PT&SCHE

Introduction of part-time and short cycle studies in Serbia

Funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

UNIVERZITET METROPOLITAN BEOGRAD

KI103 - JAVA 1: OSNOVE PROGRAMIRANJA U JAVI

Osnove programiranja

Lekcija 01

PRIRUČNIK ZA STUDENTE

KI103 - JAVA 1: OSNOVE PROGRAMIRANJA U JAVI

Lekcija 01

OSNOVE PROGRAMIRANJA

- Osnove programiranja
- Poglavlje 1: Uvod u programiranje u Javi
- Poglavlje 2: Kreiranje i prevođenje prvog Java programa
- Poglavlje 3: Jednostavni programi u Javi
- Poglavlje 4: Razvojno okruženje NetBeans
- Poglavlje 5: Domaći zadaci
- Zakljucak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

UVOD

Lekcija pravi uvod u programiranje u Programskom jeziku Java

Ovom lekcijom polaznici kursa se uvode u osnove *Java* programiranja. Shodno tome cilj je polaznici kursa ostvare sledeće ciljeve:

- upoznaju se sa programskim jezikom Java i njegovim komponentama;
- upoznaju se sa pojmovima JRE i JDK;
- nauče da instaliraju Javu i razvojno okruženje na vlastitim računarima;
- nauče da kreiraju i pokrenu jednostavne java programe na kontrolnoj liniji i u razvojnom okruženju;
- nauče da pišu programe koristeći komentare u programskom kodu;
- nauče da kreiraju Java programe u razvojnom okruženju u formi projekata, u odgovarajućem paketu.

Poglavlje 1

Uvod u programiranje u Javi

UVOD

Programske platforme obezbeđuju programske bibliotekei softverske alate neophodne za uspešan razvoj i izvršenje programa

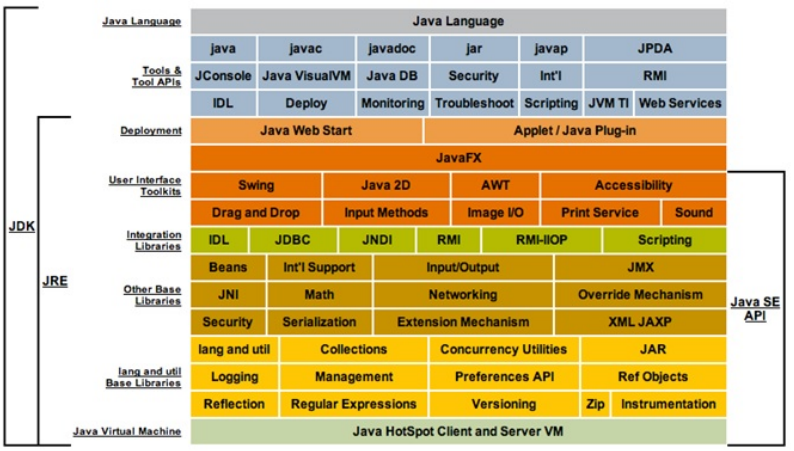
Radi lakšeg razvoja softvera i i znatnog ubrzanja celog procesa, razvijene su posebne programske platforme. U ovom delu će biti reći o :

- Java Runtime Environment - elementarnog dela Jave koji je potreban za pokretanje i izvršavanje Java programa
- Java Development Kit - skupa programskih alata za razvoj Java aplikacija
- Kreiranju i prevodjenju prvog java programa
- posebno, biće naučeno kako se instalira JDK i razvojno okruženje NetBeans IDE za razvoj Java programa.

ŠTA JE JAVA?

Java je objektno-orijentisani jezik koje se može izvršavati na svim platformama. Java je i tehnologija (sadrži pakete klasa) i sadrži i okruženje za izvršenje (JRE)

Java je objektno-orjentisani programski jezik opšte namene. Pored toga Javu čini i kompletno izvršno okruženje (JRE) koje omogućava izvršavanje programa sa pratećim komponentama.



Slika 1.1 Programski jezik Java i njegovi sastavni delovi

Jedna od najznačajnijih osobina je da je da se Java može izvršavati na više platformi. To znači da jednom napisan kod na Javi radi na svim platformama za koje postoji implementacija Jave. Moto Jave je "Write Once, Run Anywhere".

Prenosivost Jave se ne odnosi samo na računarsku platformu, već i na operativni sistem. Nebitno koji hardver je u pitanju i operativni sistem, ako postoji implementacija Jave, kod se može izvršavati bez ikakvih dodatnih koraka.

Najveći uticaj na sintaksu Jave su imali C/C++ i srodni jezici. Ideja je bila da se privuče baza programera koji već poznaju C/C++ jezik. Međutim u odnosu na C/C++ uvedene su mnoga poboljšanja koja su trebala da pojednostave programiranje.

Pre svega se misli na sledeće:

- izbačeni su pokazivači i rezervisanje i oslobađanje memorije je prepušteno Javi umesto da o tome računa vodi programer
- izbačeno je višestruko nasleđivanje i uvedeni su interfejsi
- ubačene su dodatne provere da bi se izbegle nenamerne greške, recimo kod logičkih iskaza ako se umesto znaka jednakosti upotrebi operator dodele prijaviće se greška prilikom prevođenja
- nizovi su implementirani kao objekat, a ne preko pokazivača

IZVRŠNO OKRUŽENJE JAVE

Java Rautime Environemnt (JRE) i pokreće i izvršava Java programe, a Java Development Kit (JDK) sadrži JRE i alat za razvoj, prevođenje i praćenje debagovanja aplikacija.

Java je u potpunosti objektno-orjentisan programski jezik. To znači da se **svaka funkcionalna jedinica mora predstaviti u vidu klase**. Posledica ovoga je i ta da se prilikom rešavanja problema koristi modelovanje kao mehanizam analize i projektovanja struktura realnog sveta.

Java specifikacija u stvari definiše dve stvari:

- Sam programski jezik Java
- Izvršno okruženje (Java Runtime Environment - JRE)

Java koristi kombinovani model izvršavanja. Izvorni kod se prevodi na takozvani bajtkod, a onda tako dobiveni kod se interpretira na virtuelnoj mašini. Ovo je omogućilo da se pored Jave pojave i drugi programski jezici višeg nivoa kao što su Scala, Groovy, Clojure i drugi, čiji kompajleri takođe proizvode Java bajtkod. Ne samo da se kod pisan na ovim jezicima može izvršavati na Java virtuelnoj mašini, već ovako napisani kod se može kombinovati i sa Javom.

Elementarni deo Jave je **Java Runtime Environment (JRE)** koji je potreban za pokretanje i izvršavanje Java programa. **Java Runtime Environment** se sastoji od virtuelne mašine (JVM), biblioteka i drugih komponenti koje omućavaju izvršavanje Java aplikacija (odnosno bajtkoda).

Za razvoj aplikacija potreban je **Java Development Kit (JDK)**. **Java Development Kit** se sastoji od **Java Runtime Environment**-a i alata za razvoj, prevođenje, praćenje i debagovanje aplikacija.

JAVA RUNTIME ENVIRONMENT

Java Runtime Environment (JRE) predstavlja elementarni deo Jave koji je potreban za pokretanje i izvršavanje Java programa

Radi lakšeg razvoja softvera i i znatnog ubrzanja celog procesa, razvijene su posebne programske platforme. U ovom delu će biti reći o :

- Java Runtime Environment - elementarnog dela Jave koji je potreban za pokretanje i izvršavanje Java programa
- Java Development Kit - skup programskih alata za razvoj Java aplikacija

Kreiranju i prevodjenju prvog Java programa

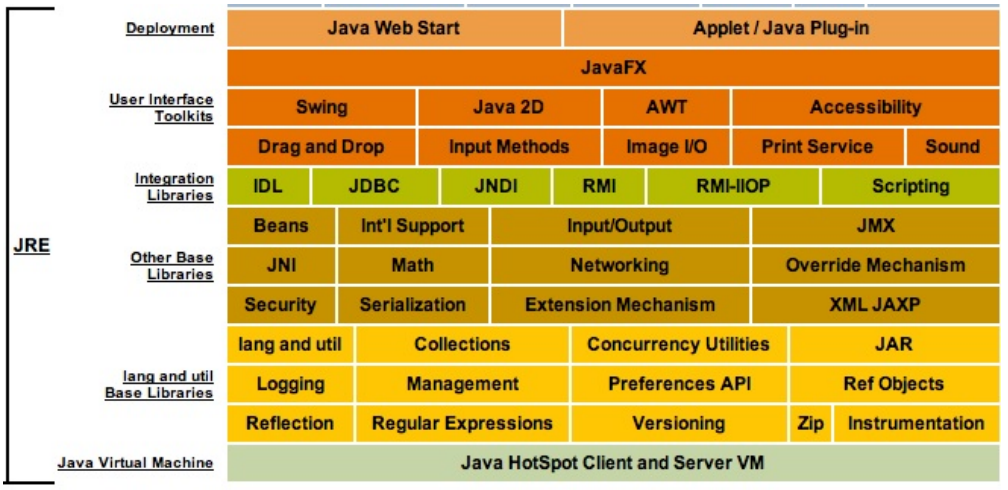
Java Runtime Environment (JRE) predstavlja elementarni deo Jave koji je potreban za pokretanje i izvršavanje Java programa.

JRE sadrži:

- **Java virtuelnu mašinu** (engl. Java Virtual Machine - JVM)
- Biblioteke i druge komponente koje omogućavaju izvršavanje Java aplikacije

Aktuelna verzija JRE je 8. Preuzimanje i instalacija se može izvršiti sa sledeće adrese: http://www.java.com/en/download/index.jsp

JRE je potreban elemernt za izvršenje Java programa, ali ne i za razvoj Java programa.



Slika 1.2 JDK -Java Development Kit - Run time - JDK izvršna verzija

JAVA DEVELOPMENT KIT (JDK)

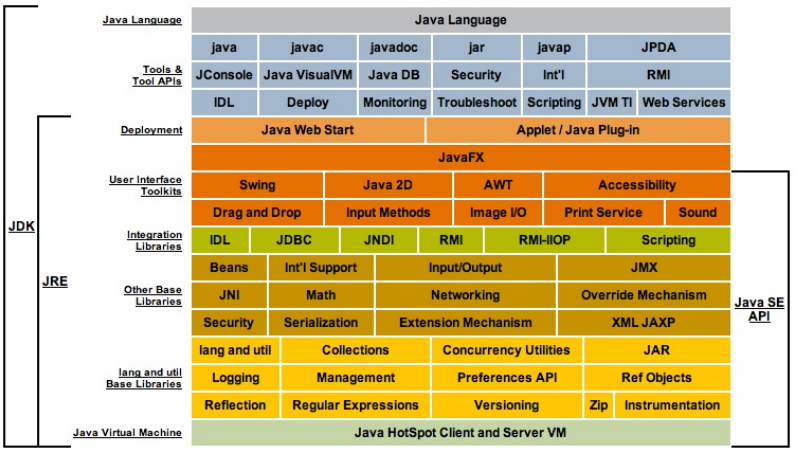
Java Development Kit (JDK) predstavlja skup programskih alata za razvoj Java aplikacija.

Java Development Kit (JDK) predstavlja skup programskih alata za razvoj Java aplikacija. JDK sadrži:

- **Java Runtime Environment**
- Alate za razvoj, prevođenje, praćenje i debagovanje aplikacija (slika 3)

Spisak alata koje sadrži možete videti na sledećem linku:

http://en.wikipedia.org/wiki/Java_Development_Kit



Slika 1.3 JDK -Java Development Kit - razvojna verzija

Aktuelna verzija je **8**. Preuzimanje i instalacija se može izvršiti sa sledećeg linka:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Napomena da je za razvoj aplikacija u Javi potrebno preuzeti **Java Development Kit (JDK**), a ne Java Runtime Environment (JRE), pošto ovaj paket omogućava samo pokretanje Java aplikacija.

Pre preuzimanja potrebno je prihvatiti uslove korišćenja (License Agreement). Prilikom preuzimanja je potrebno odabrati verziju za operativni sistem koji je instaliran na računaru na kome će se instalirati JDK. Po preuzimanju potrebno je pokrenuti instalaciju i pratiti upustva do kraja instalacije.

PRIMER 1 - INSTALIRANJE JAVE NA RAČUNARU

Preuzimanje i instaliranje neophodnih razvojnih alata i tehnologija.

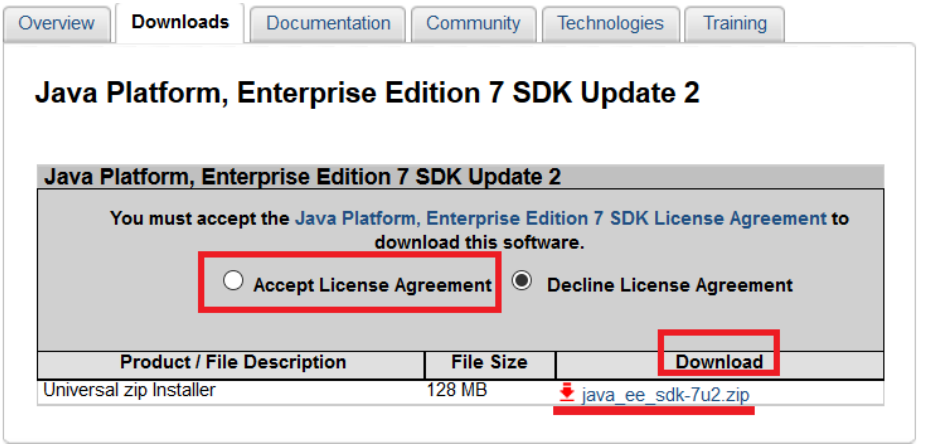
1. Instalirati Java EE poslednju JDK verziju i podesiti sistemske varijeble;
2. Instalirati razvojno okruženje NetBeans IDE - poslednju dostupnu verziju;

3. Instalirati GlassFish aplikativni server;
4. Instalirati MySQL server

Za početak je potrebno otići na link http://www.oracle.com/technetwork/java/javaee/downloads/index.html i izabrati preuzimanje Java EE 7 JDK instalacionog paketa.

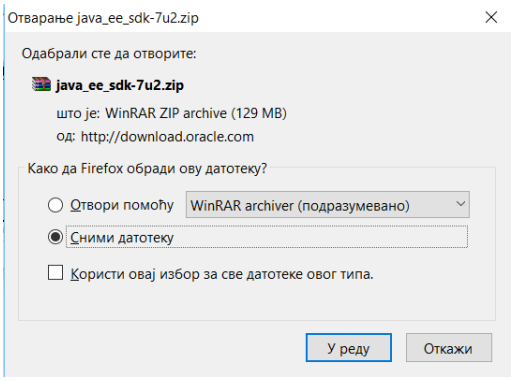
Slika 1.4 Stranica za preuzimanje Java EE 7 JDK instalacionog paketa.

Klikom na obeleženu sliku otvara se prozor u kojem se zahteva prihvatanje uslova korišćenja. Kada se uslovi prihvate vrši se klik na link za download.



Slika 1.5 Prihvatanje uslova korišćenja

Klikom na odgovarajući link preuzima se Java EE 7 JDK ZIP datoteka.

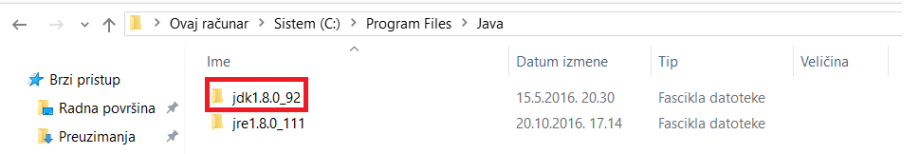


Slika 1.6 Preuzimanje Java EE 7 JDK instalacionog paketa

PRIMER 1 - ISTALIRANJE JAVE I PODEŠAVANJE SISTEMSKIH VARIJABLI

Istaliranje Jave i podešavanje sistemskih varijabli na lokalnom računaru.

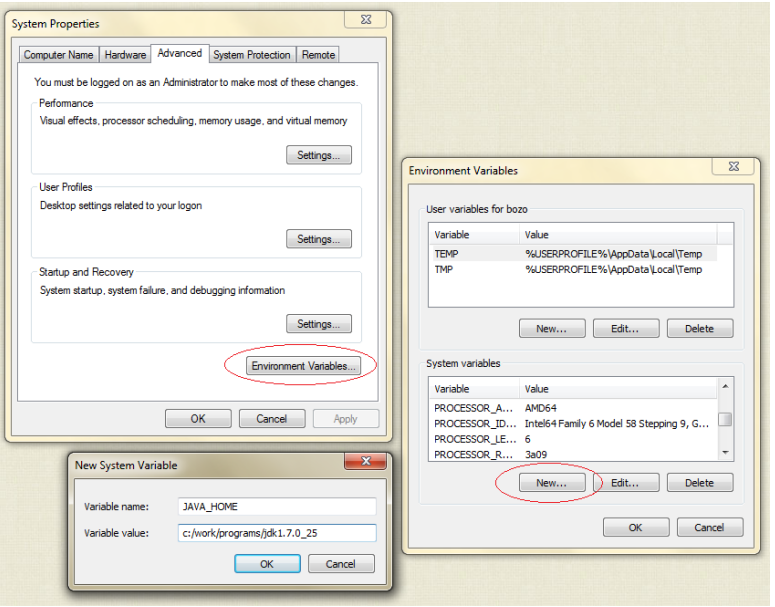
JDK datoteka je preuzeta i sačuvana u folderu Downloads. U sledećem koraku je neophodno je izvršiti njeno raspakivanje i neka to bude na lokaciji: C:\Program Files\Java kao na sledećoj slici:



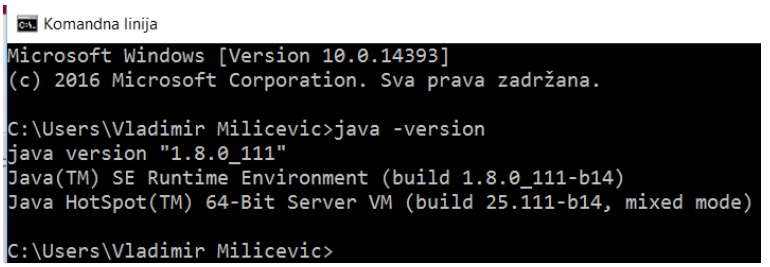
Slika 1.7 JDK je raspakovan

Sada je neophodno učiniti Javu "vidljivom" sa bilo koje lokacije na lokalnom računaru. To se omogućava podešavanjem sistemskih varijabli. Desnim klikom naMy Computer, zatim Properties, pa Advanced System Settings i Enviroment Variables, otvara se prozor u kojem će biti izvršena navedena podešavanja. Sada se definiše jedna sistemska promenljiva po imenu JAVA_HOME čija će vrednost biti putanja do JDK foldera. Nakon toga iz liste postojećih varijabli, bira se varijabla PATH u okviru koje je neophodno dodati sledeći string ;%JAVA_HOME%/bin kojim je određena putanja do JAVA prevodioca. Navedeno je prezentovano slikom 5.

Nakon podešavanja sistemskih varijabli, neophodno je proveriti da li je sve obavljeno kako treba. Otvaranjem MS DOS Command Prompt i kucanjem instrukcijejava –version vrši se provera. Ukoliko se dobije informacija kao na slici 6, posao instalacije Jave je uspešno završen.



Slika 1.8 Podešavanje sistemske varijable za Javu



Slika 1.9 Provera instalacije

ZADATAK 1

Instalirajte Javu na Vašem računaru

Instalirajte JDK paket alata, za razvoj Java programa, na Vašem računaru prateći sledeće korake:

1. Sa linka http://www.oracle.com/technetwork/java/javase/downloads/index.html preuzmite ZIP datoteku koja odgovara JDK paketu alata;
2. Raspakujte preuzetu datoteku na izabranoj lokaciji na disku (npr. c:/java);
3. Izvršite podešavanje sistemskih varijabli da bi instalirana Java bila dostupna sa bilo koje lokacije na vašem računaru;
4. Testirajte da li ste uspešno instalirali i podesili Javu.

RAZLIKE IZMEĐU JDK, JRE I JVM (VIDEO)

Vidoe pokazuje razlike između JDK, JRE i JVM.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 2

Kreiranje i prevođenje prvog Java programa

PRVI PROGRAM

Kreiranje programa MojPrviJavaProgram.java

Program pisan u Javi se može pokrenuti sa komandnog promta pozivanjem fajla koji sadrži Java aplikacije u izvornom (Java) obliku.

Sledeći kod predstavlja prvi program koji se obično daje kada se počinje sa nekim programskim jezikom. To je program koji na ekranu treba da ispiše ''Zdravo, svete!''.

```
public class MojPrviJavaProgram{
    public static void main(String[] args){
        System.out.println("Zdravo, svete!");
    }
}
```

Ime ove datoteke mora biti MojPrviJavaProgram.java i to ime mora odgovarati imenu klase. Kod definisanja imena mora se voditi računa o velikim i malim slovima (o čemu ćemo govoriti u narednom predavanju), pritom ime klase i ime datoteke moraju i po tome odgovarati jedno drugom. Prvi red programa

```
public class MojPrviJavaProgram{
```

govori da je u pitanju izvorni program koji definiše klasu po imenu MojPrviJavaProgram.

Mali programi se obično sastoje od samo jedne klase, a klasa predstavlja deo programa. (Kasnije ćemo detaljnije definisati klase.) Neke klase mogu da sadrže mnogo redova koda. Sve što pripada jednoj klasi mora da se postavi između otvorene i zatvorene velike zagrade ({ i }). Ime klase, (a time i ime datoteke) sami određujete. Ime se mora sastojati od slova ili brojeva. Prvi karakter mora biti slovo, a unutar imena ne smeju postojati praznine. Obično imena klasa počinju velikim slovima, ali to nije obavezno. Ime izvorne datoteke obavezno ima ekstenziju .java (piše se malim slovima). Obično se u jednoj datoteci nalazi samo jedna klasa, ali se u istoj datoteci može naći i više klasa. Rad:

```
public static void main(String[] args)
```

pokazuje gde program počinje.

Reč main znači da je ovo glavni metod, odnosno metod kojim virtuelna mašina počinje izvršenje programa. Taj glavni metod u ovom slučaju sadrži samo jedan iskaz:

```
System.out.println("Zdravo, svete!");
```

Ovaj iskaz prikazuje tekst koji je dat u okviru dvostrukih navodnika na ekran.

SINTAKSNE GREŠKE

Sintaksne greške se mogu javiti pri prevođenju izvornog programa (koda).

U slučaju da nismo prethodnu proceduru valjano ispoštovali, mogu se javiti greške. Sledi programski kod programa sa namerno napisanom greškom:

```
public Class MojPrviJavaProgram{
    public static void main(String[] args) {
        System.out.println("Zdravo, svete!");
    }
}
```

Ključna reč classje ovde napisana velikim početnim slovom, pa ova vrsta greške predstavlja sintaksnu grešku. Sintaksne greške predstavljaju "gramatičke greške" kod rada sa nekim programskim jezikom.

Ako se ovaj program kompajlira, prikazaće se sledeće:

```
D:\MojiPrograms>javac MojPrviJavaProgram.java
compiling: MojPrviJavaProgram.java
MojPrviJavaProgram.java:1: Class or interface declaration expected.
Class MojPrviJavaProgram
^
1 error
```

Kompajler je pokušao da prevede izvorni kod u bajkod, ali je naišao na veliko slovo **C** tamo gde to nije očekivao. Poruka o grešci koju je dao, nije baš najjasnija. Te poruke nikad nisu potpuno jasne. Ipak, ova greška bar pokazuje u kojem redu se kompajler zbunio. U ovom slučaju kompajler nije napravio novu izvršnu datoteku, pošto je prevođenje zaustavljeno.

Da bi program ponovo proradio, moramo da ispravimo grešku - potrebno je vratiti se se u editor i promeniti veliko slovo **C** u malo, a nakon toga ponoviti ciklus prevođenja i izvršavanja**.**

Potrebno je izvršavati sledeću proceduru sve dok program ne bude radio kako treba:

1. Unesite program uz pomoć nekog editora teksta i snimiti program na disk
2. Kompajlirajti program pomoću komande javac
3. Ako postoje sintaksne greške, vratite se na korak 1
4. Pokrenite program pomoću komande java
5. Ako program ne radi kako treba, vratiti se na korak 1
6. Ako sve radi kako treba - procedura je završena

LOGIČKE GREŠKE

Logičke greške su neočikavane i se javljaju u radu programa.

Ako je program uspešno preveden (kompajliran) ne mora da znači da će on ispravno raditi. Greške koje se manifestuju u radu programa, a koje niste očekivali se nazivaju **logičkim greškama,** ili **bagovima u programu.**

Ako ste, na primer, dobili zadatak da odštampate tekst **Zdravo, svete!,** a napišete sledeći program, on će se uspešno prevesti, ali nećete ostvariti ono što ste očekivali.

```
public class MojPrviJavaProgram{
    public static void main (String[] args) {
        System.out.println("Zdravo, problemi!");
    }
}
```

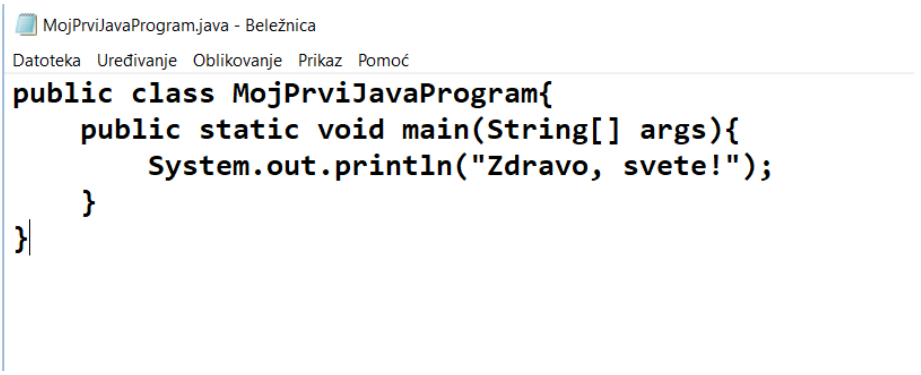
Logičke greške se mnogo češće javljaju u programima koji su duži od našeg primera. Mnogo teže ih otkriti i one se uglavnom otkrivaju tek kasnije kada program počne da se primenjuje u praksi. Zbog toga je vrlo važno izvršiti testiranje programa pre realnog puštanja u upotrebu. Testiranje treba da rade i programeri, ali i krajnji korisnici u realnim uslovima. Testiranje je posebna priča, o kojoj će kasnije biti više reči.

PRIMER 2 - KREIRANJE PRVOG JAVA PROGRAMA U TEKST EDITORU

Java program je moguće kreirati primenom bilo kojeg tekst editora

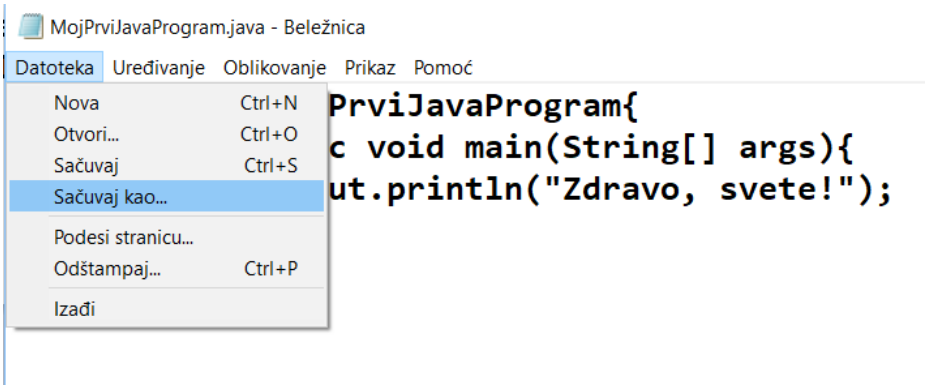
Java program je moguće kreirati primenom bilo kojeg tekst editora. Na primer, biće iskorišćen *Notepad* u kojem će biti kreiran prvi program čiji je listing već prikazan. Program će odgovarati klasi pod nazivom *MojPrviJavaProgram.java*.

Pokreće se program za rad sa tekstom i u njemu se unosi programski kod prikazan u sekciji *Prvi program*. Navedeno je moguće ilustrovati sledećom slikom.



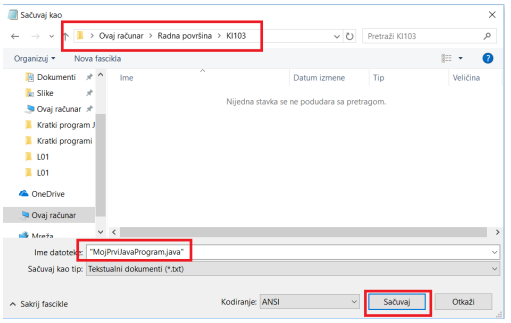
Slika 2.1 Primena tekst editora za kreiranje prvog Java programa.

Nakon što je obavljeno kodiranje, neophodno je pristupiti čuvanju kreiranog programskog koda u odgovarajućoj datoteci. Naziv datoteke mora da odgovara nazivu klase. Iz menije tekst editora se bira opcija *Datoteka* (ili File ukoliko je meni na engleskom) nakon čega je neophodno izabrati opciju Sačuvaj kao (eng. Save As). Navedeno je moguće ilustrovati sledećom slikom.



Slika 2.2 Čuvanje datoteke sa kodom

Veoma je važno da ekstenzija sačuvane datoteke bude .java. To se postiže tako što će u sledećem prozoru (naredna slika) naziv datoteke, zajedno sa ekstenzijom, biti zapisan pod znacima navoda. U suprotnom, datoteka bi bila sačuvana kao obična tekstualna datoteka, ekstezije .txt.



Slika 2.3 Izbor likacije i naziva datoteke koda

PRIMER 2 - PREVOĐENJE I POKRETANJE KREIRANOG PROGRAMA SA KOMANDNE LINIJE

Demonstracija prevođenja i izvršavanja kreiranog Java programa.

Na prethodnoj slici je moguće uočiti da je izabrana i lokacija na disku na kojoj će biti sačuvana datoteka *MojPrviJavaProgram.java*. Sada se pristupa procesu prevođenja kreiranog programa sa komandne linije. Otvara se interpretator naredbi komandne linije, u *Windows* operativnom sistemu se naziva *Command Prompt*, u kojem se kuca prva naredba koja predstavlja navigaciju ka lokaciji na disku na kojoj je sačuvan kreirani program. Na primer, u ovom konkretnom slučaju ta naredba glasi:

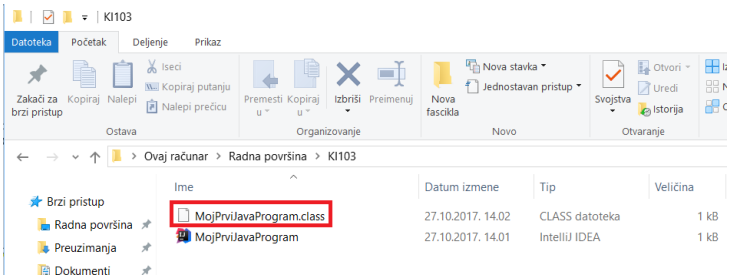
```
cd C:\Users\Vladimir Milicevic\Desktop\KI103
```

Navedeno je označeno na slici 5 plavom bojom.

Pošto se sada nalazite na lokaciji na kojoj je snimljena datoteka, odgovara kreiranom programu, moguće je pristupiti prevođenju kreirane klase. Na komandnoj liniji je neophodno otkucati sledeću naredbu:

```
javac MojPrviProgram.java
```

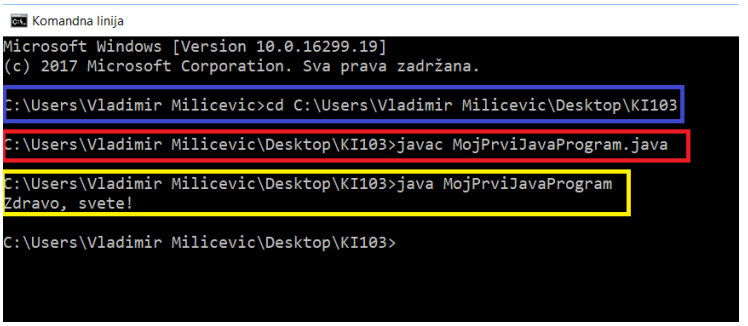
Navedeno je ilustrovano na slici 5 crvenom bojom. Ovom naredbom se poziva Java prevodilac koji od datoteke *MojPrviJavaProgram.java* pravi datoteku *MojPrviJavaProgram.class.* U ovoj datoteci se čuva *bajt kod*, skup prevedenih Java instrukcija koje se pomoću JVM izvršavaju na bilo kojoj računarskoj platformi. Bajt kod i izvorna datoteka se čuvaju na istoj lokaciji, a to je prikazano sledećom slikom.



Slika 2.4 Izvorna i bajt kod datoteka

Konačno, pokreće se datoteka sa bajt kodom, sledećom naredbom, i prikazuje rezultat izvršavanja programa (slika 5 - obeleženo žutom bojom):

```
java MojPrviJavaProgram
```



Slika 2.5 Prevođenje i pokretanje kreiranog programa sa komandne linije

```
java MojPrviJavaProgram
```

ZADATAK 2

Kreirajte, prevedite i pokrenite vaš Java program

Izvršite sledeće zadatke:

1. Otvorite neki tekst editor;
2. Kreirajte vaš Java program pod nazivom: *ImePrezimePrviProgram.java*;
3. Iskoristite prethodni primer i, umesto stringa "Zdravo, svete!", unesite Vaše ime i prezime;
4. Snimite program na željenoj lokaciji na Vašem računaru;
5. Otvorite interpretator komandne linije;
6. Izvršite navigaciju ka lokaciji na kojoj ste sačuvali datoteku sa kreiranim kodom;
7. Prevedite kreirani program sa komandne linije;
8. Pokrenite Vaš program i pratite rezultat izvršavanja.

PISANJE PRVOG JAVA PROGRAMA (VIDEO)

Video pokazuje kako da vaš prvi program pripremite i izvršite.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 3

Jednostavni programi u Javi

KAKO PROGRAM IZGLEDA I RADI?

Programi u Javi se izvršavaju iz main metoda.

Na slici 1 prikazan je listing jednostavnog programa koji pri svom izvršenju prikazuje na monitoru računara tekst “Welcome to Java”. Zbog toga, naziv datoteke koja sadrži programa, nazvana je Welcome.java

```
1 public class Welcome {
 2   public static void main(String[] args) {
 3   // Prikaz poruke  Welcome to Java! na monitoru
 4   System.out.println("Welcome to Java!");
 5   }
 6 }
```

Rezultat na monitoru:

Welcome to Java!

Prikazani brojevi nisu deo programa. Navedeni su samo radu reference, tj. pri objašnjenju programskih linija.

Linija 1 definiš)e klasu. Svaka program u Javi (nazivaćemo ih Java programima) mora da ima najmanje jednu klasu. Svaka klasa ima svoje ime, koje počinje velikim slovom. U našem primeru, naziv klase jeWelcome. Šta je klasa? To je skup podataka i metoda (malih programa) definisana između dve velike zagrade, ispre koji se nalazi ključna reč class, i naziv klase I npr. *public class Welcome { .....}*..

O konceptu klasa govorićemo u kasnijim lekcijama.

Linija 2 definiše metod koji se naziva main. Metod je mali program. Svaki program u Javi mora da ima metod main i izvršenje programa počinje i kontroliše se iz metoda main. Klasa može da ima više metoda, ali metod main mora da ima, jer on inicira početak izvršenje programa.

Šta je metod? Metod je konstrukcija (mali program) koja sadrži programske instrukcije na pisane u programskom jeziku, tj. u Javi. Metod main u ovom programu sadrži instrukciju System.out.println . Ova instrukcija prikazuje tekst “Welcome to Java” na ekranu monitora. Tekst definisan između znakova uzvika (npr. “Welcome to Java”) se u Javi naziva stringom. String je pojam u programiranju koji označava niz znakova (tj. Slova, brojeva, specijalnih

znakova...). String počinje i završava se znakom uzvika. Svaka instrukcija u Javi se završava zapetom i tačkom (;).

KOMENTARI U PROGRAMU

Komentari pomažu programerima da objasne program koji pišu, a ostalima, komentari pomažu da razumeju kako program radi.

Program pisan u Javi sadrži ključne reči, tj. rezervisane reči koje imaju specifično značenje koje kompajler (prevodilac izvornog programa) razume. Te ključne reči se ne mogu u programu koristiti za neke druge svrhe, jer imaju rezervisano značenje. Na primer, kada kompajler vidi ključnu reč class, on to razume kao ime (naziv) klase. Druge ključne reči u programu su public, static i void.

```
1 public class Welcome {
 2   public static void main(String[] args) {
 3   // Prikaz poruke  Welcome to Java! na monitoru
 4   System.out.println("Welcome to Java!");
 5   }
 6 }
```

Linja 3 sadrži komentar koji dokumentuje šta je ovaj program i kako on radi. Komentari pomažu programerima da objasne program koji pišu, a ostalima, komentari pomažu da razumeju kako program radi. Linije sa komentarima ne predstavljaju programske instrukcije, jer program može da radi i bez njih. Međutim, preporuka je da programeri koriste što više linije sa komentarima, kako bi program bio dobro dokumentovan, tj. Objašnjen. To znatno pomaže ostalim da razumeju kako program radi.

. U Javi, linija sa komentarom počinje sa dve kose crte (//) po liniji. Drugi način da se definiše jedna ili više linija sa komentarima, je da se pre komentara stavi kosa crta i zvezdica (/*), a na kraju komentara se stavi zvezdica i kosa crta (*/). Na ovaj na;in se definiše komentar u bloku ili u pausu.

Kada kompajler vidi //, on ignoriše celu liniju, tj. tekst iza // do kraja linije. Međutim, kada kompajler vidi /* on preskače jednu iliiii više linija u programu, sve dok ne dođe do */ , što je za njega oznaka za prestanak ignorisanja tog dela programa.. Primer jednog komentara u bloku se daje dole

```
// This application program displays Welcome to Java!
/* This application program displays Welcome to Java! */
/* This application program
displays Welcome to Java! */
```

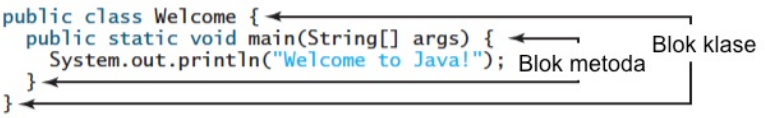
PROGRAMSKI BLOKOVI

Programski blokovi, koji počinju sa { i završavaju se sa } definišu grupu instrukcija metoda, ili grupu podataka i metoda, u slučaju klase.

Par velikih zagrada ({....}) u Java programu definiše programski blok koji grupiše komponente programa koje s enalaze između ovih zagrada. Svaki blok počinje sa otvorenom velikom zagradom ({), a završava se zatvorenom velikom zagradom (}).

Svaka klasa ima **blok klase** koji grupiše podatke i metode klase.Takođe, svaki metod sadrži **blok metoda** koji grupiše programske instrukcije (ili iskaze) metoda.

Blokovi mogu da budu povezani, tako da se jedan nalazi unutar drugog i tako redom, kao što je pokazano na slici:



Slika 3.1 Primerna programskih blokova

Svaka otvorena velika zagrada mora da se zatvara sa drugom velikom zagradom zatvaranja. Uvek kada otkucate otvorenu veliku zagradu, odmah otkucajte i zatvorenu veliku zagradu istog programskog bloka, da je ne bi zaboravili, što onda izaziva javljanje greške u programu. Najveći broja Java IDE automatski ubacuju zatvorene velike zagrade posle svake otvorene velike zagrade.

U sledećoj slici , prikazana je upotreba nekih specijalnih znakova u Javi.

Oznaka	Naziv	Opis
{}	Otvorene i zatvorene velike zagrade	Označava blok instrukcija
()	Otvorene velike i male zagrade	Upotreba kod metoda
[]	Otvorene i velike srednje zagrade	Upotreba kod nizova
//	Duple kose crte	Označavaju liniju sa komentarom
“ “	Otvorene i zatvoreni znaci navođenja	Označava string, tj. niz znakova
;	Tačka-zapeta	Označava kraj instrukcije

Slika 3.2 Primer nekih specijalnih znakova u Javi

Programi pisani u Javi su osetljivi (reaguju) na velika slova. Zato, ne smete naziv metoda, npr. main, napisati sa Main.

Java ima svoja pravila pisanja programa, tj. programskih instrukcija (sintaksa) i ako ne poštujete ta pravila sintakse, kompajler će vam javiti greške u programu. Ako dođe do javljanje grešaka sintakse pri prevođenju programa, treba da ih otklonite u izvornnom koru, i onda ponoviti prevođenje programa pozivanja kompajlerajavac.

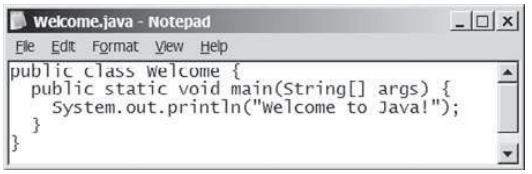
ŠIRE INFORMACIJE U VEZI SA KREIRANJEM, PREVOĐENJEM I IZVRŠAVANJEM PROGRAMA

Java program se kreira u tekst editoru ili u Java IDE, prevodi i izvršava sa komandne linije komandnog prozora računara, ili korišćenjem Java IDE.

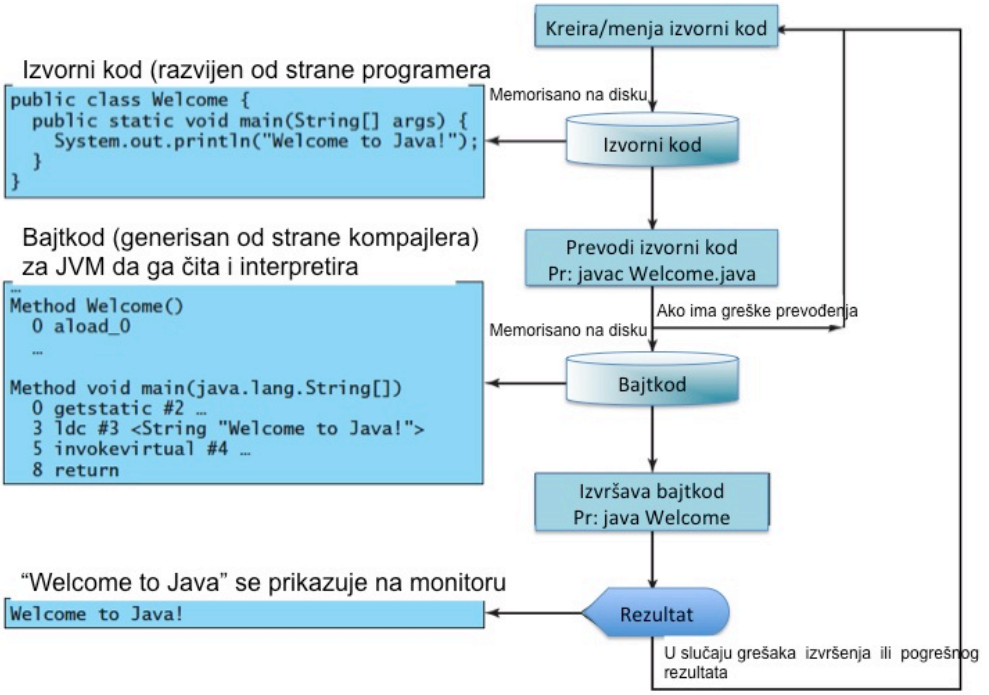
Svaki program prođe kroz aktivnosti kreiranja, prevođenja i izvršenja (slia 1). Taj proces se ponavlja, ako se jave greške pri prevođenju ili pri izvršenju (ili se dobijaju pogrešni rezultati. Proces s eponavlja posle otklanjanja uzroka greške.

Možete koristiti bilo koji editor teksta ili Java IDE za kreiranje ili menjanje izvornog koda programa. Prevođenje i izvršenje programa možete izvršiti i sa komandne linije prikazane na komandnom prozoru računara.

Na slici 2 je prikazan Notepad editor sa ispisanim programom Welcome.java.



Slika 3.3 Program Welcome.java umet preko Notepad-a.



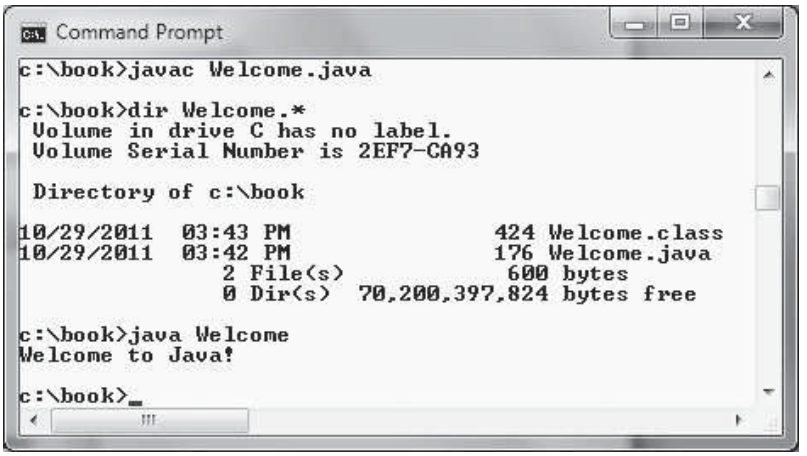
Slika 3.4 Grafički prikaz kreiranja, prevođenja i izvršavanja programa

IZVRŠENJE PROGRAMA

Program Welcome.java se prevodi sa prevodiocem javac Welcom.java u bajtkod, koji je u fajlu Welcome.class, koja se izvršava sa: java Welcome.

Na slici 3 je prikazan komandni prozor u kome se na komandnoj liniji poziva Java kompajler (prevodilac) sajavac, i onda prevodi izvorni kod u bajtkod, sa komandom java.

java



Slika 3.5 Izvršenje programa

Upozorenje: Ne koristite proširenje .class u komandnoj liniji pri izvršenju programa. Za izvršenje programa otkucajte : java ImeKlase

Kada se izvršava Java program, JVM prvo uzima bajtkod klase u memoriji, upotrebom programa koji se naziva “class loader”. Ako vaš program upotrebljava druge klase, class loader dinamički pruzima datoteke pred samu njihovu upotrebu.

Posle preuzimanja klase , JVM upotrebljava program koji se nazova **verifiktorom bajtkoda** radi provere ispravnosti bajtkoda i obezbeđuje kontrole da bajtkod ne prekrši sigurnosne restrikcije.i provere da izvršni program bez optečenja ostalih programa, i računara.

KORIŠĆENJE KOMENTARA

Pored komentara u linija (/) i u bloku linija (/....*/), treba koristiti i javadoc komentarisanje celih klasa ili metoda.*

Program treba tako pisati da se može lako čitati i razumeti. Dokumentaciju programa čine napomene i komentari koji su ovaćeni u tekst programa u izvršnom obliku. Stil programiranja i

njegova dokumentacija je isto tako važna kao i dobro programiranje, jer smanjuju mogućnost javljanja greške, i lini program čitljivijim. Ovde se daju najvažniji saveti za ispravno pisanje Java programa.

Stavite apstrakt na početku svakog programa koji objašnjava šta program radi, navedite njegova glavna svojstva i funkcije, kao i specifične tehnike koje upotrebljava. U slučaju velikih programa, uključite i komentare koji opisuju glavne korake njegovog rada i objasnite ono što je u samom programu teško čitljivo.

Pored korićenja linija sa komentarima, koje počinju sa //, i blokova komentara koje počinju a sa /* , Java podržava i komentare specifičkog tipa, koji se nazivaju **javadoc** komentarima. Komentari javadoc počinju sa sa /** a završavaju se sa */.. Oni se dobiijaju u jednoj HTML datoteci upotrebom javadoc naredbe, definisane u JDK.

Upotrebljavajte javadoc comentare (/**.....*/) za komentarisanje cele klase ili celog metoda. Ovi komentari moraju da budu ispred zaglavlja klase ili metoda da bi mogli da budu izvučeni u javadoc HTML datoteku.

Za komentarisanje unutar metoda, upotrebljavajte linije sa komentarom (//).

Primena konsistentnog stila u uvlačenju linije (indentation), čini programe jasnijim i lakše čitljivim, lakšim za ispravljanje i održavanje. Uvlačenje reda se koristi radi ilustracije strukturne odnose između programa i njegovih komponenti ili instrukcija. Java može da čita program i u slučaju da sadrži sve instrukcije u jednom vrlo dugom redu, ali ljudi lakše čitaju i održavaju program čije instrukcije su poređane sa različitim uvlaćenjem liija. Uvlaćenje (intent)svake podkomponente ili instrukcije za najmanje dva blanko otkucja, tako da se ta uvlaćenja međusobno nadomeštaju.

Jedno prazno polje bi trebalo dodati na obe strane zbog binarnog operatora , ko što j pokazano u sledečim istrukcijama:

System.out.println(3+4*4); Ovo je loš stil

System.out.println(3 + 4 * 4); Ovo je dobar stil

STILOVI ZA BLOKOVE

Postoji dva stila koji se preporučuju za pisanje blokova: stil sledeće linije i stil kraja linije. Kada izaberete jedan od njih, morate se konsistentno njega pridržavati, jer mešanje stilova ne v

Blok je grupa instrukcija (iskaza) navedenih između velikih zagrada. Postoje dva polularna stila za prikazivanje blokova (slika 1):

- stil sledeće linije, i
- stil kraja linije.

```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println("Block Styles");
  }
}
```

a) Stil sledeće linije

```
public class Test {
  public static void main(String[] args) {
    System.out.println("Block Styles");
  }
}
```

b) Stil kraja linije

Slika 3.6 Stilovi za blokove

Stil sledeće linije povezuje velike zagrade po vertzikali i čini program lakše litljivim, dok stil kraja linija štedi prostor i može da pomogne da se izbegnu neke greške pri programiranju. Oba stila su prihvatljiva. Izbor zavisi od ličnog afiniteta programera ili organizacije. Međutim, kada izaberete jedan od ova dva stila, koristite ga konsistentno, jer nije preporučljivo ih mešati.

U ovom predmetu ćemo koristiti stila kraja linije, d abi bili konsistentni sa Java API izvornim kodom.

PRIMER 3

Primeri sa više poruka i sa matematičkim izrazom.

Biće kreiran Java program koji će sada prikazivati tri poruke, umesto jedne.

U tekst editoru biće kreiran dokument pod nazivom *WelcomeWithThreeMessages.java*, sa sledećim kodom:

```
public class WelcomeWithThreeMessages {
 public static void main(String[] args) {

 //Komentar može da se piše ovako

  System.out.println("Programming is fun!");

/*Komentar može da se piše i na ovaj način*/

  System.out.println("Fundamentals First");
  System.out.println("Problem Driven");
    }
 }
```

Na poznati način, neophodno je prevesti ovaj kod. Nakon prevođenja kod se izvršava i daje sledeće rezultate:

Rezultat na monitoru:

Programming is fun!

Fundamentals First

Problem Driven

U ovom programu, moguće je primetiti da su korišćeni komentari, radi se o korisnim linijama koda koje prevodilac ignoriše i ne prikazuju se u rezultatima, a služe za opisivanje pojedinačnih linija koda ili blokova.

Sledeći program obračunava vrednost sledećeg razlomka:

$$\frac{10.5+2x3}{45-3.5}$$

Slika 3.7 Razlomak koji bi trebalo da bude uzračunat

U tekst editoru biće kreiran dokument pod nazivom *ComputeExpression.java*, sa sledećim kodom:

```
1 public class ComputeExpression {
 2   public static void main(String[] args) {
 3   System.out.println((10.5 + 2 * 3) / (45 – 3.5));
 4   }
 5 }
```

Na poznati način, neophodno je prevesti ovaj kod. Nakon prevođenja kod se izvršava i daje sledeće rezultate:

Rezultat na monitoru:

0.39759036144578314

Operator množenja u Javi je *. Kao što vidite, relativno je jednostavnoi napisati matematički izraz u Javi.

ZADATAK 3

Kreiranje Java programa koji prikazuje vaše podatke.

Izvršite sledeće zadatke:

1. Otvorite neki tekst editor;
2. Kreirajte vaš Java program pod nazivom: *LicniPodaci.java*;
3. Iskoristite prethodni primer;
4. Program bi trebalo da prikazuje sledeće poruke: Ime i prezime, adresa, mesto stanovanja i poštanski broj;
5. Svaka naredba za prikazivanje poruka bi trebalo da bude obeležena odgovarajućim komentarom;
6. Snimite program na željenoj lokaciji na Vašem računaru;

7. Otvorite interpretator komandne linije;
8. Izvršite navigaciju ka lokaciji na kojoj ste sačuvali datoteku sa kreiranim kodom;
9. Prevedite kreirani program sa komandne linije;
10. Pokrenite Vaš program i pratite rezultat izvršavanja.

Poglavlje 4

Razvojno okruženje NetBeans

INTEGRISANO RAZVOJNO OKRUŽENJE

Integrisana razvojna okruženja (engl. IDE – Integrated Development Environment) su aplikacije koji omogućavaju brz i efikasan razvoj aplikacija

Za razvoj Java aplikacija dovoljan je JDK i editor teksta. Međutim, takav razvoj bi bio naporan i spor.

Integrisana razvojna okruženja (engl. IDE – Integrated Development Environment) su aplikacije koji omogućavaju brz i efikasan razvoj aplikacija.

Prednosti korišćenja IDE:

- Editovanje koda, kompletiranje koda skraćenicama i prečicama, syntax highlighing, pomoć (hints)
- Pretraživanje koda, refaktorisanje koda
- Kompajliranje i izvršavanje, integrisan debugger
- Čarobnjaci (wizards) koji olakšavaju komplikovane zadatke
- Kreiranje izvršnih i instalacionih paketa

Neka razvojna okruženja za programski jezik Java:

NetBeans – besplatno razvojno okruženje, jednostavno za učenje i korišćenje, sadrži sve neophodne elemente potrebne za razvoj aplikacija svih obima, može se proširivati dodacima.

Eclipse – besplatno razvojno okruženje, malo složenije za korišćenje, sadrži sve neophodne elemente potrebne za razvoj aplikacija svih obima, može se prošitivati dodacima.

IntelliJ IDEA – komercijalno okruženje, veoma moćno za sve tipove aplikacija, ogroman broj dodataka, odlična podrška kako za Javu tako i za ostale jezike koje proizvode bajtkod kao što su Scala i Groovy, odlični dodaci i mogućnosti.

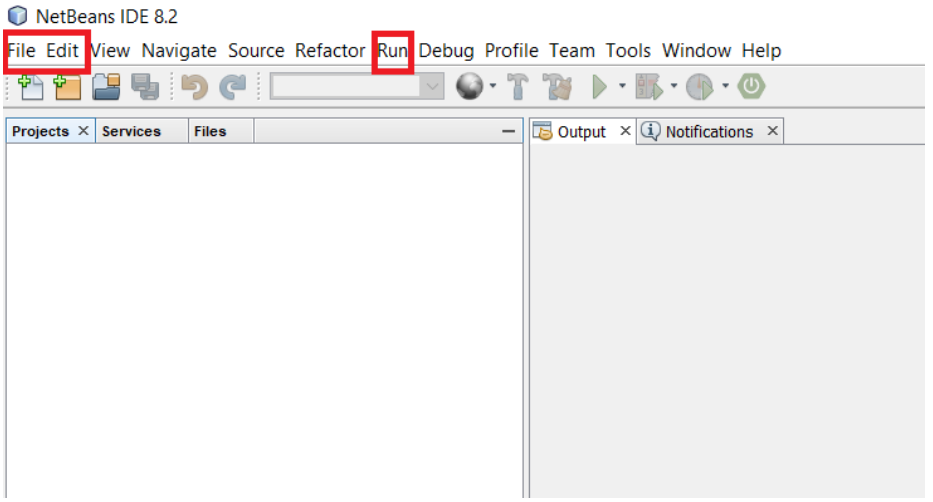
LINJA MENIJA U GLAVNOM PROZORU NETBEANS-A

Svaki deo glavnog prozora ima svoju namenu.

Namena pojedinih delova glavnog prozora okruženja NetBeans su:

Linija menija sadrži sve menije iz kojih se mogu birati opcije za postojeće funkcije NetBeans-a. Osnovni meniji su:

- **Meni File** je namenjen za formiranje novih i otvaranje postojećih projekata, za formiranje novih i otvaranje postojećih fajlova, kao i za ostale standardne aktivnosti u vezi sa fajlovima od kojih se sastoji Java aplikacija**.**
- **Meni Edit** ima potpuno istu funkciju za rad sa tekstom programa kao istoimeni meniji u ostalim grafičkim okruženjma (na primer: Undo, Redo, Copy, Paste, Find,Replace).
- **Meni Run** je namenjen za izvršavanje prevedenog programa. Pored toga, u ovom meniju se može pokrenuti debager radi otklanjanja grešaka, testiranja i detaljnog praćenja stanja promenjivih u toku rada programa.



Slika 4.1 Linja menija u glavnom prozoru netbeans-a

LINIJA PANELA GLAVNOG PROZORA NETBEANS-A

Linija sa alatima sadrži ikone alata koji omogućavaju brži pristup različitim funkcijama NetBeans-a nego preko menija

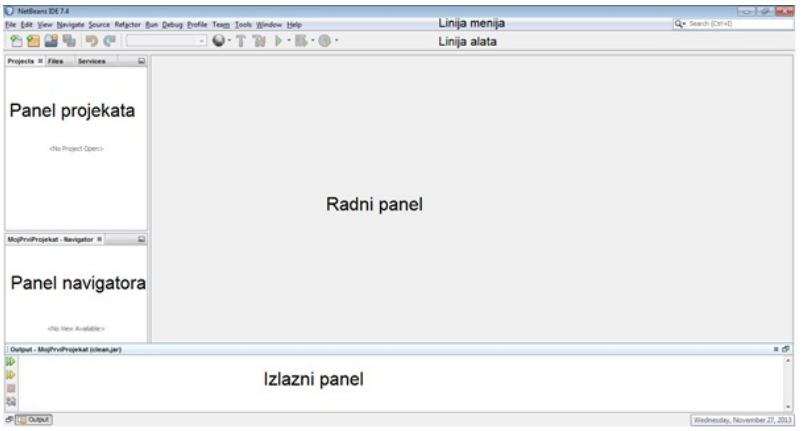
Linija sa alatima sadrži ikone alata koji omogućavaju brži pristup različitim funkcijama NetBeans-a nego preko menija.

Panel projekata sadrži kartice Projects, Files i Services u kojima se hijerarhijski prikazuju projekti, datoteke i servisi koji su trenutno aktuelni za program.

Panel navigatora prikazuje detalje stavki koje su izabrane u jednoj od tri pomenute kartice panela projekata.

U desnom, **radnom panelu** se prikazuje sadržaj datoteka koje su izabrane u jednoj od kartica panela projekata. Sadržaj otvorenih datoteka u radnom panelu se može menjati, jer se oni automatski otvaraju u integrisanom editoru NetBeans-a.

U donjem, **izlaznom panelu** se prikazuju sve važne poruke tokom prevođenja programa, kao i ulazno-izlazni podaci samog programa tokom njegovog izvršavanja.

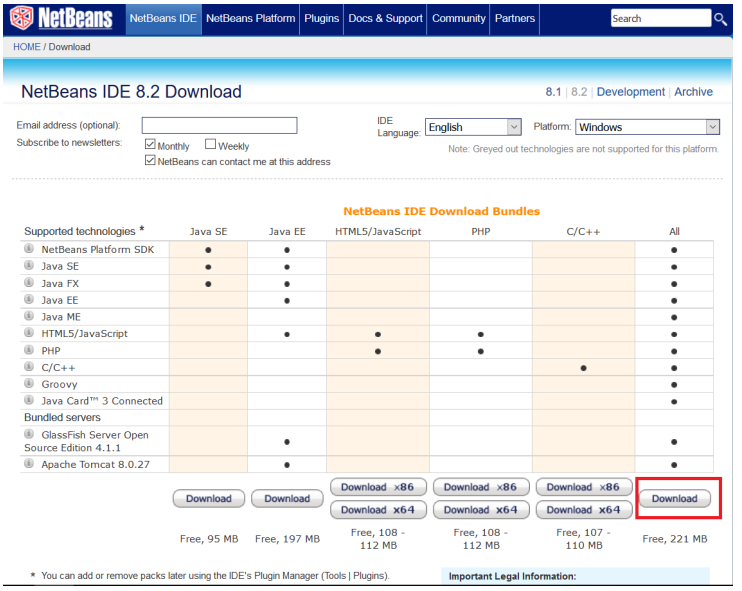


Slika 4.2 2: Glavni prozor NetBeans-a

PRIMER 4 - ISTALIRANJE RAZVOJNOG OKRUŽENJA

Preuzimanje razvojnog okruženja za razvoj Java veb aplikacija.

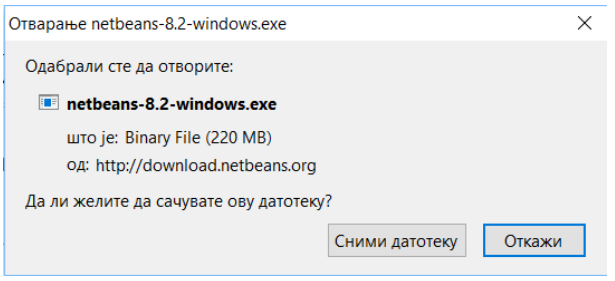
U nastavku je neophodno otići na link http://netbeans.org/downloads/index.html i preuzeti instalacionu datoteku za NetBeans IDE razvojno okruženje.



Slika 4.3 Stranica za preuzimanje instalacione datoteke

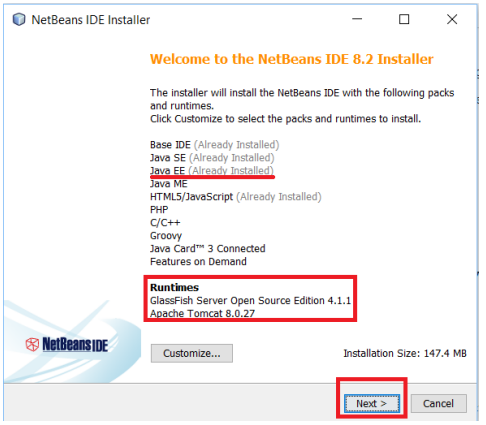
Sa slike je moguće primetiti da je Java SDK bilo moguće preuzeti zajedno, u paketu sa NetBeans.

Takođe, u ovom instalacionom paketu je moguće preuzeti i aplikacioni server GlassFish koji će u radu na razvoju veb Java aplikacija biti neizostavan u daljem radu.



Slika 4.4 Preuzimanje instalera

Pokretanjem instalera, bira se lokacija gde će razvojno okruženje biti instalirano, takođe i svi potrebni alati - GlassFish i Apache server. Praćenjem čarobnjaka za instalaciju, ceo proces se jednostavno i automatski obavlja.



Slika 4.5 Izbor tehnologija i servera za instalaciju

PRIMER 5 - NETBEANS PROJEKAT

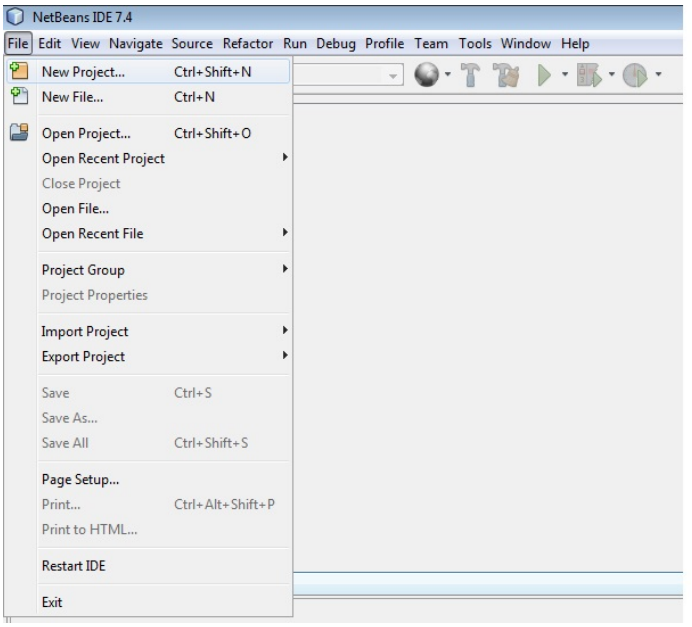
Osnovna jedinica rada u okruženju NetBeans je projekat. Projekat

Osnovna jedinica rada u okruženju NetBeans je **projekat. Projekat** koji odgovara, grubo govoreći, jednom Java programu i sastoji se od njegovih izvornih datoteka, pridruženih informacija o tome kako ih treba prevesti i izvršiti, kao i od ostalih tehničkih detalja o putanjama, dokumentaciji i slično. Sve ove podatke NetBeans smešta u direktorijumu projekta (sa poddirektorijumima) koji se pravi na osnovu imena projekta. Da bi se omogućilo grupisanje više povezanih projekata na jednom mestu, svi direktorijumi takvih projekata se inicijalno prave u drugom datom direktorijumu koji se obično naziva radni direktorijum (engl. workplace ili workbench).

Naravno, o svemu ovome vodi računa uglavnom sam softver, pa za razne vrste uobičajenih Java projekata (obična aplikacija, biblioteka klasa, Web aplikacija i tako dalje) postoje standardni šabloni od kojih se inicijalno pravi odgovarajući projekat. Nakon početnog formiranja projekta na osnovu odgovrajućeg šablona, projekat se otvara u glavnom prozoru NetBeans-a tako što se njegova **logička struktura** prikazuje u kartici Projects i struktura datoteka u kartici Files panela projekata. Pored toga, u radnom panelu se otvara glavna datoteka projekta koji sadrži početni izvorni tekst Java programa

Da biste napravili novi projekat nakon pokretanja NetBeans-a, postupite na sledeći način (pritiskom na dugme **Next** odgovarajućeg prozora u svakom koraku, prelazite na sledeći korak):

U glavnom prozoru izaberite opciju **File > New Project**, kao što je to prikazano sa slici:

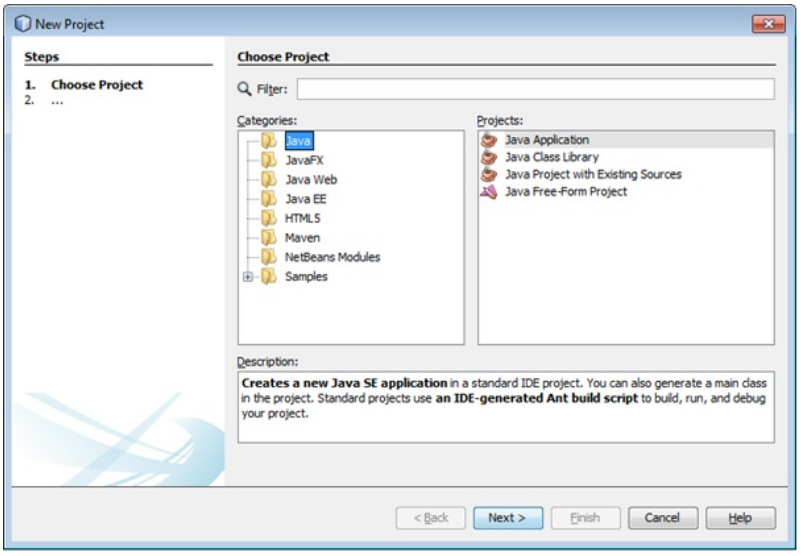


Slika 4.6 Kreiranje novog projekta

PRIMER 5 - DEFINISANJE NOVOG PROJEKTA

U prozorima NewProject i New Java Application unose se određeni podaci neophodni za kreiranje novog projekta razvoja softvera.

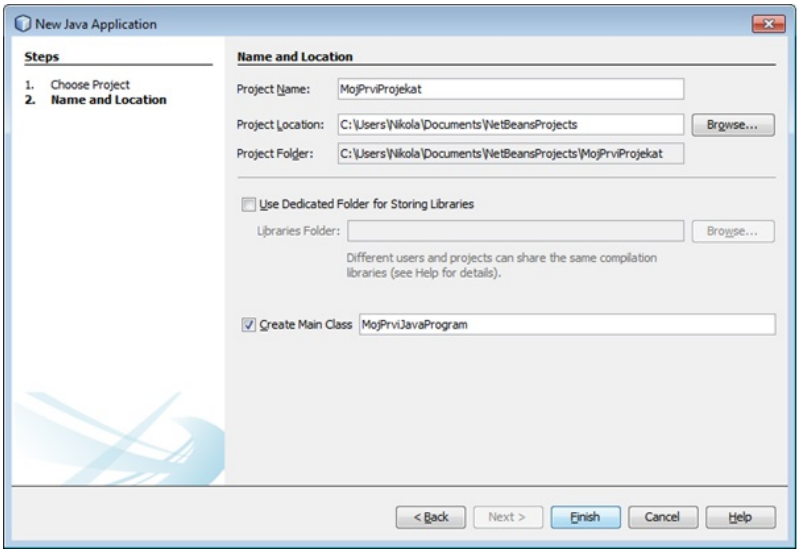
U otvorenom prozoru New Project **i**zaberite kategoriju Java i zatim opciju Java Application, kao što je to prikazano sa slici:



Slika 4.7 Izbor kategorije i tipa Java aplikacije

U otvorenom prozoru New Java Application treba upisati sledeće podatke o novom projektu (što je ilustrovano sa slici 4):

U polju Project Name upišite MojPrviProjekat za ime projekta.



Slika 4.8 Definisanje naziva Java projekta

U polju Project Location, pritiskom na dugmeBrowse izaberite direktorijum radnog prostora gde želite da se nalaze svi projekti. Obratite pažnju na to da taj direktorijum mora biti prethodno napravljen (u primeru je to NetBeansProjects, koji se automatski kreira). U polju Create Mai Class upišite MojPrviJavaProgram za ime glavne klase Java programa.

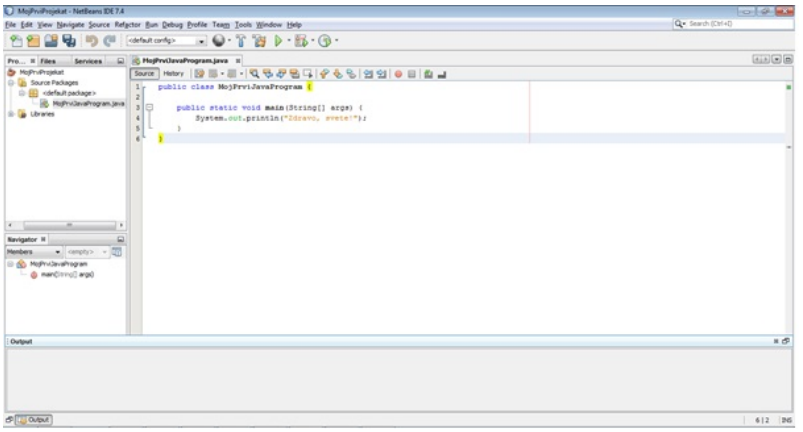
PRIMER 5 - IZGLED PRAZNOG OBJEKTA

Primena komande Create Main Class formira se kostur glavne klase MojPrviJavaProgram

Pošto ste u postupku pravljenja projekta izabrali polje Create Main Class, u okruženju NetBeans se formira kostur glavne klase MojPrviJavaProgram.

Automatski generisan tekst u datoteci datoteci MojPrviJavaProgram.java zamenite tekstom radnog programa MojPrviJavaProgram na slici .

Promene u datoteci MojPrviJavaProgram.java sačuvajte birajući opciju File > Save (prečica za ovu opciju je Ctrl + S).



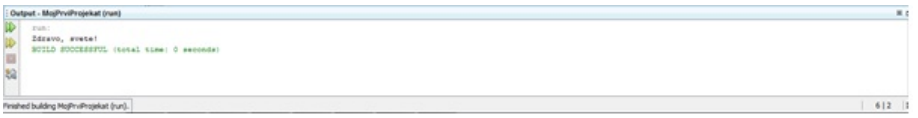
Slika 4.9 Inicijalni kod kreiranog projekta

PRIMER 5 - POKRETANJE PROJEKTA IZ RAZVOJNOG OKRUŽENJA

Primena komande Run File i Run Project

Da biste preveli Java program, izaberite opcijuRun > Run File u glavnom prozoru NetBeans-a. Prečica za ovu opciju je Shift + F6. Ulazno-izlazni podaci tokom izvršavanja programa se takođe prikazuju u izlaznom panelu tog prozora. Jedan primer izvršavanja programa MojPrviJavaProgram je prikazan na slici:

Ukoliko želite da program izvršite više puta (možda sa promenjenim ulaznim podacima), to postižete ponovnim biranjem opcije Run > Run Project.



Slika 4.10 Pokretanje projekta iz razvojnog okruženja

Ako se rezultat prevođenja završava porukom BUILD SUCCESSFUL, to znači da je prevođenje bilo uspešno i da se program može izvršiti. U suprotnom, ako se rezultat prevođenja završava porukom BUILD FAILED, program sadrži greške koje se moraju ispraviti. Greške se u izlaznom panelu prikazuju u obliku hiperveza čijim praćenjem možete preći u izvorni tekst radnog panela na mestu odgovarajuće greške. Na taj način možete ispraviti sve greške i ponovo prevesti program izborom opcije Run > Run Project. Ovaj postupak morate ponoviti sve dok prevođenje ne bude uspešno.

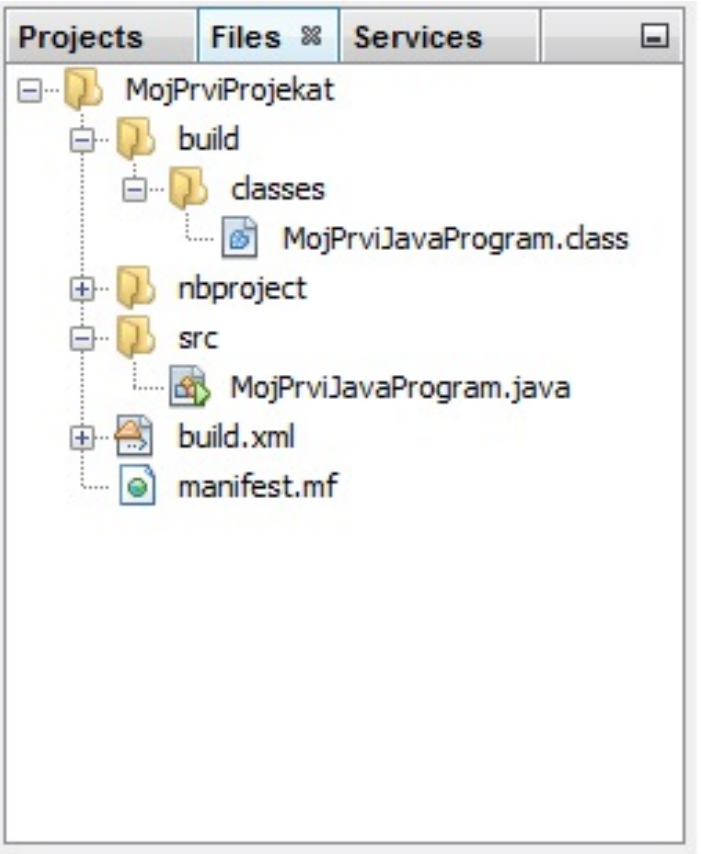
PRIMER 5 - STRUKTURA PROJEKTA

U panelu projekta Files prikazuju se komponentne datoteke projekta MojPrviJavaProjekat

U konkretnom primeru, uspešno prevedeni projekat se sastoji od bajtkoda u datoteci MojPrviJavaProgram.class. Lokacije datoteka raznih komponenti projekta MojPrviProjekat možete pregledati otvaranjem kartice Filespanela projekata i proširivanjem pojedinih čvorova prikazane strukture. Jedan primer ove mogućnosti je prikazan na slici.

Pored ovih osnovnih mogućnosti, grafičko okruženje NetBeans pruža još mnogo toga što je potrebno za razvoj složenih i profesionalnih aplikacija. Dodatne informacije o tome možete naći u priručniku sa sledećeg linka:

http://docs.oracle.com/cd/E40938_01/doc.74/e40142/toc.htm



Slika 4.11 Struktura kreiranog projekta

PRIMER 6 - PRIMER SA TRI PORUKE U RAZVOJNOM OKRUŽENJU

Kreiranje prvog projekta KI103-V01 i klase Zadatak1.java

Kreirati java projekat koji ćete zvati CS101V01. Main klasa treba da se zove Zadatak1.

Ovaj zadatak treba da prikaže 3 poruke na ekranu: Dobrodošli u Javu, Dobrodošli u kompjuterske nauke i Programiranje je zabavno.

Objašnjenje:

Važno je obratiti pažnju na samu strukturu programa. Prva linija definiše klasu programa. Svaki Java program mora imati najmanje jednu klasu čije ime se po konvenciji piše početnim velikim slovom. U ovom slučaju smo kreirali klasu Zadatak1.

Svaka klasa može sadržati nekoliko metoda, ali najvažniji je metodmain koji predstavlja prvu tačku u izvršavanju programa. Svaki program počinje da se izvršava metodom main.

Blok koji se izvršava u okviru neke metode mora počinjati otvorenom skupovom zagradom, a završavati se zatvorenom {}. Da bi program bio sintaksno ispravno napisan, zagrade moraju biti dobro uparene.

Metoda koju koristimo za štampanje poruka je **System.out.println**

Rešenje:

```
public static void main(String[] args) {
        new Zadatak1();
    }
    public Zadatak1(){
        /*
        System.out.printLn je metoda biblioteke System koja
        omogućava ispisivanje poruke na ekranu. Razlika
        između print i printLn je da li će se tekst
        prikazati u novom redu ili ne
        */
        System.out.println("Welcome to java");
        System.out.println("Welcome to Computer Science");
        System.out.println("Programming is fun");
    }
```

PRIMER 6 - FORMATIRANJE KODA U RAZVOJNOM OKRUŽENJU

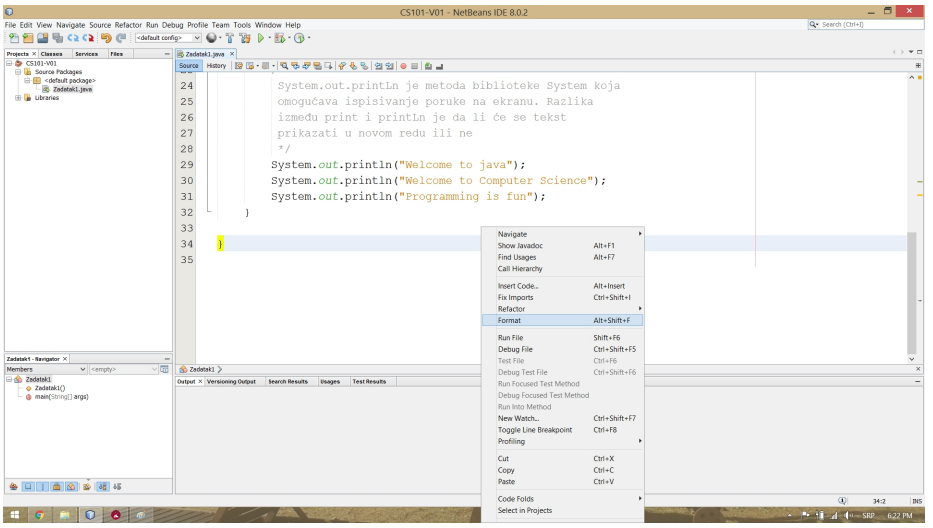
Nakon završetka pisanja koda svake klase, kod treba ispravno poravnati ili "nazubiti".

NAPOMENA:

Nakon završetka pisanja koda svake klase, kod treba ispravno poravnati ili "nazubiti". U NetBeansu postoji prečica Shift-Alt-F koja to za Vas automatski radi.

Ukoliko Vam prečica ne funkcioniše, desnim klikom na bilo kojoj liniji koda u klasi dobija se niz opcija. Format opcija formatira kod.

(slika 12)



Slika 4.12 Formatiranje koda u razvojnom okruženju

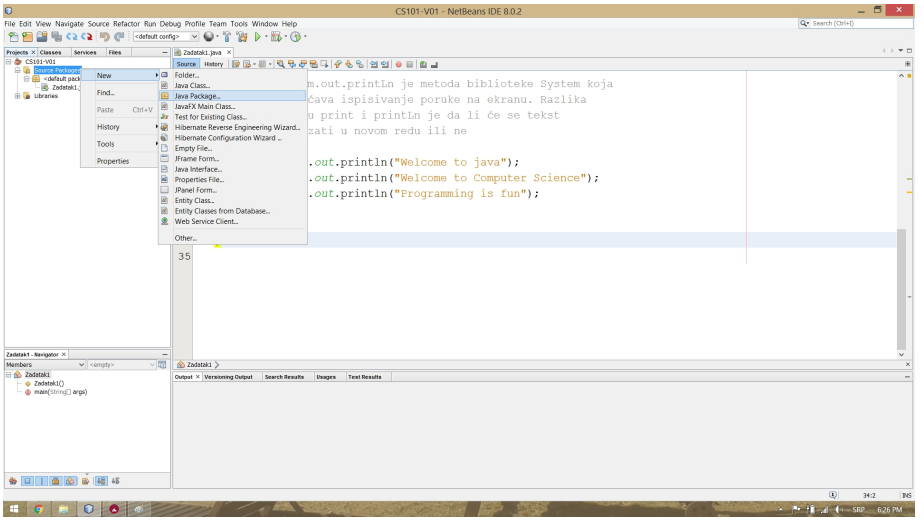
PRIMER 7 - KREIRANJE PAKETA

U okviru svakog projekta može postojati proizvoljni broj paketa

U prethodnom primeru smo koristiti podrazumevani paket u koji smo smestili Zadataka1 klasu.

Sada ćemo kreirati paket zadatak1 i u njega premestiti klasu Zadatak1.

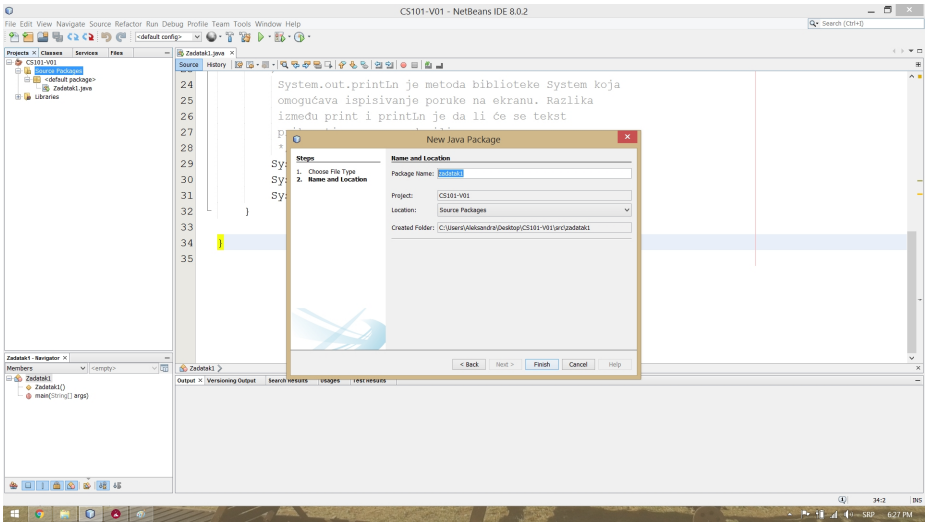
Desni klik na projekat, opsija New i odabir Package. (slika 13)



Slika 4.13 Kreiranje paketa u razvojnom okruženju

NAPOMENA:

Imena paketa se pišu malim slovom



Slika 4.14 Definisanje naziva paketa

Imenujemo projekat zadatak1 i klikom na Finish u levom meniju se pojavljuje pored podrazumevanog još jedan, nov paket.

Zadatak1 klasu prevlačimo mišem u novi projekat.

U Move Class meniju biramo opciju Refactor.

PRIMER 8 - PRIKAZIVANJE 5 PORUKA

Kreiranje klase Zadatak2.java i novog paketa

Kreirati novi paket unutar KI103-V01 koji će se zvati zadatak2.

U njemu treba da postoji pokretačka klasa Zadatak2.

Napisati program koji će da ispisuje Dobrodošli u Javu 5 puta.

Poželjno je korišćenje petlji. (kao naprednorešenje)

Objašnjenje:

Kao i u prethodnom primeru, program počinjemo definisanjem paketa i klase, a onda sam program pišemo u okviru metode main od koje počinje izvršavanje programa.

Rešenje:

```
package zadatak2;

/**
 *
 * @author Aleksandra
 */
public class Zadatak2 {
```

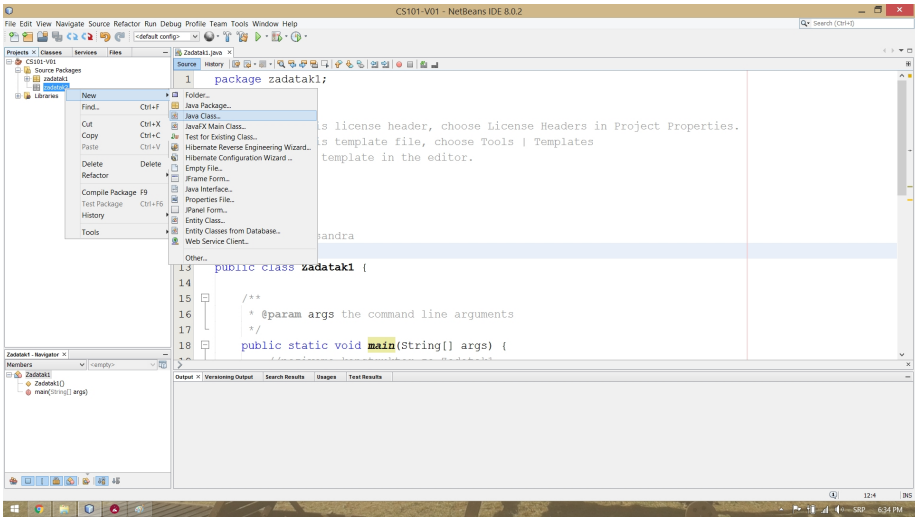
```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        new Zadatak2();
    }

    public Zadatak2() {
        System.out.println("Dobrodošli u Javu");
        System.out.println("Dobrodošli u Javu");
        System.out.println("Dobrodošli u Javu");
        System.out.println("Dobrodošli u Javu");
        System.out.println("Dobrodošli u Javu");
    }
}
```

PRIMER 8 - UPUTSTVO ZA RAD

Objašnjenje zadatka 2

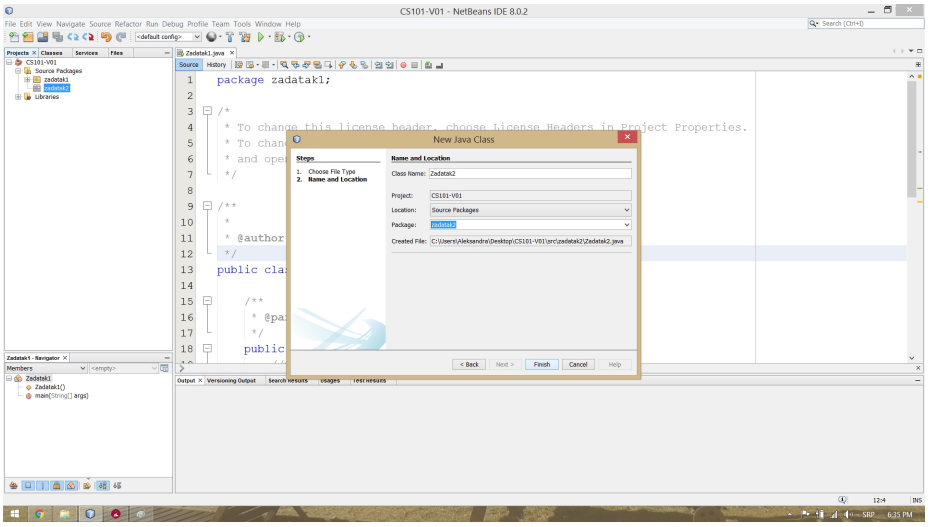
Nakon što smo kreirali paket zadatak2, kreiramo i klasu Zadatak2.

Desnim klikom na paket biramo opciju New -> Java class. (slika 15)



Slika 4.15 Kreiranje nove Java klase

Imenujemo klasu kao Zadatak2 i vodimo računa da li je smeštena u paketu zadatak2 kako smo želeli. (slika 16)



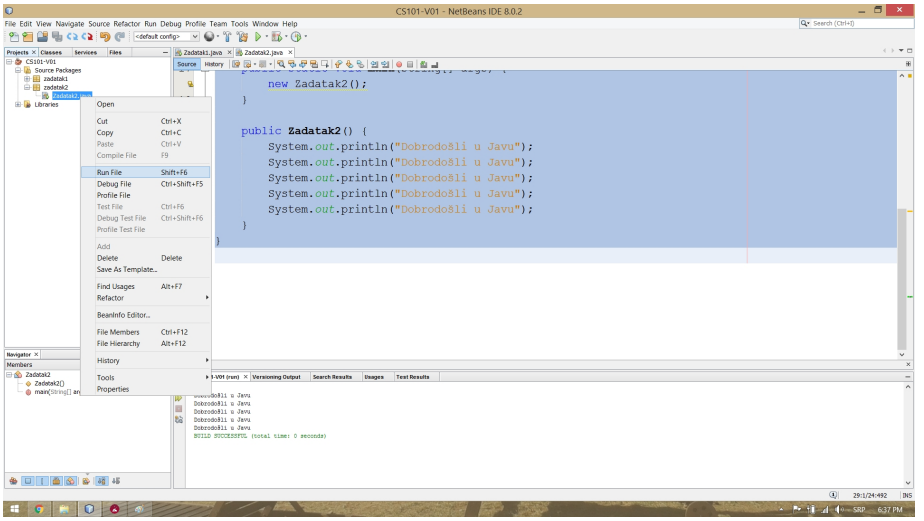
Slika 4.16 Kreiranje paketa

ZADATAK 8 - POKRETANJE KODA

Pokretanje java koda

NAPOMENA:

Program pokrećemo tako što u meniju sa leve strane na željenu klasu desnim klikom dobijamo meni iz koga biramo Run File opciju.
Rezultat rada programa se ispisuje u konzoli na dnu ekrana.
(slika 10)

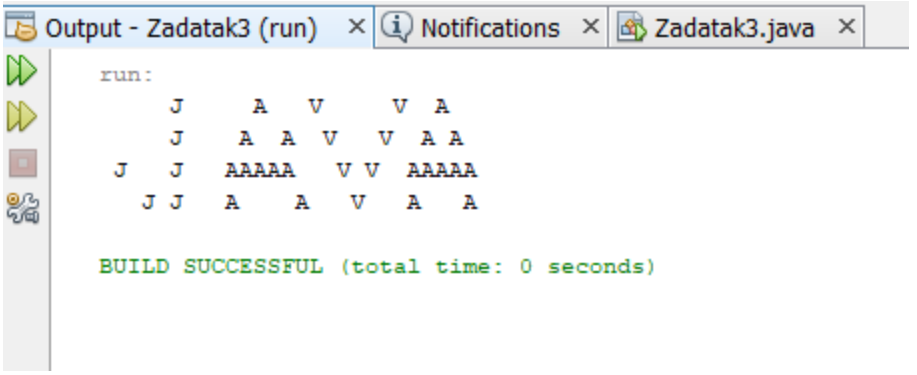


Slika 4.17 Pokretanje koda i ispisivanje rezultata

PRIMER 9 - PRIKAZIVANJE ZADATOG TEKSTA

Kreiranje klase Zadatak3.java

Kreirati novi paket unutar KI103-V01 koji će se zvatizadatak3 iu njemu treba da postoji pokretačka klasa Zadatak3. Napraviti program koji ispisuje sledeći tekst:



Slika 4.18 Očekivani izlaz na NetBeans monitoru

Objašnjenje:

Celokupna slika predstavlja kombinaciju štampanja 4 linije teksta sa odgovarajućim rasporedom karaktera, kako bi se formirao odgovarajući prikaz.

Ovde je prvi put korišćena metoda koja nosi isti naziva kao klasa. Ova metoda se naziva konstruktor i pomoću nje se kreiraju objekti klase. Kreiranjem objekta izvršava se kod definisan konstruktorom.

Rešenje:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package zadatak3;

/**
 *
 * @author Aleksandra
 */
public class Zadatak3 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        new Zadatak3();
```

```
    }

    public Zadatak3() {
        /*
         \r\n su karakteri koji nam omogućavaju da napravimo novi red
         u stringu
         */
        System.out.println("     J     A   V     V  A \r\n"
                + "     J    A  A  V   V  A A \r\n"
                + " J   J   AAAAA   V V  AAAAA\r\n"
                + "   J J   A    A   V   A   A\r\n");
    }
}
```

PRIMER 10 - REZULTAT MATEMATIČKOG IZRAZA

Kreiranje klase Zadatak4.java

Kreirati novi paket unutar KI103-V01 koji će se zvati zadatak4. U njemu treba da postoji pokretačka klasa Zadatak4. Napraviti program koji računa sledeći zadatak:

$$\frac{9.5 \times 4.5 - 2.5 \times 3}{45.5 - 3.5}$$

Objašnjenje:

U ovom primeru je prvi put upotrebljen tip podataka double**.** Tipovi podataka će biti razmatrani detaljno kasnije, ali je važno da znate da double predstavlja realni broj.

Kao rezultat operacije deljenje očekujemo realan broj, što i označavamo tako što vrednost promenljive result deklarišemo kao double tip.

> Prilikom deklaracije promenljive, svaka promenljiva mora imati svoj tip koji se navodi ispred imena. To znači da se u memoriji računara odvaja prostor od odgovarajućeg broja bajtova u koji se može smkladištiti neki podatak.

Rešenje:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package zadatak4;

/**
 *
 * @author Aleksandra
```

```
 */
public class Zadatak4 {

    public static void main(String[] args) {
        new Zadatak4();
    }

    public Zadatak4() {
        double result;
        result = ((9.5 * 4.5) - (2.5 * 3)) / (45.5 - 3.5);
        System.out.println("rezultat je = " + result);
    }
}
```

PRIMER 10 - OBJAŠNJENJE

Objašnjenje zadatka 4

Ono što je novo u ovom zadatku jesu sledeće linije koda:

```
double result;
 result = ((9.5 * 4.5) - (2.5 * 3)) / (45.5 - 3.5);
 System.out.println("rezultat je = " + result);
```

Objašnjenje:

linija

double result;
vrši deklaraciju promenljive result:
u memoriji računara rezerviše prostor od 8 bajtova i imenuje ga kao result

naredna linija
result = ((9.5 * 4.5) - (2.5 * 3)) / (45.5 - 3.5);

računa složeni aritmetički izraz i dobijeni rezultat skladišti u promenljivu result.

Drugim rečima, inicijalizuje se vrednost promenljive result.

Treća linija je zadužena da korisniku prikaže vrednost rezultata:
System.out.println("rezultat je = " + result);

Obratiti pažnju da result nije pod znacima navoda.

Probati da promenljiva result stoji pod navodnicima i uočiti razliku u prikazu rezultata nakon pokretanja programa.

ZADATAK 4

Instaliranje razvojnog okruženja i samostalno kreiranje prvog programa u NetBeans IDE

Izvršite sledeće zadatke:

1. Sa linka https://netbeans.org/ preuzmite instalacioni fajl za NetBeans IDE u zavisnosti od operativnog sistema na kojem bi trebalo da bude instaliran;
2. Instalirajte na vašem računaru NetBeans IDE;
3. Kreirajte vaš Java projekat pod nazivom: LicniPodaci;
4. Program pripada paketu licnipodaci;
5. Program bi trebalo da prikazuje sledeće poruke: Ime i prezime, adresa, mesto stanovanja i poštanski broj;
6. Pokrenite Vaš program iz razvojnog okruženja NetBeans IDE i pratite rezultat izvršavanja na konzoli.

UVOD U NETBEANS (VIDEO)

Daje se instrukcija za preuzimanje i za instaliranje NetBeans IDE na vašem računaru

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

HELLO WORLD UPOTREBOM NETBEANS (VIDEO)

Pokazuje se kreiranja programa Hello World primenom NetBeans IDE-

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 5

Domaći zadaci

ZADATAK 1

Kreiranje klase Zadatak5.java

Kreirati novi paket unutar KI103-V01 koji će se zvati domacizadatak1. U njemu treba da postoji pokretačka klasa DomaciZadatak1. Napraviti program koji računa matematičko Pi koristeći sledeću formulu:

4 * [1.0 – (1/3)+(1/5)-(1/7)+(1/9)-(1/11)]

ZADATAK 2

Kreiranje klase Zadatak6.java

Kreirati novi paket unutar KI103-V01 koji će se zvati domacizadatak2. U njemu treba da postoji pokretačka klasa DomaciZadatak2.

Napisati program koji prikazuje na ekranu površinu i obim kruga za radius od 5.5 koristeći sledeću formulu.

*Obim = 2*r*Pi.*

*Površina = radius * radius * Pi.*

Napomena:

Iako je u primeru za Pi upotrebljena direktna vrednost, to treba izbegavati jer Java sadrži biblioteku Math koja sadrži konstante poput PI, pa je češća i tačnija upotreba umesto 3.14159 koristiti *Math.PI. Pored Math biblioteke, Java sadrži mnoštvo drugih o čemu će biti reči kasnije.*

ZADATAK 3

Kreiranje klase Zadatak7.java

Kreirati novi paket unutar KI103-V01 koji će se zvati domacizadatak3. U njemu treba da postoji pokretačka klasa DomaciZadatak3.

Napraviti proram koji prikazuje površinu pravougaonika ako je visina 7.9 i širina 4.5.

Objašnjenje:

Primer je ekvivalent prethodnom primeru, osim formule po kojoj se računa tražena površina.

Zakljucak

ZAKLJUČAK

Rezime šta smo naučili u lekciji 01.

Savladavanjem ove lekcije polaznici su se upoznali sa osnovama Java programiranja. Polaznici su imali zadatak da se:

- upoznaju se sa programskim jezikom Java i njegovim komponentama;
- upoznaju se sa pojmovima JRE i JDK;
- nauče da instaliraju Javu i razvojno okruženje na vlastitim računarima;
- nauče da kreiraju i pokrenu jednostavne java programe na kontrolnoj liniji i u razvojnom okruženju;
- nauče da pišu programe koristeći komentare u programskom kodu;
- nauče da kreiraju Java programe u razvojnom okruženju u formi projekata, u odgovarajućem paketu.

Svako izlaganje je praćeno pažljivo biranim primerima, kao i zadacima za samostalni rad pomoću kojih polaznici mogu da provere praktično znanje stečeno savladavanjem demonstriranih primera.