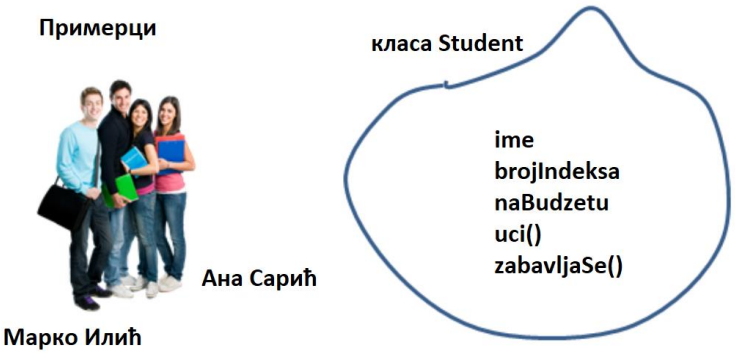
2. Објектно оријентисано програмирање

Пример 1. Слика 1 илуструје концепте објекта и класе, као и однос међу њима.□



Слика 1. Илустрација објекта и класе

Пример 2. Треба дефинисати класу `Poslediplomac` која је поткласа класе `Student` и која додатно садржи: логичко поље `zaposlen`, целобројно `brojIspita` и још два метода, један за постављање броја испита, а други за враћање његове вредности. Поткласа се генерише помоћу кључне речи `extends`.□

Пример 3. Описати објектима цртеж који на себи има два нацртана круга. Дакле, потребно је дефинисати класе `Krug` и `Crtez`. Природно је да `Krug` садржи учаурене податке који га описују, на пример центар круга и полупречник. `Crtez` садржи, као атрибуте, два круга. Објекат класе `Crtez` не треба да има директан приступ подацима који описују круг, већ да им приступа или управља њима искључиво позивањем метода над објектима класе `Krug`, на пример, позивом метода `transliraj()`. `Crtez` има могућност транслирања свих кругова за дати вектор помераја.□

Пример 4. Написати Јава програм који одређује НЗД и НЗС за три ненегативна цела броја. Рачунање НЗД и НЗС треба да буде учаурено унутар објекта.□

3. Неке програмске парадигме

Решење 1. Користећи императивну парадигму**,** програм ће бити реализован као монолитна целина. НЗД за три броја се одређује тако што се прво одреди НЗД за прва два броја, а коначни резултат се добија тако што се одреди НЗД за претходно одређени НЗД прва два броја и за трећи број.

Решење 2. Претходно постављени задатак се може решити применом структурног програмирања. НЗД се одређује коришћењем Еуклидовог алгоритма, описаног у претходном решењу. Рачунање НЗД за два и за три броја је издвојено у посебне методе, чиме је постигнуто да програмски кôд буде прегледнији, читљивији, а, такође, и лакши за одржавање и надоградњу.

Решење 3. Постављени задатак може се решити применом модуларне парадигме. НЗД се одређује по истом алгоритму као у решењу 1. Рачунање НЗД за два броја реализује се преко метода `nzd2()`, који се налази у модулу у датотеци `ModulNzd.java`. У истом модулу се налази и метод за рачунање НЗД за три броја, под називом `nzd3()`.

Решење 4. Користиће се рекурзивна верзија Еуклидовог алгоритма. Правила из логичког програмирања се симулирају преко метода. Ако, краће, са `m` и `n` означимо дате ненегативне целе бројеве онда преко Јава метода на следећи начин записујемо одговарајућа логичка правила закључивања:

$$n = 0 \quad \Rightarrow \quad nzd(m,n) = m$$

$$n > 0 \quad \Rightarrow \quad nzd(m,n) = nzd(n, m \bmod n)$$

Решење 5. Претходно постављени задатак биће решен применом функционалног стила програмирања. Користиће се рекурзивна дефиниција НЗД за два ненегативна цела броја:

$$nzd(m,n) = m, \qquad n = 0$$

$$nzd(m,n) = nzd(n, m \bmod n), \quad n > 0$$

Решење 6. У чисто објектно оријентисаним језицима оперише се само са објектима и поставља се питање да ли је могуће реализовати решење постављеног задатка у Јави, али оперишући само са објектима? Одговор је потврдан и то је урађено помоћу програма у датотеци ObjektnoNzd.java.

5. Језици и опис конструкција језика Јава

Пример 1. Наредни пример приказује шта су блокови у Јави и каква је веза видљивости локалних променљивих и блокова.□

Пример 2. Написати програм за израчунавање вредности функције:

$$sgn(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$$

при чему је x случајно генерисан цео број из интервала (-10, 10).□

Пример 3. Написати програм за израчунавање минимума два цела броја. Бројеви се учитавају преко стандардног улаза.☐

Пример 4. Написати програм за уређење три реална броја из интервала [0,1] у неопадајући поредак.☐

Пример 5. Написати програм који за унету годину и редни број месеца у години исписује број дана у том месецу.☐

Пример 6. Исписати првих 10 природних бројева применом наредбе `while`

Пример 7. Написати програм за израчунавање $S = \sum_{i=1}^{n} \frac{1}{i^3}$.☐

Пример 8. Написати програм који генерише и исписује случајне бројеве све док се не добије број већи или једнак 0.9. ☐

Пример 9. Написати програм за израчунавање дужине (броја цифара) унетог целог броја типа `long`.☐

Пример 10. Написати програм за уношење низа речи (једна реч у једном реду) са стандардног улаза све док се не препозна реч „КРАЈ“. Затим одштампати укупан број речи као и број појављивања речи: „програмирање“, „математика“ „физика“.☐

Пример 11. Написати програм који у десет редова на екрану исписује текст „Ана воли програмирање“ применом `for` наредбе.☐

Пример 12. Написати програм који израчунава факторијел унетог броја применом циклуса `for`.☐

Пример 13. Написати програм за израчунавање суме првих n природних бројева користећи `for` наредбу за бројање уназад.☐

Пример 14. Реализовати бесконачни `for` циклус.☐

Пример 15. Написати програм којим се израчунава сваки наредни степен променљиве (чија почетна вредност 0.9) све док не постане мањи или једнак 0.1. Овде наредба `for` практично замењује наредбу `while` будући да се изрази за иницијализацију и за итерацију изостављају.☐

Пример 16. Написати програм који за аргумент из интервала [0, 1] са кораком 0.1 рачуна и приказује вредност корена броја.☐

Пример 17. Написати програм који исписује индексе елемената на споредној дијагонали квадратне матрице. Димензија матрице је задата природним бројем.☐

Пример 18. Помоћу бесконачног `while` циклуса и `break` наредбе, написати програм који сумира све унете бројеве док се не унесе први негативан број. Након уноса негативног броја извршавање програма се прекида.☐

Пример 19. Помоћу бесконачног `while` циклуса, `continue` и `break` наредби написати програм који сумира све позитивне бројеве док се не унесе број 0, након чега се извршавање програма прекида.☐

Пример 20. Написати програм који проверава да ли се неки задати текст налази као под-текст (подниска) другог задатог текста.☐

6. Коришћење класа и објеката испоручених уз JDK

Пример 1. Написати Јава програм који приказује употребу форматираних и неформатираних исписа различитих типова података на стандардном излазу.□

Пример 2. Написати Јава програм који мери време извршавања метода за сабирање бројева од 1 до n. Тестирати програм за различите вредности n, као и за различите методе за мерење времена. Упоредити измерене вредности у милисекундама.□

Пример 3. Написати Јава програм који користи једну референцну променљиву за прављење великог броја нових ниски на хипу. Ово за последицу има велики број нереферисаних објеката током рада програма. Упоредити количину доступне и слободне меморије на хипу у варијантама:

1. када скупљач сам одређује кад ће се активирати и

када му корисник повремено сугерише скупљање помоћу `System.gc()` метода.□

Пример 4. Написати Јава програм који реализује и тестира метод за исписивање карактеристика монитора, при чему су аргументи метода ширина и висина. Потребно је израчунати број тачака (пиксела) и сврстати монитор у категорију стандардни или широки на основу односа ширине и висине: монитор је широк уколико је ширина два или више пута већа од висине. Такође је потребно на одговарајући начин реаговати на некоректне уносе.□

Пример 5. Написати Јава програм који задату реченицу окреће на нивоу речи. Дакле, слова унутар појединачних речи треба да остану у истом редоследу, али редослед речи унутар реченице треба да постане обрнут. Претпоставити да су једини типови сепаратора (унутар реченице) карактери за празно место. Такође, прилагодити почетно велико слово унутар новоформиране реченице.□

Пример 6. Написати Јава програм који формира ниску добијену надовезивањем задате ниске одређени број пута. За реализацију користити само класу `String`.□

Пример 7. Написати Јава програм који формира ниску добијену надовезивањем задате ниске одређени број пута. За реализацију користити и класу `StringBuilder`.□

Пример 8. Написати и тестирати Јава метод у којем се израчунава квадрат прослеђеног реалног броја. Проверити понашање у случају да је аргумент примитивног типа, као и случају да је реч о омотач типу.□

Пример 9. Написати Јава програм који из текста извлачи информације о просечној години производње, снази и запремини мотора аутомобила. Текст је сачињен од реченица, где свака има следећу структуру: „[Марка] [Модел] [Година производње] [Снага] KS [Запремина] cm3.“.□

Пример 10. Написати Јава програм који рачуна просек реалних бројева унетих са стандардног улаза. На почетку уноса корисник унапред најављује колико бројева жели да унесе, након чега следи њихов унос у сваком наредном новом реду.□

Пример 11. Написати Јава програм који врши нумеричку интеграцију неке од следећих математичких функција: `sqrt()`, `log()` или `exp()` методом Монте Карло на задатом интервалу [a, b], a>=1. Код метода Монте Карло кључну улогу има број узорака N>0, који се учитава са стандардног улаза. Што је већи број узорака, израчунавање је прецизније. Идеја Монте Карло интеграције је заснована на хоризонталном и вертикалном ограничавању графика подинтегралне функције, на задатом интервалу, на пример, правоугаоником. Након тога се генеришу координате дводимензионалних тачака, на случајан начин, тако да припадају изабраном правоугаонику. На основу односа броја тачака чије ординате су мање од вредности функције за случајно изабране апсцисе („упадају“ испод графика функције) и укупног броја генерисаних тачака, апроксимира се вредност интеграла. Ово важи у случају разматраних функција `sqrt()`, `log()` и `exp()`, али не мора важити у општем случају, тј. за произвољну функцију.□

Пример 12. Написати Јава програм који очекује од корисника да унесе месец и годину, а потом на излазу приказује одабрани месец у виду табеле где су колоне дани у недељи, а редови недеље у оквиру месеца. При том је у ћелији табеле уписан редни број дана у месецу.☐

Пример 13. Написати Јава програм који приказује сва датум-времена, до минутне прецизности, за које важи да је сума редног броја дана у месецу, месеца у години, сата (24 сатни формат) и минута једнака броју који корисник уноси на улазу. При том узети само датум-времена из текуће године. Користити формат исписа по жељи.☐

Пример 14. Написати Јава програм који рачуна просек вредности које враћа метод `nextGaussian()`. На улазу корисник уписује колико бројева жели да се генерише помоћу метода `nextGaussian()`, а на излазу се исписује просек. Семе генератора подесити на неку фиксирану вредност.☐

Пример 15. Написати Јава програм који генерише 10 предлога за „добру шифру“, где се под добром шифром подразумева она која се састоји од малих или великих слова алфабета и цифара. При том, слова у просеку треба да чине око 70% карактера, а цифре око 30%. Величина добре шифре је између 8 и 16 карактера. Семе генератора подесити на неку фиксирану вредност.☐

7. Низови у Јави

Пример 1. Написати Јава програм који одређује НЗД низа позитивних целих бројева.☐

Пример 2. Написати Јава програм који одређује НЗД низа позитивних целих бројева. У реализацији користити колекцијску `for` наредбу.☐

Пример 3. Написати Јава програм који омогућује да се оформи низ реалних бројева са задатим бројем елемената (који уноси корисник са стандардног улаза), тако да сваки члан низа добије исту (задату) вредност и да се применом колекцијског `for` циклуса прикажу вредности свих чланова низа.☐

Пример 4. Написати Јава програм који пребројава елементе у низу целих бројева из интервала 1-10 и приказује их у бројчаном облику и у облику хистограма.☐

Пример 5. Написати Јава програм који реализује стек помоћу низа. Потом га применити за учитавање реалних бројева и њихов приказ у обрнутом редоследу.☐

Пример 6. Написати Јава програм за сабирање целих бројева који су задати као аргументи командне линије. Претпоставити да су аргументи

дати коректно, тј. Да ће бити могућа конверзија задатих ниски у целе бројеве.□

Пример 7. Написати Јава програм који сабира, одузима, множи или дели два броја. Аргументи командне линије су, редом, тип операције (+, –, *, /), а потом и два реална броја.□

Пример 8. Написати Јава програм који омогућује да се коришћењем дводимензионалног низа реализују основне операције над матрицама (сабирање, одузимање, множење и израчунавање детерминанте).□

Пример 9. Написати Јава програм који задати текст трансформише у тродимензионални низ карактера такав да се на позицији (i, j, k) налази k-ти карактер j-те речи из i-те реченице.□

Пример 10. Написати Јава програм који задати текст трансформише у сортирани низ речи записаних малим словима. Након тога се у петљи уносе речи са стандардног улаза и за сваку унету реч се, применом бинарне претраге, проверава да ли постоји у претходно уређеном низу речи. Извршавање се прекида уносом речи „КРАЈ“.□

Пример 11. Написати Јава програм који приказује примену алгоритма бинарне претраге над низом целих бројева. Пре примене овог алгоритма потребно је сортирати низ.□

Пример 12. Написати Јава програм који користи метод са аргументом променљиве дужине. Метод нема повратну вредност и смисао му је да испише све прослеђене аргументе на стандардни излаз.□

8. Класе, пакети, поља, методи и објекти у Јави

Пример 1. Декларисати референце (инстанцне променљиве) за новоуведену класу `Zaposleni`, као и за предефинисану класу `String`.☐

Пример 2. Нека је на располагању класа `Zaposleni`. Направити две инстанце класе `Zaposleni` и три ниске.☐

Пример 3. Написати Јава програм који прави објекат класе `Object` и потом га исписује на стандардном излазу.☐

Пример 4. Написати Јава програм који декларише и инстанцира објекте класе `Kutija`. Ова класа се описује са три атрибута: висина, ширина и дубина.☐

Пример 5. Написати Јава програм који тестира оператор `==` применом на примитивним подацима целобројног типа и на инстанцним променљивама класе `Kutija`.☐

Пример 6. Написати Јава програм који генерише 10 псеудослучајних целих бројева. Користити наредбу `import` за увоз класе `Random`. Семе генератора псеудослучајних бројева поставити на вредност 12345.☐

Пример 7. Дефинисати класу `Ucenik` са подацима о имену и презимену и разреду. Направити објекте ове класе и обезбедити приступ пољима.☐

Пример 8. Класу `Ucenik`, из претходног примера, проширити статичким пољем које у себи садржи број направљених инстанци класе `Ucenik`. Дакле, почетна вредност статичког поља је 0, а након сваког прављења нове инстанце класе `Ucenik`, потребно је повећати је за 1. На крају програма исписати вредност овог статичког поља.☐

Пример 9. Раније дефинисану класу `Ucenik` проширити методом за приказ информација о ученику као и методом којим се, на основу

разреда, одређује да ли је ученик у старијој смени (ученици од 5. до 8. разреда су у старијој смени).□

Пример 10. Дефинисати класу `Pravougaonik` описану координатама горњег левог и доњег десног темена. Претпоставка је да су странице правоугаоника паралелне са осама координатног система па је ова репрезентација јединствена. Користити целобројне координате имајући у виду матрицу пиксела на екрану – координатни почетак (0, 0) се налази у доњем левом углу екрана. Омогућити дефинисање правоугаоника на два начина: 1) прослеђивањем два темена (била која, метод треба да процени да ли су исправна); 2) прослеђивањем горњег левог темена, ширине и висине правоугаоника. У случају да ширина или висина правоугаоника нису позитивне, пријављује се грешка.□

Пример 11. Дефинисати класу `Srednjoskolac` као поткласу класе `Ucenik`. Од додатних атрибута, средњошколац поседује и врсту средње школе. Поред тога, поседује и метод за узимање (дохватање) вредности атрибута врсте средње школе.□

Пример 12. Написати Јава програм који тестира оператор `instanceof` над објектима класе `Object` и класе `Kutija` у разним комбинацијама.□

Пример 13. Раније уведену класу `Tacka`, која представља тачку у дводимензионалном простору целобројних координата, прилагодити тако да се у класи превазилази подразумевано понашање метода `toString()`, `equals()` и `hashCode()`. Потом дефинисати класу `Duz` помоћу две тачке, и за њу, такође, редефинисати поменуте методе.□

Пример 14. Класа `Ljubimac` има потклассе `Macka` и `Pas`. У свакој од ових класа налази се метод `trci()`. Класа `TestLjubimac` служи за тестирање ових класа. У њој се праве три инстанце класе `Ljubimac` и на сваку инстанцу се примени метод `trci()`. Потребно је коректно приказати начин трчања сваке животиње, без обзира што су све декларисане као `Ljubimac`. Овде је метод `trci()` полиморфан.□

Пример 15. Прилагодити класу `Tacka` тако да садржи два конструктора, један без аргумената и други са вредностима за `x` и `y` координате. Потом преоптеретити и конструкторе за класу `Duz`. Потребно је да постоји

празан конструктор, конструктор који прихвата две тачке и конструктор који прихвата четири цела броја (редом координате тачака).☐

Пример 16. Нека је цртеж у векторској графици представљен низом полигона (претпоставити да их неће бити више од 10). Полигон је описан низом тачака (претпоставити да их неће бити више од 10). Тачка је, као и у претходним примерима, представљена помоћу две целобројне координате. Реализовати копирајући конструктор за цртеж. Да би се ово постигло, потребно је направити копирајуће конструкторе за сваку од наведених класа. Свака класа треба сама да брине о формирању сопствене копије.☐

Пример 17. Дефинисати потклasу класе `Tacka` под називом `OznacenaTacka`. Ова класа има додатно поље `oznaka` типа `String`. Приликом дефиниције конструктора класе `OznacenaTacka` искористити раније дефиниције конструктора класе `Tacka`.☐

Пример 18. Приказати примену модификатора `public` при раду са пољима, методима и класама. Написати јавну класу `A` која има јавно поље, метод и конструктор. Након тога, приказати приступ истима из тела класе `A`, затим из тела класе `B`, која се налази у истом пакету где и класа `A`, и на крају из тела класе `C`, која се не налази у истом пакету где и класе `A` и `B`.☐

Пример 19. На исти начин, као у примеру 18, приказати употребу модификатора `package`.☐

Пример 20. Слично као у примеру 18, приказати употребу модификатора `protected`. Класа `C` сада наслеђује класу `A`.☐

Пример 21. Слично као у примеру 18 приказати употребу модификатора `private`.☐

Пример 22. Дефинисати класу `Krug` која је описана центром и полупречником. Додатно, ова класа садржи и поља `obim` и `povrsina`. Потребно је дефинисати конструктор који прихвата као аргументе центар и полупречник. Такође, редефинисати метод `toString()`. Реализовати две варијанте ове класе, једну која не користи заштиту поља и другу која користи. Тестирати и упоредити ове две варијанте.☐

Пример 23. Реализовати саморастући низ ниски и приказати његову употребу. Класа `SamorastuciNizNiski` треба да учаурује низ ниски и да омогући следеће операције за рад над истим: 1) иницијализацију низа на подразумевану величину од 8 елемената; 2) метод којим се поставља вредност на задату позицију; 3) метод којим се приступа вредности на задатој позицији; 4) метод који враћа број елемената низа; 5) метод који враћа капацитет учауреног низа (алоцирани број елемената низа). Сматрати да је број елемената низа једнак позицији последњег постављеног елемента у низу увећаној за један. Елементи који су између, уколико им није експлицитно постављена вредност, представљени су `null` референцама.

Пример 24. Реализовати једноструко повезану листу ниски. Најпре дефинисати класу која описује појединачни елемент/чвор листе. Ова класа учаурује ниску (садржај) и референцу ка следећем чвору. Редефинисати метод за испис чвора. Након тога, дефинисати класу која описује једноструко повезану листу која учаурује информације о почетку и крају листе. Реализовати три конструктора: 1) конструктор који не прихвата аргумент; 2) конструктор који прихвата једну ниску и 3) конструктор који прихвата низ ниски. Од метода омогућити следеће: 1) додавање на крај; 2) додавање на почетак; 3) уклањање са краја; 4) уклањање са почетка; 5) дужина (број чворова) листе. У одвојеној класи тестирати све поменуте методе.□

9. Напредни рад са класама и објектима

Пример 1. Написати Јава класу која представља геометријски објекат у равни, тако да иста омогући једноставну надградњу и проширивање. Сваки такав геометријски објекат у перспективи треба да обезбеди могућност за проверу конвексности, проверу ограничености и мерење обима и површине.□

Пример 2. Написати Јава класе за представљање геометријских објеката у равни: тачке, дужи и праве. У класи за тачку, поред апстрактних геометријских метода, треба да се реализује и метод за рачунање удаљености до задате тачке. У класи за дуж треба да се реализују методи за рачунање дужине и проверу да ли дуж садржи задату тачку. У класи за праву треба да су реализују методи за проверу да ли права садржи задату тачку, као и методи за проверу да ли су две прослеђене тачке са истих или различитих страна праве.□

Пример 3. Написати Јава класе за геометријске објекте у равни: троугао, четвороугао и круг. У свакој од ових класа треба да се реализује и метод за испитивање да ли објекат класе садржи задату тачку.□

Пример 4. Написати Јава програм којим се, коришћењем претходно развијених класа, праве различити геометријски објекти у равни (тачке, дужи, праве, кругови, троуглови и четвороуглови), који се затим третирају на униформан начин – прикажу се њихови подаци, њихова конвексност и ограниченост, као и њихова површина.□

Пример 5. Написати Јава интерфејс `Radoznao` који ће садржати два метода.□

Пример 6. Написати Јава класу која имплементира интерфејс из претходног примера и програм који користи имплементиране методе, и то преко интерфејса и директно преко класе.□

Пример 7. Нека нам је на располагању већ развијен интерфејс `Radoznao`. Потребно је написати интерфејс `Razuman` са методима `razmotriCinjenice()` и `definisiHipotezu()`, Јава класу `Naucnik` са

ниска-атрибутом `ime`, која имплементира интерфејсе `Radoznao` и `Razuman`, као и програм који користи имплементиране методе преко променљивих типа интерфејса.☐

Пример 8. Написати интерфејс `Eksperimentator` који проширује интерфејсе `Radoznao` и `Razuman` и који садржи метод `realizujeEksperimente()`. Написати Јава класу `Istrazivac` изведену из класе `Naucnik` која имплементира интерфејс `Eksperimentator`, са ниска-атрибутом `probojUOblasti` и са превазиђеним методом за дефинисање хипотезе. Написати Јава програм који позива имплементиране методе преко променљивих типа интерфејса.☐

Пример 9. Написати Јава програм за рад са геометријским објектима у равни, тако да програм допусти једноставну надоградњу и проширивање. Треба обезебедити проверу конвексности, проверу ограничености, проверу припадности тачке објекту и рачунање обима и површине. Треба реализовати класе за геометријске објекте који представљају тачку, дуж, праву, троугао, четвороугао и круг. На крају треба направити разноврсне геометријске објекте, приказати их, срачунати њихов укупни обим и укупну површину, а потом за тачку чије се координате учитавају са стандардног улаза одредити који је од геометријских објеката садрже, а који не.☐

Пример 10. Надоградити претходни пример дефинисањем Јава метода `sadrziSeU()` у класи `Tacka` који за дату тачку проверава да ли се садржи у геометријском објекту чије се име прослеђује као аргумент метода. Написати одвојену класу која садржи `main()` метод у којем ће се, коришћењем метода `sadrziSeU()`, за тачку чије су координате учитавају са стандардног улаза, одредити сви геометријски објекти којима та тачка не припада.☐

Пример 11. Дефинисати интерфејс `StekNiski` који садржи методе карактеристичне за стек структуру података: 1) додавање ниске на врх стека (енг. `push()`); 2) уклањање елемента са врха стека (енг. `pop()`) и 3) враћање тренутне величине стека (броја елемената). Реализовати две имплементације овог интерфејса, једну која користи раније уведену динамичку структуру `SamorastuciNizNiski` (пример 23 из секције 8.9.1), и другу која користи раније уведену динамичку структуру `PovezanaListaNiski` (пример 24 из секције 8.9.2). Приказати

употребу било које од ове две имплементације „сакривањем“ имплементације иза интерфејса `StekNiski` – корисник може да одлучи коју ће имплементацију користити.□

Пример 12. Написати Јава програм за сортирање низа целих бројева, низа реалних бројева у покретном зарезу и низа ниски.□

Пример 13. Написати Јава програм за сортирање низа тачака. Критеријум сортирања је удаљеност од координатног почетка – тачке ближе координатном почетку се налазе на мањим позицијама у сортираном низу у односу на оне које су даље од координатног почетка. Специјално, ако су растојања до координатног почетка за две тачке иста, она тачка са мањом ординатом y треба да буде пре у односу на ону са већом ординатом.□

Пример 14. Написати Јава програм за сортирање низа тачака по више критеријума (по локацији од ближих према даљим у односу на координатни почетак, по ознаци, као и по локацији од даљих према ближим).□

Пример 15. Написати Јава програм за сортирање низа целих бројева на два различита начина: 1) сортирање у неопадајућем поретку по вредности и 2) сортирање тако да прво буду сви парни бројеви сортирани у неопадајућем поретку, а потом непарни бројеви сортирани у неопадајућем поректу. Програм треба да сортира и низ ниски: 1) лексикографски и 2) тако да предност имају оне ниске које садрже више самогласника, а ако ниске имају исти број самогласника онда предност треба да има краћа ниска. Сортирање реализовати помоћу метода `sort()` класе `Arrays`.□

Пример 16. Написати Јава програм за клонирање примерака класе која представља запосленог и испитује понашање оригиналног и клонираног објекта.□

Пример 17. Написати Јава програм за клонирање инстанци класе која представља запосленог и садржи променљива поља. Испитати понашање оригиналног и клонираног објекта.□

Пример 18. Написати Јава програм за клонирање инстанци класе `Zaposleni` која садржи променљива поља и у којој се испитује понашање оригиналног и клонираног објекта.☐

Пример 19. Написати класу за рад са подацима о запосленима, где ће функционалност која подржава упис информација о запосленом, у базу података, да буде реализована у оквиру класе која представља запосленог.☐

Пример 20. Написати класе за рад са подацима о запосленима, где ће функционалност, која подржава упис информација о запосленом, бити издвојена у посебан део.☐

Пример 21. Написати Јава програм за одређивање површине геометријских објеката (кругова и правоугаоника).☐

Пример 22. Написати Јава програм за одређивање површине геометријских објеката (кругова и правоугаоника), водећи рачуна о потреби његовог проширења.☐

Пример 23. Реализовати класе за правоугаоник и за квадрат које обезбеђују рачунање њихових површина.☐

Пример 24. Развити интерфејсе и класе за софтверски систем који подржава рад ресторана и води рачуна о прихватању поруџбина (лично, телефонски, он-лајн) и о плаћању (лично, он-лајн).☐

Пример 25. Развити интерфејсе и класе за софтверски систем који подржава рад ресторана и води рачуна о прихватању поруџбина (лично, телефонски, он-лајн) и плаћању (лично, он-лајн), тако да буде уважен принцип раздвајања интерфејса.☐

Пример 26. Развити класе за клијент-сервер систем где клијент, приликом реализације датог метода, користи метод сервера.☐

Пример 27. Развити интерфејсе и класе за клијент-сервер систем, где клијент, приликом реализације датог метода, користи метод сервера, водећи рачуна о принципу инверзије зависности.□

Пример 28. Развити интерфејсе и класе за клијент-сервер систем, где клијент, приликом реализације датог метода, користи метод сервера, водећи рачуна о принципу инверзије зависности. Омогућити да се током рада клијента може мењати конкретан сервис који тај клијент користи.□

Пример 29. Написати класу са информацијама о запосленом и то: име, презиме, опис посла и плата. Поред тога, написати и класу за раднике запослене преко студентске задруге.□

10. Угнежђене класе

Пример 1. Написати Јава програм који рачуна плату радника на месечном нивоу. Плата је дефинисана као умножак броја радних сати и цене сата. Минималан број сати које радник мора да оствари је 160 (норма), а све преко тога се рачуна као прековремени рад. Прековремени сати су реткост и постоји засебан механизам рачунања прековременог додатка. За сваки наредни сат преко норме увећава се цена по сату за 2%. На пример, ако је радник радио 1 прековремени сат добиће 102% цене сата као додатак. Ако је радио два сата, добиће 102%+ 104.04%=206.04% итд. Максимална цена по сату је 2000 РСД, било да је реч о основној цени рада или прековременој.☐

Пример 2. У секцији 8.9.2, примеру 24, имплементирана је једноструко повезана листа ниски. Проблем са том имплементацијом је могућност формирања чворова листе и ван тела класе `PovezanaListaNiski`. Ово непожељно понашање може се онемогућити применом концепта унутрашње класе.☐

Пример 3. Написати Јава класу и метод за проверу да ли број мобилног телефона има исправан запис у мрежама мобилних оператера у Србији (то не гарантује и постојање броја). Поред тога, број је потребно довести до стандардног формата облика [позивни број]/[број]. Исправан запис подразумева да број телефона почиње неким од доступних позивних бројева мрежа: 060, 061, 062, 063, 064, 065, 066, 068, 069. Након тога се могу појављивати цифре, симболи ‘/’, ‘-’ и празан простор. Претпоставити (поједностављено) да је број исправан ако поред позивног броја има још 6 или 7 цифара. За потребе исправности и довођења у стандардни формат користити локалну унутрашњу класу.☐

Пример 4. Написати Јава програм за сортирање низа ниски у растућем поретку према суми ASCII вредности знакова који чине ниску. Ако две ниске имају исту суму ASCII вредности, као секундарни критеријум поређења користити стандардно растуће лексикографско уређење. За сортирање користити уграђени метод `Arrays.sort()`, уведен у секцији 7.6. Проследити методу одговарајући функционал за поређење.☐

11. Изузеци и тврдње

Пример 1. Написати Јава програм који покушава да приступи елементу низа ван дозвољених граница. Ухватити изузетак и обрадити га унутар `main()` метода.☐

Пример 2. Написати Јава програм који приказује рад нетачне рекурзивне имплементације метода за рачунање факторијела (нема изласка из рекурзије).☐

Пример 3. Написати Јава програм у којем се спречава појава реферисања на елемент чији индекс је ван граница индекса низа, тј. спречава се појава изузетка класе `ArrayIndexOutOfBoundsException`.☐

Пример 4. Написати Јава програм у којем се показује покушај конверзије `Integer` објекта у `Boolean` објекат.☐

Пример 5. Написати Јава програм који у петљи прихвата текст са стандардног улаза. У свакој итерацији петље покушати парсирање унетог текста у датум-формату „dd.MM.yyyy“. Корисник треба да уноси текст све док га не унесе у исправном формату – након тога прекинути извршавање петље, односно програма.☐

Пример 6. Написати Јава програм за решавање проблема из примера 5, при чему парсирање треба издвојити у засебан статички метод. С обзиром да овај метод неће имати контролу над главном петљом, у којој се уноси текст од стране корисника, пропагација изузетка у `main()` метод има више смисла него реаговање на изузетак унутар метода за парсирање.☐

Пример 7. Написати Јава програм који реализује сабирање одговарајућих елемената два дводимензионална низа. Елементи су одговарајући ако имају исте индексе. Уколико се сабирање односи на два низа који немају идентичне димензије, потребно је избацити нови тип изузетка.☐

Пример 8. Написати Јава програм који помоћу тврдње проверава да ли је број ненегативан.☐

Пример 9. Написати Јава програм који ради исто што и програм у примеру 8 са том разликом што помоћу тврдње треба да се прикаже и опис грешке.☐

Пример 10. Написати Јава програм који, помоћу тврдње, проверава да ли метод за сортирање низа заиста сортира низ (постуслов).☐

12. Набројиви (енумерисани) тип

Пример 1. Написати Јава програм који приказује дефинисање и употребу набројивог типа за дане у недељи.☐

Пример 2. Написати Јава програм за испис броја дана у одабраном месецу одабране године. Одабир месеца и године се остварује при покретању програма, навођењем као аргумената командне линије. Месец се описује називом записаним словима латиничног писма, док се година задаје као број.☐

Пример 3. Написати Јава програм који ради са набројивим типом за представљање планета Сунчевог система. Поред назива и редног броја планете, потребно је предвидети и додатне информације о маси и пречнику. На стандардни излаз потом исписати информације за сваку планету. Додатно, дефинисати метод који рачуна тежину тела на површини сваке планете. Као аргумент командне линије се задаје маса тела.☐

Пример 4. Написати Јава програм који приказује употребу набројивог типа за представљање основних аритметичких операција: сабирања, одузимања, множења и дељења.☐

Пример 5. Написати Јава програм у којем се реализује набројиви тип `DanUNedelji` употребом класе.☐

13. Генерички тип

Пример 1. Написати Јава програм за рад са класом која представља кутију. У кутију је могуће убацити произвољни објекат. Дефинисати методе за убацивање и извлачење објекта из кутије и тестирати њихов рад.☐

Пример 2. Написати Јава програм за рад са класом која представља кутију, попут оне у примеру 1. За разлику од примера 1, класу је потребно реализовати као генеричку.☐

Пример 3. Написати Јава програм за рад са генеричком класом која представља уређени пар вредности генеричких типова. Поред конструктора, који прихвата обе вредности, потребно је реализовати и методе за приступ првом и другом члану уређеног пара. Такође је потребно редефинисати метод `toString()`.☐

Пример 4. У примеру 11 из секције 9.2.5 дефинисан је интерфејс `StekNiski`, након чега су реализоване његове две различите имплементације: `StekNiskiPrekoNiza` и `StekNiskiPrekoListe`. Прилагодити те дефиниције тако да се може оперисати са произвољним типовима података, а не само са нискама. Након тога, приказати употребу обе имплементације додавањем елемената на стек и накнадним сабирањем свих елемената стека.☐

Пример 5. Написати Јава програм који реализује статички самостални генерички метод за линеарну претрагу генеричког низа и тестирати његове могућности.☐

Пример 6. Написати Јава програм који реализује статички самостални генерички метод за тражење минималног елемента у низу. Након тога, применити метод на низ објеката типа `String` и низ објеката раније уведеног генеричког типа уређени пар. Потребно је „дорадити“ класу за уређени пар тако да имплементира генерички интерфејс `Comparable`. Поређење се врши на основу прве координате, а потом на основу друге ако су прве координате једнаке.☐

Пример 7. Написати програм који реализује негенеричку варијанту генеричке класе `UredjeniParUporediv<...>`. Слично као у примеру 6, реализовати статички метод за тражење минималног елемента низа и применити га на низ уређених парова.□

Пример 8. Написати Јава програм који реализује генеричку потклasу претходно уведене класе `KutijaGenericka<T>`. Потклаcа треба да омогући прослеђивање боје кутије, третирајући је као аргумент конструктора. Након тога, тестирати имплицитну конверзију између инстанцних променљивих из смера потклаcе ка наткласи.□

14. Колекције и речници

Пример 1. Дефинисати генерички интерфејс за ред који чине методи за додавање елемента на крај реда, узимање са почетка и добијање информације о броју елемената реда. Након тога, два пута имплементирати уведени интерфејс за генеричке типове података: употребом кружног низа и повезане листе.☐

Пример 2. Приказати употребу итератора над произвољном генеричком колекцијом. Урадити исто и помоћу колекцијске наредбе `for`.☐

Пример 3. Приказати рад са класом `LinkedList`: додавање и уклањање елемената, приступ елементима на случајној позицији и набрајање свих елемената листе у оба смера.☐

Пример 4. Илустровати на примеру рад са класом `ArrayList` користећи операције за додавање и уклањање елемената, приступ елементима на задатој позицији и набрајање свих елемената листе у оба смера.☐

Пример 5. Упоредити брзине рада низовне и повезане листе. Конкретно, потребно је: 1) упоредити брзине додавања великог броја елемената у листе; 2) набрајање елемената тако формираних листи и 3) одредити брзину случајног приступа одређеном броју елемената тако формираних листи.☐

Пример 6. Реализовати генеричку класу за уређени пар уз редефинисање метода `equals()` и `hashCode()`. Након тога, приказати поређење уређених парова као и добијене хеш-кодове. Као основ за класу користити раније уведену генеричку класу `UredjeniPar` из поглавља 13, пример 3.☐

Пример 7. Демонстрирати употребу `HashSet` за уграђене типове `Double`, `String`, као и за раније уведене типове `UredjeniPar` и `UredjeniParUporediv`.☐

Пример 8. Приказати употребу `TreeSet` класе за уграђене типове `Double`, `String`, као и за раније уведени тип `UredjeniPar` уз употребу `Comparator` интерфејса.☐

Пример 9. Упоредити употребу класа `HashSet` и `LinkedHashSet`.☐

Пример 10. Приказати употребу класе `BitSet`.☐

Пример 11. Приказати употребу класе `EnumSet` над раније уведеним енумерисаним типом за представљање дана у недељи `DanUNedelji` (пример 1 у поглављу 12).☐

Пример 12. Приказати употребу класе `ArrayDeque` у FIFO и LIFO режиму.☐

Пример 13. Приказати употребу класе `PriorityQueue` у уређивању редоследа рада процеса. Процес је описан називом и приоритетом.☐

Пример 14. Приказати употребу класе `HashMap` у одржавању информација о особама. Особа се описује ЈМБГ-ом, именом, презименом и годином рођења. Потребно је омогућити претрагу скупа особа на основу ЈМБГ-а.☐

Пример 15. Пребројати појављивања речи у задатом тексту. Након тога исписати на конзоли првих 10 речи и број њихових појављивања у складу са лексикографским уређењем речи. За реализацију користити уграђену класу `TreeMap`. Игнорисати знаке интерпункције и величину слова.☐

Пример 16. Дефинисати и тестирати метод за сумирање листе бројева при чему елементи листе могу бити цели или реални бројеви.☐

Пример 17. Дефинисати класу `OsobaUporediva` као потклasу класе `Osoba` из примера 14. Ова класа треба да имплементира интерфејс `Comparable<OsobaUporediva>`. Поређење треба да буде засновано на вредности ЈМБГ. Такође, дефинисати класу `Student` као потклasу класе `OsobaUporediva`. Студент, поред тога што је особа, има и број индекса и просечну оцену, а начин поређења је исти као код особе. Испитати употребу уграђеног метода за сортирање `Collections.sort()` над листом студената. Да ли класа `Student` може да имплементира

интерфејс `Comparable<Student>`? Како се постиже да метод за сортирање ради над листом студената? Које је заглавље овог метода и која ограничења постоје на џокер типу који се овде користи?□

Пример 18. У примеру 4 (секција 13.2.1) је дефинисан генерички интерфејс за стек и две имплементације овог интерфејса: помоћу саморастућег низа и повезане листе. Сада је потребно реализовати исти интерфејс и класе, са том разликом да генерички интерфејс `Stek<T>` проширује (наслеђује) основни колекцијски интерфејс `Collection<T>`. Ово ће омогућити бољу спрегнутост између будућих имплементација интерфејса `Stek<T>` и великог броја механизама (услужних библиотека) које раде са колекцијама.

Да би се једноставније достигле функционалности које прописује `Collection<T>` интерфејс, класе које имплементирају `Stek<T>` могу да наследе класе `AbstractCollection<T>`.□

15. Улаз и излаз

Пример 1. Приказати употребу `FileInputStream` за читање података из датотеке чија је путања задата као аргумент командне линије. Прочитане бајтове исписати у формату карактера на конзоли. Тестирати читање из бинарне датотеке, на пример, слике у PNG формату, као и из текстуалне TXT датотеке у ASCII и UTF-8 кодним странама.□

Пример 2. Приказати употребу `DataOutputStream` за писање различитих примитивних типова на стандардни излаз, тј. конзолу. Такође, у одвојеној класи, приказати употребу `DataOutputStream` и `FileOutputStream` за писање различитих примитивних типова у бинарну датотеку. Потом ту датотеку отворити и прочитати употребом `FileInputStream` и `DataInputStream` (инверзни кораци).□

Пример 3. За дату ниску, применом `StringReader` класе, приказати само оне карактере дате ниске такве да се у N наредних карактера, након њих, не налази ознака за празан простор. Поред метода `read()`, користити и методе `mark()` и `reset()`.□

Пример 4. Реализовати програм који задати низ ниски уписује у датотеку чија је путања дата као аргумент командне линије. За испис у датотеку користити класу `PrintWriter`.□

Пример 5. Прочитати карактере из задате текстуалне датотеке у UTF-8 формату. У реализацији користити уланчавање класа `FileInputStream` (за приступ току бајтова датотеке), `InputStreamReader` (за конверзију бајтова у карактере) и `BufferedReader` (за побољшање перформанси).□

Пример 6. Корисник уноси путању датотеке или директоријума као аргумент командне линије. Програм треба да изврши рекурзивни обилазак система датотека по дубини, почев од те путање. При том је потребно исписивати пуну апсолутну путању свих датотека (не и директоријума).□

Пример 7. Из датотеке „ostalo/studenti.txt“ која има следећи садржај:

```
1009987567890 Марко Петровић 1987 23 9.33
2001967567890 Ана Ковачевић 1967 13 8.43
1009997567890 Марија Мирковић 1997 111 9.36
```

учитати студенте у листу објеката класе `Student`, који се описују редом ЈМБГ-ом, именом, презименом, годином рођења, бројем индекса и просечном оценом. Потом исписати елементе листе.

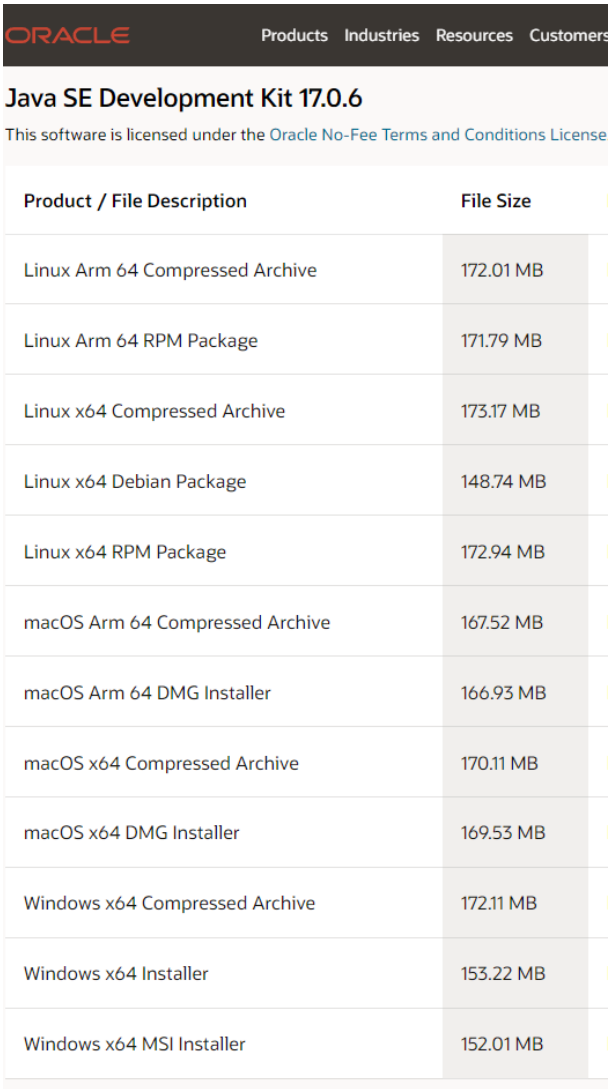
За реализацију програма може се користити раније дефинисана класа `Student` из примера 17 секције 14.6. За учитавање података из датотеке користити класу `Scanner` са подешеном кодном страном UTF-8. За испис на конзолу користити `PrintWriter` са подешеном кодном страном UTF-8.□

Додатак А – Инсталација Јаве и развојног окружења

Инсталација JDK

1. Први корак је преузимање одговарајуће JDK инсталације са Oracle веб сајта. Конкретно, за верзију Јаве 17 LTS адреса је:

https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html

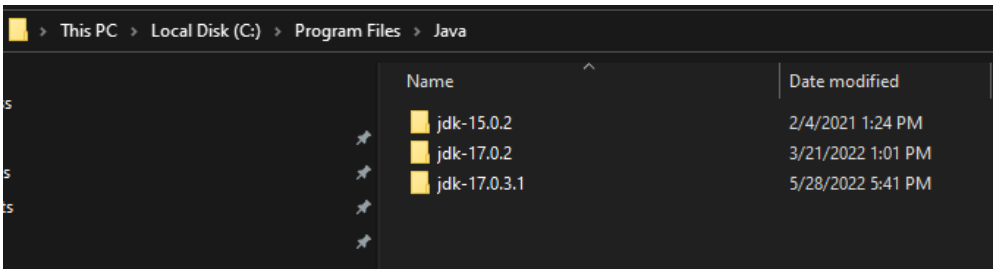


Windows 64-битни

2а. Преузети датотеку са називом „Windows x64 msi Installer“.

3а. Након тога, покренути преузету датотеку и довршити њену инсталацију.

4а. По успешној инсталацији требало би да је направљен поддиректоријум за одговарајућу инсталацију у оквиру директоријума „C://Program Files/Java“, на пример:



На слици се види да је могуће имати истовремено више инсталираних JDK. Која од њих ће се користити приликом развоја програма зависи од подешавања развојног окружења.

Linux 64-битни

2б. Преузети датотеку са називом „Linux x64 Compressed Archive“.

3б. Распаковати је у директоријум „/opt“ помоћу наредбе:

```
tar xvzf  jdk-17.0.3.1_linux-x64_bin.tar.gz -C /opt
```

(Верзија коју преузмете не мора нужно бити 17.0.3.1 па је у складу са тим потребно и ажурирати наредбу изнад.)

4б. Подесити JAVA_HOME променљиву окружења тако да показује на направљени директоријум, а потом извршни директоријум „bin“ из JAVA_HOME укључити у променљиву PATH:

```
export JAVA_HOME=/opt/jdk-17.0.3.1
export PATH=$PATH:$JAVA_HOME/bin
```

5б. Проверити да ли је Јава успешно инсталирана наредбом:

```
java -version
```

Ако јесте, требало би да се добије испис попут следећег:

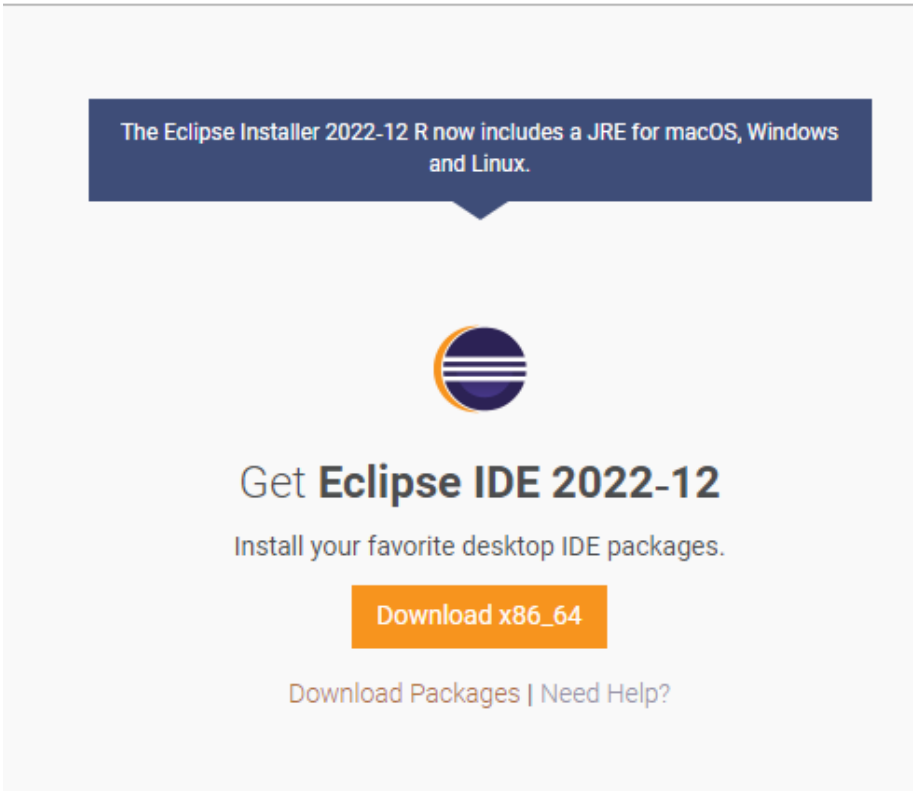
```
java 17.0.3.1 2022-05-02 LTS
Java(TM) SE Runtime Environment…
```

Инсталација развојног окружења Eclipse под Windows ОС

1. Посетити сајт Eclipse организације за преузимање програма, конкретно адресу:

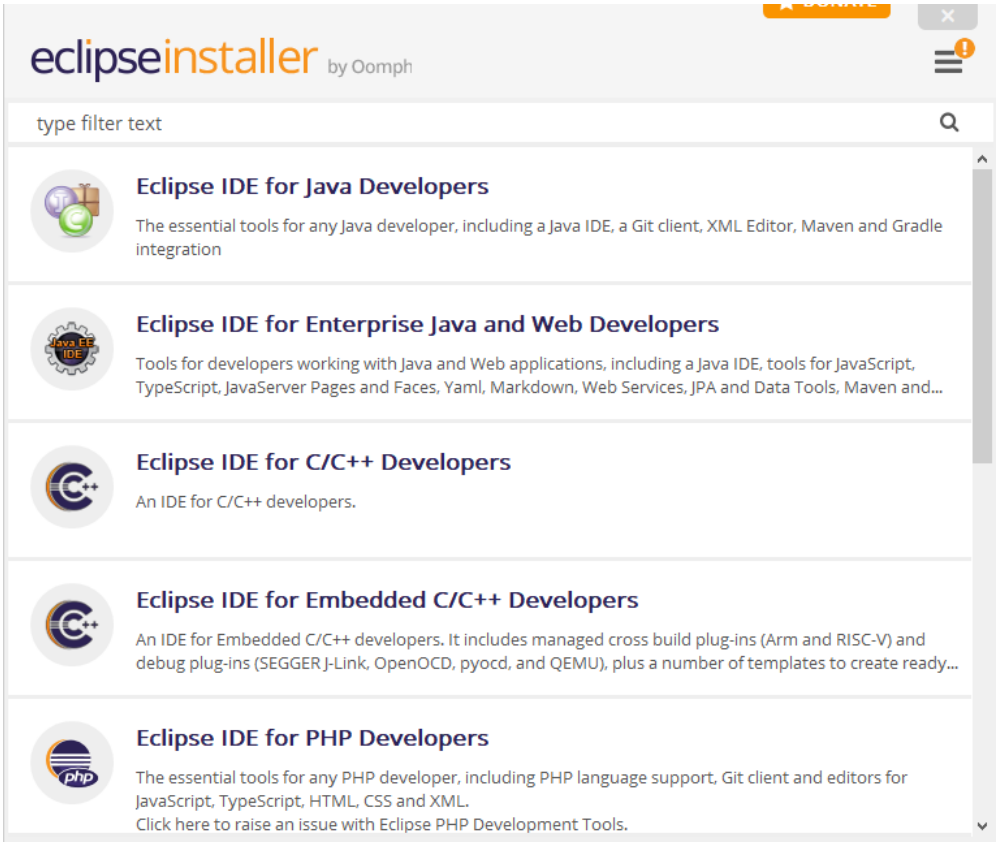
https://www.eclipse.org/downloads/

2. Кликнути на дугме „Download X86_64“.



3. Након тога покренути инсталацију.

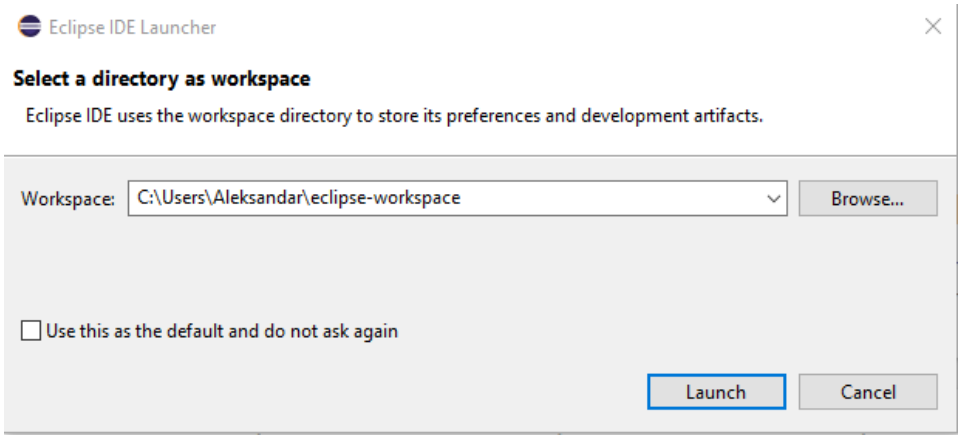
4. У оквиру првог инсталационог прозора одабрати опцију „Eclipse IDE for Java Developers“:



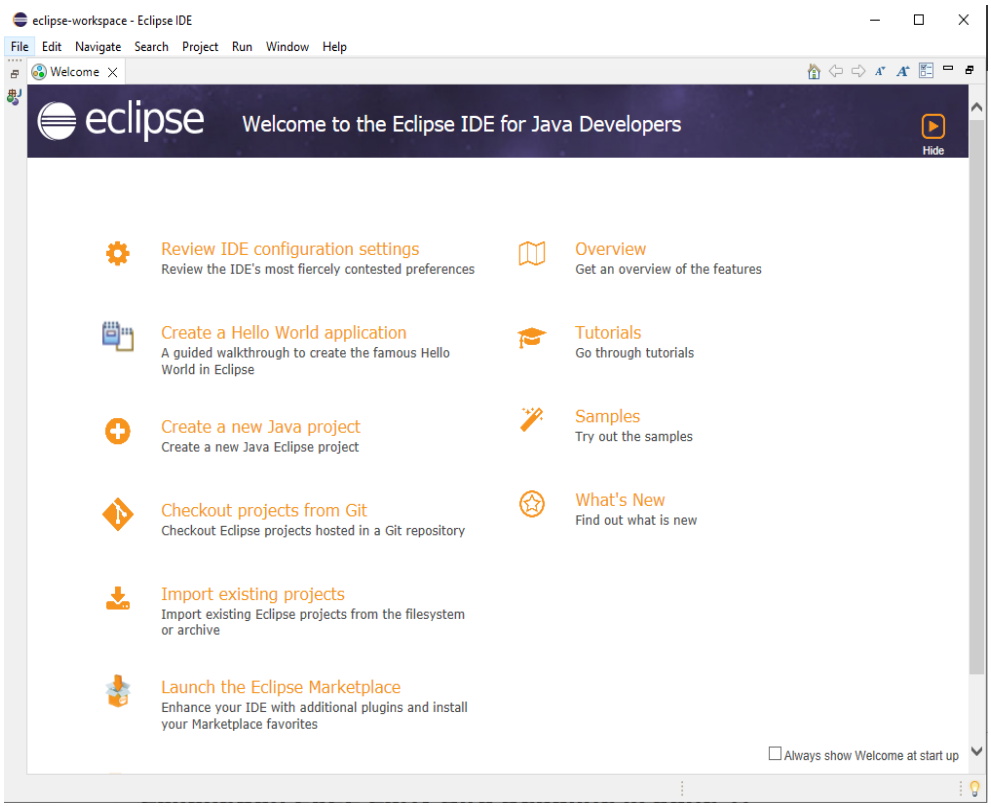
5. Следећи инсталациони прозор ће укључивати неколико питања. Најважније је питање где се налази претходно инсталирана Јава.

6. По успешној инсталацији могуће је покренути Eclipse развојно окружење кликом на дугме „LAUNCH“ или алтернативно кликом на иконицу на радној површини.

7. Приликом покретања Eclipse, биће постављено питање где корисник жели да позиционира директоријум са кодовима (енг. workspace). Није погрешно прихватити предложену локацију.



8. Покренуто Eclipse окружење изгледа овако:

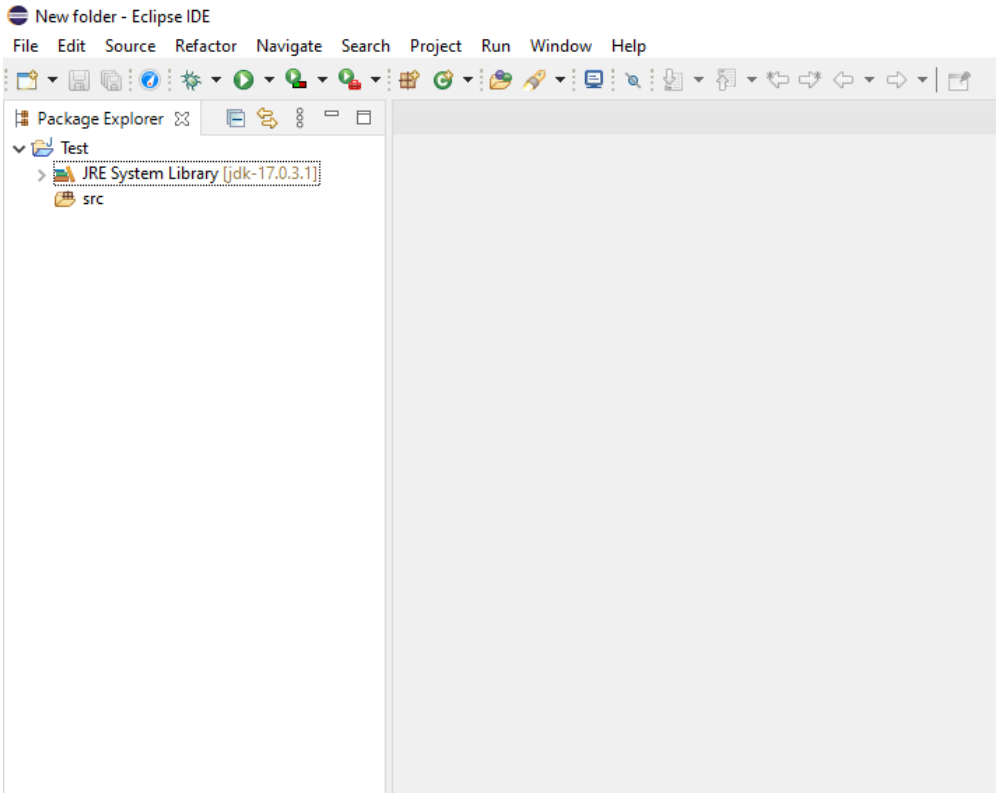


9. Угасити прозор добродошлице (енг. welcome).

10. Сада опцијом „File→New→Java Project“ и задавањем имена пројекта направити први пројекат.

11. У следећем прозору може се поставити питање у вези употребе система модула, уколико се не планира њихово коришћење, може се одабрати опција „Don't Create“.

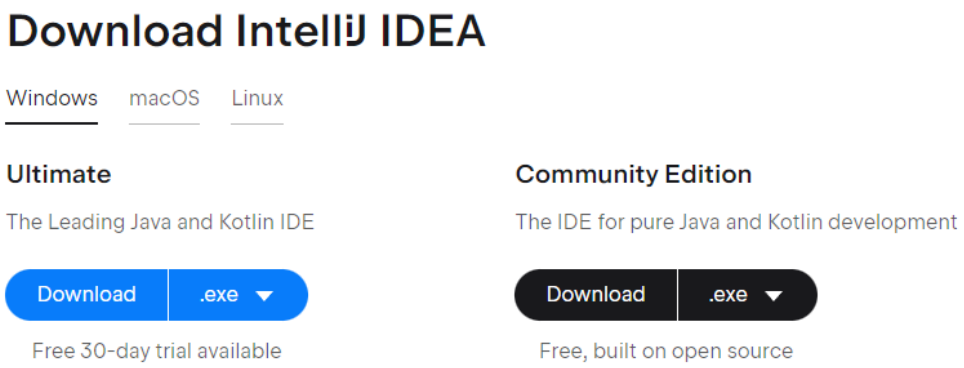
12. Тиме је завршена инсталација развојног окружења и покренут је пројекат:



Инсталација развојног окружења IntelliJ под Windows ОС

1. Посетити сајт компаније JetBrains на следећој веб локацији: https://www.jetbrains.com/idea/download/

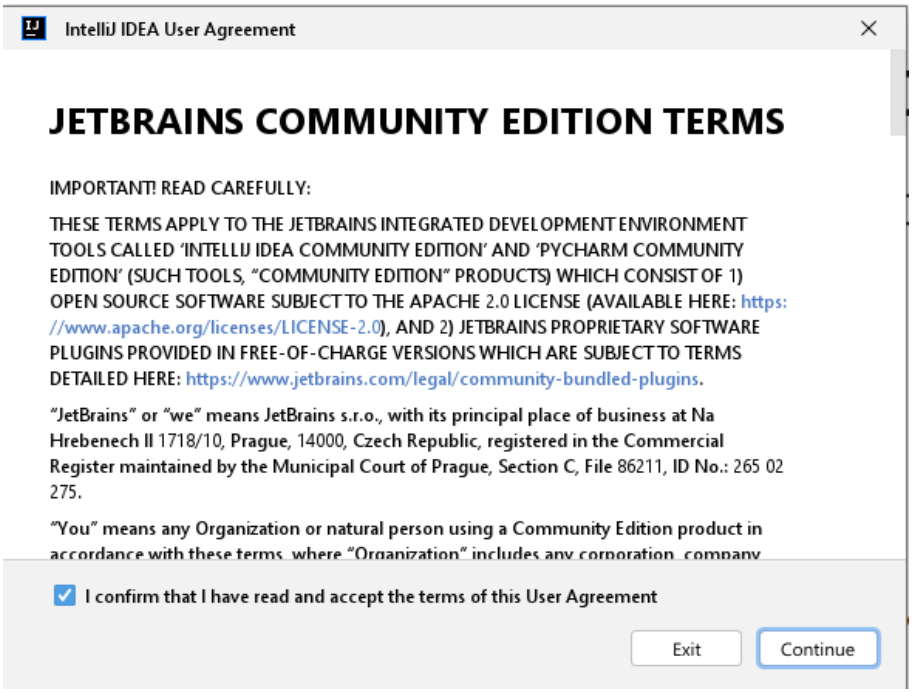
2. Преузети „Community“ верзију IntelliJ развојног окружења.



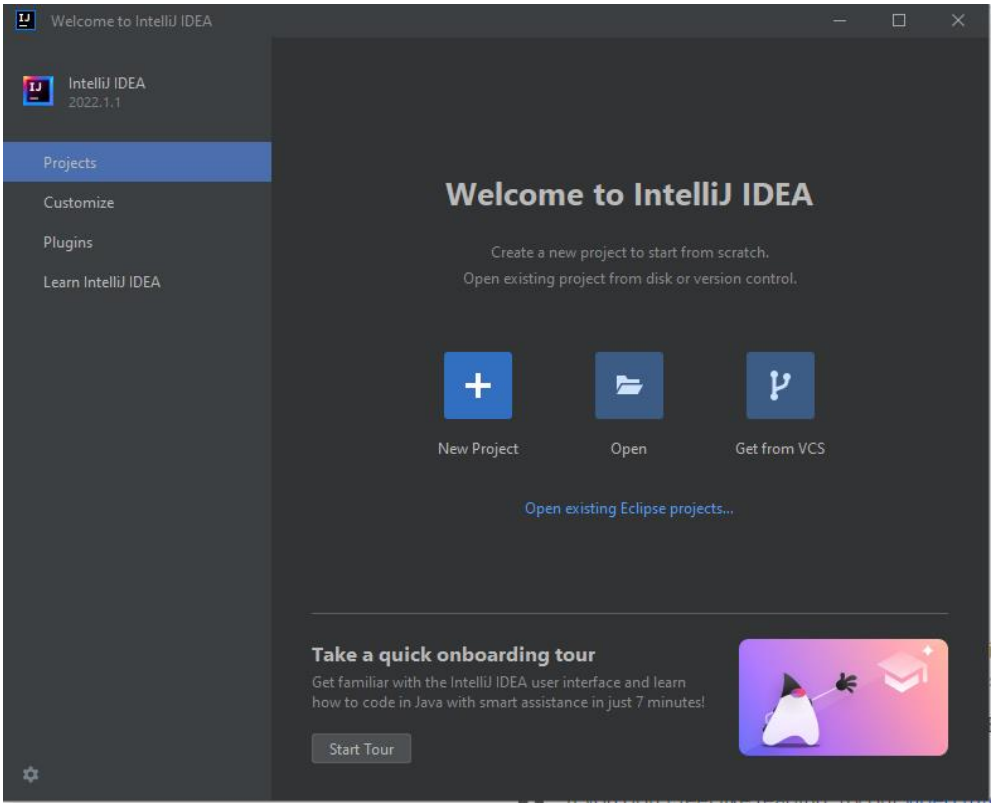
3. Након преузимања, покренути инсталациону датотеку и проћи кроз процес инсталације (не би требало да буде нејасних питања).

4. По завршетку инсталације могуће је одмах покренути развојно окружење, а могуће је и накнадно путем иконице на радној површини и слично.

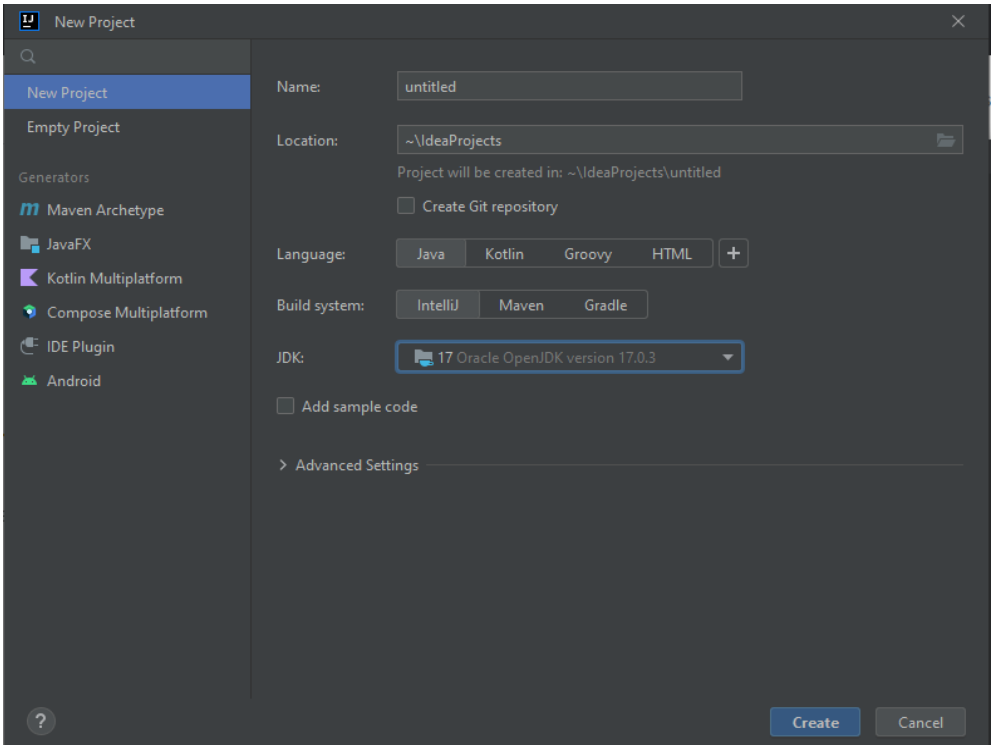
5. Појавиће се следеће питање приликом покретања:



6. Почетни екран након покретања изгледа овако:

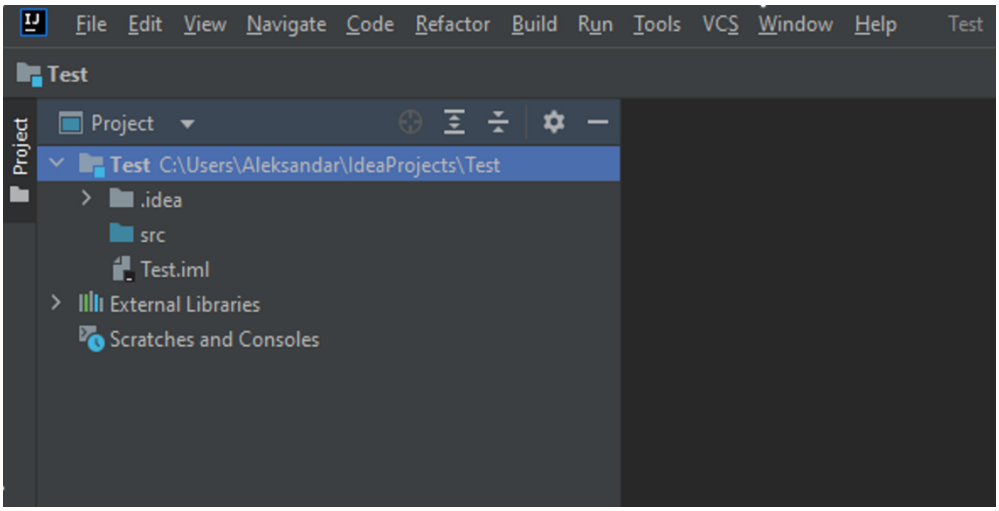


7. Кликнути на опцију „New Project“. Након тога ће се појавити следећи дијалог:



8. Унети назив пројекта и проверити да ли је JDK адекватно препознат од стране IntelliJ окружења. Потом кликнути на дугме „Create“.

9. Тиме је завршена инсталација развојног окружења и покренут је пројекат:



Додатак Б – Упутство за употребу GitHub репозиторијума и подешавање ћирилице

Сви примери коришћени у књизи налазе се на GitHub репозиторијуму: https://github.com/matf-oop-java/tom-1

Следи упутство за преузимање (клонирање) овог репозиторијума и његову интеграцију са претходно наведеним развојним окружењима Eclipse или IntelliJ.

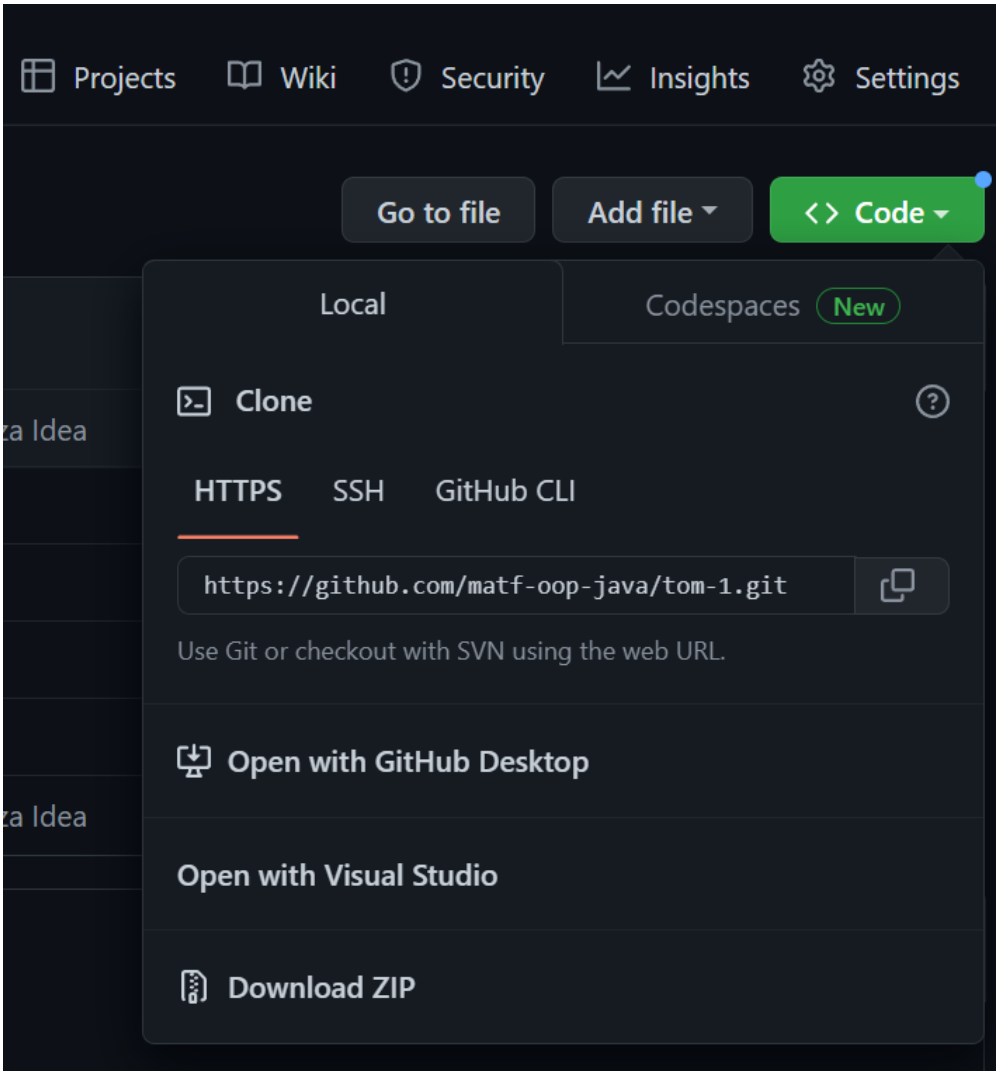
Преузимање (клонирање) GitHub репозиторијума

1. Посетити веб сајт: https://github.com/matf-oop-java/tom-1

2. Кодови и пратећи подаци се могу преузети на најмање два начина: у виду .zip архиве или применом одговарајћег Git клијента.

2a. За преузимање у виду .zip архиве, кликнути на дугме „Code“ и након тога одабрати опцију „Download ZIP“.

Након преузимања, архиву распаковати на погодној локацији.



2б. За преузимање путем Git клијента (клонирање) користити одговарајућег клијента, на пример, TortoiseGit који се може преузети са адресе: https://tortoisegit.org.
(Развојна окружења обично нуде и интегрисану подршку за Git, али то овде није објашњено.)

- Помоћу TortoiseGit алата позиционирати се у одговарајући директоријум где треба сместити датотеке.

- Потом, користећи десни клик мишем, било где у отвореном директоријуму, изабрати опцију „Git Clone“.

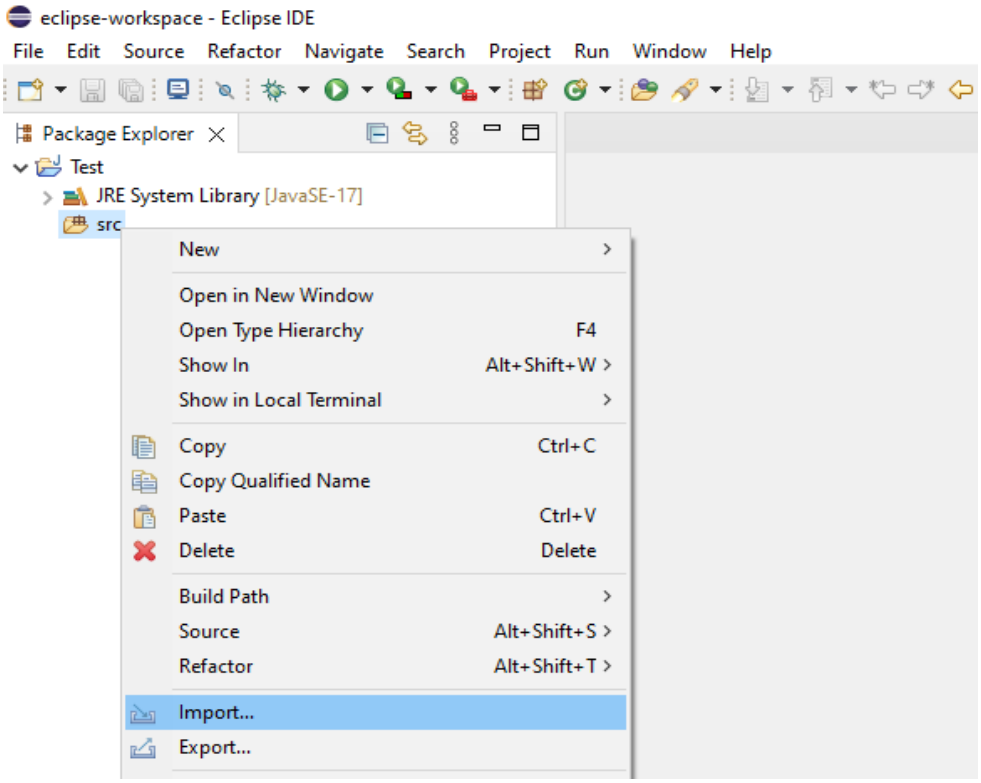
- Појавиће се следећи прозор у који треба унети Git адресу репозиторијума https://github.com/matf-oop-java/tom-1 и кликнути на дугме „OK“.

3. Преузети (или клонирани) директоријум GitHub пројекта би требало да садржи поддиректоријуме: „ostalo“ и „src“. У оквиру директоријума „src“ се налазе програмски кодови свих примера из књиге, док директоријум „ostalo“ садржи неке пропратне датотеке које се користе у неким од примера.

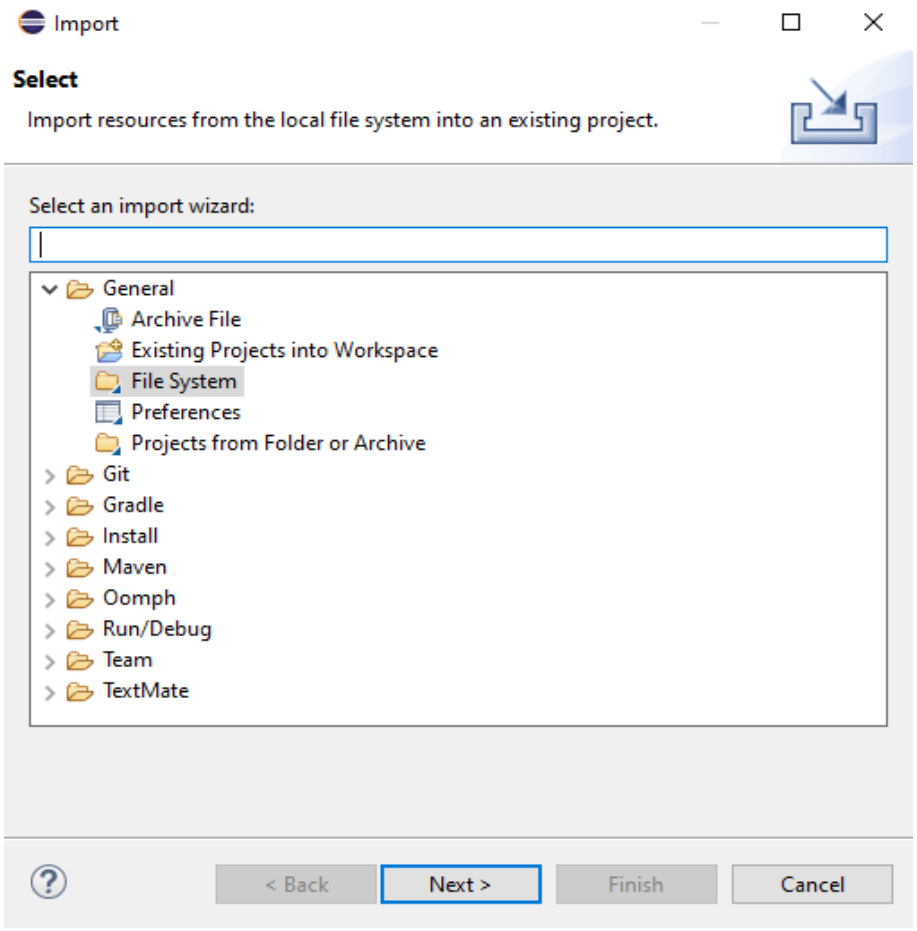
4. Након што је преузет и распакован или клониран материјал са GitHub-а, исти се може уградити у постојећи Eclipse или IntelliJ пројекат на начине описане у следећим секцијама.

Уграђивање GitHub материјала у постојећи Eclipse пројекат

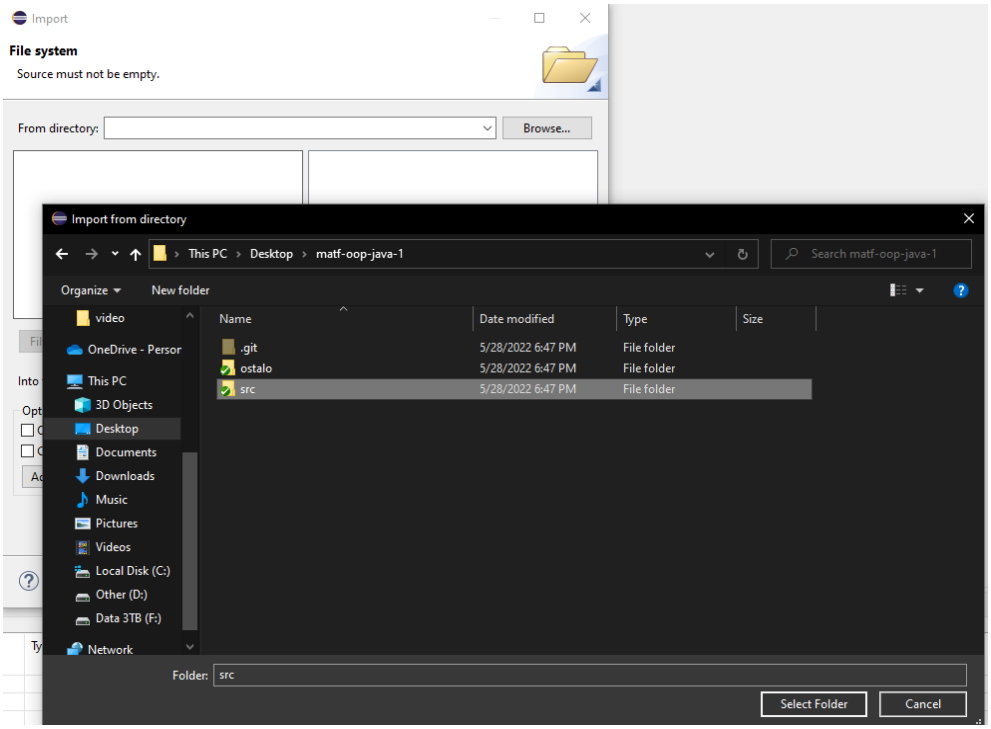
1. У оквиру раније направљеног Јава пројекта обележити директоријум „src“, потом кликнути десним дугметом миша и изабрати опцију „Import“.



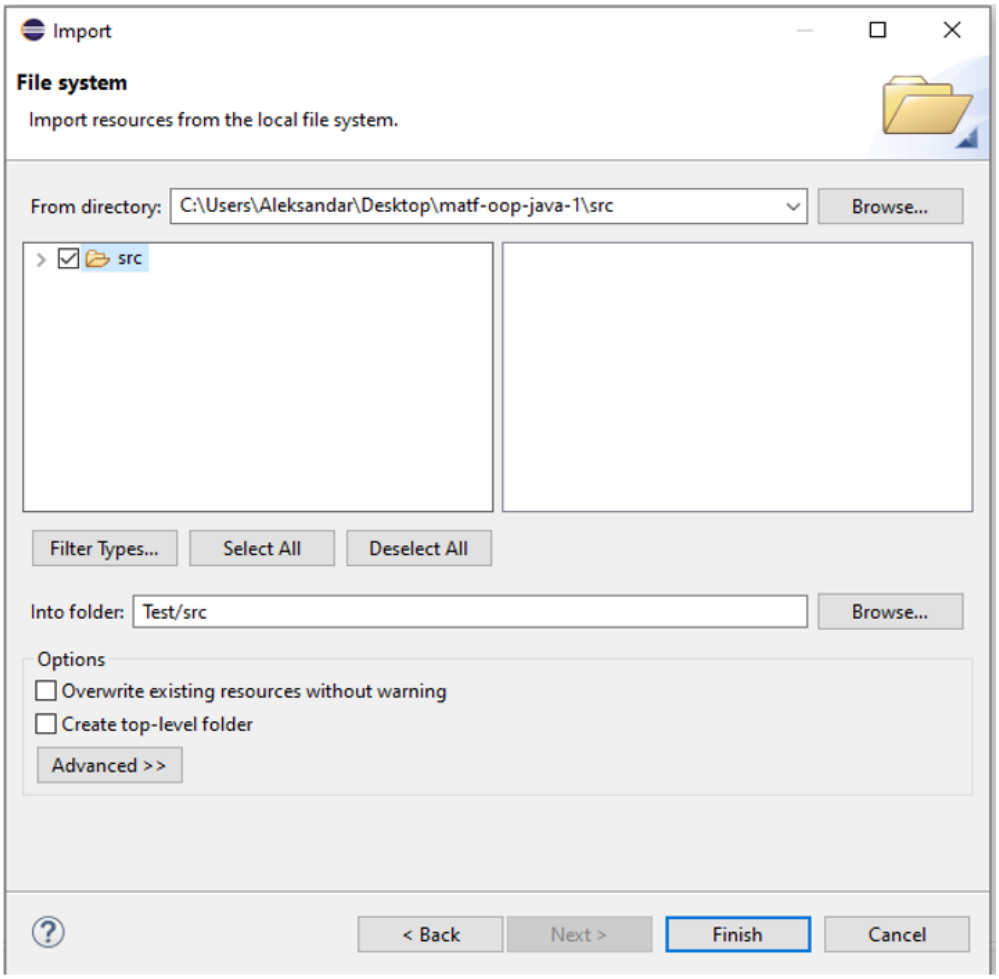
2. Одабрати, потом, извор одакле се увози „General→File System“.



3. За опцију „From directory:“ поставити вредност која упућује на „src“ директоријум преузетог GitHub репозиторијума.

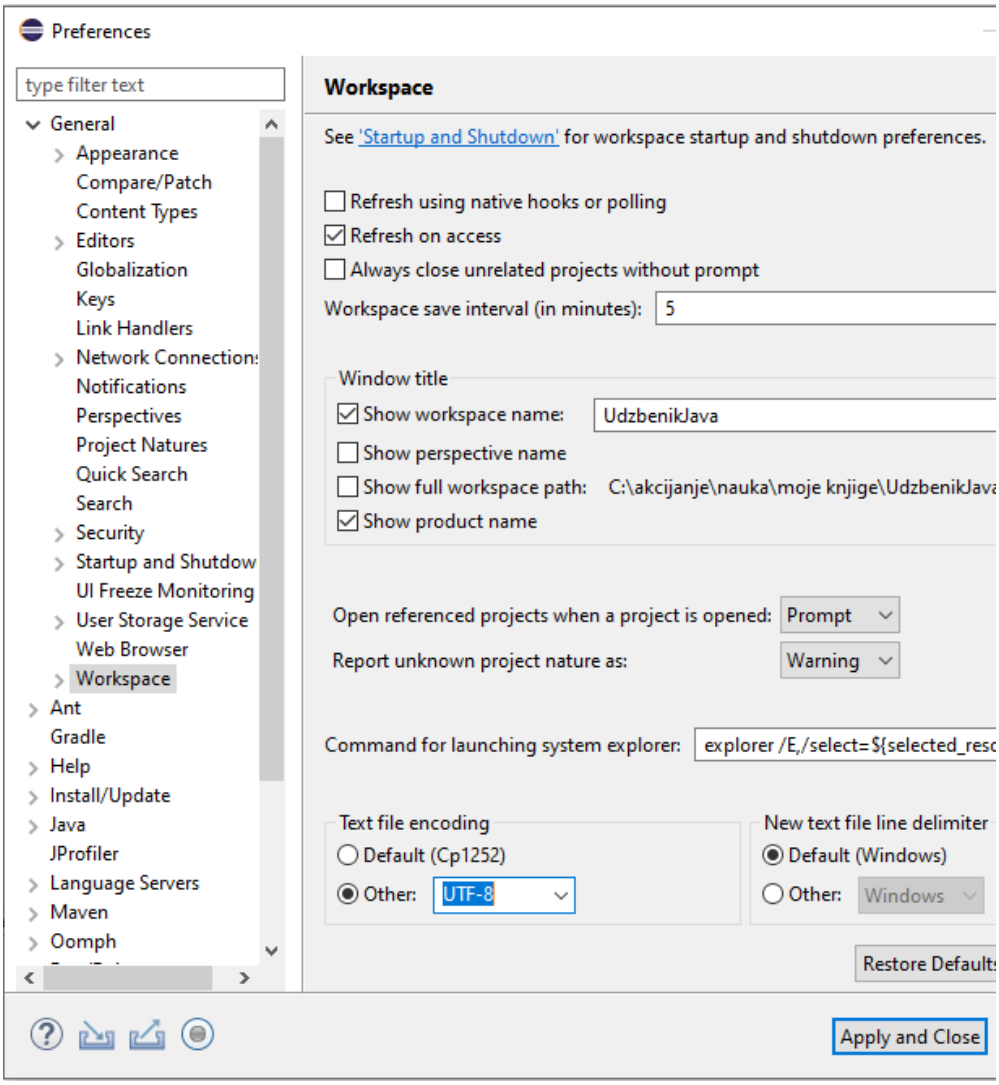


4. Чекирати опцију поред ознаке „src“ и кликнути на дугме „Finish“.



5. Након успешног учитавања свих примера, вероватно, ће се појавити одређене грешке. Разлог је у што се у већини примера користи ћирилично писмо које није подразумевано подешено у оквиру Eclipse пројекта.

Да би се решио овај проблем, потребно је отићи на опцију „Window→Preferences→General→Workspace“ и након тога поставити „Text file encoding“ на UTF-8, затим сачувати подешавање кликом на „Apply and Close“.



6. Након тога би требало да су сви примери функционални, да нема грешака и да се ћирилично писмо уредно приказује у Јава текстуалном едитору.

Уграђивање GitHub материјала у постојећи IntelliJ пројекат

1. Обележити директоријум „rs“ који се налази у директоријуму „src“ и копирати га (на тастатури CTRL+C или десни клик мишем па опција „Copy“).

2. У оквиру раније направљеног Јава пројекта означити мишем директоријум „src“, и у њега налепити претходно копирани директоријум „rs“ (на тастатури CTRL+V или десни клик мишем па опција „Paste“) .

3. Требало би да су, након тога, сви примери (из свих поглавља) учитани и да је кодна страна аутоматски подешена.