством за склапање. Други пример може да се односи на упутства која добијамо од *GPS* уређаја када желимо да, користећи навигацију, стигнемо из тачке **А** у тачку **Б**.

Наставник тражи од ученика да осмисле, и напишу на папиру, сопствени алгоритам, након чега неки од ученика представљају своја решења.

Co-funded by the Erasmus+ Programme of the European Union

3.2.2. СЦЕНАРИО: **Позивање метода у *Greenfoot* окружењу**

Табела 6. Позивање метода у Greenfoot окружењу

Назив	Позивање метода у *Greenfoot* окружењу
Образовни циљеви и исходи учења	Након ове лекције, ученици би требало да су суштински разумели начин позивања метода, уз посебан нагласак на методе *Greenfoot* окружења: `act()` и `move(int)`. Требало би да умеју да користе кључну реч `this` када желе да референцирају актуелни објекат одређене класе. Поред тога, требало би да су у потпуности овладали позивањем метода и да су разумели синтаксу и параметре које је неопходно обезбедити приликом позива методе. Даље, ученици би требало да су увидели значај документовања кôда и требало би да су способни да ефикасно документују *Java* кôд, и то на јасан и читљив начин. Коначно, требало би да су овладали контролама *Greenfoot* окружења које омогућавају прецизну манипулацију и интеракцију са елементима игре како би се осигурао занимљив и функционалан ток игре.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **110** минута. 1. Метода `act()` (10 минута) 2. Метода `move(int)` (20 минута) 3. Кључна реч `this` (5 минута) 4 .Задатак **3.3.** (10 минута) 5. Аутоматско довршавање кôда (5 минута) 6. Значај документовања програмског кôда (15 минута) 7. Задатак **3.4.** (5 минута) 8. Задатак **3.5.** (5 минута) 9. Задатак **3.6.** (10 минута) 10. Задатак **3.7.** (20 минута) 11. Дискусија: Алгоритам, особине, развој алгоритама, *Greenfoot* дугмад (5 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње блоком од 10 минута који је посвећен разумевању сврхе методе `act()`, а затим је током наредних 20 минута фокус на појашњавању методе `move(int)`, два кључна елемента за развој програма у *Greenfoot* окружењу. Ученици ће затим провести 5 минута разматрајући значај кључне речи `this` када се жели референцирати актуелни објекат у контексту одређене класе.

Co-funded by the Erasmus+ Programme of the European Union

	Након тога, ученици ће током 10 минута решавати **Задатак 3.3.** који има за циљ увежбавање начина позивања метода, уз коришћење одговарајуће синтаксе и параметара. Следи петоминутно упознавање са опцијом аутоматског довршавања кôда у *Greenfoot* окружењу, са нагласком на повећање ефикасности програмирања. Током наредних 15 минута ученици треба да схвате значај документовања кôда, као и то да јасна и концизна документација резултује бољом читљивошћу и лакшим одржавањем програмског кôда. Следи петоминутни задатак (**Задатак 3.4.**) у оквиру којег се од ученика тражи да документују свој кôд, водећи рачуна о томе да документација треба да буде разумљива, не само њима, већ и другима. Надовезујући се на овај задатак, ученици ће провести још 5 минута додајући детаље у документацију свог кôда (**Задатак 3.5.**). Током следећег десетоминутног блока, ученици имају задатак да прегледају и прочитају документацију других ученика, како би стекли увид у различите приступе и стилове програмирања (**Задатак 3.6.**). Пре завршне петоминутне дискусије, ученици ће провести 20 минута проучавајући контроле *Greenfoot* окружења, које омогућавају управљање током извршавања програма и прецизну манипулацију и интеракцију са елементима игре, како би се осигурао занимљив и функционалан ток игре (**Задатак 3.7.**). Током целе лекције, ученици ће радити самостално или у мањим групама, како би се подстакла сарадња и заједничко учење. Активним учешћем у дефинисању и анализи алгоритама, ученици развијају критичко мишљење и способност алгоритамског решавања проблема. До краја лекције, ученици ће стећи дубље разумевање начина позивања метода у програмском језику *Java*, документовања кôда и управљања током извршавања програма у *Greenfoot* окружењу, али и развити вештине кључне за развој игара, али и у другим доменима.
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

3.2.2.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Метода `act()` (10 минута)

Циљ: Разумевање сврхе методе `act()` и начина позивања метода.

Концепти за дискусију: метода `act()`.

Активности: Наставник отвара последњу верзију `TowerDefense` пројекта. Наставник поставља инстанцу `Enemy` класе на позицију `0`, `0`. Наставник пита ученике шта мисле да ће се догодити када се позове `act()` метода над инстанцом класе `Enemy`. Да ли је то очекивано понашање?

2. Метода `move(int)` (20 минута)

Циљ: Употреба методе `move(int)` за померање објекта унапред и уназад**.**

Концепти за дискусију: методе за померање објекта.

Активности: Наставник објашњава `move(int)` методу. Наставник мења позицију инстанце `Enemy` класе и позива методу прослеђујући јој различите позитивне вредности, на пример **1** и **3**. Наставник пита ученике шта мисле да ће се догодити уколико позову `move` методу и проследе јој негативну вредност, на пример `-1`.

Наставник задаје ученицима задатак да имплементирају методу `backward()` која треба да обезбеди да се објекат помери за **1** корак уназад. Да ли је могуће померити објекат за **2** корака или **X** корака уназад? Шта је потребно променити да би се то обезбедило?

3. Кључна реч `this` (5 минута)

Циљ: Схватање улоге кључне речи `this`**.**

Концепти за дискусију: кључна реч `this`.

Активности: Наставник објашњава кључну реч `this`. Помоћу кључне речи `this` добија се референца на актуелни објекат, путем које се може приступити атрибутима који су специфични за дати објекат или се могу позивати методе у контексту датог објекта.

4 .Задатак **3.3.** (10 минута)

Циљ: Савладавање начина имплементације методе за вертикално померање објекта.

Концепти за дискусију: позивање методе.

Активности: Наставник треба да замоли ученике да му помогну да реши следећи задатак:

> **Задатак 3.3. (Позивање метода):** Потребно је померити инстанцу `Enemy` класе за два корака унапред, када се позове `act()` метода. Затим креирајте додатне инстанце класе `Enemy` и позовите ову методу над сваком од инстанци. Да ли сте очекивали такав ефекат?
>
> **[Commit: `7ba327ebeba6a13be68d9d21cc7e74b0da376132`]**

Наставник показује ученицима како да реше овај задатак (додавањем позива `move(int)` методе у `act()` методу), док ученици прате упутства наставника и дају своје предлоге.

Наставник пита ученике како би померили објекат вертикално, тј. нагоре или надоле. Наставник затим тражи од ученика да пронађу погодну методу за померање нагоре/надоле тако што ће, након десног клика на дати објекат, одабрати опцију `Inherited from Actor`. Ученици би требало да препознају следеће методе: `turn`, `setRotation`, `setLocation`, `getLocation`, `getRotation`. Наставник затим задаје ученицима задатак да имплементирају методе `up()` и `down()`. Док ученици решавају задатак, наставник прати њихов рад, и када заврше са радом појашњава како се дате методе могу имплементирати. Методе `up()` и `down()` мењају вредност `y` координате, при чему је повећавају односно смањују, респективно. Наставник сада може увести концепт параметара методе, али без залажења у детаље.

Co-funded by the Erasmus+ Programme of the European Union

5. Аутоматско довршавање кôда (5 минута)

Циљ: Упознавање са опцијом аутоматског довршавања кôда (енгл. *autocomplete*) у *Greenfoot* окружењу.

Концепти за дискусију: аутоматско довршавање кôда (**CTRL+SPACE**).

Активности: Наставник показује ученицима како могу пронаћи методе које се могу позвати над одређеним објектом. Наставник истиче да ће им ова опција бити посебно корисна ако су заборавили како се нека метода тачно зове или ако су тек почели да програмирају и желе да „питају" *Greenfoot* окружење за помоћ како би брже написали кôд.

6. Значај документовања програмског кôда (15 минута)

Циљ: Разумевање улоге коментара и документације**.**

Концепти за дискусију: коментари и документација, прозор са документацијом.

Активности: Наставник упознаје ученике са специјалним линијама кôда које нису део самог програма, тј. које се не извршавају (коментари, коментари за документовање).

Наставник треба да истакне разлику између обичних коментара и документације (као посебног типа коментара). Наставник показује ученицима изворни кôд класе `Enemy`, а затим и генерисани *HTML* документ који представља документацију дате класе. Наставник овом приликом објашњава и да постоје одређена правила којих се треба придржавати када се пише документација за класе или методе. Наставник задаје ученицима задатак да истраже како се пише документација за *Java* класе и методе.

7. Задатак **3.4.** (5 минута)

Циљ: Упознавање са начином документовања метода**.**

Концепти за дискусију: ознаке (енгл. *tags*) за документовање метода.

Активности: Наставник задаје ученицима задатак:

> **Задатак 3.4. (Документовање метода):** Додајте коментар за документовање методе `act()`.
>
> **[Commit***:* `68b1c82c7df2c7826f2d3f78373498569adab7e9`**]**

8. Задатак **3.5.** (5 минута)

Циљ: Упознавање са начином документовања класа**.**

Концепти за дискусију: ознаке (енгл. *tags*) за документовање класа.

Активности: Наставник задаје ученицима задатак:

> **Задатак 3.5. (Документовање класа):** Измените коментар за документовање `Enemy` класе. Додајте верзију класе и аутора. До којих промена је дошло у генерисаној *HTML* страници?
>
> **[Commit:** `1a7a9f83c5271a7c0dfa46ce3b1ee65682b0c5e5`**]**

9. Задатак **3.6.** (10 минута)

Циљ: Овладавање начином прегледања документације класе, како би се разумело понашање њених метода.

Концепти за дискусију: проучавање документације.

Активности: Наставник задаје ученицима задатак:

> **Задатак 3.6. (Преглед документације):** Истражите документацију класа `Actor` и `World`.

Наставник наглашава значај прегледања документације како би се пронашле методе које би могле бити од користи.

Co-funded by the Erasmus+ Programme of the European Union

10. Задатак **3.7.** (20 минута)

Циљ: Упознавање са контролама доступним у *Greenfoot* окружењу**.**

Концепти за дискусију: дугмад у *Greenfoot* окружењу.

Активности: Наставник наставља рад на последњој верзији пројекта. Наставник тражи од ученика да креирају и додају два објекта класе `Enemy` и да позову методу `act()` сваке од инстанци.

Наставник затим указује на дугме `Act`. Наставник треба да кликне на `Act` дугме које се налази у главном прозору. Наставник поставља питања, како би ученици сами закључили и објаснили шта се догодило.

Наставник такође тражи од ученика да кликну на дугме `Run` и да закључе шта се догодило.

Наставник тражи од ученика да кликну на дугме `Reset` и објасне шта се догодило.

> **Задатак 3.7. (Проучавање контрола *Greenfoot* окружења):** Испробајте дугмад у главном прозору *Greenfoot* окружења. Креирајте више инстанци класе `Enemy`. Кликните на дугме `Act` - шта се догађа? Кликните на дугме `Run` - шта се догађа? Након што први пут кликните на дугме `Run`, кликните на дугме `Pause` - шта се догађа? Какав ефекат има померање клизача `Speed` на позивање `act()` методе, након клика на дугме `Run`? Шта се догађа када кликнете на дугме `Reset`?

Након тога, наставник објашњава шта је потребно урадити како би се, сваки пут када се кликне на `Reset` дугме, на екрану појавила два објекта класе `Enemy` и то на позицијама `(0,3)` и `(3,3)`. Наставник треба да објасни ученицима да је, уколико желе да се објекти појаве на сцени сваки пут када се кликне на `Reset` дугме, потребно изменити конструктор класе `World` тако да се унутар њега креирају објекти и постављају на жељене позиције.

11. Дискусија: Алгоритам, особине, развој алгоритама, *Greenfoot* дугмад (5 минута)

Циљ: Резимирање лекције**.**

Концепти за дискусију: методе за управљање кретањем, дугмад у *Greenfoot* окружењу, документација.

Активности: Наставник даје резиме лекције. Наставник затим може да истакне значај документовања кôда, као и да се осврне на конвенције за именовање метода и класа.

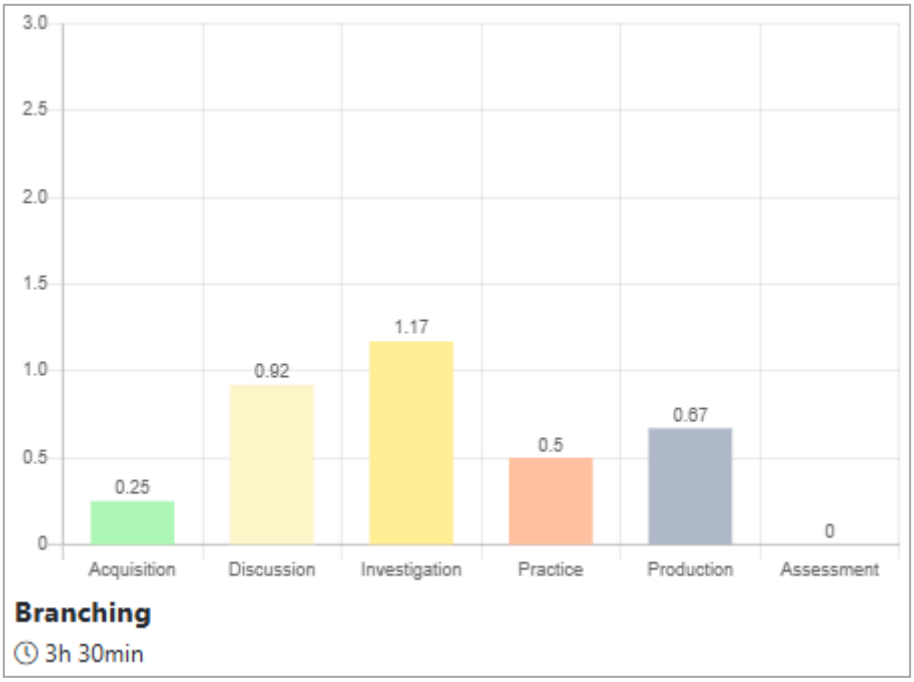
Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the Erasmus+ Programme of the European Union

4. ГРАНАЊЕ

4.1. Структура наставних активности и задаци

На следећој слици је дат приказ расподеле ангажовања ученика према типу активности, на основу искуства наставника средњих школа у Словачкој Републици, а који је генерисан у алату за креирање наставних сценарија.



Слика 4. Предложена расподела ангажовања ученика према типу активности за тематску целину „Гранање“

Задатак 4.1. Праћење промена интерног стања објекта

Креирајте инстанцу класе `Enemy` и поставите је у центар сцене. Отворите прозор у којем се приказује интерно стање дате инстанце и поставите га тако да буде видљив док је апликација покренута. Затим покрените апликацију и пратите промене у вредностима атрибута `x`, `y` и `rotation` дате инстанце `Enemy` приликом позивања различитих метода. Како се ове вредности мењају током померања (нагоре, надоле, улево и удесно) и окретања?

Задатак 4.2. Детектовање ивица света

Додајте кôд у тело методе `act()` како би се, када објекат класе `Enemy` стигне до ивице света, он окренуо за **180°** (позивањем методе `setRotation(int)`).

[Commit: `4927c3ff7eb39b51ba2738f2ab500fd6c32e3bb4`**]**

Задатак 4.3. Увођење нових класа

Уведите две нове класе, `Direction` и `Orb`, као изведене класе `Actor` класе. Припремите одговарајуће слике (максималне величине `50 x 50` пиксела) у неком графичком едитору. Затим придружите дате слике новим класама.

[Commit: `4ed6b37e6d481181d8b340639aa03391406b6c2e`**]**

Co-funded by the Erasmus+ Programme of the European Union

Задатак 4.4. Детектовање колизије

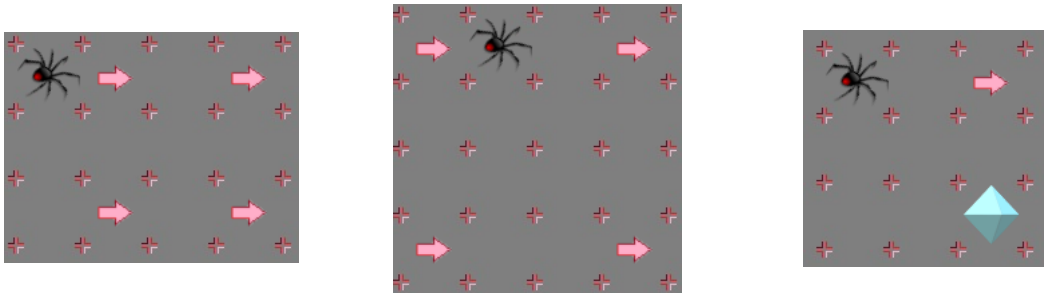
Додајте кôд у тело `act()` методе `Enemy` класе, како би се обезбедило да се:

- непријатељ окрене за **90°**, у смеру кретања казаљке на сату, ако се затекне у истој ћелији у којој се налази инстанца класе `Direction`.
- непријатељ окрене за **90°**, у смеру супротном од кретања казаљке на сату, ако се затекне у истој ћелији у којој се налази инстанца класе `Orb`.

[Commit: `968e6f195e3def25e11bc41b664ba1715f7da11d`**]**

Задатак 4.5. Предвиђање начина кретања инстанце непријатеља (произвољна почетна поставка)

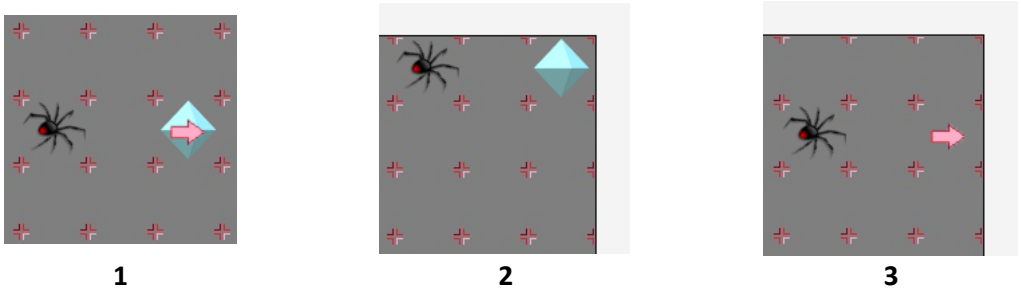
Припремите различите почетне поставке игре, инспирацију можете пронаћи на слици испод. Како ће се померати непријатељ? Покрените апликацију. Да ли је ваша претпоставка тачна? Ако није, зашто?



Слика 5. Примери почетних позиција инстанци

Задатак 4.6. Предвиђање начина кретања инстанце непријатеља (задата почетна поставка)

Размотрите почетну поставку на свакој од следећих слика. Како ће се померати непријатељ? Подесите апликацију (тј. поставите инстанце на одговарајуће почетне позиције) и покрените је. Да ли је ваша претпоставка тачна? Ако није, зашто?



Слика 6. Изазовније почетне поставке игре

Задатак 4.7. Имплементација потпуног и вишеструког гранања

Измените тело `act()` методе `Enemy` класе тако што ћете увести потпуно и угњеждено гранање. Потребно је дефинисати угњеждене услове. Детектовање да ли је непријатељ дошао до ивице света треба да буде примарни услов, затим треба проверити да ли додирује инстанцу класе `Direction` и, на крају, да ли додирује инстанцу класе `Orb`.

[Commit: `f017de8b49d4fc77f62afac4d842429560bcfb8b`**]**

Задатак 4.8. Предвиђање начина кретања инстанце непријатеља (промењено понашање)

Прођите поново кроз **Задатак 4.5.** и **Задатак 4.6.** Шта се променило?

Co-funded by the Erasmus+ Programme of the European Union

4.2. Наставни сценарији

Припремљена су **два** наставна сценарија за тематску целину „Гранање“.

4.2.1. СЦЕНАРИО: **Непотпуно гранање у *Greenfoot* окружењу**

Табела 7. Непотпуно гранање у Greenfoot окружењу

Назив	Непотпуно гранање у *Greenfoot* окружењу
Образовни циљеви и исходи учења	Наставни сценарио покрива непотпуно гранање (вишеструко гранање је намерно изостављено). Ученици ће умети да пишу кôд користећи услове. Након ове лекције, ученици ће знати како да употребе једноставне `if` наредбе и како да користе услове у кôду, како би контролисали понашање своје игре. Стећи ће основно знање о програмском језику *Java*, савладати неопходну синтаксу и научити како да анализирају и разумеју програмски кôд, што ће им помоћи да разумеју зашто се њихова игра понаша на одређени начин и како да разреше проблеме. Ученици ће бити способни да развију сопствене пројекте игара примењујући стечена знања о објектно-оријентисаном програмирању.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **50** минута. 1. Увод (5 минута) 2. Тумачење кôда (15 минута) 3. Непотпуно гранање (10 минута) 4. Задатак **4.1.** (10 минута) 5. Задатак **4.2.** (10 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње краћим уводом, након ког следи петнаестоминутни блок посвећен тумачењу кôда метода `turn(int)` и `setRotation(int)` класе `Actor`, чиме се успоставља основа за увођење концепта непотпуног гранања. Наставник усмерава дискусију са циљем да осигура да сви ученици истоветно разумеју улогу ове методе. Након тога, следи десетоминутни блок посвећен савладавању основа непотпуног гранања. Овај блок је кључан како би ученици усвојили суштинске принципе пре него што се упусте у писање кôда. Ученици затим учествују у десетоминутном истраживачком задатку (**Задатак 4.1.**), у оквиру којег анализирају интерно стање протагонисте, како би развили способност праћења и разумевања тока извршавања програма на основу изворног кôда.

Co-funded by the Erasmus+ Programme of the European Union

	Наредни десетоминутни блок посвећен је практичном раду (**Задатак 4.2.**) и од ученика се очекује да у свој пројекат додају кôд који омогућава детектовање ивица света.
Вредновање	Активности у оквиру ове лекције ће омогућити наставницима да дају ученицима повратне информације на основу формативног вредновања њиховог учешћа у спроведеним дискусијама, али и њиховог рада у окружењу обрнуте учионице (енгл. *flipped classroom*), као и рада у тиму. Међусобно вредновање ће се спровести *online*, као део домаћег задатка. Кроз ову активност ученици ће се подсетити важних аспеката лекције, биће подстакнути да критички вреднују решења других ученика, имаће увид у добра и мање добра решења итд., што доприноси и повећању општег достигнућа постављених образовних циљева и исхода учења целе групе. Исходе учења и стечена знања ће ученици даље примењивати и током развоја тимског пројекта на којем раде. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

4.2.1.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Увод (5 минута)

Циљ: Утврђивање знања о претходно обрађеним, концептима. Представљање циљева лекције.

Концепти за дискусију: алгоритам, документовање програмског кôда.

Активности: Током увода у лекцију наставник се, заједно са ученицима, осврће на претходно обрађене концепте. Наставник затим представља нову област коју ће са ученицима обрађивати током лекције. Како би илустровао предмет лекције, наставник представља пример алгоритма са једноставним гранањем. Пример би могао бити алгоритам за прелазак улице на пешачком прелазу без семафора. Пешак не прелази одмах улицу, већ најпре проверава да ли има возила која долазе са леве или десне стране. Ако нема возила, пешак прелази улицу.

2. Тумачење кôда (15 минута)

Циљ: Анализа појединих метода класе `Actor`.

Концепти за дискусију: интерно стање објекта.

Активности: Наставник преузима најновију верзију пројекта:

- са *Moodle* платформе или
- из *git* репозиторијума.

Наставник креира и поставља објекат класе `Enemy` на неко место на сцени. Наставник затим појашњава поједине методе класе `Actor`:

- `move(int)`
- `turn(int)`
- `setRotation(int)`

Током појашњавања самих метода, наставник указује на то како се мењају одређени атрибути објекта (на пример, позиција објекта на сцени, тј. вредности `x` и `y` координата). Наставник затим заједно са ученицима разматра како би се могла допунити метода `act()` да би се, сваки пут када се она позове, објекат класе `Enemy` померио за два корака унапред.

3. Непотпуно гранање (10 минута)

Циљ: Усвајање концепта непотпуног гранања**.**

Концепти за дискусију: гранање, непотпуно гранање.

Активности: Наставник наставља рад на пројекту. Поставља објекат класе `Enemy` на сцену. Наставник објашњава ученицима како могу да провере да ли се дати објекат налази у горњој половини сцене, а онда да прикажу поруку: "**`Pronadjen`**" (енгл. "**`Found`**").

Наставник ученицима показује методу `showText` која служи за приказивање текста на екрану.

4. Задатак **4.1.** (10 минута)

Циљ: Развијање способности праћења и разумевања тока извршавања програма.

Концепти за дискусију: интерно стање објекта.

Активности: Наставник показује ученицима како се могу пратити промене интерног стања објекта:

> **Задатак 4.1. (Праћење промена интерног стања објекта)**: Креирајте инстанцу класе `Enemy` и поставите је у центар сцене. Отворите прозор у којем се приказује интерно стање дате инстанце и поставите га тако да буде видљив док је апликација покренута.

Co-funded by the Erasmus+ Programme of the European Union

Затим покрените апликацију и пратите промене у вредностима атрибута `x`, `y` и `rotation` дате инстанце `Enemy` приликом позивања различитих метода. Како се ове вредности мењају током померања (нагоре, надоле, улево и удесно) и окретања?

Ученици одговарају на постављена питања.

5. Задатак **4.2.** (10 минута)

Циљ: Осмишљавање и имплементација поступка детектовања досезања ивица света**.**

Концепти за дискусију: детектовање ивица света (`World`).

Активности: Наставник разговара са ученицима о томе како могу утврдити да ли се неки објекат налази, или не налази, на ивици (енгл. *edge*) света (`World`). На пример, ако су познате тачне димензије света, на основу позиције `(x,y)` може се утврдити да ли се објекат налази или не налази на његовој ивици.

Наставник поставља инстанцу класе `Enemy` било где на сцени (али не на њеној ивици) и позива методу `isAtEdge()`. Наставник са ученицима дискутује о томе шта се догодило. Наставник затим премешта објекат на ивицу и указује на резултат извршавања методе `isAtEdge()`, која би сада требало да врати вредност `true`.

Наставник појашњава методу `isAtEdge()`.

Наставник задаје ученицима задатак:

> **Задатак 4.2. (Детектовање ивица света)**: Додајте кôд у тело методе `act()` како би се, када објекат класе `Enemy` стигне до ивице света, он окренуо за **180°** (позивањем методе `setRotation(int)`).

Наставник разговара са ученицима о томе како објекат који стигне до ивице света, може да настави своје кретање:

- уназад (без окретања)
- уназад (са окретањем).

Наставник, заједно са ученицима, допуњује програмски кôд `act()` методе.

Након тога, потребно је покренути `act()` методу и прокоментарисати са ученицима шта се догађа са објектом класе `Enemy` када он стигне до ивице света.

[Commit*:* `4927c3ff7eb39b51ba2738f2ab500fd6c32e3bb4`**]**

Co-funded by the Erasmus+ Programme of the European Union

4.2.2. СЦЕНАРИО: **Потпуно гранање у *Greenfoot* окружењу**

Табела 8. Потпуно гранање у Greenfoot окружењу

Назив	Потпуно гранање у *Greenfoot* окружењу
Образовни циљеви и исходи учења	Наставни сценарио покрива непотпуно и потпуно гранање, као и вишеструко гранање. Ученици ће умети да пишу кôд користећи сложеније услове. Након ове лекције, ученици ће знати како да употребе сложене `if-else` наредбе и како да користе специфичније и софистицираније услове да би управљали понашањем своје игре. Стећи ће додатно знање о програмском језику *Java*, савладати неопходну синтаксу и научити како да анализирају и разумеју сложенији програмски кôд, што ће им помоћи да разумеју зашто се њихова игра понаша на одређени начин и како да разреше проблеме. Ученици ће бити способни да развију сопствене пројекте игара примењујући нова сазнања.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **170** минута. 1. Задатак **4.3.** (30 минута) 2. Детектовање колизије (30 минута) 3. Задатак **4.4.** (10 минута) 4. Задатак **4.5.** (15 минута) 5. Задатак **4.6.** (15 минута) 6. Тумачење кôда: Потпуно гранање (15 минута) 7. Задатак **4.7.** (20 минута) 8. Задатак **4.8.** (30 минута) 9. Обнављање теоријских концепата (5 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње 30-минутном практичном вежбом (**Задатак 4.3.**) током које ученици уводе нове класе: `Direction` и `Orb`. Овај задатак помаже ученицима да боље разумеју објектно-оријентисану природу програмског језика *Java* и увиде значај правилног структурирања програма. Током наредног блока од 20 минута разматра се начин детектовања колизије, како би се помогло ученицима да боље разумеју начин интеракције објеката у окружењу игре. Током наредних 10 минута ученици имају задатак да додају детекцију колизије у свој пројекат, и практично примене стечено знање о гранању (**Задатак 4.4.**). Следе истраживачки задаци од по 15 минута, у оквиру којих се од ученика тражи да предвиде начин кретања објеката класе `Enemy` са произвољном (**Задатак 4.5.**) и изазовнијом почетном поставком игре (**Задатак 4.6.**), кроз које развијају способност решавања проблема и аналитичке вештине.

Co-funded by the Erasmus+ Programme of the European Union

	Наредна петнаестоминутна дискусија је усмерена на комплетно гранање, како би се осигурало да ученици увиђају разлику између непотпуног и потпуног гранања кôда. Наредни блок од 20 минута је посвећен практичном раду и од ученика се очекује да имплементирају сложено гранање са детекцијом колизије, како би кроз практичну примену сложенијих концепата утврдили стечено знање (**Задатак 4.7.**). Лекција се приводи крају изазовним истраживачким задатком (**Задатак 4.8.**) од 30 минута у оквиру којег се од ученика тражи да поново предвиде начин кретања непријатеља, овог пута са додатним искуством које су стекли решавањем претходних задатака, како би применили целокупно стечено знање. Лекција се завршава петоминутним теоријским осврtом, односно рекапитулацијом обрађених концепата.
Вредновање	Активности у оквиру ове лекције ће омогућити наставницима да дају ученицима повратне информације на основу формативног вредновања њиховог учешћа у спроведеним дискусијама, али и њиховог рада у окружењу обрнуте учионице (енгл. *flipped classroom*), као и рада у тиму. Међусобно вредновање ће се спровести *online*, као део домаћег задатка. Кроз ову активност ученици ће се подсетити важних аспеката лекције, биће подстакнути да критички вреднују решења других ученика, имаће увид у добра и мање добра решења итд., што доприноси и повећању општег достигнућа постављених образовних циљева и исхода учења целе групе. Исходе учења и стечена знања ће ученици даље примењивати и током развоја тимског пројекта на којем раде. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

4.2.2.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Задатак **4.3.** (30 минута)

Циљ: Увежбавање начина додавања нових класа у пројекат.

Концепти за дискусију: класа.

Активности: Наставник задаје ученицима задатак:

> **Задатак 4.3. (Увођење нових класа)**: Уведите две нове класе, `Direction` и `Orb`, као изведене класе `Actor` класе. Припремите одговарајуће слике (максималне величине `50 x 50` пиксела) у неком графичком едитору. Затим придружите дате слике новим класама.

Док ученици решавају задатак наставник прати њихов рад, а затим бира једног ученика који треба да представи и објасни своје решење.

[Commit*:* `4ed6b37e6d481181d8b340639aa03391406b6c2e`**]**

2. Детектовање колизије (30 минута)

Циљ: Разматрање начина детектовања колизије**.**

Концепти за дискусију: детектовање колизије.

Активности: Наставник поставља инстанцу класе `Enemy` на одређену позицију, а затим поставља инстанцу класе `Direction` у исти ред сцене тј. света. Наставник затим у `act()` методу додаје кôд путем које се објекат помера један корак унапред.

Наставник објашњава ученицима како да утврде да ли се два, или више, објеката („протагониста“) налазе на истој позицији (тј. у истој ћелији) света. Наставник разјашњава методу `isTouching()`.

Наставник, заједно са ученицима, мења `act()` методу `Enemy` класе како би се обезбедило да се непријатељ окрене за **90°**, у смеру кретања казаљке на сату, ако се затекне у истој ћелији у којој се налази инстанца класе `Direction`.

Наставник, заједно са ученицима, прати шта се догађа са вредношћу атрибута `rotation`.

3. Задатак **4.4.** (10 минута)

Циљ: Утврђивање начина детектовања колизије.

Концепти за дискусију: /.

Активности: Наставник задаје ученицима задатак:

> **Задатак 4.4. (Детектовање колизије)**: Додајте кôд у тело `act()` методе `Enemy` класе, како би се обезбедило да се:
>
> - непријатељ окрене за **90°**, у смеру кретања казаљке на сату, ако се затекне у истој ћелији у којој се налази инстанца класе `Direction`.
> - непријатељ окрене за **90°**, у смеру супротном од кретања казаљке на сату, ако се затекне у истој ћелији у којој се налази инстанца класе `Orb`.

Док ученици решавају задатак наставник прати њихов рад, а затим бира једног ученика који треба да представи и објасни своје решење.

[Commit*:* `968e6f195e3def25e11bc41b664ba1715f7da11d`**]**

Co-funded by the Erasmus+ Programme of the European Union

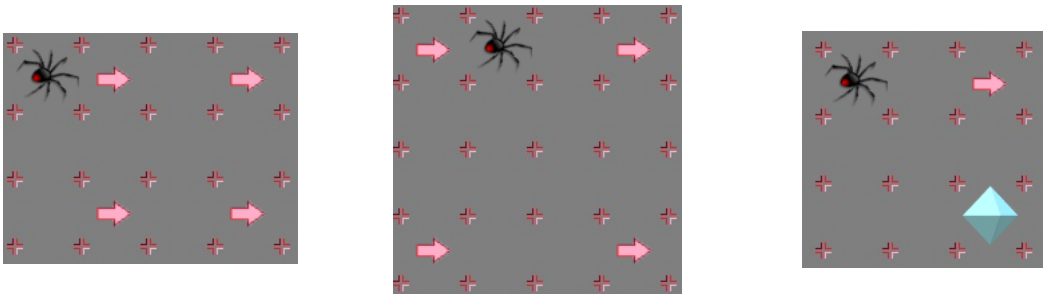
4. Задатак **4.5.** (15 минута)

Циљ: Разматрање начина кретања инстанце непријатеља.

Концепти за дискусију: начин кретања инстанце непријатеља.

Активности: Наставник задаје ученицима задатак:

Задатак 4.5. (Предвиђање начина кретања инстанце непријатеља): Припремите различите почетне поставке игре, инспирацију можете пронаћи на слици испод. Како ће се померати непријатељ? Покрените апликацију. Да ли је ваша претпоставка тачна? Ако није, зашто?



1 **2** **3**

Слика 7. Примери почетних позиција инстанци

Док ученици решавају задатак наставник прати њихов рад, а затим бира једног ученика који треба да представи и објасни своје решење.

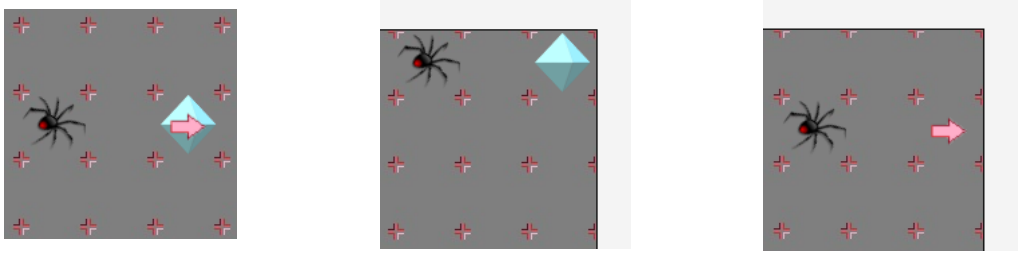
5. Задатак **4.6.** (15 минута)

Циљ: Утврђивање знања о начину кретања инстанце непријатеља.

Концепти за дискусију: начин кретања инстанце непријатеља.

Активности: Наставник задаје ученицима задатак:

Задатак 4.6. (Предвиђање начина кретања инстанце непријатеља): Размотрите почетну поставку на свакој од следећих слика. Како ће се померати непријатељ? Подесите апликацију (тј. поставите инстанце на одговарајуће почетне позиције) и покрените је. Да ли је ваша претпоставка тачна? Ако није, зашто?



1 **2** **3**

Слика 8. Изазовније почетне поставке игре

Док ученици решавају задатак наставник прати њихов рад, а затим бира једног ученика који треба да представи и објасни своје решење.

Co-funded by the Erasmus+ Programme of the European Union

6. Тумачење кôда: Потпуно гранање (15 минута)

Циљ: Упознавање са потпуним и вишеструким гранањем**.**

Концепти за дискусију: `if-else` наредба, угњеждена `if-else` наредба, `switch` наредба.

Активности: Наставник тражи од ученика да опишу, на папиру, како пешак прелази улицу. Наставник прати рад ученика. Након одређеног времена, наставник напомиње да је потребно обратити пажњу на то да ли на месту преласка улице постоји семафор. Уколико је неки ученик у међувремену већ поставио ово, или неко слично, питање, наставник га похваљује и наглашава да је веома важно да се, пре него што се приступи имплементацији, увек прво изанализира проблем и идентификују све могуће ситуације. Кроз ову активност би ученици требало да савладају потпуно и угњеждено гранање.

Наставник ће затим замолити ученике да поставе објекте класе `Orb` и класе `Direction` на различите ивице света. Наставник треба да укаже на то да се може десити да истовремено буду испуњена два различита услова: непријатељ додирује инстанцу класе `Orb` или `Direction`, али додирује и ивицу света. Цртањем овакве ситуацију на табли или на папиру, ученици треба да увиде да је потребно увести потпуно и угњеждено гранање, након чега треба да прилагоде понашања непријатеља (тј. да промене методу `act()` класе `Enemy`).

7. Задатак **4.7.** (20 минута)

Циљ: Савладавање начина имплементације потпуног и вишеструког гранања**.**

Концепти за дискусију: угњеждена `if-else` наредба, `switch` наредба.

Активности: Наставник задаје ученицима задатак:

> **Задатак 4.7. (Имплементација потпуног и вишеструког гранања)**: Измените тело `act()` методе `Enemy` класе тако што ћете увести потпуно и угњеждено гранање. Потребно је дефинисати угњеждене услове. Детектовање да ли је непријатељ дошао до ивице света треба да буде примарни услов, затим треба проверити да ли додирује инстанцу класе `Direction` и, на крају, да ли додирује инстанцу класе `Orb`.

Док ученици решавају задатак наставник прати њихов рад, а затим бира једног ученика који треба да представи и објасни своје решење.

[Commit*:* `f017de8b49d4fc77f62afac4d842429560bcfb8b`**]**

8. Задатак **4.8.** (30 минута)

Циљ: Предвиђање кретања непријатеља са произвољном почетном поставком**.**

Концепти за дискусију: понашање објеката.

Активности: Наставник поставља објекте на сцену насумично, а ученици треба да објасне њихово кретање и понашање (појединачно или у паровима).

Наставник задаје ученицима задатак:

> **Задатак 4.8. (Предвиђање начина кретања инстанце непријатеља)**: Прођите поново кроз **Задатак 4.5.** и **Задатак 4.6.** Шта се променило?

9. Обнављање теоријских концепата (5 минута)

Циљ: Резимирање целокупне области**.**

Концепти за дискусију: гранање.

Активности: Наставник даје резиме лекције и осврће се, заједно са ученицима, на све претходно обрађене концепте.

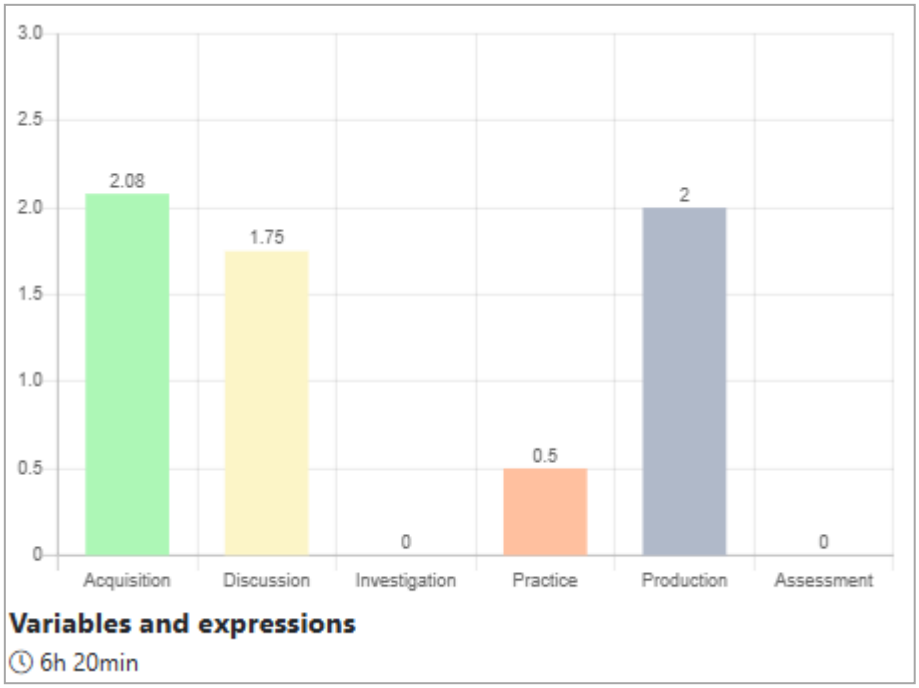
Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the Erasmus+ Programme of the European Union

5. ПРОМЕНЉИВЕ И ИЗРАЗИ

5.1. Структура наставних активности и задаци

На следећој слици је дат приказ расподеле ангажовања ученика према типу активности, на основу искуства наставника средњих школа у Словачкој Републици, а који је генерисан у алату за креирање наставних сценарија.



Слика 9. Предложена расподела ангажовања ученика према типу активности за тематску целину „Променљиве и изрази“

Задатак 5.1. Ротирање објеката у правцу који најпре треба одредити

Промените тело `act()` методе класе `Enemy` тако да обезбедите да се њена инстанца ротира у правцу који је дефинисан у инстанци класе `Direction` коју додирује (оне треба да имају исту ротацију). Позовите методу `getOneIntersectingObject(_cls_)` и сачувајте резултујућу инстанцу у погодној локалној променљивој (нпр. `Direction direction`) – неопходно је пре тога извршити конверзију, јер метода заправо враћа објекат типа `Actor`; дакле, методу треба позвати на следећи начин: `(Direction)getOneIntersectingObject(_cls_)`. Ако је добијен одговарајући резултат, позовите методу `getDirection()` како бисте добили неопходну информацију о ротацији (сачувајте је, ако желите), а затим подесите ротацију инстанце непријатеља (`this`) помоћу методу `setDirection(int)`. Тестирајте своје решење.

[Commit: `97dddc4beba40ac785c7413bb245ba849cd956d2`**]**

Задатак 5.2. Промена назива класе `MyWorld` у `Arena`

Преименујте претходно дефинисану класу `MyWorld`. Нов назив класе треба да буде `Arena`. Поред тога, неопходно је да преименујете и конструктор класе `MyWorld()` у `Arena()`.

[Commit: `aaf73c9bfd9f76a2a1e504f5e78d2976f1cada12`**]**

Co-funded by the Erasmus+ Programme of the European Union

Задатак 5.3. Креирање сопствене почетне поставке протагониста у арени

Креирајте сопствену почетну поставку протагониста у арени. Протагонисти треба да буду постављени на одговарајуће позиције унутар конструктора класе `Arena`. Потребно је креирати једну инстанцу класе `Enemy`, једну инстанцу класе `Orb` и барем једну инстанцу класе `Direction`. Након што декларишете и иницијализујете променљиве жељеног типа, потребно је да подесите атрибуте креираних објекта позивањем одговарајућих метода. Коначно, креиране објекте треба да поставите на жељене позиције у арени позивањем методе `addObject(Actor,int,int)`. Тестирајте своје решење.

[Commit*:* `8b105ea2eaf697f08c321efe687ddd31e2d0a041`**]**

Задатак 5.4. Идентификовање проблема везаног за кретања непријатеља и предлог решења

Да ли уочавате неки проблем везан за кретања непријатеља? Шта је узрок? Како би се могао решити?

Задатак 5.5. Додавање атрибута `moveDelay` у класу `Enemy`

Потребно је додати атрибут `moveDelay` (типа `int`) у класу `Enemy`. Потребно је и имплементирати параметризовани конструктор који треба да обезбеди да се дати атрибут иницијализује са вредношћу параметра. Коначно, потребно је ажурирати и кôд у класи `Arena` у складу са уведеним изменама.

[Commit*:* `6092489ce57541e77ae4e2ee886b20853df9f8a4`**]**

Задатак 5.6. Имплементација кретања непријатеља уз задршку

Потребно је изменити методу `act()` класе `Enemy`, тако да се обезбеди да ће се непријатељ померити тек након што се метода `act()` изврши одређени број (`moveDelay`) пута. Такође је потребно увести нов атрибут `nextMoveCounter` типа `int` и иницијализовати га са **0**. Затим треба изменити методу `act()` тако да позива `this.move(1)` само ако атрибут `nextMoveCounter` има вредност **0**. Након померања, атрибут `nextMoveCounter` добија вредност атрибута `moveDelay`. Када непријатељ не може да се помери (јер `nextMoveCounter` још није достигао вредност **0**), потребно је смањити вредност атрибута `nextMoveCounter` за **1**.

[Commit*:* `bf26e6ed23911ccb712fae3e243cdedff3a89a7f`**]**

Задатак 5.7. Имплементација параметризованог конструктора класе `Direction`

Дефинишите параметризовани конструктор у класи `Direction`. Конструктор треба да прима један параметар: `rotation` типа `int`. У телу конструктора је потребно ротирати креирану инстанцу у складу са вредношћу параметра. Потребно је ажурирати и кôд у класи `Arena` у складу са уведеним изменама.

[Commit*:* `3c4b9ef57ab17bac2a0abc7fc5e76ea4b6e27e4b`**]**

Задатак 5.8. Преклапање конструктора у класи `Direction`

Дефинишите додатни непараметризовани конструктор у класи `Direction`. У телу овог конструктора је потребно позвати претходно дефинисани параметризовани конструктор са аргументом постављеним на **0** (ово одговара ротацији од **0°**). Потребно је ажурирати и програмски кôд у класи `Arena` позивајући непараметризовани конструктор класе `Direction`, где је то могуће.

[Commit*:* `1e67e67523c66acea4e93363c9a3173302f424c8`**]**

У овој фази пројекта отвара се простор за дефинисање додатних домаћих задатака. У том смислу, могу се увести нове класе, изрази и вредности, како би се обезбедило додатно понашање. Наставник може да поразговара са ученицима о различитим могућностима, а затим им може задати, као домаћи задатак, да имплементирају одређено понашање.

Co-funded by the Erasmus+ Programme of the European Union

5.2. Наставни сценарији

Припремљено је **пет** наставних сценарија за тематску целину „Промењиве и изрази“.

5.2.1. СЦЕНАРИО: **Увод у промењиве и типове података у *Greenfoot* окружењу**

Табела 9. Увод у промењиве и типове података у Greenfoot окружењу

Назив	Увод у промењиве и типове података у *Greenfoot* окружењу
Образовни циљеви и исходи учења	После ове лекције, ученици ће умети правилно да користе промењиве и типове података. Разумевање ових концепата се продубљује кроз развој игре у *Greenfoot* окружењу, чиме се подстиче креативност, тимски дух и страственији приступ програмирању.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **45**-минута. 1. Увод (10 минута) 2. Идентификација промењивих (5 минута) 3. Типови података (15 минута) 4. Декларација промењивих (10 минута) 5. Иницијализација промењивих (5 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Кроз овај наставни сценарио ученици ће се упознати са принципима програмирања који се односе на промењиве и типове података, а из перспективе развоја игара у *Greenfoot* окружењу. Лекција почиње десетоминутним уводом током којег наставник успоставља контекст, надовезујући се на претходне лекције, који ће бити основа за увођење и дефинисање концепта промењиве. Након тога следи петоминутни блок, током којег ученици, заједно са наставником, промишљају и идентификују промењиве за своју игру. Будући да свака промењива мора бити неког типа, током наредних 15 минута се обрађују различити типови података. Кључне активности обухватају десетоминутни блок, који води наставник, а током којег ученици декларишу промењиве за своју игру и уче о значају имена и типа промењиве; након чега следи петоминутни блок посвећен иницијализацији промењивих, тј. додељивању вредности промењивима. У том контексту, може се променити понашање објеката у игри (нпр. ротација објекта, кретање објекта). Ученици ће наставити рад на игри коју су започели током претходних лекција. Сходно томе, до краја ове лекције у игру ће бити уведени нови концепти, који се односе на промењиве и типове података.

Co-funded by the Erasmus+ Programme of the European Union

Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

5.2.1.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Увод (10 минута)

   У уводном делу се успоставља контекст у односу на претходне лекције. Наставник уводи појам **променљиве**.

2. Идентификација променљивих (5 минута)

   Циљ: Идентификовање променљивих кроз дискусију, са нагласком на улогу променљивих у програмирању.

   Концепти за дискусију: променљиве и вредности.

   Активности:
   - Наставник уводи појам променљиве.
   - Од ученика се може затражити да промисле и идентификују променљиве за своју игру.
   - Наставник и ученици могу дискутовати о променљивима.
   - Тип променљиве се, током ове активности, може занемарити (или начелно споменути).

3. Типови података (15 минута)

   Циљ: Разумевање концепта типа података и препознавање примера његове примене у свакодневном животу, а затим фокусирање на типове променљивих који су потребни за даљи развој игре.

   Концепти за дискусију: променљиве, типови података, примери типова података који се могу пронаћи у свакодневном животу, типови променљивих потребни за игру.

   Активности:
   - Наставник уводи појам **типа података**.
   - Можете поразговарати са ученицима о примерима из свакодневног живота (нпр. цели бројеви се могу довести у везу са бројем присутних ученика, реални бројеви се могу довести у везу са ценом неког производа, текстуални тип се може довести у везу са именом ученика итд.).
   - Размотрити типове података у контексту *Greenfoot* окружења и програмског језика *Java*.
   - Детаљно продискутовати типове променљивих који су потребни за даљи развој игре.

4. Декларација променљивих (10 минута)

   Циљ: Примена знања о типовима података и променљивима како би се декларисале променљиве неопходне за развој игре.

   Концепти за дискусију: променљиве, типови података, типови променљивих потребни за игру.

   Активности:
   - Разрадити типове података у контексту *Greenfoot* окружења и програмског језика *Java*.
   - Наставник треба да појасни разлику између декларације и иницијализације променљиве:
     - Када се декларише нека променљива, специфицира се да је променљива одређеног типа података, док сама вредност може, али не мора, бити обезбеђена.
     - Може се појаснити аналогијом (нпр. предвиђено је поље за унос броја телефона, али број телефона још увек није унет).
   - Декларисати променљиве које су потребне за даљи развој игре.
   - Могу се размотрити и додатни примери. На пример, ако је реч о методи `act()`, може се декларисати променљива за текст који треба да буде приказан.

Co-funded by the Erasmus+ Programme of the European Union

5. Иницијализација променљивих (5 минута)

Циљ: Употреба типова података и променљивих и иницијализација променљивих неопходних за развој игре.

Концепти за дискусију: променљиве, типови података, вредности променљивих.

Активности:

- Представити могуће вредности и распоне (енгл. *range*) вредности, претходно уведених типова података.
- Обрадити вредности и распоне типова података у контексту *Greenfoot* окружења и програмског језика *Java*.
- Наставник треба да понови разлику између декларације и иницијализације променљиве:
  - Када се иницијализује нека променљива, задаје јој се почетна вредност која се у наставку програма може променити. Променљива се може иницијализовати и приликом њене декларације (тј. истовремено са декларацијом).
- Иницијализовати променљиве које су потребне за даљи развој игре.
- Могу се размотрити и додатни примери. На пример, ако је реч о методи `act()`, може се иницијализовати променљива за текст који треба да буде приказан.

Co-funded by the Erasmus+ Programme of the European Union

5.2.2. СЦЕНАРИО: **Увод у операторе и изразе у *Greenfoot* окружењу**

Табела 10. Увод у операторе и изразе у Greenfoot окружењу

Назив	Увод у операторе и изразе у *Greenfoot* окружењу
Образовни циљеви и исходи учења	После ове лекције, ученици ће умети правилно да примене операторе у изразима програмског језика. Уводе се различите врсте оператора (тј. аритметички оператори, логички оператори, релациони оператори) и одговарајући изрази. Поред тога, даје се осврт и на изразе који укључују објекте (нпр. концепт конверзије објеката), као и на референтне променљиве. Наведени концепти се разматрају у контексту развоја игре у *Greenfoot* окружењу, чиме се подстиче креативност, тимски дух и страственији приступ програмирању.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **95**-минута. 1. Оператори (15 минута) 2. Аритметички оператори и изрази (10 минута) 3. Логички оператори (15 минута) 4. Релациони оператори (10 минута) 5. Логички изрази (10 минута) 6. Изрази који укључују објекте (5 минута) 7. Референтне променљиве (15 минута) 8. Задатак **5.1.** (15 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Кроз овај наставни сценарио ученици ће се упознати са принципима програмирања који се односе на операторе и изразе, а из перспективе развоја игара у *Greenfoot* окружењу. Лекција почиње десетоминутним уводом током којег наставник успоставља контекст, надовезујући се на претходне лекције, који ће бити основа за увођење и дефинисање концепта оператора. Након тога следи десетоминутни блок, током којег се разматрају аритметички оператори и изрази. Различити оператори и изрази у *Greenfoot* окружењу се илуструју и анализирају кроз практичне примере. Следећи блок од 15 минута посвећен је логичким операторима, који се користе за рад са логичким вредностима, након чега се још 10 минута посвећује релационим операторима који се користе за поређење вредности. Надовезујући се на претходно обрађене концепте, у оквиру следећег десетоминутног блока се разматрају логички изрази у контексту *Greenfoot* окружења.

Co-funded by the Erasmus+ Programme of the European Union

	Током следећих 5 минута пажња се усмерава на изразе који укључују објекте, док је се током наредног блока од 15 минута разматрају референтне променљиве. Последњих 15 минута је посвећено практичном раду, па ученици решавају **Задатак 5.1.**, при чему их наставник усмерава, а по завршетку пружа повратне информације. У оквиру ове активности предвиђено је и међусобно вредновање. Ученици ће наставити рад на игри коју су започели током претходних лекција. Сходно томе, до краја ове лекције у игру ће бити уведени нови концепти, који се односе на операторе и изразе.
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

5.2.2.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Оператори (15 минута)

Циљ: Увођење концепта оператора, повезивање са примерима из свакодневног живота и упознавање са различитим врстама оператора**.**

Концепти за дискусију: променљиве, типови података, оператори, примери примене оператора у свакодневном животу.

Активности: Наставник уводи појам **оператора**. Наставник треба да приближи овај концепт ученицима коришћењем примера из свакодневног живота (нпр. куповина производа на пијаци). Након тога, наставник представља различите врсте оператора.

2. Аритметички оператори и изрази (10 минута)

Циљ: Разумевање концепта аритметичких оператора, повезивање са примерима из свакодневног живота и упознавање са различитим аритметичким операторима**.**

Концепти за дискусију: променљиве, типови података, оператори, примери примене аритметичких оператора у свакодневном животу.

Активности: Наставник може да укаже на операторе који су ученицима већ познати из других предмета (нпр. аритметички оператори у математици). Ови оператори се користе за извршавање основних рачунских операција. У програмском језику *Java* често се користе следећи аритметички оператори: сабирање (`+`), одузимање (`-`), множење (`*`), дељење (`/`), модуо (`%`). Важно је објаснити појмове оператора, операнада и приоритета оператора. Такође, може се указати на проблем који може настати приликом дељења нулом. Може се објаснити да се оператор `/` приликом дељења два цела броја понаша као оператор целобројног дељења (тј. остатак ће бити занемарен), као и начин на који се овај проблем може превазићи (може се повезати за типовима података, у конкретном случају са `double` типом података, као и са конверзијом). Могу се размотрити и додатни примери. У оквиру *Greenfoot* окружења ови оператори се могу користити за рад са атрибутима објеката (нпр. позиција, кретање и сл.). На пример, могу се дефинисати локалне променљиве, како би се омогућило памћење позиције објекта (`x`,`y`), а затим њено мењање повећавањем вредности датих променљивих.

3. Логички оператори (15 минута)

Циљ: Разумевање концепта логичких оператора, повезивање са примерима из свакодневног живота и упознавање са различитим логичким операторима**.**

Концепти за дискусију: променљиве, типови података, оператори, примери примене логичких оператора у свакодневном животу.

Активности: Логички оператори користе се за грађење логичких израза. Ови оператори се могу повезати за концептом логичких оператора у математици. У програмском језику *Java* често се користе следећи логички оператори: логичко „И“ (оператор `&&` или оператор `&`), логичко „ИЛИ“ (оператор `||` или оператор `|`), логичка негација (оператор `!`). Наставник појашњава операторе, операнде и приоритет оператора. У том смислу, важно је напоменути да логички оператор негације има највиши приоритет. Такође, може се објаснити концепт „одложене евалуације“ (енгл. *lazy evaluation*) код примене логичких оператора `&&` и `||`, где се десна страна израза евалуира само ако се на основу леве стране израза не може одредити резултат читавог израза. На овај начин омогућава се ефикасно извршавање израза. Примери се разматрају у контексту *Greenfoot* окружења, где се ови оператори користе за проверу услова и управљање током извршавања програма. Додатни пример може подразумевати дефинисање локалних променљивих да би се проверило да ли је `x`-координата позиције објекта једнака њеној `y`-координати, уз примену логичког оператора како би се утврдило да ли се објекат налази на дијагонали.

Co-funded by the Erasmus+ Programme of the European Union

4. Релациони оператори (10 минута)

Циљ: Разумевање концепта релационих оператора, повезивање са примерима из свакодневног живота и упознавање са различитим релационим операторима**.**

Концепти за дискусију: променљиве, типови података, оператори, примери примене релационих оператора у свакодневном животу.

Активности: Наставник може да укаже на операторе који су ученицима већ познати из других предмета (нпр. релациони оператори у математици). Ови оператори се користе за поређење вредности и враћају резултат у виду логичке вредности `true` или `false`. Могу се објаснити различити оператори: провера једнакости (`==`), провера неједнакости (`!=`), провера да ли је одређена вредност мања (`<`) или већа (`>`) од друге вредности, провера да ли је одређена вредност мања или једнака (`<=`), односно већа или једнака (`>=`) другој вредности. Применом ових оператора могу се донети одређене одлуке у току извршавања програма (нпр. у оквиру гранања). Наставник појашњава операторе, операнде и приоритет оператора. Додатни пример може подразумевати дефинисање локалних променљивих да би се проверило да ли је `y`-координата позиције неког објекта мања од `y`-координате неког другог објекта, уз примену релационог оператора како би се утврдио однос између позиција та два објекта.

5. Логички изрази (10 минута)

Циљ: Разумевање концепта логичких израза, повезивање са примерима из свакодневног живота и упознавање са различитим логичким изразима**.**

Концепти за дискусију: променљиве, типови података, оператори, примери примене логичких израза у свакодневном животу.

Активности: Логички изрази се граде коришћењем логичких оператора. Као резултат логичких израза добија се логичка вредност `true` или `false`. Наставник може објаснити логичке изразе у контексту претходно обрађених оператора. Наставник затим разматра операторе, операнде и приоритете оператора у контексту логичких израза. На пример, помоћу логичког израза `isTouching(_cls_) && !isAtEdge()` се може проверити да ли је дошло до колизије два објекта (ове методе су обрађене у претходним сценаријима). У том смислу, наставник се још једном може осврнути на концепт „одложене евалуације“ (енгл. *lazy evaluation*) код примене логичких оператора `&&` и `||`, у контексту ефикасног извршавања логичких израза. Могу се размотрити и додатни примери. На пример, логички изрази се могу користити да би се потврдило да ли се објекат налази унутар света (чија је димензија задата).

6. Изрази који укључују објекте (5 минута)

Циљ: Разумевање израза који укључују објекте и компатибилности објеката.

Концепти за дискусију: променљиве, типови података, оператори, изрази који укључују објекте, конверзија.

Активности: Наставник може разматрати изразе који укључују објекте у контексту објектно-оријентисаног програмирања, али и у контексту *Greenfoot* окружења и програмског језика *Java*. Наставник затим разматра операторе, операнде, приоритете оператора, компатибилност типова и конверзију (енгл. *casting*) у контексту израза коју укључују објекте.

Наставник треба да објасни концепт компатабилности типова у контексту хијерархије класа. Имајући у виду природу наслеђивања, тј. чињеницу да изведена класа наслеђује сва својства своје наткласе, изведена класа је компатибилна са натклasом. Другим речима, ако би се дефинисала референтна променљива типа наткласе (нпр. `Actor`), она би могла референцирати и инстанцу било класе која је из ње изведена, на пример:. `Actor obj = new Enemy();`. У овом случају, преко променљиве `obj` се може приступити само својствима класе `Actor` (не може се приступати својствима која су карактеристична искључиво за класу `Enemy`).

Co-funded by the Erasmus+ Programme of the European Union

Наставник затим треба да појасни концепт експлицитне конверзије (енгл. *casting*). Наиме, ако су типови компатибилни, експлицитна конверзија омогућава да се референца једног типа, претвори у референцу другог типа (нпр.: `Enemy enemy = (Enemy)obj;`) при чему треба нагласити да се сам објекат не мења, већ се искључиво мења тип референце на дати објекат. Сада се преко променљиве `enemy` може приступити свим својствима изведене класе (`Enemy`). Наставник треба да објасни и зашто конверзија у обрнутом правцу није могућа. Наставник треба да укаже на то да је важно проверити да ли је одређена експлицитна конверзија могућа, односно да је пожељно да се пре конверзије провери да ли су релевантни типови уопште компатибилни. Сходно томе, наставник може да уведе и оператор `instanceof`.

Могу се размотрити и додатни примери. На пример, могу се упоредити две референтне променљиве (`==` или `equals`) како би се проверило да ли се поклапају, тј. да ли заправо референцирају један исти објекат. Наставник може да, касније, када се детаљније обрађује наслеђивање, објасни како се метода `equals` може и другачије имплементирати.

7. Референтне променљиве (15 минута)

Циљ: Разумевање концепта референтних променљивих, повезивање са примерима из праксе и примена у развој игре**.**

Концепти за дискусију: променљиве, типови података, оператори, примери примене референтних променљивих у пракси.

Активности: Наставник треба да објасни референтне променљиве у контексту објектно-оријентисаног програмирања. Наставник треба да објасни да вредност саме променљиве представља референцу која упућује на одређени објекат, при чему тип променљиве заправо прописује које типове објеката дата променљива може да референцира. Наставник треба да појасни улогу оператора доделе (`=`), помоћу којег се обезбеђује да променљива референцира одређени објекат, што може довести и до тога да две променљиве референцирају исти објекат. Наставник треба да појасни и концепт `null` вредности.

8. Задатак **5.1.** (15 минута)

Циљ: Продубљивање разумевање концепата променљивих, типова података, оператора и израза, и њихова практична примена у развој игре**.**

Концепти за дискусију: променљиве, типови података, оператори, изрази.

Активности: Наставник треба разјасни разлику између коришћења `this.rotation` и коришћења само `rotation`. Наставник треба да опише понашање методе `getOneIntersectingObject(_cls_)`. Уколико одговарајући објекат не постоји, метода враћа `null` вредност.

Наставник излаже задатак:

> **Задатак 5.1. (Ротирање објеката у правцу који најпре треба одредити)**: Промените тело `act()` методе класе `Enemy` тако да обезбедите да се њена инстанца ротира у правцу који је дефинисан у инстанци класе `Direction` коју додирује (оне треба да имају исту ротацију). Позовите методу `getOneIntersectingObject(_cls_)` и сачувајте резултујућу инстанцу у погодној локалној променљивој (нпр. `Direction direction`) – неопходно је пре тога извршити конверзију, јер метода заправо враћа објекат типа `Actor`; дакле, методу треба позвати на следећи начин: `(Direction)getOneIntersectingObject(_cls_)`. Ако је добијен одговарајући резултат, позовите методу `getDirection()` како бисте добили неопходну информацију о ротацији (сачувајте је, ако желите), а затим подесите ротацију инстанце непријатеља (`this`) помоћу методу `setDirection(int)`. Тестирајте своје решење.

Наставник усмерава ученике и прати њихов рад, а затим дискутује са ученицима о решењима.

[Commit*:* 97dddc4beba40ac785c7413bb245ba849cd956d2**]**

Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the Erasmus+ Programme of the European Union

5.2.3. СЦЕНАРИО: **Увод у конструкторе у *Greenfoot* окружењу**

Табела 11. Увод у конструкторе у Greenfoot окружењу

Назив	Увод у конструкторе у *Greenfoot* окружењу
Образовни циљеви и исходи учења	После ове лекције, ученици ће суштински разумети концепт конструктора. У оквиру овог наставног сценарија обрађују се основни теоријски концепти везани за конструкторе, тумачи се одговарајући програмски кôд, а затим се, кроз практичан рад, примењује стечено знање. Разумевање се продубљује кроз развој игре у *Greenfoot* окружењу, чиме се подстиче креативност, тимски дух и страственији приступ програмирању.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **65** минута. 1. Основни теоријски концепти (10 минута) 2. Тумачење кôда (20 минута) 3. Задатак **5.2.** (5 минута) 4. Задатак **5.3.** (30 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Кроз овај наставни сценарио ученицима ће се приближити концепт конструктора, а из перспективе развоја игара у *Greenfoot* окружењу. Лекција почиње десетоминутним уводом у оквиру којег наставник уводи основне теоријске концепте. Након тога следи блок од 20 минута, током којег се тумаче и дискутују различити примери кôда у *Greenfoot* окружењу. Током наредних 5 минута ученици решавају **Задатак 5.2.**, у оквиру којег се од њих очекује да преименују претходно дефинисану класу `MyWorld`. Нов назив класе треба да буде `Arena`. Важно је обратити пажњу на то да је такође потребно преименовати и конструктор класе. Последњих 30 минута је посвећено практичном раду, па ученици решавају **Задатак 5.3.** који се односи на креирање сопствене почетне поставке протагониста у Арени, при чему их наставник усмерава, а по завршетку им пружа повратне информације. У оквиру ове активности предвиђено је и међусобно вредновање. Ученици ће наставити рад на игри коју су започели током претходних лекција. Сходно томе, до краја ове лекције у игру ће бити уведени нови концепти, који се односе на конструкторе.

Co-funded by the Erasmus+ Programme of the European Union

Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

5.2.3.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Основни теоријски концепти (10 минута)

Циљ: Разумевање концепта конструктора**.**

Концепти за дискусију: конструктори, класе, објекти, кључне речи: `super`, `new`, `this`.

Активности: Наставник уводи концепт **конструктора** у контексту концепата класе и објекта у објектно-оријентисаном програмирању. Наставник објашњава да се конструктори користе за иницијализацију конкретне инстанце (тј. објекта) класе.

2. Тумачење кôда (20 минута)

Циљ: Продубљивање разумевање концепта конструктора и повезивање са примерима из праксе**.**

Концепти за дискусију: конструктори, класе, објекти, методе, параметри, кључне речи: `super`, `new`, `this`, примери примене конструктора у пракси.

Активности: Наставник разматра конструкторе у контексту концепата класе и објекта: конструктори се користе за иницијализацију конкретних инстанци (тј. објеката) класе. Поред тога, конструктори се аутоматски позивају, кад год се креира неки објекат, а могу бити имплицитно или експлицитно дефинисани. Постоје подразумевани конструктори (који су имплицитно дефинисани), као и параметризовани и непараметризовани конструктори (који су експлицитно дефинисани од стране програмера). Наставник такође треба да појасни разлику између параметризованих и непараметризованих конструктора. Како би приближио овај концепт ученицима, наставник треба да користи реалне примере.

3. Задатак **5.2.** (5 минута)

Циљ: Промена назива класе**.**

Концепти за дискусију: конструктори, класе, објекти.

Активности: Наставник задаје ученицима задатак:

> **Задатак 5.2. (Промена назива класе `MyWorld` у `Arena`)**: Преименујте претходно дефинисану класу `MyWorld`. Нов назив класе треба да буде `Arena`. Поред тога, неопходно је да преименујете и конструктор класе `MyWorld()` у `Arena()`.
>
> **[Commit***:* aaf73c9bfd9f76a2a1e504f5e78d2976f1cada12**]**

4. Задатак **5.3.** (30 минута)

Циљ: Разматрање односа конструктора и стања игре**.**

Концепти за дискусију: конструктори, класе, објекти, методе, параметри, кључне речи: `super`, `new`, `this`.

Активности: Наставник задаје ученицима задатак:

> **Задатак 5.3. (Креирање сопствене почетне поставке протагониста у арени)**: Креирајте сопствену почетну поставку протагониста у арени. Протагонисти треба да буду постављени на одговарајуће позиције унутар конструктора класе `Arena`. Потребно је креирати једну инстанцу класе `Enemy`, једну инстанцу класе `Orb` и барем једну инстанцу класе `Direction`. Након што декларишете и иницијализујете променљиве жељеног типа, потребно је да подесите атрибуте креираних објекта позивањем одговарајућих метода. Коначно, креиране објекте треба да поставите на жељене позиције у арени позивањем методе `addObject(Actor,int,int)`. Тестирајте своје решење.
>
> **[Commit***:* 8b105ea2eaf697f08c321efe687ddd31e2d0a041**]**

Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the Erasmus+ Programme of the European Union

5.2.4. СЦЕНАРИО: **Увод у атрибуте у *Greenfoot* окружењу**

Табела 12. Увод у атрибуте у Greenfoot окружењу

Назив	Увод у атрибуте у *Greenfoot* окружењу
Образовни циљеви и исходи учења	После ове лекције, ученици ће суштински разумети концепт атрибута. У оквиру овог наставног сценарија обрађују су основни теоријски концепти који се односе на атрибуте, а затим се, кроз практичан рад, примењује стечено знање. Разумевање се продубљује кроз развој игре у *Greenfoot* окружењу, чиме се подстиче креативност, тимски дух и страственији приступ програмирању.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **100**-минута. 1. Задатак **5.4.** (30 минута) 2. Атрибути (10 минута) 3. Параметри конструктора (10 минута) 4. Задатак **5.5.** (20 минута) 5. Задатак **5.6.** (30 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Кроз овај наставни сценарио ученицима ће се приближити концепт атрибута, а из перспективе развоја игара у *Greenfoot* окружењу. Лекција почиње 30-минутним уводом у оквиру којег се од ученика очекује да идентификују проблем везан за кретања непријатеља и дају предлог решења (**Задатак 5.4.**). Надовезујући се на претходни задатак, наставник током наредних 10 минута уводи концепт атрибута. Следећих 10 минута обрађују се параметри конструктора. Током наредног 20-минутног блока, који води наставник, ученици треба да у класу `Enemy` додају нов атрибут да би се омогућило увођење задршке у кретању непријатеља, али и да имплементирају одговарајући параметризовани конструктор (**Задатак 5.5.**). По завршетку наставник пружа повратне информације. У оквиру ове активности предвиђено је и међусобно вредновање. Током последњег 30-минутног блока, који води наставник, ученици треба да имплементирају кретања непријатеља уз задршку, и последично да ажурирају тело `act()` методе (**Задатак 5.6.**), а по завршетку наставник пружа повратне информације. У оквиру ове активности предвиђено је и међусобно вредновање. Ученици ће наставити рад на игри коју су започели током претходних лекција. Сходно томе, до краја ове лекције у игру ће бити уведени нови концепти, који се односе на атрибуте.

Co-funded by the Erasmus+ Programme of the European Union

Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

5.2.4.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Задатак **5.4.** (30 минута)

   Циљ: Разматрање проблема везаног за померање непријатеља и могућег решења у контексту конструктора и атрибута**.**

   Концепти за дискусију: конструктори, атрибути, методе.

   Активности: Наставник излаже задатак:

   > **Задатак 5.4. (Идентификовање проблема везаног за кретања непријатеља и предлог решења)**: Да ли уочавате неки проблем везан за кретања непријатеља? Шта је узрок? Како би се могао решити?

   Наставник указује на то да се објекти класе `Enemy` тренутно померају за по две ћелије одједном. Да би се ово решило, брзина непријатеља се може другачије моделовати тако да се непријатељ увек помера само за по једну ћелију. Међутим, може се додатно увести нов атрибут `moveDelay`, који ће омогућити увођење задршке, тако да ће се непријатељ померити тек након што се метода `act()` изврши одређени број пута.

2. Атрибути (10 минута)

   Циљ: Увођење концепта атрибута**.**

   Концепти за дискусију: конструктори, класе, објекти, атрибути.

   Активности: Наставник уводи концепт **атрибута** у контексту концепата класе и објекта у објектно-оријентисаном програмирању.

3. Параметри конструктора (10 минута)

   Циљ: Увођење концепта параметара конструктора**.**

   Концепти за дискусију: конструктори, класе, објекти, атрибути, параметри, кључне речи: `super`, `new`, `this`.

   Активности: Наставник уводи концепт **параметара конструктора** у контексту концепата класе и објекта у објектно-оријентисаном програмирању.

4. Задатак **5.5.** (20 минута)

   Циљ: Утврђивање знања о атрибутима и параметрима конструктора**.**

   Концепти за дискусију: конструктори, класе, објекти, атрибути, параметри, кључне речи: `super`, `new`, `this`.

   Активности: Наставник задаје ученицима задатак:

   > **Задатак 5.5. (Додавање атрибута `moveDelay` у класу `Enemy`)**: Потребно је додати атрибут `moveDelay` (типа `int`) у класу `Enemy`. Потребно је и имплементирати параметризовани конструктор који треба да обезбеди да се дати атрибут иницијализује са вредношћу параметра. Коначно, потребно је ажурирати и кôд у класи `Arena` у складу са уведеним изменама.

   [Commit: 6092489ce57541e77ae4e2ee886b20853df9f8a4**]**

5. Задатак **5.6.** (30 минута)

   Циљ: Продубљивање разумевања атрибута, као и параметара конструктора**.**

   Концепти за дискусију: конструктори, класе, објекти, атрибути, параметри, кључне речи: `super`, `new`, `this`.

   Активности: Наставник задаје ученицима задатак:

Co-funded by the Erasmus+ Programme of the European Union

Задатак 5.6. (Имплементација кретања непријатеља уз задршку): Потребно је изменити методу `act()` класе `Enemy`, тако да се обезбеди да ће се непријатељ померити тек након што се метода `act()` изврши одређени број (`moveDelay`) пута. Такође је потребно увести нов атрибут `nextMoveCounter` типа `int` и иницијализовати га са **0**. Затим треба изменити методу `act()` тако да позива `this.move(1)` само ако атрибут `nextMoveCounter` има вредност **0**. Након померања, атрибут `nextMoveCounter` добија вредност атрибута `moveDelay`. Када непријатељ не може да се помери (јер `nextMoveCounter` још није достигао вредност **0**), потребно је смањити вредност атрибута `nextMoveCounter` за **1**.

[Commit*:* `bf26e6ed23911ccb712fae3e243cdedff3a89a7f`**]**

Co-funded by the Erasmus+ Programme of the European Union

5.2.5. СЦЕНАРИО: **Увод у преклапање конструктора у *Greenfoot* окружењу**

Табела 13. Увод у преклапање конструктора у Greenfoot окружењу

Назив	Увод у преклапање конструктора у *Greenfoot* окружењу
Образовни циљеви и исходи учења	После ове лекције, ученици ће суштински разумети идеју преклапања конструктора. У оквиру овог наставног сценарија обрађују су основни теоријски концепти везани за преклапање конструктора, а затим се, кроз практичан рад, примењује стечено знање. Разумевање се продубљује кроз развој игре у *Greenfoot* окружењу, чиме се подстиче креативност, тимски дух и страственији приступ програмирању.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **75** минута. 1. Основни теоријски концепти (5 минута) 2. Задатак **5.7.** (25 минута) 3. Задатак **5.8.** (25 минута) 4. Обнављање теоријских концепата (20 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Кроз овај наставни сценарио ученицима ће се приближити механизам преклапања конструктора, а из перспективе развоја игара у *Greenfoot* окружењу. Лекција почиње петоминутним уводом у оквиру којег наставник објашњава основне теоријске концепте. Током наредних 25 минута ученици решавају **Задатак 5.7.,** у оквиру којег се од њих очекује да дефинишу параметризовани конструктор, при чему их наставник усмерава, а по завршетку им пружа повратне информације. Следи блок од 25 минута током којих ученици треба да практично примене стечена знања о преклапању конструктора (**Задатак 5.8.**). У оквиру обе ове активности предвиђено је и међусобно вредновање. Последњи блок од 20 минута, посвећен је теоријском осврту, односно рекапитулацији свих обрађених концепата (као што су променљиве, оператори, изрази, конструктори, атрибути и преклапање конструктора).
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.

Co-funded by the Erasmus+ Programme of the European Union

Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

5.2.5.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Основни теоријски концепти (5 минута)

Циљ: Усвајање концепта преклапања конструктора**.**

Концепти за дискусију: конструктори.

Активности: Наставник обрађује концепт преклапања конструктора.

2. Задатак **5.7.** (25 минута)

Циљ: Утврђивање знања о параметризованим конструкторима**.**

Концепти за дискусију: конструктори, параметри, атрибути, параметризовани конструктори.

Активности: Наставник задаје ученицима задатак:

> **Задатак 5.7. (Имплементација параметризованог конструктора класе `Direction`)**: Дефинишите параметризовани конструктор у класи `Direction`. Конструктор треба да прима један параметар: `rotation` типа `int`. У телу конструктора је потребно ротирати креирану инстанцу у складу са вредношћу параметра. Потребно је ажурирати и кôд у класи `Arena` у складу са уведеним изменама.
>
> **[Commit***:* `3c4b9ef57ab17bac2a0abc7fc5e76ea4b6e27e4b`**]**

3. Задатак **5.8.** (25 минута)

Циљ: Практична примена преклапања конструктора**.**

Концепти за дискусију: конструктори, преклапање конструктора.

Активности: Наставник задаје ученицима задатак:

> **Задатак 5.8. (Преклапање конструктора у класи `Direction`)**: Дефинишите додатни непараметризовани конструктор у класи `Direction`. У телу овог конструктора је потребно позвати претходно дефинисани параметризовани конструктор са аргументом постављеним на **0** (ово одговара ротацији од **0°**). Потребно је ажурирати и програмски кôд у класи `Arena` позивајући непараметризовани конструктор класе `Direction`, где је то могуће.
>
> **[Commit***:* `1e67e67523c66acea4e93363c9a3173302f424c8`**]**

4. Обнављање теоријских концепата (20 минута)

Циљ: Резимирање целокупне области**.**

Концепти за дискусију: променљиве, оператори, изрази, конструктори, атрибути, преклапање конструктора.

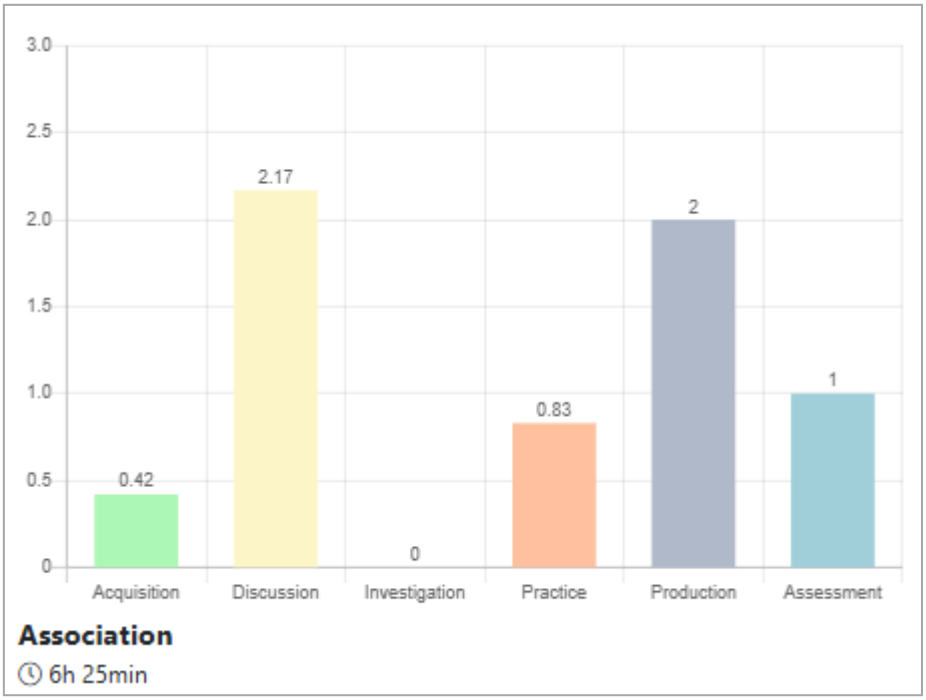
Активности: Наставник даје резиме лекције и осврће се, заједно са ученицима, на све претходно обрађене концепте.

Co-funded by the Erasmus+ Programme of the European Union

6. АСОЦИЈАЦИЈЕ

6.1. Структура наставних активности и задаци

На следећој слици је дат приказ расподеле ангажовања ученика према типу активности, на основу искуства наставника средњих школа у Словачкој Републици, а који је генерисан у алату за креирање наставних сценарија.



Слика 10. Предложена расподела ангажовања ученика према типу активности за тематску целину „Асоцијације“

Задатак 6.1. Осмишљавање динамике игре

Шта би требало да се деси када непријатељ, током игре, стигне до сфере (тј. на који начин би она могла бити оштећена) и шта би били могући исходи?

Задатак 6.2. Дефинисање начина интеракције између објеката

Детаљно разрадите начин интеракције између инстанце класе `Enemy` и инстанце класе `Orb` када дође до њихове колизије, тако што ћете дефинисати тачан редослед корака (односно метода које треба да буду позване над различитим објектима).

Задатак 6.3. Увођење атрибута који су кључни за имплементацију логике игре

Потребно је додати целобројни атрибут `attack` у класу `Enemy`. Потребно је променити и конструктор класе `Enemy` како би се обезбедило да се дати атрибут иницијализује путем параметра. Потребно је додати целобројни атрибут `hp` у класу `Orb`. Потребно је променити и конструктор класе `Orb` како би се обезбедило да се дати атрибут иницијализује путем параметра. Коначно, потребно је ажурирати и кôд у класи `Arena` у складу са уведеним изменама.

[Commit: `4ca1e9f25685990d2bdfe5b610c28422e0944f95`**]**

Co-funded by the Erasmus+ Programme of the European Union

Задатак 6.4. Имплементација „*get*“ методе

Имплементирајте одговарајућу ***„get“*** методу за атрибут `attack` класе `Enemy`, водећи рачуна да обезбедите да она има одговарајућу повратну вредност.

[Commit*:* `72b7456ea4cc11416c57d72c89b6a7f7e9266e3e`**]**

Задатак 6.5. Савладавање начина имплементације методе

Додајте класи `Arena` методу `respawn`, која треба да има један параметар (типа `Enemy`). Ова метода неће имати повратну вредност. У телу методе је потребно подесити локацију и ротацију непријатеља. тако да одговарају њиховим првобитним вредностима, тј. вредностима на које су били иницијализовани у конструктору.

[Commit*:* `43a221876b8acb4fd507175ec4c8f520121d1ab1`**]**

Задатак 6.6. Увежбавање начина дефинисања методе

Додајте класи `Orb` методу `hit`, која треба да има један параметар (типа `Enemy`). Ова метода неће имати повратну вредност. Тело методе треба да остане празно. Тестирајте методу и обратите пажњу на израз који се приказује у одговарајућем прозору.

[Commit*:* `fe03d520260f172066be35055a901487bf7c2ff7`**]**

Задатак 6.7. Позивање методе `hit(Enemy)` класе `Orb` из класе `Enemy`

Преправите тело `act()` методе класе `Enemy`, како бисте обезбедили да се унутар ње позове метода `hit(Enemy)` класе `Orb` када дође до колизије инстанце класе `Enemy` и инстанце класе `Orb`. Уклоните сегменте програмског кôда који доводе до непотребног понашања, као што је ротирање након колизије са сфером или одбијања од ивица света.

[Commit*:* `63f9c96717d9d2587b60095e3b249b0158c8587b`**]**

Задатак 6.8. Имплементација методе `hit(Enemy)` у класи `Orb`

Имплементирајте методу `hit(Enemy)` у класи `Orb`, у складу са алгоритмом дефинисаним кроз **Задатак 6.2.** Тестирајте методу.

[Commit*:* `84bcd7c128faaa9313b507f7438f826ae2f47d2c`**]**

Задатак 6.9. Увођење класа `Bullet` и `Tower`

Дефинишите класу `Bullet` како бисте представили пројектиле које ће куле лансирати. Затим дефинишите класу `Tower`, како бисте представили стационарне објекте (куле) који ће лансирати пројектиле ка непријатељима.

[Commit*:* `ece4df70042c8f60098e14ad2cee55514897d825`**]**

Задатак 6.10. Разрада понашања пројектила

Размислите о томе како би пројектил (тј. инстанца класе `Bullet`) требало да се понаша током игре. Како би требало да се помера? Којом брзином? Шта би требало да се деси када погоди непријатеља? Шта би требало да се деси када стигне до ивице света?

Задатак 6.11. Имплементација кретања пројектила

Имплементирајте кретање пројектила и његово уклањање из арене, ако стигне до ивице света. Почните тако што ћете у тело `act()` методе класе `Bullet` додати позив методе `move(int)`, како бисте обезбедили да се пројектил непрекидно помера унапред.

[Commit*:* `d372827a831381b2254f838041fa4d9a42e53b82`**]**

Co-funded by the Erasmus+ Programme of the European Union

Задатак 6.12. Разрада логике лансирања пројектила

Размислите о томе како ће кула створити и лансирати пројектил.

Задатак 6.13. Формализовање логике лансирања пројектила

Дефинишите редослед позива метода.

Задатак 6.14. Имплементација логике лансирања пројектила

Имплементирајте логику лансирања пројектила.

[Commit*:* `62aec085954beacf996865a55bed312a09c675f2`**]**

Задатак 6.15. Преклапање конструктора у класи `Tower`

Имплементирајте додатни конструктор у класи `Tower`, који ће имати и целобројни параметар `rotation`. Потребно је ажурирати и кôд у класи `Arena`, како бисте на различите начине позиционирали куле, позивајући сваки пут одговарајући конструктор класе `Tower`.

[Commit*:* `bfb6a271f490c341c760e654b3f86a87111c54cb`**]**

Задатак 6.16. Дефинисање начина интеграције пројектила у игру

Детаљно разрадите начин на који инстанце класе `Bullet` треба да интерагују са другим објектима, тако што ћете дефинисати тачан редослед корака (односно метода које треба да буду позване над различитим објектима). Шта би требало да се деси када пројектил погоди непријатеља?

Задатак 6.17. Имплементација интеракције пројектила и непријатеља

Имплементирајте логику обраде колизије пројектила и непријатеља, укључујући ефекте колизије.

[Commit*:* `dcfe31bc006b7f3dcd8b8b759cc1be901c32913c`**]**

Задатак 6.18. Имплементација

Имплементирајте процес креирања непријатеља, при чему је потребно имплементирати и механизам задршке, како би се контролисала учесталост креирања непријатеља. Потребно је имплементирати и услове за завршетак игре. Игра се завршава ако је и последњи непријатељ уклоњен. У том случају је потребно зауставити игру и приказати одговарајућу поруку на екрану.

[Commit*:* `d48341a095561500af6032d5c8f56e201060f9a4`**]**

Co-funded by the Erasmus+ Programme of the European Union

6.2. Наставни сценарији

Припремљена су **четири** наставна сценарија за тематску целину „Асоцијације“.

6.2.1. СЦЕНАРИО: **Имплементација начина интеракције између објеката**

Табела 14. Имплементација начина интеракције између објеката

Назив	Имплементација начина интеракције између објеката
Образовни циљеви и исходи учења	После ове лекције, ученици би требало да су суштински разумели како објекти међусобно интерагују, што представља један од кључних аспекта објектно-оријентисаног програмирања. Ученици би требало да су се, кроз развој интерактивне логике саме игре, извештили у дефинисању, имплементирању и позивању метода у програмском језику *Java* . Поред тога, ученици би требало да су схватили сврху метода за контролисање приступа атрибутима (енгл. *accessor methods*) и њихов значај у контексту енкапсулације и скривања информација.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **75** минута. 1. Задатак **6.1.** (10 минута) 2. Задатак **6.2.** (15 минута) 3. Задатак **6.3.** (10 минута) 4. Методе (15 минута) 5. Задатак **6.4.** (5 минута) 6. Задатак **6.5.** (10 минута) 7. Задатак **6.6.** (10 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње 10-минутном дискусијом о могућој динамици игри, са фокусом на то шта би требало да се деси када непријатељ стигне до сфере (**Задатак 6.1.**). Ова активност треба да успостави основу за разумевање начина интеракције између објеката у контексту игре. Следи 15-минутни задатак (**Задатак 6.2.**) у оквиру којег се од ученика очекује да прецизно дефинишу начин интеракције између инстанце класе `Enemy` и инстанце класе `Orb`, када дође до њихове колизије, тако што ће дефинисати тачан редослед метода које треба да буду позване над одговарајућим објектима. У наредном 10-минутном задатку (**Задатак 6.3.**) фокус је на атрибутима `attack` (у класи `Enemy`) и `hp` (у класи `Orb`), који су кључни за управљање логиком игре.

Co-funded by the Erasmus+ Programme of the European Union

	Кроз овај задатак, ученици ће разумети улогу датих атрибута и научити како да их дефинишу и користе. Током следећег 15-минутног блока, детаљно се обрађују методе. Овај блок има за циљ утемељивање знања о начину дефинисања, имплементирања и позивања метода. У оквиру наредног петоминутног задатка (**Задатак 6.4.**) од ученика се очекује да имплементирају одговарајућу ***„get“*** методу за атрибут `attack` класе `Enemy`, како би утемељили своје знање о енкапсулацији и скривању информација. Следећих 10 минута биће посвећено имплементацији и тестирању методе `respawn(Enemy)` класе `Arena.` Ученици у оквиру дате методе треба да имплементирају логику враћања инстанце непријатеља на његову почетну поставку (**Задатак 6.5.**). Последњи 10-минутни задатак (**Задатак 6.6.**) се односи на имплементацију и тестирање методе `hit(Enemy)` класе `Orb`, која ће управљати логиком интеракције између објеката приликом колизије непријатеља и сфере.
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

6.2.1.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Задатак **6.1.** (10 минута)

Циљ: Подстицање ученика да осмисле динамику игре**.**

Концепти за дискусију: оштећење сфере, премештање непријатеља, изазивање догађаја у игри (нпр. смањивање енергије, репродукција звучних ефеката, завршетак игре).

Активности: Током увода у лекцију наставник најпре даје преглед претходно обрађених концепата, како би се осигурало да ученици имају темељну основу за ново градиво. Наставник, кроз дискусију, појашњава да објекти међусобно интерагују путем метода.

Наставник излаже ученицима задатак:

> **Задатак 6.1. (Осмишљавање динамике игре)**: Шта би требало да се деси када непријатељ, током игре, стигне до сфере (тј. на који начин би она могла бити оштећена) и шта би били могући исходи?

Ученици, у тимовима, осмишљавају могуће ефекте. Наставник усмерава дискусију како би подстакао ученике да ускладе своје предлоге са следећим алгоритмом.

Опис решења:

> Након колизије са непријатељем, сфери се може умањити енергија (**HP** — *Health Points*), све док потпуно не изгуби енергију (**HP=0**), када игра треба да се заврши. Са друге стране, ако након колизије сфера и даље има енергије (**HP>0**), непријатеља треба вратити на његову почетну поставку .

Потребно је размотрити и друге догађаје који могу бити изазвани као последица интеракције између објеката, као што су, на пример, звучни ефекти или приказ различитих порука на екрану.

Ову активност треба да помогне ученицима да своје идеје практично примене и утврде своје разумевање динамике, и интеракција, у оквиру игре.

2. Задатак **6.2.** (15 минута)

Циљ: Дефинисање начина интеракције између објеката**.**

Концепти за дискусију: позивање метода, пренос параметара, референце на објекте.

Активности: Наставник задаје ученицима задатак:

> **Задатак 6.2. (Дефинисање начина интеракције између објеката)**: Детаљно разрадите начин интеракције између инстанце класе `Enemy` и инстанце класе `Orb` када дође до њихове колизије, тако што ћете дефинисати тачан редослед корака (односно метода које треба да буду позване над различитим објектима).

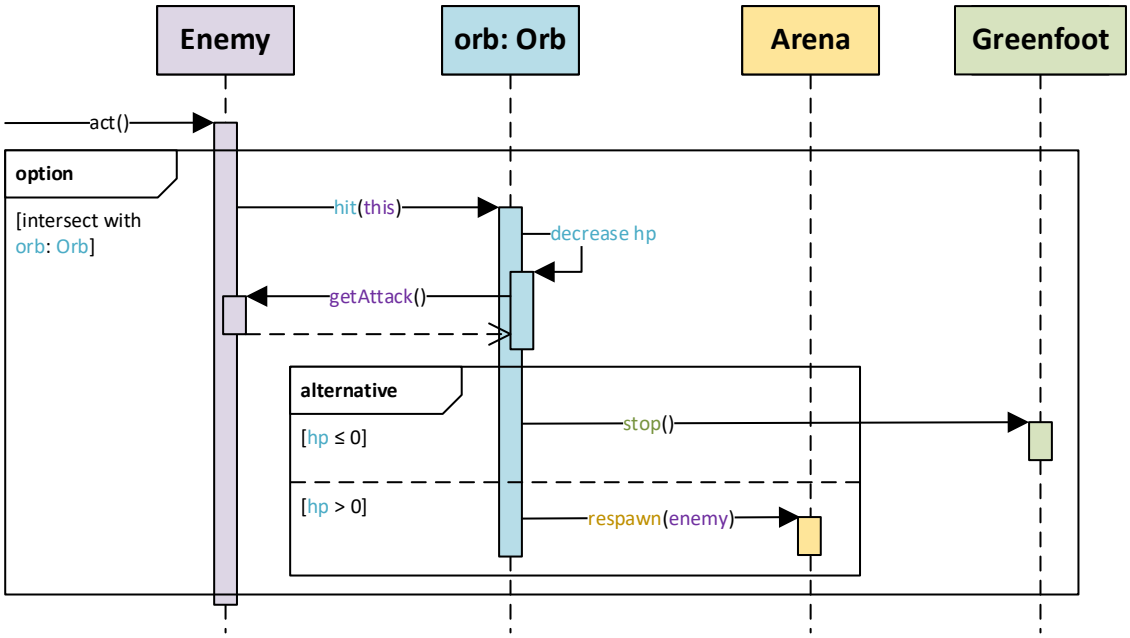
Ученици решавају задатак у паровима, а наставник прати њихов рад и усмерава их како би се обезбедило да дефинисани редослед интеракција између објеката следи претходно утврђен алгоритам.

Овај задатак има за циљ продубљивање разумевања начина дефинисања редоследа позивања метода, преношења параметара и референци на објекте.

Како би помогао ученицима да лакше савладају поступак дефинисања тока интеракције између објеката, односно прецизирање редоследа позива метода, наставник може да представи и *UML* (*Unified Modeling Language*) дијаграм секвенце који омогућава визуелни приказ интеракција између објеката.

Co-funded by the Erasmus+ Programme of the European Union

Решење:



Слика 11. UML дијаграм секвенце (решење задатка 6.2.)

3. Задатак **6.3.** (10 минута)

Циљ: Увођење атрибута кључних за имплементацију логике игре `attack` у класу `Enemy` и `hp` у класу `Orb`**.**

Концепти за дискусију: атрибути, енкапсулација.

Активности: Наставник уводи концепте атрибута и енкапсулације и указује на то како атрибути `attack` (у класи `Enemy`) и `hp` (у класи `Orb`) могу представљати карактеристике објеката које су од значаја за утврђивање исхода интеракција у оквиру игре.

Наставник задаје ученицима задатак:

> **Задатак 6.3. (Увођење атрибута који су кључни за имплементацију логике игре)**: Потребно је додати целобројни атрибут `attack` у класу `Enemy`. Потребно је променити и конструктор класе `Enemy` како би се обезбедило да се дати атрибут иницијализује путем параметра. Потребно је додати целобројни атрибут `hp` у класу `Orb`. Потребно је променити и конструктор класе `Orb` како би се обезбедило да се дати атрибут иницијализује путем параметра. Коначно, потребно је ажурирати и кôд у класи `Arena` у складу са уведеним изменама.
>
> **[Commit***:* `4ca1e9f25685990d2bdfe5b610c28422e0944f95`**]**

Док ученици решавају задатак наставник прати њихов рад, а затим им постепено објашњава како да ажурирају кôд у класи `Arena` да би се прилагодио уведеним изменама. Ученици прате упутства наставника и раде заједно са њим, корак по корак.

Овај практичан задатак треба да помогне ученицима да боље разумеју улогу атрибута у интеракцијама између објеката и значај енкапсулације када је реч о чувању интегритета програмског кôда.

Co-funded by the Erasmus+ Programme of the European Union

4. Методе (15 минута)

Циљ: Упознавање са начином дефинисања метода и начином њиховог позивања**.**

Концепти за дискусију: начин дефинисања методе, начин позивања методе, параметри, повратне вредности.

Активности: Наставник тумачи концепт методе као енкапсулирани (обједињени, учаурени) скуп радњи и понашања. Наставник кроз практичне примере представља синтаксу и структуру дефиниције методе, и илуструје како се методе позивају над објектима.

Наставник објашњава разлику између различитих врста метода, тј. оних које искључиво треба да обаве неку радњу (тзв. `void` методе, односно методе које немају повратну вредност) и оних које поред обављања неке радње треба и да врате одређени резултат (методе које имају одређену повратну вредност).

Наставник објашњава начин на који се методи прослеђују параметри и наглашава да су типови параметара, као и њихов редослед, од суштинског значаја. Кроз вођене практичне задатке, ученици увежбавају начин дефинисања метода које имају различите типове параметара и повратне вредности, али и начин позивања датих метода над инстанцама класе. Обрађују се различите варијанте (тј. методе које обављају одређене радње, које модификују стања објеката или које враћају специфичне вредности), како би се утврдило разумевање функционалности метода унутар класе.

5. Задатак **6.4.** (5 минута)

Циљ: Увођење концепта метода за контролисање приступа атрибутима (енгл. *accessor methods*), а посебно за преузимање тренутне вредности атрибута (***get***).

Концепти за дискусију: методе за контролисање приступа атрибутима (енгл. *accessor methods*), метода за преузимање тренутне вредности атрибута (***get***), енкапсулација и скривање информација.

Активности: Наставник објашњава сврху метода за контролу приступа атрибутима и наглашава њихов значај у контексту енкапсулације и скривања информација. Наставник указује на то да је важно обезбедити да атрибут буде приватан, као и да је, сходно томе, потребно обезбедити одговарајућу *„**get**“* методу како би се омогућило преузимање тренутне вредности приватног атрибута. Оваквим приступом се може осигурати контролисани приступ подацима.

Наставник представља синтаксу и структуру *„**get**“* метода, као и начин њихове употребе.

Наставник задаје ученицима задатак:

> **Задатак 6.4. (Имплементација *„get“* методе)**: Имплементирајте одговарајућу ***„get“*** методу за атрибут `attack` класе `Enemy`, водећи рачуна да обезбедите да она има одговарајућу повратну вредност.
>
> **[Commit***:* 72b7456ea4cc11416c57d72c89b6a7f7e9266e3e**]**

Кроз ову активност ученици ће продубити разумевање концепата енкапсулације и скривања информација и њихове примене у циљу контролисања приступа подацима у контексту објектно-оријентисаног програмирања.

Co-funded by the Erasmus+ Programme of the European Union

6. Задатак **6.5.** (10 минута)

Циљ: Имплементирање и тестирање методе `respawn(Enemy)` класе `Arena`, која је задужена за враћање инстанце непријатеља на његову почетну поставку, након колизије са сфером**.**

Концепти за дискусију: имплементација метода, тестирање, логика игре.

Активности: Наставник објашњава начин имплементације метода како би се обезбедило жељено понашање објекта.

Наставник задаје ученицима задатак:

> **Задатак 6.5. (Савладавање начина имплементације методе)**: Додајте класи `Arena` методу `respawn`, која треба да има један параметар (типа `Enemy`). Ова метода неће имати повратну вредност. У телу методе је потребно подесити локацију и ротацију непријатеља. тако да одговарају њиховим првобитним вредностима, тј. вредностима на које су били иницијализовани у конструктору.
>
> **[Commit***:* `43a221876b8acb4fd507175ec4c8f520121d1ab1`**]**

Ученици затим треба да тестирају методу како би се уверили да она исправно функционише. Наставник постепено објашњава како се тестира метода. Ученици прате упутства наставника и раде заједно са њим, корак по корак. Наставник најпре објашњава ученицима да је прво потребно да креирају инстанцу класе `Arena` и инстанцу класе `Enemy`, али да не треба да одмах затим покрену апликацију, већ је потребно да превуку `Enemy` објекат на неку локацију у арени. Након тога је потребно приступити контекстном менију инстанце класе `Arena` (десним кликом на дату инстанцу на сцени) како би се позвала метода `respawn`. Наставник затим указује на прозор који омогућава задавање конкретних вредности параметара. Да би се обезбедила вредност за параметар, потребно је одабрати неку инстанцу класе `Enemy` (кликом на одговарајућу инстанцу на сцени), али пре тога треба проверити да ли је апликација паузирана и да ли је поље за унос вредности датог параметра активно (потребно је да се у њему налази курсор који трепери). Ученици треба да обрате пажњу на израз који ће након тога бити формиран у прозору. Коначно, потребно је кликнути на дугме `OK` и пратити ефекте методе.

Наставник води рачуна о томе да ли су ученици разумели сваки од корака и, уколико је потребно, помаже ученицима и даје додатна објашњења, како би осигурао да су ученици у потпуности савладали начин имплементације методе и логике игре.

7. Задатак **6.6.** (10 минута)

Циљ: Дефинисање методе `hit(Enemy)` у класи `Orb.`

Концепти за дискусију: /.

Активности: Наставник објашњава сврху методе `hit(Enemy)` класе `Orb`, наглашавајући то да bi дата метода требalo да енкапсулира логику интеракције између сфере и непријатеља након њихове колизије.

Наставник задаје ученицима задатак:

> **Задатак 6.6. (Увежбавање начина дефинисања методе)**: Додајте класи `Orb` методу `hit`, која треба да има један параметар (типа `Enemy`). Ова метода неће имати повратну вредност. Тело методе треба да остане празно. Тестирајте методу и обратите пажњу на израз који се приказује у одговарајућем прозору.
>
> **[Commit***:* `fe03d520260f172066be35055a901487bf7c2ff7`**]**

Co-funded by the Erasmus+ Programme of the European Union

Да би тестирали ову методу, ученици прате поступак, сличан поступку који је изложен у претходном задатку. Ученици креирају инстанцу класе `Orb` и инстанцу класе `Enemy`. Ученици затим треба да приступе контекстном менију `Orb` објекта (без претходног покретања апликације) и да одаберу методу `hit`. Наставник проверава да ли сви ученици разумеју како да обезбеде вредност за параметар (кликом на одговарајућу инстанцу класе `Enemy`, док је апликација паузирана). На овај начин припрема се одговарајући израз за позив методе, који ученици затим извршавају кликом на дугме `OK`.

Наставник наглашава да је важно обратити пажњу на формирани израз, јер се тако може проверити да ли ће метода бити позвана на исправан начин.

Кроз овај задатак, ученици стичу практично искуство и увежбавају начин дефинисања и тестирања метода. Наставник води рачуна о томе да ли су ученици разумели сваки од корака и, уколико је потребно, помаже ученицима и даје додатна објашњења.

Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the Erasmus+ Programme of the European Union

6.2.2. СЦЕНАРИО: **Увод у асоцијације**

Табела 15. Увод у асоцијације

Назив	Увод у асоцијације
Образовни циљеви и исходи учења	До краја ове лекције, ученици треба да стекну солидно схватање улоге коју асоцијације играју у пројектовању и имплементацији интерактивних система у *Greenfoot* окружењу. Они ће моћи да идентификују различите типове асоцијација и да примене ово знање за ефикасно моделовање и имплементацију сложених интеракција између *Greenfoot* објеката. Ово практично разумевање јача њихову способност да пројектују кохезивне и интерактивне сценарије игара користећи објектно-оријентисане принципе.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **70** минута. 1. Асоцијације (10 минута) 2. Задатак **6.7.** (15 минута) 3. Тумачење кôда (15 минута) 4. Задатак **6.8.** (30 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње 10-минутном дискусијом о односима између класа, кроз коју се уводи концепт асоцијација. Након тога следи 15-минутни задатак (**Задатак 6.7.**) у оквиру којег је потребно из класе `Enemy` позвати методу `hit(Enemy)` класе `Orb`. Током следећег 15-минутног блока наставник детаљно објашњава методу `Greenfoot.stop()` и методу `getWorldOfType(_cls_)` класе `Actor`. Кључна активност је 30-минутни **Задатак 6.8.** у оквиру којег се од ученика очекује да имплементирају методу `hit(Enemy)` класе `Orb`.
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.

Co-funded by the Erasmus+ Programme of the European Union

Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

6.2.2.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Асоцијације (10 минута)

Циљ: Увођење концепта асоцијације**.**

Концепти за дискусију: асоцијације, односи између класа.

Активности: Наставник уводи концепт **асоцијације** између класа. Наставник подстиче дискусију, користећи практичне примере из *Greenfoot* окружења како би илустровао овај концепт. Ученици треба да разумеју да асоцијације заправо дефинишу како класе утичу једна на другу, као што је, на пример, утицај класе `Enemy` на класу `Orb` у игри.

Наставник треба да појасни различите типове асоцијација:

- један-према-један,
- један-према-више и
- више-према-више.

Користећи *UML* дијаграме класа, наставник може да прикаже ове односе, како би помогао ученицима да боље разумеју како су асоцијације структуриране и имплементиране у њиховим пројектима. На пример, дијаграм би могао да прикаже како је класа `Enemy` повезана са вишеструким инстанцама класе `Orb`, указујући на однос један-према-више, где сваки непријатељ може утицати на неколико сфера.

Ученици треба да учествују у идентификацији и мапирању асоцијација у оквиру својих пројеката. Они треба да истраже како објекти интерагују и да продискутују могуће импликације на логику игре. Наставник даје конкретне примере из контекста развоја игре, илуструјући како инстанце класе `Enemy` ступају у интеракцију са инстанцама класе `Orb` и како овим интеракцијама управљају асоцијације.

2. Задатак **6.7.** (15 минута)

Циљ: Позивање методе `hit(Enemy)` класе `Orb` из класе `Enemy`**.**

Концепти за дискусију: позивање метода, референце на објекте.

Активности: Наставник задаје ученицима задатак:

> **Задатак 6.7. (Позивање методе `hit(Enemy)` класе `Orb` из класе `Enemy`)**: Преправите тело `act()` методе класе `Enemy`, како бисте обезбедили да се унутар ње позове метода `hit(Enemy)` класе `Orb` када дође до колизије инстанце класе `Enemy` и инстанце класе `Orb`. Уклоните сегменте програмског кôда који доводе до непотребног понашања, као што је ротирање након колизије са сфером или одбијања од ивица света.
>
> **[Commit***:* `63f9c96717d9d2587b60095e3b249b0158c8587b`**]**

Наставник, кроз практичне примере, демонстрира како правилно позвати методу. Ученици уче како да користе референцу на текући објекат (`this`) да би позвали методу `hit(Enemy)` над инстанцом класе `Orb`, након што дође до колизије са непријатељем. Ученици треба да анализирају ток извршавања програма у *Greenfoot* окружењу, разумевајући како позивање метода заправо усмерава ток њихове игре.

Наставник даје упутства за отклањање грешака и тестирање имплементације, како би се осигурало да се метода `hit(Enemy)` класе `Orb` понаша на предвиђен начин.

Co-funded by the Erasmus+ Programme of the European Union

3. Тумачење кôда (15 минута)

Циљ: Разумевање понашања методе `Greenfoot.stop()` и методе `getWorldOfType(_cls_)` класе `Actor`.

Концепти за дискусију: статичке методе *Greenfoot* окружења.

Активности: Наставник представља две значајне методе *Greenfoot* окружења: `Greenfoot.stop()` и `getWorldOfType(_cls_)` класе `Actor`.

Метода `Greenfoot.stop()` је од суштинског значаја за управљање током извршавања *Greenfoot* сценарија. Када се позове, она зауставља симулацију и замрзава све протагонисте и интеракције у окружењу. Наставник показује како се `Greenfoot.stop()` метода може интегрисати у сценарио игре.

Позивање методе `getWorldOfType(_cls_)` над одређеним протагонистом, враћа инстанцу света у којем се налази дати протагониста, под условом да је тај свет задатог типа (тј. `_cls_`) или неког његовог подтипа.

4. Задатак **6.8.** (30 минута)

Циљ: Имплементирање методе `hit(Enemy)` класе `Orb`.

Концепти за дискусију: имплементација метода, промена стања објекта, логика игре.

Активности: Наставник задаје ученицима задатак:

> **Задатак 6.8. (Имплементација методе `hit(Enemy)` у класи `Orb`)**: Имплементирајте методу `hit(Enemy)` у класи `Orb`, у складу са алгоритмом дефинисаним кроз **Задатак 6.2.** Тестирајте методу.
>
> **[Commit***:* `84bcd7c128faaa9313b507f7438f826ae2f47d2c`**]**

Опис могућег решења:

Најпре треба умањити енергију сфере (`hp`), чиме се симулира обим штете коју непријатељ наноси сфери приликом њихове колизије. Након тога, је потребно проверите да ли је енергија сфере (`hp`) пала на нулу или испод нуле. Ако јесте, игра треба да се заврши, па је потребно позвати методу `Greenfoot.stop()` да би се игра зауставила. Тиме је игра завршена. Потребно је обрадити и случај када је енергија сфере, након колизије са непријатељем, и даље изнад нуле. У том случају је потребно имплементирати логику која ће обезбедити враћање непријатеља на његову почетну позицију, како би се игра наставила.

Ученици треба да, кроз дискусију, договоре конкретне финесе које желе да имплементирају, као што је обим штете који одређени тип непријатеља наноси сфери и шта се дешава када је енергија сфере потпуно исцрпљена. Наставник ученицима пружа смернице у погледу имплементације методе, начина коришћења параметра (`Enemy`) и ажурирања стања сфере (`hp`).

Наставник подстиче ученике да темељно тестирају своја решења, како би осигурали да ће се метода понашати на предвиђени начин у различитим сценаријима игре.

Co-funded by the Erasmus+ Programme of the European Union

6.2.3. СЦЕНАРИО: **Дефинисање и имплементација стратешких елемената игре**

Табела 16. Дефинисање и имплементација стратешких елемената игре

Назив	Дефинисање и имплементација стратешких елемената игре
Образовни циљеви и исходи учења	Лекција се фокусира на развијање вештина ученика у дефинисању и имплементацији стратешких елемената игре како би се створила занимљива и интерактивна игра. Ученици ће осмислити, формализовати и имплементирати сложени алгоритам који представља срж саме игре. На овај начин, стичу практичан увид у принципе пројектовања игре и продубљују своје разумевање стратешких интеракција између различитих објеката.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **130** минута. 1. Задатак **6.9.** (10 минута) 2. Задатак **6.10.** (10 минута) 3. Задатак **6.11.** (30 минута) 4. Задатак **6.12.** (15 минута) 5. Задатак **6.13.** (15 минута) 6. Задатак **6.14.** (30 минута) 7. Задатак **6.15.** (20 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње 10-минутним задатком (**Задатак 6.9.**) у оквиру којег се од ученика тражи да дефинишу класе `Bullet` и `Tower`. Након тога, следи 10-минутна дискусија о томе како инстанце класе `Bullet` треба да се крећу и како треба да се понашају када пројектил погоди непријатеља или стигне до ивице света (**Задатак 6.10.**). Ученици затим посвећују 30 минута имплементацији кретања инстанци класе `Bullet` (**Задатак 6.11.**). Затим следи 15-минутна дискусија о томе како инстанце класе `Tower` треба да стварају и лансирају пројектиле (**Задатак 6.12.**). Циљ ове дискусије је да се разради логика лансирања пројектила из кула. Следећих 15 минута посвећено је формализацији ове логике (**Задатак 6.13.**). Коначно, ученици проводе 30 минута имплементирајући дату логику (**Задатак 6.14.**). Лекција се завршава 20-минутним блоком који је посвећен интегрисању кула и пројектила у игру, са акцентом на правилно позиционирање кула позивањем одговарајућих конструктора (**Задатак 6.15.**).

Co-funded by the Erasmus+ Programme of the European Union

Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

6.2.3.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Задатак **6.9.** (10 минута)

Циљ: Увођење класа `Bullet` и `Tower`**.**

Концепти за дискусију: дефинисање класа, улоге класа.

Активности: Наставник даје преглед основних принципа везаних за начин креирања објеката, начин њиховог кретања и међусобне интеракције у *Greenfoot* окружењу. Наставник подстиче дискусију како би разјаснио улоге различитих класа у игри и њихове међусобне односе.

Наставник уводи нове протагонисте у игру и појашњава њихове улоге. Класом `Tower` ће бити представљени стационарни објекти (куле) који могу лансирати пројектиле ка непријатељима, док ће класом `Bullet` бити представљени пројектили које ће куле лансирати.

Наставник задаје ученицима задатак:

> **Задатак 6.9. (Увођење класа `Bullet` и `Tower`)**: Дефинишите класу `Bullet` како бисте представили пројектиле које ће куле лансирати. Затим дефинишите класу `Tower`, како бисте представили стационарне објекте (куле) који ће лансирати пројектиле ка непријатељима.
>
> **[Commit***:* `ece4df70042c8f60098e14ad2cee55514897d825`**]**

Наставник, уколико је потребно, подсећа ученике како се уводи нова класа, као изведена класа класе `Actor` — десним кликом на класу `Actor` и избором опције `New subclass...` у њеном контекстном менију.

Наставник води рачуна о томе да су ученици разумели различите улоге нових класа и начин на који ће оне интераговати у током игре. До краја ове активности, ученици би требало да су успешно дефинисали и успоставили основну структуру класа `Bullet` и `Tower`, припремајући се за имплементацију сложенијих интеракција у наредним задацима.

2. Задатак **6.10.** (10 минута)

Циљ: Дискусија о очекиваном понашању пројектила**.**

Концепти за дискусију: логика кретања, детекција колизије.

Активности: Наставник задаје ученицима задатак:

> **Задатак 6.10. (Разрада понашања пројектила)**: Размислите о томе како би пројектил (тј. инстанца класе `Bullet`) требало да се понаша током игре. Како би требало да се помера? Којом брзином? Шта би требало да се деси када погоди непријатеља? Шта би требало да се деси када стигне до ивице света?

Ученици заједно разрађују могуће понашање пројектила. Наставник усмерава дискусију како би подстакао ученике да ускладе своје предлоге са следећим алгоритмом.

Опис решења:

> Пројектил би требало да се креће у правцу у којем је лансиран, и то праволинијски, без промене правца. Брзина пројектила треба да буде управљива и конзистентна. За имплементацију овог понашања могу се користити методе за кретање које су обезбеђене у самом *Greenfoot* окружењу. Наставник треба да истакне улогу конструктора када је реч о иницијализацији вредности атрибута приликом креирања инстанци класе.
>
> Ако пројектил стигне до ивице света, треба га уклонити из игре. Ако погоди непријатеља, пројектил би требало да нестане, а непријатељ би требао да буде оштећен или уклоњен.

Co-funded by the Erasmus+ Programme of the European Union

Сама логика понашања пројектила би се могла реализовати на следећи начин:

Пројектил је најпре потребно померити унапред, и то константном брзином, а затим је потребно проверити да ли је можда стигао до ивице света. Ако јесте, потребно је уклонити пројектил из арене. Ако није, потребно је проверити да ли је дошло до колизије пројектила и непријатеља. Ако дође до колизије, то значи да је пројектил погодио непријатеља, па је потребно уклонити пројектил из арене и нанети штету непријатељу или га чак и уклонити.

Наставник демонстрира како се наведена логика може имплементирати коришћењем *Greenfoot* метода: `move(int)`, `isAtEdge()` и `getOneIntersectingObject(_cls_)`.

Наставник подстиче ученике да даље разраде своје идеје и да размисле о додатним детаљима, као што је подешавање брзине у зависности од тежине саме игре или додавање визуелних ефеката када пројектил погоди непријатеља. Ова активност помаже ученицима да схвате принципе кретања и детекције колизије у развоју игре, припремајући их за њихову имплементацију.

3. Задатак **6.11.** (30 минута)

Циљ: Имплементирање кретања инстанце класе `Bullet`.

Концепти за дискусију: померање протагониста, детекција ивица света.

Активности: Ученици најпре треба да се осврну на принципе кретања протагониста и детекције досезања ивице света, фокусирајући се на то како се ови принципи примењују на класу `Bullet`. Наставник треба да подсети ученике на неке од претходних задатака, истичући знање које овде треба да примене. Потребно је да се подсете како да, у тело `act()` методе, додају кôд за интеракцију са ивицом света (**Задатак 4.2.**). Потребно је да се подсете како да детектују да ли је дошло до колизије (**Задатак 4.4.**). Коначно, треба да се подсете како да користе бројаче и механизам задршке, да би у методи `act()` могли да контролишу брзину кретања (**Задатак 5.6.**).

Наставник задаје ученицима задатак:

> **Задатак 6.11. (Имплементација кретања пројектила)**: Имплементирајте кретање пројектила и његово уклањање из арене, ако стигне до ивице света. Почните тако што ћете у тело `act()` методе класе `Bullet` додати позив методе `move(int)`, како бисте обезбедили да се пројектил непрекидно помера унапред.

[Commit*:* `d372827a831381b2254f838041fa4d9a42e53b82`**]**

4. Задатак **6.12.** (15 минута)

Циљ: Дискусија о логици лансирања пројектила из куле**.**

Концепти за дискусију: креирање објеката, позивање метода.

Активности: Наставник задаје ученицима задатак:

> **Задатак 6.12. (Разрада логике лансирања пројектила)**: Размислите о томе како ће кула створити и лансирати пројектил.

Ученици заједно разрађују могућа решења, док наставник усмерава дискусију. Ученици најпре треба да продискутују општу логику лансирања пројектила у игри. Потребно је да се фокусирају на кључне аспекте као што су креирање инстанци класе `Bullet` и позивање метода за њихово лансирање. Наставник наглашава да кула не би требало да лансира пројектиле при сваком позиву `act()` методе. Ученици треба да размисле о томе како је могуће обезбедити то да кула само повремено лансира пројектиле. Наставник их наводи да размисле о коришћењу механизма задршке, као и када се радило о управљању кретањем непријатеља (**Задатак 5.6.**). Наставник треба да појасни улогу конструктора у имплементација овог механизма.

Co-funded by the Erasmus+ Programme of the European Union

Наставник објашњава да ће бити потребно да се класи `Tower` додају нови атрибути: `shootDelay` и `nextShootCounter`. Путем ових атрибута ће се контролисати учесталост лансирања.

Наставник разрађује детаље класе `Tower`. Најпре је потребно у класи `Tower` дефинисати атрибут `shootDelay`, као и атрибут `nextShootCounter`, који треба иницијализовати са вредношћу **0**. Након тога, потребно је промените тело `act()` методе класе `Tower` како би се имплементирала логика лансирања пројектила. У оквиру методе би требало створити и лансирати пројектил само када се атрибут `nextShootCounter` спусти на **0**. Након лансирања пројектила, атрибут `nextShootCounter` треба ресетовати на вредност атрибута `shootDelay`. Међутим, ако атрибут `nextShootCounter` још није достигао вредност **0**, требало би смањити његову вредност за **1**. Коначно, потребно је дефинисати засебну `fire()` методу која ће бити задужена за стварање и лансирање пројектила. У овој методи је потребно креирати инстанцу класе `Bullet` и додати је у арену.

Кроз овај процес, ученици ће разумети како да имплементирају логику лансирања пројектила, раздвајањем релевантних корака у различите методе класе `Tower`. Наставник треба да се постара да ученици увиде значај позивања метода и креирања објеката, продубљујући њихово разумевање датих концепата у контексту властите игре.

5. Задатак **6.13.** (15 минута)

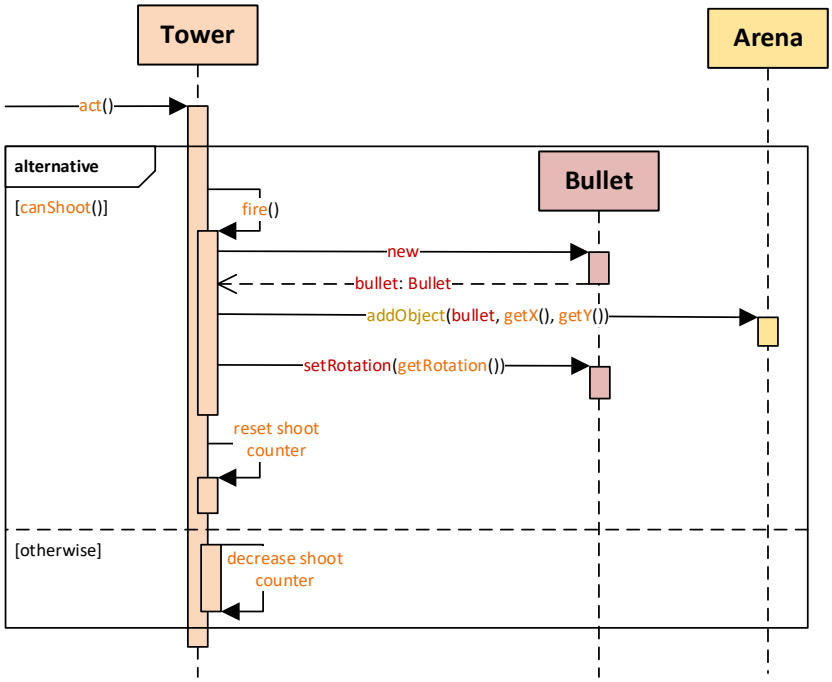
Циљ: Дискусија о механизму позивања метода за лансирање пројектила.

Концепти за дискусију: позивање метода.

Активности: Наставник понавља како објекти у игри међусобно комуницирају путем метода. Наставник задаје ученицима задатак:

> **Задатак 6.13. (Формализовање логике лансирања пројектила)**: Дефинишите редослед позива метода.

Наставник може да користи *UML* дијаграм секвенце како би приказао интеракције између објеката `Tower`, `Bullet` и `Arena`. Дијаграм визуелно представља ток позива метода, помажући ученицима да разумеју редослед интеракција. Наставник указује на то како различити објекти учествују у реализацији делова алгоритма.



Слика 12. UML дијаграм секвенце (задатак 6.13.)

Co-funded by the Erasmus+ Programme of the European Union

6. Задатак **6.14.** (30 минута)

Циљ: Имплементирање логике лансирања пројектила**.**

Концепти за дискусију: креирање објеката, постављање инстанци протагониста у арену.

Активности: Наставник задаје ученицима задатак:

> **Задатак 6.14. (Имплементација логике лансирања пројектила)**: Имплементирајте логику лансирања пројектила.
>
> **[Commit***:* `62aec085954beacf996865a55bed312a09c675f2`**]**

Наставник затим разлаже задатак на кораке и води ученике кроз сваки од њих.

- Потребно је дефинисати неопходне атрибуте и конструкторе класе `Tower`. Наставник подсећа ученике на то да је класи `Tower` потребан атрибут путем којег ће детектовати тренутак у којем може да лансира пројектил.
- Потребно је дефинисати две методе:
  - `boolean canShoot()` (иницијално може враћати `false`) и
  - `void fire()` (иницијално може имати празно тело)
- Потребно је имплементирати тело `act()` методе и у њему, на адекватан начин, позвати претходно дефинисане методе.
- Потребно је имплементирати методу `canShoot()` тако да враћа `true` ако је атрибут `shootCounter` стигао до **0**.
- Потребно је имплементирати методу `fire()` тако да се најпре креира инстанца класе `Bullet`, а затим се поставља на исту позицију у арени на којој се налази дата кула и њено усмерење (`rotation`) се усклађује се кулом.

Наставник треба да се постара да ученици разумеју сваки део програмског кôда.

Ученици затим тестирају своја решења тако што покрећу игру, постављају `Tower` објекат у арену и проверавају да ли лансира пројектиле у одговарајућим интервалима. Наставник охрабрује ученике да решавају евентуалне проблеме, како би осигурали предвиђено понашање.

7. Задатак **6.15.** (20 минута)

Циљ: Интегрисање кула и пројектила у игру**.**

Концепти за дискусију: /.

Активности: Наставник објашњава циљ: створити потпуно функционалну игру. Сходно томе, потребно је интегрисати куле у арену и осигурати да оне исправно интерагују са пројектилима и непријатељима.

Ученици постављају инстанце кула на различите позиције у арени и тестирају њихово понашање у окружењу игре. Наставник објашњава како је могуће додати кулу путем *Greenfoot* интерфејса, чиме се осигурава да ће свака кула бити правилно позиционирана. Наставник затим разматра механизам преклапања конструктора и указује на то како је овакав приступ посебно погодан за иницијализацију `Tower` објеката са различитим ротацијама, при чему он омогућава већу контролу над постављањем и оријентацијом кула у арени.

Наставник задаје ученицима задатак:

> **Задатак 6.15. (Преклапање конструктора у класи `Tower`)**: Имплементирајте додатни конструктор у класи `Tower`, који ће имати и целобројни параметар `rotation`. Потребно је ажурирати и кôд у класи `Arena`, како бисте на различите начине позиционирали куле, позивајући сваки пут одговарајући конструктор класе `Tower`.
>
> **[Commit***:* `bfb6a271f490c341c760e654b3f86a87111c54cb`**]**

Co-funded by the Erasmus+ Programme of the European Union

6.2.4. СЦЕНАРИО: **Даља разрада стратешких елемената игре**

Табела 17. Даља разрада стратешких елемената игре

Назив	Даља разрада стратешких елемената игре
Образовни циљеви и исходи учења	До краја лекције, ученици заокружити логику игре и дефинисати јасне услове за завршетак игре. Након ове лекције, ученици ће имати боље разумевање асоцијација и интеракција између објеката у *Greenfoot* окружењу. Ученици ће у потпуности овладати тиме како различити објекти у игри интерагују и биће у стању да примене ове концепте на сопствене пројекте развоја игара.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **110** минута. 1. Задатак **6.16.** (15 минута) 2. Задатак **6.17.** (30 минута) 3. Тумачење кôда (15 минута) 4. Задатак **6.18.** (30 минута) 5. Обнављање теоријских концепата (20 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње 15-минутном дискусијом о томе како инстанце класе `Bullet` треба да интерагују са релевантним објектима да би се остварило жељено понашање игре (**Задатак 6.16.**). Након тога, ученици ће посветити 30 минута имплементацији претходно дефинисане функционалности (**Задатак 6.17.**). Следећи 15-минутни блок је посвећен статичким методама *Greenfoot* класе: `Greenfoot.showText(String,int,int)` и `Greenfoot.getRandomNumber(int)`. Током наредних 30 минута ученици раде на задатку (**Задатак 6.18.**) који се односи на дефинисања начина креирања непријатеља и услова за завршетак игре. Лекција се завршава 20-минутним теоријским осврtом, односно рекапитулацијом обрађених концепата.
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.

Co-funded by the Erasmus+ Programme of the European Union

Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

6.2.4.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Задатак **6.16.** (15 минута)

Циљ: Дискусија о томе како инстанца класе `Bullet` треба да интерагује са релевантним објектима.

Концепти за дискусију: интеракција између објеката.

Активности: Лекција почиње дискусијом о интеракцијама између објеката у *Greenfoot* окружењу, фокусирајући се на то како инстанце различитих класа комуницирају и утичу једна на другу. Дискусију треба усмерити на то како инстанце класе `Bullet` интерагују са другим објектима, као што су непријатељи и арена.

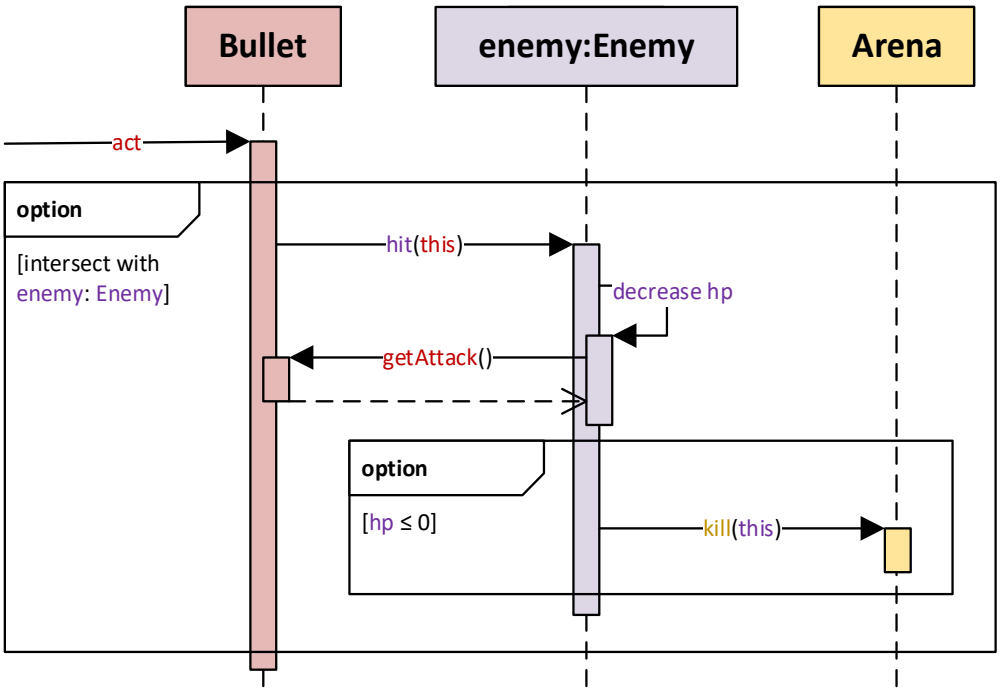
Наставник задаје ученицима задатак:

> **Задатак 6.16. (Дефинисање начина интеграције пројектила у игру)**: Детаљно разрадите начин на који инстанце класе `Bullet` треба да интерагују са другим објектима, тако што ћете дефинисати тачан редослед корака (односно метода које треба да буду позване над различитим објектима). Шта би требало да се деси када пројектил погоди непријатеља?

Ученици дискутују о логици интеракције пројектила и о методама које треба да буду позване. Ученици се подстичу да размишљају и поделе своје идеје о логици интеракције пројектила.

Наставник подсећа ученике на механизам детекције колизије, у контексту колизије пројектила и непријатеља. Потребно је дефинисати ефекте детектоване колизије, као што је смањење енергије непријатеља или уклањање непријатеља из арене.

Наставник може да користи *UML* дијаграм секвенце како би приказао интеракције између објеката `Bullet`, `Enemy` и `Arena`.



Слика 13. UML дијаграм секвенце (решење задатка 6.16.)

Co-funded by the Erasmus+ Programme of the European Union

2. Задатак **6.17.** (30 минута)

Циљ: Имплементирање интеракције пројектила и непријатеља**.**

Концепти за дискусију: /.

Активности: Наставник задаје ученицима задатак:

> **Задатак 6.17. (Имплементација интеракције пројектила и непријатеља)**: Имплементирајте логику обраде колизије пројектила и непријатеља, укључујући ефекте колизије.
>
> **[Commit***:* `dcfe31bc006b7f3dcd8b8b759cc1be901c32913c`**]**

Наставник затим разлаже задатак на кораке и води ученике кроз сваки од њих.

- Потребно је дефинисати неопходне атрибуте и методе класа `Bullet` и `Enemy`.
- Потребно је у класи `Bullet` детектовати да је дошло до колизије са инстанцом непријатеља и позвати методу `hit(Bullet)` класе `Enemy`.
- Потребно је у класи `Enemy` имплементирати методу `hit(Bullet)` путем које ће се одредити ефекти колизије, као што је умањивање енергије непријатеља или његово уклањање из игре.

Наставник треба да се постара да ученици разумеју сваки део програмског кôда.

Ученици затим тестирају своја решења тако што покрећу игру, како би осигурали да се интеракција између пројектила и непријатеља понаша онако како је предвиђено.

3. Тумачење кôда (15 минута)

Циљ: Разумевање понашања специфичних *Greenfoot* метода које ће бити корисне у игри**.**

Концепти за дискусију: приказ текста, генерисање насумичних бројева.

Активности: Наставник уводи методе `Greenfoot.showText(String,int,int)` и `Greenfoot.getRandomNumber(int)`.

Метода `Greenfoot.showText(String,int,int)` се користи за приказивање текста на екрану на одређеним координатама. Наставник објашњава да је ова метода корисна за приказивање информација о игри као што су резултати, ниво енергије или упутства директно на екрану игре.

Метода `Greenfoot.getRandomNumber(int)` генерише насумични број између **0** (укључујући и нулу) и наведене вредности (која неће бити укључена). Наставник дискутује о томе како се ова метода може користити за увођење случајности у игру, као што је стварање непријатеља на насумичним локацијама или генерисање насумичних образаца кретања.

4. Задатак **6.18.** (30 минута)

Циљ: Креирање непријатеља и дефинисање услова за завршетак игре**.**

Концепти за дискусију: логика игре, услови за завршетак игре.

Активности: Наставник задаје ученицима задатак:

> **Задатак 6.18. (Имплементација)**: Имплементирајте процес креирања непријатеља, при чему је потребно имплементирати и механизам задршке, како би се контролисала учесталост креирања непријатеља. Потребно је имплементирати и услове за завршетак игре. Игра се завршава ако је и последњи непријатељ уклоњен. У том случају је потребно зауставити игру и приказати одговарајућу поруку на екрану.
>
> **[Commit***:* `d48341a095561500af6032d5c8f56e201060f9a4`**]**

Co-funded by the Erasmus+ Programme of the European Union

Опис решења:

У телу `act()` методе класе `Arena` ће се периодично покретати процес креирања непријатеља. Требало би увести механизам задршке за управљање учесталошћу креирања непријатеља. Потребно је имплементирати методу `spawn()`у класи `Arena` која ће заправо креирати инстанцу непријатеља, а коју треба позвати из `act()` методе класе `Arena`. Метода `spawn()`треба да креира инстанцу класе `Enemy`, постави одговарајуће вредности атрибута и дода инстанцу у арену.

Такође је потребно дефинисати и имплементирати услове за крај игре. Када број инстанци непријатеља падне на нулу, игра је готова и играч је победио. Да би се то постигло, потребно је увести атрибут у класу `Arena` који ће пратити број креираних инстанци непријатеља. Овај атрибут треба повећати сваки пут када се непријатељ појави у арени, а смањити приликом уклањања из арене. Методу `kill(Enemy)` класе `Arena` треба прилагодити тако да се, унутар ње, проверава да ли су сви непријатељи уклоњени. Ако јесу, потребно је зауставити игру позивом методе `Greenfoot.stop()` и приказати поруку о победи. Стога, позивање методе `Greenfoot.stop()` треба да буде последња наредба у методи `kill(Enemy)`.

5. Обнављање теоријских концепата (20 минута)

Циљ: Резимирање целокупне области.

Концепти за дискусију: асоцијације, интеракција између објеката.

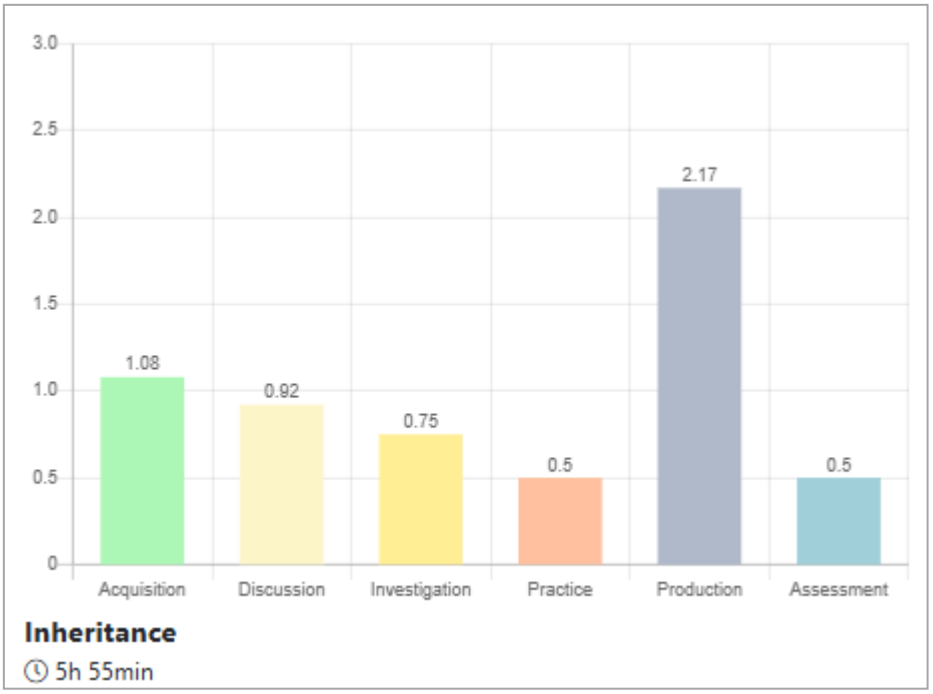
Активности: Наставник даје резиме лекције и осврће се, заједно са ученицима, на кључне појмове.

Co-funded by the Erasmus+ Programme of the European Union

7. НАСЛЕЂИВАЊЕ

7.1. Структура наставних активности и задаци

На следећој слици је дат приказ расподеле ангажовања ученика према типу активности, на основу искуства наставника средњих школа у Словачкој Републици, а који је генерисан у алату за креирање наставних сценарија.



Слика 14. Предложена расподела ангажовања ученика према типу активности за тематску целину „Наслеђивање“

Задатак 7.1. Идентификовање заједничких карактеристика класа

Уочите сличности у понашању класа `Orb` и `Direction`.

Задатак 7.2. Увођење наткласе

Дефинишите нову класу `PassiveActor` са методом `act()`. Класе `Orb` и `Direction` треба сада да буду изведене из класе `PassiveActor`. Промените програмски кôд класа `Orb` и `Direction`, тако што ћете из њих уклонити `act()` методе (будући да ће ову методу наследити од класе `PassiveActor`).

[Commit: `afe617814c07a5d885ed06479bf71deda8725f19`]

Задатак 7.3. Увођење апстрактне класе

Прогласите класу `PassiveActor` апстрактном.

[Commit: `f7a5702cae29bf21c9c88620d01ef64e4127c21c`]

Задатак 7.4. Анализа понашања протагониста

Анализирајте класе `Bullet` и `Enemy`, како бисте препознали сличности у њиховом понашању.

Co-funded by the Erasmus+ Programme of the European Union

Задатак 7.5. Увођење нове апстрактне класе

Дефинишите нову апстрактну класу `MovingActor` са методом `act()`. Класу `MovingActor` треба дефинисати као натклasu класа `Bullet` и `Enemy`.

[Commit: `43e53b533563ce0a860b294ad9009f77409c48d4`**]**

Задатак 7.6. Идентификовање заједничких својстава протагониста

Идентификујте заједничке атрибуте класа `Bullet` и `Enemy`. Анализирајте `act()` методе класа `Bullet` и `Enemy`.

Задатак 7.7. Рефакторисање дела програмског кôда који се односи на кретање инстанци

Преместите идентификоване опште атрибуте `moveDelay` и `nextMoveCounter` из класа `Bullet` и `Enemy` у натклasu `MovingActor`. Додајте параметризовани конструктор у класу `MovingActor` да бисте обезбедили да ови атрибути буду иницијализовани. Позовите овај конструктор из изведених класа `Bullet` и `Enemy`, коришћењем кључне речи `super` и обезбеђивањем одговарајућих вредности параметара. Преместите у класу `MovingActor` онај део програмског кôда који је идентичан у `act()` методама класа `Bullet` и `Enemy` (друге делове програмског кôда ових метода не треба мењати). На крају је потребно да прва наредба у `act()` методама класа `Bullet` и `Enemy`, буде позив `act()` методе класе `MovingActor`.

[Commit: `ca1f010a63445c1847b74259a1c6cd4817121db3`**]**

Задатак 7.8. Увођење сопствених протагониста

Дефинишите неколико изведених класа класе `Enemy`, које ће представљати различите врсте непријатеља, нпр. `Frog` (жаба) и `Spider` (паук). Придружите им одговарајуће слике, водећи рачуна да буду адекватних димензија (тј. да нису веће од једне ћелије). У свакој од класа дефинишите непараметризовани конструктор, у чијем телу ће се позивати конструктор њихове заједничке натклase (тј. класе `Enemy`) са вредностима параметара које су својствене тој класи. У изведеним класама имплементирајте `act()` методу или просто позовите `act()` методу натклase.

[Commit: `b0ac1fbe793548a32f7700c292aed631918c8388`**]**

Задатак 7.9. Динамичко креирање инстанци протагониста и илустрација принципа супституције

Преправите тело методе `spawn()` у класи `Arena`. Потребно је да креирате инстанцу, било класе `Frog`, било класе `Spider` и сачувате њену референцу у променљивој типа `Enemy`. При томе је потребно да употребите гранање да бисте дефинисали која ће од инстанци бити креирана ако је услов испуњен, а која ако није. Критеријум, тј. услов гранања може бити произвољан (нпр. на основу насумичне вредности или неког егзактног израза). Покрените апликацију и проверите да ли ова измена утиче на њено понашање.

[Commit: `8cd4397f585ec957bbc18ca98e01823f434a13a6`**]**

Задатак 7.10. Пројектовање нове хијерархије класа

Потребно је пројектовати хијерархију арена. Свака појединачна класа, која ће бити изведена из класе `Arena`, може представљати другачији ниво игре.

Co-funded by the Erasmus+ Programme of the European Union

Задатак 7.11. Дефинисање универзалне арене

Класи `Arena` је потребно додати атрибуте `spawnPositionX`, `spawnPositionY` и `spawnRotation`, Потребно је променити конструктор класе како би се обезбедило да се ови атрибути иницијализују путем параметара. Ове нове атрибуте је затим потребно употребити у методама `spawn()` и `respawn(Enemy)` да би се задала почетна позиција и усмерење одређеног непријатеља. Потребно је конструктору класе `Arena` додати још два параметра (`width` и `height`) путем којих ће се задати димензија саме арене. Вредности ових параметара треба проследити конструктору натклase класе `Arena`. Коначно, водите рачуна о томе да класа `Arena` представља само „костур“ за пројектовање других класа, због чега је потребно да се она дефинише као апстрактна.

[Commit*:* `e9844d7d9b5f19969618b469ebc907d0fe3c1357`**]**

Задатак 7.12. Увођење изведене класе `DemoArena`

Дефинишите нову класу `DemoArena` као изведену класу класе `Arena`. Имплементирајте одговарајући конструктор који ће позвати конструктор натклase са вредностима параметара које ће осигурати креирање арене одговарајућих димензија и одговарајуће почетно позиционирање и усмеравање инстанци непријатеља. Поред тога, потребно је у тело конструктора класе `DemoArena`, преместити онај део програмског кôда конструктора класе `Arena` који се односи на дефинисање почетне поставке (тј. распореда) инстанци класа `Orb`, `Direction` и `Tower`. Креирајте нову инстанцу класе `DemoArena` (десним кликом на класу `DemoArena` и избором опције `new DemoArena()` у њеном контекстном менију).

[Commit*:* `6a6569774b5735f453a56c7cb2cdbf19d228eae9`**]**

Задатак 7.13. Самостални даљи развој хијерархије класа

Осмислите и уведите нове интересантне класе које би се могле извести из класе `Arena`. Своје решење можете поделити са осталим ученицима.

С обзиром на значај концепта наслеђивања у објектно-оријентисаном програмирању, тренутна структура пројекта отвара простор за даљу дискусију и проширења. У том смислу, могу се размотрити додатне класе и хијерархије, и могу се увести нове класе, методе и атрибути. Са друге стране, наставник може и да прилагоди ову тему и да користи само хијерархије класа предложене у овом материјалу, како би представио добробити коришћења наслеђивања.

Co-funded by the Erasmus+ Programme of the European Union

7.2. Наставни сценарији

Припремљена су **четири** наставна сценарија за тематску целину „Наслеђивање“.

7.2.1. СЦЕНАРИО: **Увод у наслеђивање у *Greenfoot* окружењу**

Табела 18. Увод у наслеђивање у Greenfoot окружењу

Назив	Увод у наслеђивање у *Greenfoot* окружењу
Образовни циљеви и исходи учења	Након ове лекције, ученици би требало да разумеју основне концепте наслеђивања. Разумевање ових концепата се продубљује кроз развој игре у *Greenfoot* окружењу, чиме се подстиче креативност, тимски дух и страственији приступ програмирању.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **75** минута. 1. Концепт наслеђивања (15 минута) 2. Хијерархија класа и наслеђивање (15 минута) 3. Задаци **7.1.** и **7.2.** (15 минута + 15 минута) 4. Увод у апстрактне класе (5 минута) 5. Задатак **7.3.** (10 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње 15-минутним уводом током којег наставник успоставља контекст у односу на претходне лекције, али и будући развој игре. Након тога следи 15-минутни блок, током којег ученици и наставници дискутују о хијерархији класа у оквиру игре. Да би се објаснили концепти везани за наслеђивање, анализирају се класе `Orb` и `Direction`. Током следећег 15-минутног задатка, ученици треба да идентификују заједничка својстава ових класа (**Задатак 7.1.**). Потребно је да уоче да ове класе заправо немају никакво активно дејство, тј. метода `act()` у обема класама има празно тело. Сходно томе, потребно је да ученици идентификују да им је метода `act()` заједничка. На основу идентификованих заједничких својстава, током наредних 15 минута ученици треба да имплементирају нову класу `PassiveActor`, са методом `act()` (**Задатак 7.2.**). Током следећег петоминутног блока наставник уводи концепт апстрактне класе. Апстрактне класе служе као „костури“ за друге класе и, као такве, се не могу инстанцирати, али могу бити од користи приликом пројектовања хијерархије класа.

Co-funded by the Erasmus+ Programme of the European Union

	Током наредних 10 минута ученици решавају **Задатак 7.3.** Наиме, с обзиром на то да класа `PassiveActor` заправо представља „костур“, потребно је да она буде апстрактна. Поред тога, `PassiveActor` треба да буде наткласа класа `Orb` и `Direction`. Будући да је `act()` метода већ дефинисана у класи `PassiveActor`, потребно ју је уклонити из класа `Orb` и `Direction`. Кроз ове активности ученици ће се упознати са напреднијим техникама објектно-оријентисаног програмирања, а посебно са наслеђивањем, хијерархијом класа и апстрактним класама.
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

7.2.1.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Концепт наслеђивања (15 минута)

Циљ: Успостављање контекста, увођење концепта наслеђивања кроз примере из свакодневног живота, дискусија о добробити његовог коришћења**.**

Концепти за дискусију: наслеђивање, примери наслеђивања у свакодневном животу.

Активности: У уводном делу се успоставља контекст у односу на претходне лекције. Наставник уводи појам **наслеђивања**. Наставник треба да приближи овај концепт ученицима коришћењем примера из свакодневног живота (нпр. посматрањем односа родитељ-дете: деца наслеђују карактеристике од својих родитеља, као што су тип косе, боја очију итд.). Потребно је продискутовати добробити коришћења наслеђивања у програмирању. Дати концепти се разматрају у контексту *Greenfoot* окружења и програмског језика *Java*.

2. Хијерархија класа и наслеђивање (15 минута)

Циљ: Увођење концепта хијерархије класа у контексту наслеђивања и наслеђивању карактеристика, дискусија о примерима из свакодневног живота, указивање на добробити коришћења хијерархије класа**.**

Концепти за дискусију: наслеђивање, хијерархија класа, примери хијерархија класа у свакодневном животу, добробити коришћења наслеђивања и хијерархија класа.

Активности: Наставник уводи концепт **хијерархије класа** у контексту наслеђивања тако што објашњава концепте класа предака (оваква класа се понекад назива и: наткласа, базна класа, класа родитељ) и класа потомака (оваква класа се понекад назива и: поткласа, изведена класа, класа дете), при чему се може продискутовати о претходно разматраним примерима класа из свакодневног живота. Неопходно је истаћи да изведене класе наслеђују карактеристике (тј. атрибуте и методе) од наткласе, али и напоменути да изведене класе могу да поседују додатна својства, која нису доступна наткласи.

Потребно је поразговарати и о добробити коришћења хијерархија класа у контексту наслеђивања. Такође ученицима треба појаснити да у програмском језику *Java* класа може имати више поткласа, али само једну наткласу. Примера ради, може се размотрити улога класе `Object` у контексту хијерархије класа и наслеђивања.

3. Задаци **7.1.** и **7.2.** (15 минута + 15 минута)

Циљ: Идентификовање заједничких карактеристика у класама игре, успостављање заједничке наткласе, имплементација хијерархије класа**.**

Концепти за дискусију: наслеђивање, хијерархија класа, имплементација наслеђивања и хијерархије класа.

Активности: Наставник задаје ученицима задатак:

> **Задатак 7.1. (Идентификовање заједничких карактеристика класа)**: Уочите сличности у понашању класа `Orb` и `Direction`.

Потребно је уочити да ниједна од класа нема активно дејство, тј. да метода `act()` у обема класама има празно тело. Сходно томе, може се увести заједничка наткласа (нпр. `PassiveActor`) чија ће `act()` метода бити празна, што ће учинити понашање изведених класа очигледнијим.

Ове класе (`PassiveActor`, `Orb`, и `Direction`) треба да послуже за представљање хијерархије класа у контексту наслеђивања, а наставник може визуелно да представи хијерархију класа користећи хијерархијски дијаграм.

Co-funded by the Erasmus+ Programme of the European Union

Наставник затим задаје ученицима задатак:

> **Задатак 7.2. (Увођење натклase)**: Дефинишите нову класу `PassiveActor` са методом `act()`. Класе `Orb` и `Direction` треба сада да буду изведене из класе `PassiveActor`. Промените програмски кôд класа `Orb` и `Direction`, тако што ћете из њих уклонити `act()` методе (будући да ће ову методу наследити од класе `PassiveActor`).

[Commit: afe617814c07a5d885ed06479bf71deda8725f19]

Наставник треба да истакне шта се променило у *Greenfoot* окружењу када је класа `Actor` замењена класом `PassiveActor`.

4. Увод у апстрактне класе (5 минута)

Циљ: Увођење концепта апстрактне класе, дискусија о примерима из свакодневног живота како би се илустровала њихова примена и специјализација у потклaсе.

Концепти за дискусију: наслеђивање, хијерархија класа, апстрактне класе, примери апстрактних класа.

Активности: Наставник уводи појам **апстрактне класе**. Потребно је објаснити да апстрактне класе представљају генерализације на веома високом нивоу апстракције, тј. дефинишу „костур“ других класа, и да се, као такве, не могу инстанцирати, али да су од значаја када је реч о пројектовању хијерархије класа. Наставник може са ученицима продискутовати примере апстрактних класа и њихових потклаcа (нпр. класа **Рачунар** може се дефинисати као апстрактна класа, која се даље може специјализовати у класе **Десктоп**, **Лаптоп** и **МобилниТелефон**, свака са одређеним скупом њој специфичних својстава итд.). Други пример могу бити геометријске фигуре. **Правоугаоник** или **Троугао** могу наследити апстрактну класу **ГеометријскиОблик**. Треба истаћи да не постоји формула за израчунавања обима или површине општег геометријског облика. Међутим, формуле за обим и површину правоугаоника и троугла су познате. **Квадрат** може наследити **Правоугаоник**. Ученици треба да осмисле додатне примере геометријских облика и тела.

5. Задатак **7.3.** (10 минута)

Циљ: Дискусија о улози класе `PassiveActor`, дефинисање класе `PassiveActor` као апстрактне класе.

Концепти за дискусију: наслеђивање, хијерархија класа, апстрактне класе, дефинисање апстрактне класе.

Активности: Наставник разрађује концепт апстрактне класе у контексту *Greenfoot* окружења и програмског језика *Java*. Наставник указује на то да у контексту развоја игре, класа `PassiveActor` представља „костур“, због чега је потребно да се она дефинише као апстрактна.

Наставник задаје ученицима задатак:

> **Задатак 7.3. (Увођење апстрактне класе)**: Прогласите класу `PassiveActor` апстрактном.

[Commit: f7a5702cae29bf21c9c88620d01ef64e4127c21c]

Наставник указује на то да се приликом дефинисања метода класе `PassiveActor` може наићи на методе које се заправо не могу имплементирати у овој заједничкој класи, па се стога њихова имплементација мора препустити изведеним класама. На пример, може се посматрати класа **ГеометријскиОблик** и изведене класе **Правоугаоник** или **Троугао**. Иако ће сваки од конкретних облика имплементирати методе за рачунање свог обима и површине, ове методе није могуће имплементирати у заједничкој натклаcи. Наставник објашњава да, када се нека метода класе дефинише као апстрактна, да се тиме у суштини указује на то да ће она бити имплементирана у изведеним класама. Наставник истиче то да класа која садржи апстрактну методу мора бити апстрактна.

Co-funded by the Erasmus+ Programme of the European Union

7.2.2. СЦЕНАРИО: **Овладавање концептом наслеђивања у *Greenfoot* окружењу (део 1)**

Табела 19. Овладавање концептом наслеђивања у Greenfoot окружењу (део 1)

Назив	Овладавање концептом наслеђивања у *Greenfoot* окружењу (део 1)
Образовни циљеви и исходи учења	Након ове лекције, ученици би требало да разумеју напредније концепте наслеђивања. Разумевање ових концепата се продубљује кроз развој игре у *Greenfoot* окружењу, чиме се подстиче креативност, тимски дух и страственији приступ програмирању.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **95** минута. 1. Задатак **7.4.** (15 минута) 2. Задатак **7.5.** (15 минута) 3. Задатак **7.6.** (15 минута) 4. Кључна реч `super` (20 минута) 5. Задатак **7.7.** (30 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње 15-минутним задатком (**Задатак 7.4.**) у оквиру којег се од ученика очекује да анализирају понашање инстанци класа `Bullet` и `Enemy`, а пре свега у погледу тога како се померају и како реагују на своје окружење. Потребно је уочити да се инстанце ових класа слично понашају, тј. померају се на исти начин и након тога на одређени начин реагују на своје окружење. На основу идентификованог заједничког понашања, у следећем 15-минутном задатку (**Задатак 7.5.**) се од ученика очекује да имплементирају нову апстрактну класу `MovingActor` са методом `act()`. Ова класа заправо треба да представља заједничку натклasu, а путем њене `act()` методе ће се касније имплементирати заједничко понашање (тј. начин кретања инстанци класа `Bullet` и `Enemy`), док ће се у свакој од ових класа онда разрадити понашање које је својствено за сврху те класе. Сходно томе, потребно је да ученици класу `MovingActor` дефинишу као натклasu класа `Bullet` и `Enemy`. Наредни 15-минутни задатак подразумева даљу анализу класа `Bullet` и `Enemy` са нагласком на уочавању сличности и разлика у погледу њихових атрибута, као и начина имплементације њиховог кретања (**Задатак 7.6.**). У том контексту, анализирају се `act()` методе датих класа, али и атрибути `moveDelay` и `nextMoveCounter`. Потребно је препознати да је програмски кôд `act()` метода класа `Bullet` и `Enemy`, у делу у којем се имплементира начин кретања самих инстанци, идентичан.

Co-funded by the Erasmus+ Programme of the European Union

	Након тога следи 20-минутни блок посвећен увођењу кључне речи `super` у контексту наслеђивања. Током последњег 30-минутног блока врши се рефакторисање дела програмског кôда који се односи на кретање протагониста (**Задатак 7.7.**). Сходно томе, претходно идентификовани заједнички атрибути `moveDelay` и `nextMoveCounter` треба да се преместе из изведених класа `Bullet` и `Enemy` у натклasu `MovingActor`. Поред тога, како би се обезбедила иницијализација ових атрибута, у класи `MovingActor` је потребно дефинисати параметризовани конструктор. Дати конструктор треба позвати из изведених класа `Bullet` и `Enemy`, коришћењем кључне речи `super` и обезбеђивањем одговарајућих вредности параметара. Након тога потребно је у класу `MovingActor` преместити онај део програмског кôда `act()` метода који је идентичан у класама `Bullet` и `Enemy` (док друге делове програмског кôда ових метода не треба мењати). Коначно потребно је обезбедити да прва наредба у `act()` методама класа `Bullet` и `Enemy`, буде позив `act()` методе класе `MovingActor`. Кроз ове активности ученици ће се упознати са напреднијим техникама објектно-оријентисаног програмирања, а посебно са апстрактним класама и наслеђивањем.
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

7.2.2.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Задатак **7.4.** (15 минута)

Циљ: Анализирање класа `Bullet` и `Enemy` са нагласком на уочавању сличности у њиховом понашању, посебно у погледу тога како се померају и како реагују на своје окружење**.**

Концепти за дискусију: наслеђивање, хијерархија класа, апстрактне класе.

Активности: Наставник задаје ученицима задатак

> **Задатак 7.4. (Анализа понашања протагониста)**: Анализирајте класе `Bullet` и `Enemy`, како бисте препознали сличности у њиховом понашању.

Потребно је уочити да се инстанце ових класа слично понашају, тј. померају се на исти начин и након тога на одређени начин реагују на своје окружење.

2. Задатак **7.5.** (15 минута)

Циљ: Имплементирање апстрактне класе**.**

Концепти за дискусију: наслеђивање, хијерархија класа, апстрактне класе, начин имплементације апстрактних класа.

Активности: Потребно је ученике подсетити на то да изведене класе наслеђују карактеристике од наткласе. Сходно опсервацијама из претходног задатка, потребно је закључити да би се могла увести нова наткласа са методом у оквиру које ће се имплементирати заједничко понашање инстанци класа `Bullet` и `Enemy`, док ће се у свакој од изведених класа касније разрадити понашање које је својствено за сврху те класе. Треба истаћи то да класа `MovingActor` дефинише само „костур“ за пројектовање других класа, због чега је потребно да се она дефинише као апстрактна.

Наставник задаје ученицима задатак:

> **Задатак 7.5. (Увођење нове апстрактне класе)**: Дефинишите нову апстрактну класу `MovingActor` са методом `act()`. Класу `MovingActor` треба дефинисати као натклasу класа `Bullet` и `Enemy`.

[Commit*:* `43e53b533563ce0a860b294ad9009f77409c48d4`**]**

3. Задатак **7.6.** (15 минута)

Циљ: Даља анализа класа `Bullet` и `Enemy` са нагласком на уочавању сличности и разлика у погледу начина имплементације њиховог кретања**.**

Концепти за дискусију: наслеђивање, хијерархија класа, апстрактне класе.

Активности: Наставник задаје ученицима задатак..

> **Задатак 7.6. (Идентификовање заједничких својстава протагониста)**: Идентификујте заједничке атрибуте класа `Bullet` и `Enemy`. Анализирајте `act()` методе класа `Bullet` и `Enemy`.
>
> Потребно је приметити да је програмски кôд `act()` метода класа `Bullet` и `Enemy`, у делу у којем се имплементира начин кретања самих инстанци, идентичан. Такође треба примети да обе класе имају атрибуте `moveDelay` и `nextMoveCounter`, те се они могу посматрати као општи атрибути.

Co-funded by the Erasmus+ Programme of the European Union

4. Кључна реч `super` (20 минута)

Циљ: Увођење кључне речи `super` у контексту наслеђивања, демонстрирање начина њене употребе за приступање својствима натклассе**.**

Концепти за дискусију: наслеђивање, хијерархија класа, кључна реч `super` у контексту наслеђивања, и место њеног навођења у програмском кôду.

Активности: Наставник уводи кључну реч `super`, која се у контексту наслеђивања може користити:

- како би се позвао конструктор натклассе,
- како би се позвала метода натклассе или
- како би се приступило атрибуту натклассе.

Наставник треба да истакне то да се, у сваком од ових случајева, кључна реч `super` мора навести као прва реч у одговарајућој наредби.

5. Задатак **7.7.** (30 минута)

Циљ: Рефакторисање дела програмског кôда који се односи на кретање инстанци**.**

Концепти за дискусију: наслеђивање, хијерархија класа, кључна реч `super` у контексту наслеђивања.

Активности: Наставник задаје ученицима задатак:

> **Задатак 7.7. (Рефакторисање дела програмског кôда који се односи на кретање инстанци)**: Преместите идентификоване опште атрибуте `moveDelay` и `nextMoveCounter` из класа `Bullet` и `Enemy` у натклассу `MovingActor`. Додајте параметризовани конструктор у класу `MovingActor` да бисте обезбедили да ови атрибути буду иницијализовани. Позовите овај конструктор из изведених класа `Bullet` и `Enemy`, коришћењем кључне речи `super` и обезбеђивањем одговарајућих вредности параметара. Преместите у класу `MovingActor` онај део програмског кôда који је идентичан у `act()` методама класа `Bullet` и `Enemy` (друге делове програмског кôда ових метода не треба мењати). На крају је потребно да прва наредба у `act()` методама класа `Bullet` и `Enemy`, буде позив `act()` методе класе `MovingActor`.
>
> **[Commit***:* `ca1f010a63445c1847b74259a1c6cd4817121db3`**]**

Потребно је продискутовати то да изведене класе могу поседовати додатне карактеристике које нису доступне натклассе (нпр. другачија имплементација `act()` метода).

Co-funded by the Erasmus+ Programme of the European Union

7.2.3. СЦЕНАРИО: **Овладавање концептом наслеђивања у *Greenfoot* окружењу (део 2)**

Табела 20. Овладавање концептом наслеђивања у Greenfoot окружењу (део 2)

Назив	Овладавање концептом наслеђивања у *Greenfoot* окружењу (део 2)
Образовни циљеви и исходи учења	Након ове лекције, ученици би требало да разумеју напредније концепте наслеђивања. Разумевање ових концепата се продубљује кроз развој игре у *Greenfoot* окружењу, чиме се подстиче креативност, тимски дух и страственији приступ програмирању.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **70** минута. 1. Задатак **7.8.** (30 минута) 2. Увод у принцип супституције (20 минута) 3. Задатак **7.9.** (20 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње 30-минутним задатком (**Задатак 7.8.**) у оквиру којег се од ученика очекује да уведу нове протагонисте, изведене из класе `Enemy`, нпр. `Frog` и `Spider`. Потребно је и да им придруже одговарајуће слике, али и да у свакој од класа дефинишу непараметризовани конструктор у чијем телу ће се позивати конструктор њихове заједничке натклase (тј. класе `Enemy`). Током наредних 20 минута наставник уводи принцип супституције Барбаре Лисков. Овај принцип, као један од пет *SOLID* принципа објектно-оријентисаног пројектовања, наводи да би метода која као параметар очекује референцу на инстанцу одређене натклase, требало да може да, уместо ње, прими и референцу на инстанцу неке од њених изведених класа. Последњи 20-минутни блок је посвећен динамичком увођењу у игру инстанци нових протагониста (**Задатак 7.9.**) У том контексту разматра се метода `spawn()` класе `Arena` а инстанце нових класа се креирају према различитим критеријумима и чувају у променљивој типа `Enemy`. Указује се на то да се понашање апликације неће променити , тј. да нису потребне никакве додатне измене у кôду, чиме се илуструје примена принципа супституције. Кроз ове активности ученици ће се упознати са напреднијим техникама објектно-оријентисаног програмирања, а посебно са практичном применом наслеђивања.

Co-funded by the Erasmus+ Programme of the European Union

Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

7.2.3.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Задатак **7.8.** (30 минута)

Циљ: Дефинисање изведених класа класе `Enemy`**.**

Концепти за дискусију: наслеђивање, хијерархија класа, кључна реч `super`.

Активности: Наставник задаје ученицима задатак:

> **Задатак 7.8. (Увођење сопствених протагониста)**: Дефинишите неколико изведених класа класе `Enemy`, које ће представљати различите врсте непријатеља, нпр. `Frog` (жаба) и `Spider` (паук). Придружите им одговарајуће слике, водећи рачуна да буду адекватних димензија (тј. да нису веће од једне ћелије). У свакој од класа дефинишите непараметризовани конструктор, у чијем телу ће се позивати конструктор њихове заједничке натклase (тј. класе `Enemy`) са вредностима параметара које су својствене тој класи. У изведеним класама имплементирајте `act()` методу или просто позовите `act()` методу натклase.

[Commit*:* `b0ac1fbe793548a32f7700c292aed631918c8388`**]**

2. Увод у принцип супституције (20 минута)

Циљ: Увођење принципа супституције, анализа предности примене овог принципа**.**

Концепти за дискусију: наслеђивање, хијерархија класа, референтне променљиве, принцип супституције.

Активности: Наставник уводи принцип супституције Барбаре Лисков, као принцип који је један од пет *SOLID* принципа објектно-оријентисаног пројектовања. Наставник објашњава да принцип подразумева да би метода, која као параметар очекује референцу на инстанцу одређене натклase, требало да може да, уместо ње, прими и референцу на инстанцу неке од њених изведених класа. Како би појаснио овај принцип, наставник треба да са ученицима продискутује примере. На пример, нека је класа **Рачунар** дефинисана као натклаcа, а **Десктоп**, **Лаптоп** и **МобилниТелефон** као изведене класе. Према принципу супституције, методе које очекују инстанцу класе **Рачунар** ће такође моћи да приме и инстанцу неке од њених изведених класа, без потребе за било каквом изменом кôда. Наставник затим треба да поразговара са ученицима о предностима примене принципа супституције у контексту наслеђивања.

3. Задатак **7.9.** (20 минута)

Циљ: Демонстрирање примене принципа супституције и динамичко креирање инстанци протагониста.

Концепти за дискусију: наслеђивање, хијерархија класа, референтне променљиве, принцип супституције.

Активности: Наставник задаје ученицима задатак. Треба напоменути да ниједан други део програмског кôда у апликацији не треба мењати.

> **Задатак 7.9. (Динамичко креирање инстанци протагониста и илустрација принципа супституције)**: Преправите тело методе `spawn()` у класи `Arena`. Потребно је да креирате инстанцу, било класе `Frog` било класе `Spider` и сачувате њену референцу у променљивој типа `Enemy`. При томе је потребно да употребите гранање да бисте дефинисали која ће од инстанци бити креирана ако је услов испуњен, а која ако није. Критеријум, тј. услов гранања може бити произвољан (нпр. на основу насумичне вредности или неког егзактног израза). Покрените апликацију и проверите да ли ова измена утиче на њено понашање.

[Commit*:* `8cd4397f585ec957bbc18ca98e01823f434a13a6`**]**

Наставник треба да истакне то да чињеница да се понашање апликације није променило, тј. да нису потребне никакве додатне измене у програмском кôду, доказује принцип супституције.

Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the Erasmus+ Programme of the European Union

7.2.4. СЦЕНАРИО: **Овладавање концептом наслеђивања у *Greenfoot* окружењу (део 3)**

Табела 21. Овладавање концептом наслеђивања у Greenfoot окружењу (део 3)

Назив	Овладавање концептом наслеђивања у *Greenfoot* окружењу (део 3)
Образовни циљеви и исходи учења	Након ове лекције, ученици би требало да суштински разумеју основне и напредније концепте наслеђивања. Разумевање ових концепата се продубљује кроз развој игре у *Greenfoot* окружењу, чиме се подстиче креативност, тимски дух и страственији приступ програмирању.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **115** минута. 1. Задатак **7.10.** (20 минута) 2. Задаци **7.11.** (30 минута) и **7.12.** (15 минута) 3. Задатак **7.13.** (30 минута) 4. Обнављање теоријских концепата (20 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње 20-минутном дискусијом о могућој хијерархији арена (**Задатак 7.10.**). Током наредног 30-минутног задатка (**Задатак 7.11.**) дефинише се универзална арена. Потребно је да ученици класи `Arena` додају атрибуте `spawnPositionX`, `spawnPositionY` и `spawnRotation`, осигурају њихову иницијализују унутар тела конструктора и употребе их у методама `spawn()` и `respawn(Enemy)`. Такође је потребно да самом конструктору додају још два параметра (`width` и `height`) путем којих ће се задати димензија арене. Будући да класа `Arena` представља само „костур“ за пројектовање других класа, потребно је да је дефинишу као апстрактну. Следећи 15-минутни задатак (**Задатак 7.12.**) фокусиран је на дефинисање конкретне арене. Сходно томе, ученици треба да уведу изведену класу `DemoArena`. Потребно је да имплементирају одговарајући конструктор који ће позвати конструктор наткласе. Поред тога, потребно је да у тело конструктора класе `DemoArena`, преместе онај део програмског кôда конструктора класе `Arena` који се односи на дефинисање почетне поставке (тј. распореда) инстанци класа `Orb`, `Direction` и `Tower`. Коначно, потребно је да креирају нову инстанцу класе `DemoArena`. У оквиру последњег 30-минутног задатка (**Задатак 7.13.**) се од ученика очекује да осмисле, а затим и уведу у игру, нове занимљиве конкретне арене (тј. класе које би могле бити изведене из класе `Arena`), при чему своје решење могу поделити са другим ученицима.

Co-funded by the Erasmus+ Programme of the European Union

	Лекција се завршава 20-минутним теоријским осртом, односно рекапитулацијом обрађених концепата. Кроз ове активности ученици ће темељно савладати напредније технике објектно-оријентисаног програмирања, а посебно практичну примену наслеђивања.
Вредновање	Увођење елемената игре и надметања у наставу (енгл. *gamification*) омогућава само неформално вредновање, али ће повећати ниво заинтересованости и истинске мотивисаности, као и степен усвојеног знања целокупне групе. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

7.2.4.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Задатак **7.10.** (20 минута)

Циљ: Пројектовање нове хијерархије класа**.**

Концепти за дискусију: наслеђивање, хијерархија класа, конструктори.

Активности: Наставник излаже ученицима задатак и покреће дискусију:

Задатак 7.10. (Пројектовање нове хијерархије класа): Потребно је пројектовати хијерархију арена. Свака појединачна класа, која ће бити изведена из класе `Arena`, може представљати другачији ниво игре.

2. Задаци **7.11.** (30 минута) и **7.12.** (15 минута)

Циљ: Увођење универзалне (апстрактне) арене, извођење и дефинисање конкретне арене, анализа хијерархије класа арена**.**

Концепти за дискусију: наслеђивање, хијерархија класа, апстрактне класе, конструктори.

Активности: Потребно је увести универзалну арену, па наставник задаје ученицима задатак:

Задатак 7.11. (Дефинисање универзалне арене): Класи `Arena` је потребно додати атрибуте `spawnPositionX`, `spawnPositionY` и `spawnRotation`. Потребно је променити конструктор класе како би се обезбедило да се ови атрибути иницијализују путем параметара. Ове нове атрибуте је затим потребно употребити у методама `spawn()` и `respawn(Enemy)` да би се задала почетна позиција и усмерење одређеног непријатеља. Потребно је конструктору класе `Arena` додати још два параметра (`width` и `height`) путем којих ће се задати димензија саме арене. Вредности ових параметара треба проследити конструктору натклase класе `Arena`. Коначно, водите рачуна о томе да класа `Arena` представља само „костур“ за пројектовање других класа, због чега је потребно да се она дефинише као апстрактна.

[Commit*:* e9844d7d9b5f19969618b469ebc907d0fe3c1357**]**

Наставник затим задаје ученицима задатак:

Задатак 7.12. (Увођење изведене класе `DemoArena`): Дефинишите нову класу `DemoArena` као изведену класу класе `Arena`. Имплементирајте одговарајући конструктор који ће позвати конструктор натклase са вредностима параметара које ће осигурати креирање арене одговарајућих димензија и одговарајуће почетно позиционирање и усмеравање инстанци непријатеља. Поред тога, потребно је у тело конструктора класе `DemoArena`, преместити онај део програмског кôда конструктора класе `Arena` који се односи на дефинисање почетне поставке (тј. распореда) инстанци класа `Orb`, `Direction` и `Tower`. Креирајте нову инстанцу класе `DemoArena` (десним кликом на класу `DemoArena` и избором опције `new DemoArena()` у њеном контекстном менију).

[Commit*:* 6a6569774b5735f453a56c7cb2cdbf19d228eae9**]**

3. Задатак **7.13.** (30 минута)

Циљ: Дефинисање сопствених изведених класа**.**

Концепти за дискусију: наслеђивање, хијерархија класа, апстрактне класе, конструктори.

Активности: Наставник задаје ученицима задатак:

Задатак 7.13. (Самостални даљи развој хијерархије класа): Осмислите и уведите нове интересантне класе које би се могле извести из класе `Arena`. Своје решење можете поделити са осталим ученицима.

Co-funded by the Erasmus+ Programme of the European Union

4. Обнављање теоријских концепата (20 минута)

Циљ: Резимирање целокупне области.

Концепти за дискусију: наслеђивање, хијерархија класа, апстрактне класе, кључна реч `super`, принцип супституције.

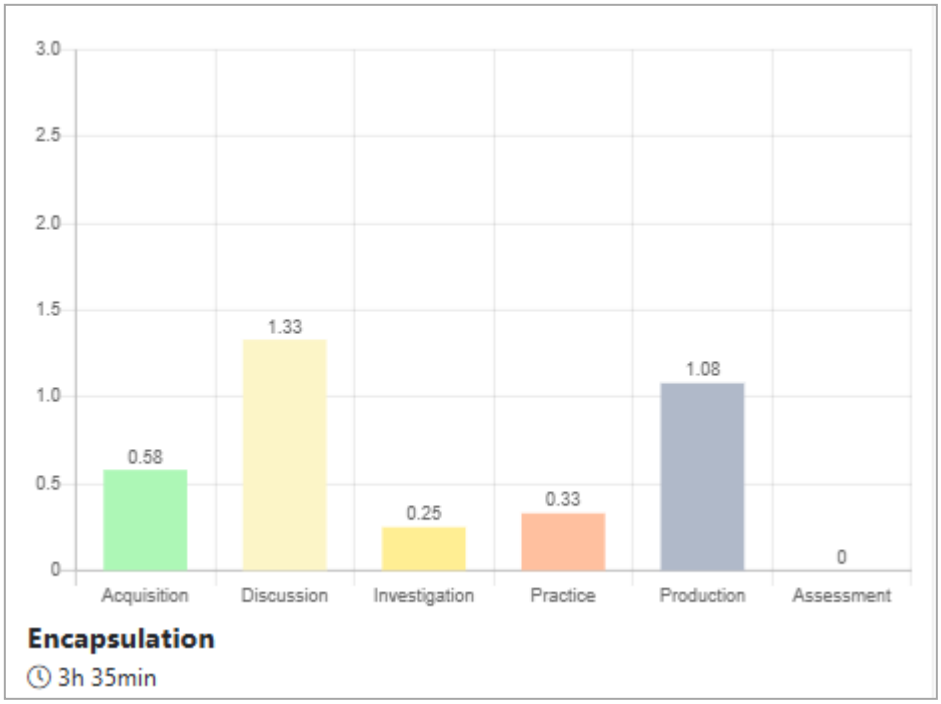
Активности: Наставник даје резиме лекције и осврће се, заједно са ученицима, на све претходно обрађене концепте.

Co-funded by the Erasmus+ Programme of the European Union

8. ЕНКАПСУЛАЦИЈА И СКРИВАЊЕ ИНФОРМАЦИЈА

8.1. Структура наставних активности и задаци

На следећој слици је дат приказ расподеле ангажовања ученика према типу активности, на основу искуства наставника средњих школа у Словачкој Републици, а који је генерисан у алату за креирање наставних сценарија.



Слика 15. Предложена расподела ангажовања ученика према типу активности за тематску целину „Енкапсулација и скривање информација“

Задатак 8.1. Увођење класе `ManualTower`

Дефинишите класу `ManualTower` као изведену класу класе `Tower`. Потребно је да имплементирате конструкторе и да осигурате да ће конструктори наткласе бити позвани. Имплементирајте и методу `act()` тако да она позива `act()` методу своје наткласе. Креирајте неколико инстанци дате класе и додајте их у арену.

[Commit: `63a02fa0c5080165cba8b467da08c4b65f31d0a8`**]**

Задатак 8.2. Имплементација методе `changeControl(boolean)`

Уведите логички атрибут `isManuallyControlled` и иницијализујте га са вредношћу `false`. Имплементирајте методу `changeControl(boolean)` путем које се мења вредност атрибута `isManuallyControlled` и, у складу са њим, иконица дате куле.

[Commit: `2257746b7dac5eaab7acc55d6493319230338f3a`**]**

Задатак 8.3. Праћење промена интерног стања објекта

Директно (тј. ручно) покрените методу `changeControl(boolean)` над неком инстанцом класе `ManualTower` и посматрајте како се мењају њено интерно стање и визуелни приказ.

Co-funded by the Erasmus+ Programme of the European Union

Задатак 8.4. Увођење методе `processUserControl()`

Када корисник кликне мишем на одређену кулу, потребно је преиначити ко њоме управља. При томе, ако корисник преузима управљање кулом, онда она треба да се ротира како би се усмерила према положају курсора миша. Повежите имплементирану методу са методом `act()`. Тестирајте своје решење.

[Commit*:* `6ec1f489576019a6493490f9e97797920b923869`**]**

Задатак 8.5. Идентификовање проблема и предлог решења

Идентификујте шта је проблематично са тренутном имплементацијом управљања кулама. Како би се ови проблеми могли решити?

Задатак 8.6. Увођење статичког атрибута

Додајте приватни атрибут `controlledInstance` (типа `ManualTower`) у класу `ManualTower`, који ће референцирати инстанцу куле којом, у датом тренутку, управља корисник и иницијализујте га са вредношћу `null`. Овај атрибут мора бити заједнички свим инстанцама класе `ManualTower`, тако да мора бити означен кључном речи `static`. Анализирајте интерно стање класе? Шта је додато?

[Commit*:* `c4739460bed583d2126de066acc6b1149d022990`**]**

Задатак 8.7. Увођење статичке методе

Имплементирајте методу `changeControlledInstance(ManualTower)` путем које корисник може да преузме контролу над неком кулом. Методу имплементирајте као методу класе `ManualTower` (тј. означите је кључном речи `static`). Параметар методе треба да буде инстанца куле којом корисник жели да управља. У телу методе, најпре треба проверити да ли је ова инстанца већ изабрана. Ако јесте, ништа не би требало да се промени, али ако је методи прослеђена нека друга инстанца, потребно је осигурати да ће доћи до промене.

[Commit*:* `9dc6d8dd4dcbbd71edb8009c1a72403dea1a0ee0`**]**

Задатак 8.8. Довршавање игре

Тестирајте решење, директним (тј. ручним) позивањем методе `changeControlledInstance(ManualTower)` да бисте потврдили да она обезбеђује тражено понашање. На ком месту у кôду би ова метода требало да буде позвана?

[Commit*:* `c052bbb6aa4c7e690d4d8cf55d3831028fa2b9e3`**]**

Co-funded by the Erasmus+ Programme of the European Union

8.2. Наставни сценарији

Припремљена су **два** наставна сценарија за тематску целину „Енкапсулација и скривање информација“.

8.2.1. СЦЕНАРИО: **Увод у енкапсулацију *Greenfoot* окружењу**

Табела 22. Увод у енкапсулацију Greenfoot окружењу

Назив	Увод у енкапсулацију *Greenfoot* окружењу
Образовни циљеви и исходи учења	Циљ овог сценарија је да ученицима приближи концепте енкапсулације и скривања информација кроз даљи развој игре `TowerDefense`.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **125** минута. 1. Увод (5 минута) 2. Задатак **8.1.** (20 минута) 3. Задаци **8.2.** и **8.3.** (30 минута) 4. Дискусија (35 минута) 5. Тумачење кôда (25 минута) 6. Задатак **8.4.** (10 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Кроз овај наставни сценарио ученици ће се упознати са концептима енкапсулације и скривања информација у контексту објектно-оријентисаног програмирања, а из перспективе развоја игара у *Greenfoot* окружењу. Лекција почиње концизним уводом од 5 минута, у оквиру којег наставник даје преглед циљева лекције. Ученици у тимовима крећу са развојем класе `ManualTower`, коју изводе из класе `Tower` (**Задатак 8.1.**). Током ове активности, која траје 20 минута, ученици треба да се усредсреде на дефинисање два конструктора, који морају бити усклађени са конструкторима натклассе, како би се осигурала исправна иницијализација. Потребно је и да ученици имплементирају методу `act()` која ће позивати `act()` методу своје натклассе. Након што ученици дефинишу класу, следи полусатни блок током којег ће се ученици упознати са практичном применом енкапсулације и скривања информација за управљањем стањем објекта. Наставник најпре уводи логички атрибут `isManuallyControlled`, који иницијализује са вредношћу `false`.

Co-funded by the Erasmus+ Programme of the European Union

	Ученици затим имплементирају методу `changeControl(boolean)` путем које се мења вредност атрибута `isManuallyControlled` и, у складу са њим, иконица дате куле (**Задатак 8.2.**). Кроз ову активност се демонстрира енкапсулирање и скривање информација, јер се путем метода контролише приступ стању објекта. Сваки ученик би затим требало да директно (тј. ручно) покрене методу `changeControl(boolean)` над инстанцама класе `ManualTower` и обрати пажњу на то како се мењају интерна стања и визуелни прикази кула (**Задатак 8.3.**). Следећи блок од 35 минута је посвећен дискусији која има за циљ да ученици разумеју да се суштина овог сценарија огледа у имплементацији приватне методе `processUserControl()` чији је задатак да детектује када се мишем кликнуло на инстанцу одређене куле. Наредних 25 минута разматра се начин имплементације ове методе. Када корисник кликне на одређену кулу, метода мења стање куле и ажурира њену оријентацију на основу позиције миша, при чему се користи енкапсулација за сакривање сложеније логика управљања кулом. Током последњих десет мунута ученици треба да имплементирају дату методу и повежу је са методом `act()` (**Задатак 8.4.**), али и да тестирају интеракцију са окружењем игре како би потврдили да метода обезбеђује тражено понашање, утврђујући на тај начин знање о томе како приватне методе омогућавају да се контролише промена стања објекта.
Вредновање	Активности у оквиру ове лекције ће омогућити наставницима да дају ученицима повратне информације на основу формативног вредновања њиховог учешћа у спроведеним дискусијама, али и њиховог рада у окружењу обрнуте учионице (енгл. *flipped classroom*), као и рада у тиму. Међусобно вредновање ће се спровести *online*, као део домаћег задатка. Кроз ову активност ученици ће се подсетити важних аспеката лекције, биће подстакнути да критички вреднују решења других ученика, имаће увид у добра и мање добра решења итд., што доприноси и повећању општег достигнућа постављених образовних циљева и исхода учења целе групе. Исходе учења и стечена знања ће ученици даље примењивати и током развоја тимског пројекта на којем раде. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

8.2.1.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Увод (5 минута)

Наставник треба да покрене претходно развијену игру и да са ученицима размотри понашање различитих протагониста. Наставник предлаже да се, у циљу лакшег одстрањивања непријатеља, уведе нова врста куле којом може управљати корисник. Корисник би требало да може да, у једном тренутку, управља само једном кулом. Када кликне на кулу, корисник би требало да преузме контролу над њом. Да би се указало на то којом кулом управља корисник, потребно је да кула, којом у датом тренутку управља корисник, има другачији изглед.

2. Задатак **8.1.** (20 минута)

Циљ: Припрема класе `ManualTower`.

Концепти за дискусију: наслеђивање, класе, конструктори.

Активности: Будући да ученици већ умеју да дефинишу изведене класе, замолите их да у тимовима реше следећи задатак:

> **Задатак 8.1. (Увођење класе `ManualTower`)**: Дефинишите класу `ManualTower` као изведену класу класе `Tower`. Потребно је да имплементирате конструкторе и да осигурате да ће конструктори наткласе бити позвани. Имплементирајте и методу `act()` тако да она позива `act()` методу своје наткласе. Креирајте неколико инстанци дате класе и додајте их у арену.
>
> **[Commit**: `63a02fa0c5080165cba8b467da08c4b65f31d0a8`**]**

Током овог дела лекције, ученици ће обновити градиво, применити га, и унапредити практично знање о наслеђивању.

3. Задаци **8.2.** и **8.3.** (30 минута)

Циљ: Увиђање потребе за приватним атрибутима.

Концепти за дискусију: методе, класе, атрибути, модификатори приступа.

Активности: Наставник треба да припреми иконице за кулу којом управља корисник. Потребно је објаснити ученицима како да користе методу `Actor.setImage(String)`, како би променили иконицу објекта програмским путем. Оставити ученицима пар минута да испробају ову методу.

Наставник треба да покрене дискусију о томе како се може утврдити да ли кулом управља корисник. Наставник треба да истакне то да је, поред тога што је потребно променити стање објекта, једнако важно ажурирати и његову иконицу. Неопходно је нагласити да, уколико корисник жели да промени стање неког `Tower` објекта и директно мења искључиво његов атрибут, иконица неће бити промењена. Ова дискусија треба да помогне ученицима да увиде потребу за променом, и вредности атрибута и пратеће иконице, путем методе, али и да разумеју зашто атрибути треба да буду приватни, а не јавни. Објасните ученицима да се овакав приступ назива енкапсулација, и да он омогућава да интерно стање објекта буде сакривено, а да се јавне методе користе како би се дато стање променило на контролисан начин.

Пружите ученицима прилику да, у тимовима, имплементирају логику саме методе.

> **Задатак 8.2. (Имплементација методе `changeControl(boolean)`)**: Уведите логички атрибут `isManuallyControlled` и иницијализујте га са вредношћу `false`. Имплементирајте методу `changeControl(boolean)` путем које се мења вредност атрибута `isManuallyControlled` и, у складу са њим, иконица дате куле.
>
> **[Commit**: `2257746b7dac5eaab7acc55d6493319230338f3a`**]**

Co-funded by the Erasmus+ Programme of the European Union

Наставник затим тражи од ученика да директно (тј. ручно) покрену своју методу и прате промену интерног стања објекта.

> **Задатак 8.3. (Праћење промена интерног стања објекта)**: Директно (тј. ручно) покрените методу `changeControl(boolean)` над неком инстанцом класе `ManualTower` и посматрајте како се мењају њено интерно стање и визуелни приказ.

4. Дискусија (35 минута)

Циљ: Разумевање потребе за енкапсулирањем логике унутар засебне методе.

Концепти за дискусију: методе, гранање.

Активности: Наставник треба да истакне то да се стање куле за сада може променити искључиво директним (тј. ручним) позивањем методе `changeControl(boolean)`. Циљ је да се обезбеди да се ова метода аутоматски позове након клика мишем на одређену кулу.

Међутим, потребно је указати на то да се миш може налазити и ван подручја света и да ће тада информације о тренутном статусу миша имати вредност `null`. Наставник подсећа ученике и на то да се метода `act()` непрестано извршава током трајања игре и да она заправо треба да провери да ли је корисник кликнуо на кулу и да само у том случају позове методу `changeControl(boolean)`. Потребно је нагласити да логика која се односи на интеракцију са корисником треба да буде енкапсулирана унутар засебне методе `processUserControl()`.

5. Тумачење кôда (25 минута)

Циљ: Увођење методе неопходне за решавање проблема.

Концепти за дискусију: методе, *Greenfoot* окружење.

Активности: Наставник са ученицима разматра на који начин би се могло променити стање инстанце, када се кликне на одређени објекат. Да би овакво понашање могло да се имплементира, потребно је увести и појаснити методу `GreenFoot.mouseClicked(Object)`. Поред тога, потребно је увести и класу `MouseInfo`, чија се инстанца може користити за добијање информација о текућој позицији миша.

6. Задатак **8.4.** (10 минута)

Циљ: Продубљивање разумевања приватних метода и енкапсулације кроз практичан задатак.

Концепти за дискусију: методе, модификатори приступа, класе.

Активности: Наставник распоређује ученике у тимове, и тражи од њих да имплементирају логику методе `processUserControl()`.

> **Задатак 8.4. (Увођење методе `processUserControl()`)**: Када корисник кликне мишем на одређену кулу, потребно је преиначити ко њоме управља. При томе, ако корисник преузима управљање кулом, онда она треба да се ротира како би се усмерила према положају курсора миша. Повежите имплементирану методу са методом `act()`. Тестирајте своје решење.

[Commit*:* `6ec1f489576019a6493490f9e97797920b923869`**]**

Наставник подсећа ученике на то да је могуће и да миш буде ван подручја света.

Када ученици заврше са радом, наставник бира тим (или два тима) који ће представити своје решење. Наставник дискутује са ученицима о предложеном решењу. До краја лекције сви ученици би требало да у потпуности разумеју начин имплементације ове методе.

Co-funded by the Erasmus+ Programme of the European Union

8.2.2. СЦЕНАРИО: **Даљи развој игре са фокусом на енкапсулацију и скривање информација**

Табела 23. Даљи развој игре са фокусом на енкапсулацију и скривање информација

Назив	Даљи развој игре са фокусом на енкапсулацију и скривање информација
Образовни циљеви и исходи учења	Циљ овог сценарија је продубљивање разумевања концепата енкапсулације и скривања информација кроз даљи развој игре `TowerDefense`.
Циљна група	Ученици средњих школа који учествују у *OOP4Fun* курсу и који желе да савладају основе објектно-оријентисаног програмирања.
Временски план	Укупно време потребно за реализацију сценарија: **95** минута. 1. Сесија у „обрнутој“ учионици (30 минута) 2. Атрибути класе / Статички атрибути (5 минута) 3. Задатак **8.6.** (5 минута) 4. Методе класе / Статичке методе (10 минута) 5. Задатак **8.7.** (20 минута) 6. Задатак **8.8.** (15 минута) 7. Обнављање теоријских концепата (10 минута)
Материјали и ресурси	Приручник *OOP4Fun* пројекта. Ресурси *OOP4Fun* пројекта. Изворни кôд пројекта доступан на *Github/Gitlab* платформи. Ресурси на Интернету.
Опис	Лекција почиње 30-минутним блоком током којег ће ученици кроз међусобну дискусију покушати да идентификују проблеме везане за управљањем кулом од стране корисника, као што је немогућност поништавања избора претходно одабране куле. Ученици треба да се укључе у дискусију и да предложе решења за евидентирање статуса кула (**Задатак 8.5.**). Током следећих 5 минута, наставник уводи концепт атрибута класе, тј. статичких атрибута. Након тога ученици треба да посвете 5 минута прилагођавању класе `ManualTower` како би се увео механизам за поништавања избора претходно одабране куле (**Задатак 8.6.**). Током наредног десетоминутног блока наставник уводи концепт методе класе тј. статичке методе. Након чега, следе два блока (од 20 и 15 минута) током којих треба имплементирати статичку методу `changeControlledInstance(ManualTower)` која ће омогућити преузимање контроле над кулама на јединственом централном месту (**Задатак 8.7**) и довршити пројекат (**Задатак 8.8.**). Кроз ове задатке и демонстрацију тога како статичке методе могу управљати заједничким стањем, тј. стањем које деле све инстанце класе, ученици ће стећи дубље разумевање концепта енкапсулације и скривања информација. Лекција се завршава десетоминутним теоријским осврtом, односно рекапитулацијом обрађених концепата.

Co-funded by the Erasmus+ Programme of the European Union

	Кроз овај опсежан образовни приступ ученици овладавају концептима енкапсулације и скривања информација, увиђају њихов значај и применљивост у реалним апликацијама, развијајући при томе вештине решавања проблема и сарадње.
Вредновање	Активности у оквиру ове лекције ће омогућити наставницима да дају ученицима повратне информације на основу формативног вредновања њиховог учешћа у спроведеним дискусијама, али и њиховог рада у окружењу обрнуте учионице (енгл. *flipped classroom*), као и рада у тиму. Исходе учења и стечена знања ће ученици даље примењивати и током развоја тимског пројекта на којем раде. На наставницима је да, у складу са могућностима и расположивим ресурсима, изаберу одговарајући начин вредновања рада ученика.
Дељење решења	За дељење решења са наставницима и другим ученицима користиће се уобичајена поставка *Github/Gitlab* платформе и образовне платформе (нпр. *Moodle*). Ученици могу да наставе дискусију на дату тему путем форума који ће им бити расположив у оквиру образовне платформе.

Co-funded by the Erasmus+ Programme of the European Union

8.2.2.1. СМЕРНИЦЕ ЗА ПРИПРЕМУ НАСТАВЕ

1. Сесија у „обрнутој“ учионици (30 минута)

Циљ: Увиђање потребе за иницијализацијом атрибута на једном месту.

Концепти за дискусију: атрибути класе / статички атрибути.

Активности: На почетку лекције наставник подстиче ученике да уоче проблем са тренутно имплементираним начином управљања кулама:

> **Задатак 8.5. (Идентификовање проблема и предлог решења)**: Идентификујте шта је проблематично са тренутном имплементацијом управљања кулама. Како би се ови проблеми могли решити?

За сада није могуће да се поништи избор већ одабране куле. Наставник подсећа ученике да би корисник требало да може да, у једном тренутку, управља само једном кулом, па сходно томе у сваком датом тренутку само једна кула може бити одабрана. Наставник подстиче ученике да размисле о томе како би овај проблем решили. Дискусија би требало да доведе до идеје да би било добро да постоји неки запис о тренутно одабраној инстанци, који би се ажурирао када се одабере нека друга инстанца, тј. једно централно место које би се иницијализовало само једанпут, а којем би се затим могло приступити и из других делова програма.

2. Атрибути класе / Статички атрибути (5 минута)

Циљ: Упознавање са концептом атрибута класе / статичких атрибута.

Концепти за дискусију: атрибути, атрибути класе / статички атрибути, класе.

Активност: Наставник објашњава шта су атрибути класе, тј. статички атрибути: променљиве које припадају самој класи, а не инстанцама те класе. Наставник повезује овај концепт са претходном дискусијом, где би постојање заједничког атрибута могло да реши проблем.

3. Задатак **8.6.** (5 минута)

Циљ: Практична примена статичких атрибута и `null` вредности.

Концепти за дискусију: атрибути, статички атрибути, класе, модификатори приступа.

Активност: Наставник ученицима задаје задатак:

> **Задатак 8.6. (Увођење статичког атрибута)**: Додајте приватни атрибут `controlledInstance` (типа `ManualTower`) у класу `ManualTower`, који ће референцирати инстанцу куле којом, у датом тренутку, управља корисник и иницијализујте га са вредношћу `null`. Овај атрибут мора бити заједнички свим инстанцама класе `ManualTower`, тако да мора бити означен кључном речи `static`. Анализирајте интерно стање класе? Шта је додато?
>
> **[Commit**: c4739460bed583d2126de066acc6b1149d022990**]**

Како би се током игре пратило која је кула тренутно одабрана, потребно је додати приватни статички атрибут `controlledInstance` у класу `ManualTower` и иницијализовати га са вредношћу `null`. Статички атрибути су заједнички за целу класу, а не односе се на појединачне објекте те класе. Дакле, дефинисање статичке променљиве ће омогућити да се одреди да ли је нека кула одабрана и ако јесте, која кула, референцирањем назива класе, без потребе приступања одређеној инстанци. Наставник треба да нагласи да постоји само један једини `controlledInstance` атрибут у целој игри. На самом почетку, дати `controlledInstance` атрибут треба да буде иницијализован са вредношћу `null`, јер није изабрана ниједна кула.

Током анализе интерног стања класе, наставник разјашњава разлику између статичких и нестатичких атрибута. Наставник разговара са ученицима о предностима коришћења статичких атрибута у имплементацији игара. Наставник такође треба да спомене статичке методе и да са ученицима продискутује када коришћење статичких метода може бити од користи.

Co-funded by the Erasmus+ Programme of the European Union

4. Методе класе / Статичке методе (10 минута)

Циљ: Упознавање са концептом метода класе / статичких метода.

Концепти за дискусију: методе класе / статичке методе, класе, објекти.

Активност: Наставник уводи концепт метода класе / статичких метода које могу да раде над подацима који су дефинисани на нивоу класе. Наставник дискутује са ученицима о потреби за методом као што је нпр. `changeControlledInstance`, која би требало да омогући ажурирање информације о томе којом кулом, у датом тренутку, управља корисник. Наставник наглашава да се овакве методе заправо могу позвати без потребе за постојањем инстанци дате класе.

5. Задатак **8.7.** (20 минута)

Циљ: Практична примена статичких метода.

Концепти за дискусију: методе, статичке методе класе, статички атрибути.

Активност: Наставник излаже задатак:

> **Задатак 8.7. (Увођење статичке методе)**: Имплементирајте методу `changeControlledInstance(ManualTower)` путем које корисник може да преузме контролу над неком кулом. Методу имплементирајте као методу класе `ManualTower` (тј. означите је кључном речи `static`). Параметар методе треба да буде инстанца куле којом корисник жели да управља. У телу методе, најпре треба проверити да ли је ова инстанца већ изабрана. Ако јесте, ништа не би требало да се промени, али ако је методи прослеђена нека друга инстанца, потребно је осигурати да ће доћи до промене.

Опис решења:

> Наставник уводи методу `changeControlledInstance(ManualTower controlledInstance)`. У телу методе, најпре треба проверити да ли се прослеђена вредност разликује од вредности која ће запамћена у статичком атрибуту `controlledInstance`. Ако се разликује, потребно је ажурирати атрибут `controlledInstance`, тј. референцу тренутно одабране инстанце. Наставник директно (тј. ручно) позива методу и указује ученицима на то да се иконице кула не мењају. Наставник наглашава да само ажурирање референце неће довести и до промене куле којом се управља, већ да ово понашање мора да се додатно имплементира. Дакле, потребно је додати програмски кôд путем којег ће се најпре ослободити текућа кула, а затим након ажурирања референце, кориснику препустити управљање новом кулом. Сходно томе, потребно је најпре позвати методу `changeControl(boolean)` са одговарајућим вредностима параметара. Најпре треба позвати ову методу над тренутно запамћеном инстанцом, како би се њен атрибут `isManuallyControlled` поставио на вредност `false`. Затим је потребно ажурирати статички атрибут `controlledInstance` класе `ManualTower` прослеђеном вредношћу. Коначно, потребно је поново позвати методу `changeControl(boolean)`, овог пута над новозапамћеном инстанцом, са вредношћу `true`. Наставник треба да нагласи да је могуће је да у одређеним тренуцима корисник не управља ниједном кулом, тако да је неопходно обрадити потенцијалне `null` референце (које се могу појавити ако корисник тренутно не управља ниједном кулом, и/или ако не жели да управља ниједном кулом, тј. када се методи проследи вредност `null`).

[Commit*:* `9dc6d8dd4dcbbd71edb8009c1a72403dea1a0ee0`**]**

Co-funded by the Erasmus+ Programme of the European Union

6. Задатак **8.8.** (15 минута)

Циљ: Практична примена статичких метода.

Концепти за дискусију: методе, статичке методе, статички атрибути.

Активност: Наставник излаже задатак:

> **Задатак 8.8. (Довршавање игре):** Тестирајте решење, директним (тј. ручним) позивањем методе `changeControlledInstance(ManualTower)` да бисте потврдили да она обезбеђује тражено понашање. На ком месту у кôду би ова метода требало да буде позвана?

Опис решења:

> Наставник треба да продискутује са ученицима, на ком месту у кôду би ова метода требало да буде позвана. Методу треба позвати унутар `act()` методе класе `Arena`, као и унутар методе `processUserControl()`.
>
> На крају, потребно је прогласити методу `ManualTower.changeControl(Boolean)` приватном. Потребно је размотрити какав ће утицај то имати на интерна стања инстанци класе `ManualTower`.

[Commit*:* `c052bbb6aa4c7e690d4d8cf55d3831028fa2b9e3`**]**

7. Обнављање теоријских концепата (10 минута)

Наставник даје резиме лекције, наглашавајући значај статичких атрибута и метода. Наставник треба да подстакне ученике да наставе да истражују тако што ће применити научене концепте у сопственим пројектима.

Co-funded by the Erasmus+ Programme of the European Union

ЛИТЕРАТУРА

Conole, G., Cross, S., Brasher, A., Weller, M., Nixon, S., & Clark, P. (2008). *A learning design methodology to foster and support creativity in design*. Proceedings of the 6th International Conference on Networked Learning, Greece, p. 46-53.

Hernández-Leo, D., Asensio-Pérez, J. I., Derntl, M., Pozzi, F., Chacón, J., Prieto, L. P., & Persico, D. (2018). *An integrated environment for learning design*. Frontiers in ICT, 5 (9): p. 1-19.

Sharpe, R., Beetham, H., & De Freitas, S. (2010). *Rethinking learning for a digital age: How learners are shaping their own experiences*. Routledge.

Kölling, M. (2016). *Introduction to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations 2nd Ed.* Pearson Education.

Clark, N. (2017). *Java: Programming Basics for Absolute Beginners*. CreateSpace Independent Publishing Platform.

Clark, N. (2018). *Java: A Detailed Approach to Practical Coding (Step-By-Step Java) 2nd Ed*. CreateSpace Independent Publishing Platform.

Kennedy, S., & van Putten, M. (2023). *Learn Java with Projects: A concise practical guide to learning everything a Java professional really needs to know*. Packt Publishing.

Liskov, B. & Wing, J. (1994). *A behavioral notion of subtyping*. ACM Transactions on Programming Languages and Systems. 16 (6): p. 1811–1841. [https://dl.acm.org/doi/pdf/10.1145/197320.197383]

Mohan, P. (2013). *Fundamentals of object-oriented programming*. CreateSpace Independent Publishing Platform.

Samoylov, N. (2022). *Learn Java 17 Programming - Second Edition: Learn the fundamentals of Java Programming with this updated guide with the latest features*. Packt Publishing.

https://www.greenfoot.org

https://docs.oracle.com/javase/specs/

Oracle Corporation. (1999). *Code Conventions for the Java Programming Language: 9. Naming Conventions*. https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html

Oracle Corporation. (2004). *How to Write Doc Comments for the Javadoc Tool*. https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html

ИНФОРМАЦИЈЕ О ПРОЈЕКТУ

Назив пројекта
Object Oriented Programming for Fun

Акроним пројекта
OOP4FUN

Број уговора
2021-1-SK01-KA220-SCH-00027903

Координатор пројекта
Žilinska univerzita v Žiline (Словачка Република)

Партнери на пројекту
Sveučilište u Zagrebu (Република Хрватска)
Srednja škola Ivanec (Република Хрватска)
Univerzita Pardubice (Чешка Република)
Gymnázium Pardubice (Чешка Република)
Obchodna akademia Povazska Bystrica (Словачка Република)
Hochschule fuer Technik und Wirtschaft Dresden (Савезна Република Немачка)
Gymnasium Dresden-Plauen (Савезна Република Немачка)
Универзитет у Београду (Република Србија)
Гимназија Ивањица (Република Србија)

Додатне информације о пројекту су доступне на: **https://www.oop4fun.eu/**