

1. laboratorijska vježba: Osnove rada s razvojnom okolinom za programski jezik Java

Cilj prve laboratorijske vježbe je upoznavanje s osnovnim funkcionalnostima razvojne okoline (engl. integrated development environment, IDE) za programski jezik Java. Ove upute su napisane za obje najpopularnije razvojne okoline: *Eclipse* i *NetBeans*. Preduvjet za izvođenje ove laboratorijske vježbe je instalirana zadnja inačica Java JDK (inačica 8) i barem jedna od dvije navedene okoline. Za instalaciju i postavljanje navedenih okolina konzultirajte izvore na webu ili predavanje *1. Osnove programskog jezika Java* (http://www.fer.unizg.hr/download/repository/1_Programski_jezik_Java.pdf). Prije nego što počnete rješavati zadatke, upoznajte se s osnovama odabrane razvojne okoline:

- *Eclipse* - <http://help.eclipse.org/mars/index.jsp?nav=%2F0> (Concepts)
- *NetBeans* - <https://netbeans.org/features/ide/index.html>.

Zadatak 1: Moj prvi program

Napravite novi Java projekt imena `oop-lab1`. Unutar projekta napravite razred imena `MyFirstProgram` unutar paketa `hr.fer.oop.lab1`. Unutar razreda `MyFirstProgram` napraviti metodu `main` koja ispisuje poruku „Moj prvi program!“.

Izvorni kod metode `main`:

```
public static void main(String[] args) {  
    System.out.println("Moj prvi program!");  
}
```

Prilikom pisanja koda, služite se automatskom nadopunom koda (engl., *code completion*) pomoću prečice `Ctrl+Space`.

Opaska:

Eclipse ima „dosadne“ prečace koji vas sprječavaju u unosu uglate zagrade „]“ i vitičaste „{“ pomoću hrvatske tipkovnice. Predlažemo modifikaciju prečaca:

- *Window – Preferences*
- *General – Keys*
 - Odabrati naredbu *Find Text in Workspace* – pritisnuti *Unbind Command*
 - Isto napraviti i za *Skip All Breakpoints*.

Pokrenite program i pogledajte ispis na konzoli odabrane razvojne okoline (*Eclipse: Console view*, *NetBeans: Output window*).

Pomoć:

- *Eclipse*
 - Novi projekt: *File – New – Java Project* – upišite ime projekta - „`oop-lab1`“
 - Novi razred: „Selektirajte“ upravo kreirani projekt – *File* (ili desna tipa miša) – *New – Class* – upišite ime razreda „`MyFirstProgram`“ te ime paketa `hr.fer.oop.lab1`
Uočite da vam IDE nudi opciju automatskog kreiranja funkcije `main` (možete, ali i ne morate ju iskoristiti)
 - Pokretanje programa: pritisnuti zeleni „*play*“ gumb u gornjoj alatnoj traci.
- *NetBeans*
 - Novi projekt: *File – New Project – (Java – Java Application, Next)* – (odznačiti *Create Main Class*)

- Novi razred: File – New File – (Java – Java Class) – upišite ime razreda „MyFirstProgram“ te ime paketa `hr.fer.oop.lab1`
- Pokretanje programa: (Projects) – desni klik na `MyFirstProgram.java` – Run File

Istražite alternativne načine stvaranja nove klase i pokretanje programa.

Dodajte sljedeći kod unutar metode `main`:

```
for (int i = 0; i < args.length; i++) {
    int argNo = i + 1;
    System.out.println(" " + argNo + ". argument programa = " + args[i]);
}
```

Metoda `main` kao argument prima polje `String`-ova. Koristeći odabranu razvojnu okolinu, pokrenite program sa sljedećim argumentima: „prvi drugi treci cetvrti peti sesti sedmi osmi deveti deseti“.

Pomoć:

- Eclipse
 - (strelica pokraj zelenog “*play*“ gumba) – Run Configurations
 - Arguments – Program Arguments – Apply – Close
- NetBeans
 - (strelica pokraj <default config>) – customize – New – OK – Arguments – OK

Pokrenite program s novom konfiguracijom i proučite ispis na konzoli. Istražite alternativne načine konfiguracije pokretanja programa.

U svim ostalim zadacima koristite navedenu konfiguraciju za pokretanje programa!

Zadatak 2: *Debugging* – analiza koda radi otklanjanja neispravnosti

Unutar razreda `MyFirstProgram` dodajte sljedeću metodu:


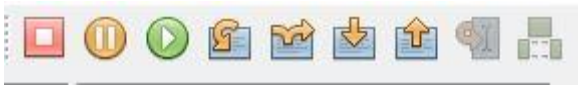
```
public static void cluelessMethod(String[] args) {
    String result = "";
    int step = 0;
    for (int i = 0; i < args.length; i++) {
        step++;
        String arg = args[i];
        int argLength = arg.length();
        String upperCase = arg.toUpperCase();
        result += " " + arg;

        if ((i + 1) < args.length) {
            System.out.print ("Korak " + step + ": ");
            System.out.println("Ne znam zasto ali resetirat cu brojac i...");
            result = "";
            i = 0;
        }
    }
    System.out.println("Argumenti programa su: " + result);
}
```

Na kraju metode `main` dodajte poziv nove metode: `cluelessMethod(args);`

Pokrenite program. Proučite ispis na konzoli. Program bi na kraju izvođenja trebao ispisati argumente programa. Ako izvršavanje programa traje beskonačno, analizirajte kod pomoću ugrađenog *debuggera* te otklonite neispravnost (*warning*, *error*, funkcioniranje).

Pomoć:

- Zaustavite izvođenje programa pritiskom na crveni „Stop“ gumb koji se nalazi u blizini konzole
- Dodajte breakpoint uz naredbu `String result = ""`; te uz `for (int i = 0; i < args.length; i++)`;
 - Eclipse: dvostruki klik na broj linije koda
 - NetBeans: klik na broj linije koda
- Pokrenuti program u debug modu
 - Eclipse: Run – Debug (ili ikona „bube“ kraj ikone za normalno pokretanje programa) (uočite promjenu „perspektive“ tj. rasporeda prozora unutar Eclipse – ovaj raspored prilagođen je za debuggiranje – pojavili su se prozori za praćenje izvođenja, stanja varijabli i sl. Trenutni „Perspective“ možete mijenjati ikonama u gornjem desnom kutu ekrana ili kroz *Window - Perspective*)
 - NetBeans: Debug – Debug Project
- Koristeći standardne opcije debuggera pratite promjenu stanja lokalnih varijabli `argLength`, `i`, `upperCase` i `result` te identificirajte neispravnost
 - Eclipse: 
 - NetBeans: 
- Uklonite neispravnost te ponovno pokrenite program.
(Napomena: istražite koje su i pokušajte zapamtiti kratice za „Step into“, „Step over“ i „Resume“ u odabrane razvojne okoline)

Istražite kako možete dodati vlastiti izraz čiju vrijednost ćete pratiti tijekom debugginga (Eclipse – Expressions, NetBeans – Watch expression). Istražite kako dodati „uvjetni breakpoint“, tj. dodati uvjet zaustavljanja izvođenja programa tek kad je zadovoljne neki uvjet (npr. zaustavite petlju tek kad varijabla korak bude veća ili jednaka 1337)

(Eclipse – desna tipka nad odgovarajućom točkom prekida izvođenja – Breakpoint Properties – Conditional – uvjet, NetBeans - desna tipka nad odgovarajućom točkom prekida izvođenja - Breakpoint – Properties – Condition – uvjet)

Zadatak 3: Dokumentacija koda *Javadoc*, pregled izvornog koda Javinih klasa iz JDK-a

Pročitajte čemu služi metoda `toUpperCase` razreda `String`.

Pomoć:

- Eclipse: (kursor na metodu `toUpperCase`) – F2
- NetBeans: (držati tipku Control i staviti kursor na metodu `toUpperCase`) – kliknuti more... za cjelokupno objašnjenje

Otvorite izvorni kod razreda `String`: (kursor na `String`) – držati Control + klik mišem.

Istražite alternativne načine pregledavanja izvornog koda te pripadajuće dokumentacije.

Što ako izvorni kod nije dostupan?

Ako ste slijedeći gornje upute dobili poruku „Source not found“ ili sl. (npr. koristite JRE, a ne JDK), alat će vam ponuditi mogućnost povezivanja izvornog koda (Attach Source). Izvorni kod za razred `String` je

moguće pronaći u zip datoteci src.zip u korijenskom JDK direktoriju (npr. „C:\Program Files\Java\jdk1.8.0_60\src.zip“). U nekom drugom slučaju, izvorni kod će možda biti potrebno skinuti s interneta ili sl.

Zadatak 4: Izvoz, brisanje i uvoz projekta u razvojnoj okolini

Izvezite svoj projekt na disk u obliku ZIP arhive.

Pomoć:

- Eclipse: File – Export – General – Archive File
- NetBeans: File – Export Project – To ZIP

Izbrišite projekt iz odabrane razvojne okoline.

Pomoć:

- Eclipse: (desni klik na projekt) – Označiti „Delete project contents on disk (cannot be undone)“ - Delete
- NetBeans: (desni klik na projekt) – Označiti „Also delete sources under <putanja> folder.“ – Delete

Uvezite projekt u odabranu razvojnu okolinu.

Pomoć:

- Eclipse: File – Import – Existing Projects into Workspace – Select archive file
- NetBeans: File – Import Project – From ZIP

Zadatak 5: Rad s vanjskim Javinim bibliotekama

Preuzmite binarnu inačicu biblioteke Apache Commons Math 3.5 (commons-math3-3.5-bin) sa sljedeće poveznice: http://commons.apache.org/proper/commons-math/download_math.cgi

Otpakirajte preuzetu arhivu te dodajte commons-math3-3.5.jar u projekt.

Pomoć:

- Eclipse:
 - (Package Explorer) – desni klik – New – Folder – upisati ime „lib“
 - Kopirajte i zalijepite commons-math3-3.5.jar u mapu „lib“
 - desni klik na projekt – Build Path – Configure Build Path
 - Libraries – Add JARs – Označiti commons-math3-3.5.jar
 - Apply – OK
- NetBeans:
 - (Projects) desni klik na projekt – New – Folder – upisati ime „lib“
 - (Files) Kopirajte i zalijepite commons-math3-3.5.jar u mapu „lib“
 - desni klik na projekt – Properties
 - Libraries – Compile – Add JAR/Folder – odabrati putanju kopiranog commons-math3-3.5.jar – OK
- (Napomena: uočite da se JAR-ovi mogu grupirati u knjižnice JAR-ova – tzv. „User libraries“. Ako imate više JAR datoteka koje često zajedno koristite, umjesto da jednu po jednu dodajete u svaki projekt koji želite, skupite ih u User library i dodajte sve odjednom)

Koristeći vanjsku biblioteku, potrebno je izračunati i ispisati minimalnu, maksimalnu te prosječnu duljinu znakova argumenata. Za navedeno je nužno koristiti razred `DescriptiveStatistics`. Dodajte izvorni kod i Javadoc za vanjsku biblioteku te proučite navedeni razred (otvorite izvorni kod i pročitajte

dokumentaciju preko odabrane razvojne okoline).

Preuzmite inačicu biblioteke Apache Commons Math 3.5 s izvornim kodom (commons-math3-3.5-src). Kopirajte commons-math3-3.5-javadoc.jar (iz binarne inačice biblioteke) te commons-math3-3.5-src.zip u mapu „lib“.

Pomoć:

- Eclipse:
 - (Package Explorer) – desni klik na projekt – Build Path – Configure Build Path
 - Libraries – commons-math3-3.5.jar
 - Javadoc location – Edit – (Javadoc in archive) iz mape „lib“ odabrati commons-math3-3.5-javadoc.jar
 - Source attachment – Edit – Path – (Workspace location) iz mape „lib“ odabrati arhivu commons-math3-3.5-src.zip
 - Apply – OK
- NetBeans:
 - desni klik na projekt – Properties
 - Libraries – Compile – commons-math3-3.5.jar – Edit
 - Javadoc: iz mape „lib“ odabrati commons-math3-3.5-javadoc.jar
 - Source: iz mape „lib“ odabrati commons-math3-3.5-src.zip

Korištenje razreda iz vanjskog paketa (biblioteke) potrebno je najaviti naredbom `import` koja se piše na vrhu datoteke `MyFirstProgram.java`:

```
import org.apache.commons.math3.stat.descriptive.DescriptiveStatistics;
```

Provjerite radi li vanjska biblioteka uz pomoć metode `calculateStatistics`.

```
public static void calculateStatistics(String[] args) {
    DescriptiveStatistics statistics = new DescriptiveStatistics();
    for (int i = 0; i < args.length; i++) {
        String arg = args[i];
        int argLength = arg.length();
        statistics.addValue(argLength);
    }
    System.out.println("Prosječna duljina znakova argumenata: " + statistics.getMean());
    System.out.println("MIN: " + statistics.getMin() + " MAX: " + statistics.getMax());
}
```

Zadatak 6: Izvoz programa u Java biblioteku

Izvezite vaš program u neizvršnu JAR datoteku.

Pomoć:

- Eclipse:
 - Unutar projekta napravite novu mapu „dist“
 - File – Export – Java – JAR File
 - unutar „Select the resources to export“ odaberite „src“ i „lib“
 - odaberite putanju do mape „dist“ te nazovite datoteku kao oop-lab1.jar
- NetBeans: JAR datoteka se prilikom operacije „Build“ generira i smješta u „dist“ direktorij unutar

direktorija projekta (npr. C:\Users\Mirko\Documents\NetBeansProjects\oop-lab1\dist). Provjerite samo da li je „označena“ opcija „Build JAR after compiling“ u Project Properties – Build – Packaging.

(Uočite i opciju „compress JAR file“ tj. „Compress the contents of the JAR file“ u Eclipse)

Sada se vaša biblioteka može koristiti na sličan način kao što ste vi koristili Apache Commons Math.

Uočite da se `MyFirstProgram` (tj. njegova metoda `main`) zapakiran u JAR datoteku može pokrenuti iz komandne linije. Da biste to napravili potrebno je uz ime JAR datoteke navesti i naziv razreda, npr. :

- na Linuxu/Unixu/Macu:
`java -cp oop-lab1.jar:../lib/commons-math3-3.5.jar
hr.fer.oop.lab1.MyFirstProgram param1 drugiParam treci`
- na Windowsima:
`java -cp oop-lab1.jar;..\lib\commons-math3-3.5.jar
hr.fer.oop.lab1.MyFirstProgram param1 drugiParam treci`

Otvorite JAR datoteku (npr. s programom 7zip ili sl.) i proučite njen sadržaj. Otvorite datoteku `MANIFEST.MF` pomoću uređivača teksta. Proučite unose u navedenoj datoteci.

Izvezite vaš program u izvršnu JAR datoteku.

Pomoć:

- Eclipse: File – Export – Java – Runnable JAR File. Postavite Launch configuration i Export destination
- NetBeans: desni klik na projekt – Properties – Run – Main Class:
`hr.fer.oop.lab1.MyFirstProgram` pa ponovite isto kao u gornjem primjeru, nakon toga desni klik na projekt – Build

Otpakirajte izvršnu JAR datoteku i otvorite njen `MANIFEST.MF` pomoću uređivača teksta i pogledajte koje su razlike.

Ako smo kreirali ovakvu arhivu, pokretanje programa može se obaviti bitno jednostavnije: umjesto navođenja putanje u kojoj se pretražuju izvršne datoteke i navođenja razreda koji je potrebno pokrenuti, dovoljno je iskoristiti opciju `-jar` i predati naziv JAR arhive: `java -jar oop-lab1.jar`. Java će, zahvaljujući informacijama u `MANIFEST.MF`, pokrenuti `main` iz `MyFirstProgram`.