



CS101 - UVOD U OBJEKTNO- ORIJENTISANO PROGRAMIRANJE

Uvod u Javu

Lekcija 01

PRIRUČNIK ZA STUDENTE

CS101 - UVOD U OBJEKTNO-ORIJENTISANO PROGRAMIRANJE

Lekcija 01

UVOD U JAVU

- ✓ Uvod u Javu
- ✓ Poglavlje 1: Specifikacija programskog jezika Java
- ✓ Poglavlje 2: Jednostvan program u Javi
- ✓ Poglavlje 3: Test za samotestiranje 1
- ✓ Poglavlje 4: Kreiranje, kompilacija i izvršavanje programa
- ✓ Poglavlje 5: Stil programiranja i dokumentacija
- ✓ Poglavlje 6: Greške programiranja
- ✓ Poglavlje 7: Razvijanje Java programa pomoću NetBeans-a
- ✓ Poglavlje 8: Test za samotestiranje 2
- ✓ Poglavlje 9: Pokazna vežba
- ✓ Poglavlje 10: Individualna vežba: Zadaci za vežbu
- ✓ Zaključak

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

CILJEVI LEKCIJE

Cilje lekcije je da skrene pažnju studenta na sintaksu, tj. na način kako se pišu programske instrukcije u Javi.

Ova lekcija ima sledeće ciljeve:

- Da opiše odnos između Jave i World Wide Veba
- Da razume značenje specifikacije jezika Java, API, JDK™, JRE™ i IDE (§1.6).
- Za pisanje jednostavnog Java programa (§1.7).
- Za prikaz izlaza na konzoli (§1.7).
- Da objasni osnovnu sintaksu Java programa (§1.7).
- Za kreiranje, kompajliranje i pokretanje Java programa (§1.8) Da pravilno koristite stil Java programiranja i dokumente (§1.9).
- Da objasne razlike između sintaksičkih grešaka, grešaka u izvršavanju i logičkih grešaka (§1.10)
- Za razvoj Java programa koristeći NetBeans™ (§1.11).

▼ Poglavlje 1

Specifikacija programskog jezika Java

SPECIFIKACIJA JAVE

Java je moćan i svestran programski jezik za razvoj softvera koji radi na mobilnim uređajima, desktop računarima i serverima.

Java je jednostavna, objektno orijentisana, distribuirana, interpretirana, robusna, bezbedna, arhitektonski neutralna, prenosiva, visokih performansi, višenitna i dinamična

Java je programski jezik opšte namene sa punim funkcijama koji se može koristiti za razvoj robusnih aplikacija kritičnih za misiju. Koristi se ne samo na desktop računarima, već i na serverima i mobilnim uređajima. Java je, međutim, sada veoma popularna za razvoj aplikacija na veb serverima. Ove aplikacije obrađuju podatke, vrše proračune i generišu dinamičke veb stranice. Mnoge komercijalne veb stranice su razvijene pomoću Jave na pozadini.

Java sintaksa je definisana u specifikaciji Java jezika, a Java biblioteka je definisana u interfejsu Java aplikacijskog programa (API). JDK je softver za kompajliranje i pokretanje Java programa. IDE je integrisano razvojno okruženje za programe koji se brzo razvijaju

Specifikacija Java jezika je tehnička definicija sintakse i semantike Java programskog jezika. Kompletnu specifikaciju Java jezika možete pronaći na docs.oracle.com/javase/specs/

Interfejs aplikacijskog programa (API), takođe poznat kao biblioteka, sadrži unapred definisane klase i interfejse za razvoj Java programa. API se i dalje širi. Najnoviju Java API dokumentaciju možete pogledati na <https://docs.oracle.com/en/java/javase>.

Java je punopravan i moćan jezik koji se može koristiti na mnogo načina. Dolazi u tri izdanja

- Java Standard Edition (**Java SE**) za razvoj aplikacija na strani klijenta. Aplikacije mogu da rade na desktopu.
- Java Enterprise Edition (**Java EE**) za razvoj aplikacija na strani servera, kao što su Java servleti, JavaServer Pages (JSP) i JavaServer Faces (JSF).
- Java Micro Edition (**Java ME**) za razvoj aplikacija za mobilne uređaje, kao što su mobilni telefoni.

JDK I JRE

JDK se sastoji od skupa zasebnih programa, od kojih se svaki poziva iz komandne linije, za kompajliranje, pokretanje i testiranje Java programa.

U ovom predmetu izučavamo korišćenje Java SE za uvođenje Java programiranja. Java SE je osnova na kojoj se zasnivaju sve druge Java tehnologije. Postoji mnogo verzija Java SE. Najnovija, Java SE 11 (ili jednostavno Java 11), je verzija koju koristimo u ovom predmetu. Oracle izdaje svaku verziju sa **Java Development Toolkit-om (JDK)**. Za Javu 11, **Java Development Toolkit** se zove JDK 11.

JDK se sastoji od skupa zasebnih programa, od kojih se svaki poziva iz komandne linije, za kompajliranje, pokretanje i testiranje Java programa. Program za pokretanje Java programa poznat je kao Java Runtime Environment (JRE).

Umesto korišćenja JDK-a, možete da koristite **Java razvojni alat** (npr. NetBeans, Eclipse i TektPad)—softver koji obezbeđuje integrisano razvojno okruženje (IDE) za brz razvoj Java programa. Uređivanje, kompajliranje, pravljenje, otklanjanje grešaka i pomoć na mreži integrisani su u jedan grafički korisnički interfejs. *Jednostavno unesete izvorni kod u jedan prozor ili otvorite postojeću datoteku u prozoru, a zatim kliknete na dugme ili stavku menija ili pritisnete funkcijski taster da biste kompajlirali i pokrenuli program.*

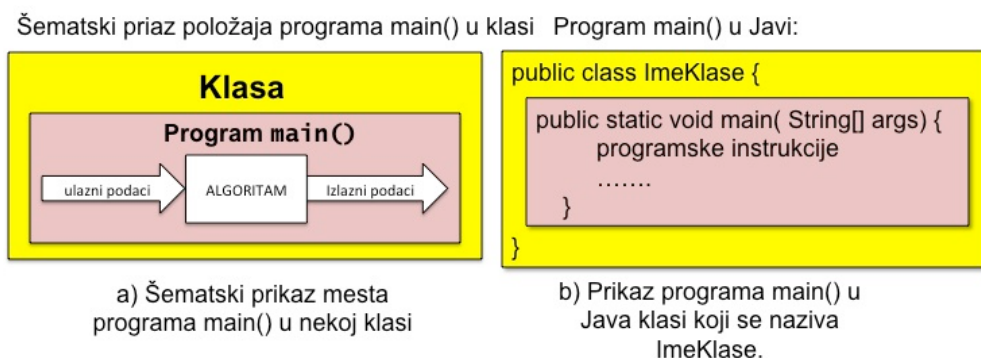
▼ Poglavlje 2

Jednostvan program u Javi

METOD MAIN ()

U ovoj i nekoliko narednih lekcija, koristiće metod main, da bi napisali jednostavni Java programi u primerima i zadacima koje ćemo sada kostiti

Java je objektno-orijentisan programski jezik. **Klasa** definiše strukturu objekata (koji sadrže podatke, a klasa definiše tipove podataka kojenjeni objekti koriste i koji se nazivaju **atributima**. **Klasa** sadrži i tzv. **metode**, tj. male programe koji koriste podatke definisane atributima klase. O klasama će biti mnogo više reči u kasnijim lekcijama. Za sada, nas ovde interesuje samo jedan metod, i to tzv., glavni metod koji se naziva **main**. U njemu možemo da napišemo jednostavne Java program u kome želimo da prikažemo način pisanja programa u Javi (slika 2.1)



Slika 2.1 Metod main() definisan u jednoj klasii

Jedan objektno-orijentisan program, najčešće koristi veći broj objekata koje definiše više klasa, i jedan metod **main** (). Međutim, samo JEDNA klasa može da sadrži metod **main()**, jer iz njega staruje ceo program.

LISTING KLASA WELCOME SA PORUKOM WELCOME TO JAVA


Svaki Java program mora imati najmanje jednu klasu i glavni metod main.

Java program se izvršava iz glavne metode u klasi koja se naziva **main**..

Počnimo sa jednostavnim Java programom koji prikazuje poruku "Welcome to Java"! na konzoli. (Reč konzola je stari računarski termin koji se odnosi na uređaj za unos teksta i prikaz na računaru. Ulaz konzole znači primanje unosa sa tastature, a izlaz konzole prikazivanje izlaza na monitoru.) Program je dat u Listingu. 1.1

```
1 public class Welcome {  
2     public static void main(String[] args) {  
3         // Display message Welcome to Java! on the console  
4         System.out.println("Welcome to Java!");  
5     }  
6 }
```

klasa
main metod
komentar
prikaz poruke



Slika 2.2 Listing 2.1 klase Welcome sa porukom "Welcome to Java"

Imajte na umu da su brojevi redova samo u referentne svrhe; oni nisu deo programa. Dakle, nemojte unositi brojeve redova u svoj program.

Linija 1 definiše klasu. Svaki Java program mora imati najmanje jednu klasu. Svaka klasa ima ime. Po konvenciji, imena klasa počinju velikim slovom. U ovom primeru, ime klase je Welcome

Linija 2 definiše glavni metod. Program se izvršava iz glavnog metoda. Klasa može da sadrži nekoliko metoda. Glavni metod je ulazna tačka gde program počinje da se izvršava.

OBJAŠNJENJE LISTINGA KLASSE WELCOME

Ključne reči imaju specifično značenje za kompajler i ne mogu se koristiti u druge svrhe u programu

Metod je konstrukcija koja sadrži iskaze. Glavni metod u ovom programu sadrži naredbu **System.out.println**. Ova izjava prikazuje string "Welcome to Java!" na konzoli (red 4). String je programski termin koji označava niz znakova. String mora biti stavljen u dvostruke navodnike. Svaka izjava u Javi završava se tačkomi zarezom (;), poznatom kao terminator iskaza.

Ključne reči imaju specifično značenje za kompajler i ne mogu se koristiti u druge svrhe u programu. Na primer, kada prevodilac vidi reč *class*, razume da je reč posle klase naziv za klasu. Ostale ključne reči u ovom programu su javne, statične i nevažne.

Treći red je komentar koji dokumentuje šta je program i kako je konstruisan.

Komentari pomažu programerima da komuniciraju i razumeju program. Oni nisu programski izrazi, pa ih kompajler ignoriše. U Javi, komentarima prethode dve kose crte (//) na liniji, koje se nazivaju linijski komentar, ili su zatvorene između /* i */ na jednom ili više redova, što se naziva blok komentar ili komentar paragrafa. Kada kompajler vidi //, ignoriše sav tekst posle // u istom redu. Kada vidi /*, skenira sledeći */ i ignoriše svaki tekst između /* i */. Evo primera komentara

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

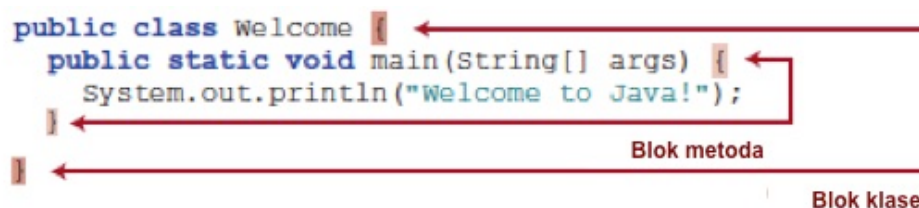
```
/* This application program  
displays Welcome to Java! */
```

SPECIFIKACIJA BLOKOVA ISKAZA KLASA I METODA

Par zagrada u programu formira blok koji grupiše komponente programa

Par zagrada u programu formira blok koji grupiše komponente programa. U Javi, svaki blok počinje otvorom ({) i završava se zagradom (}). Svaka klasa ima blok klase koji grupiše podatke i metode klase. Slično tome, svaki metod ima blok metoda koji grupiše iskaze u metodi. Blokovi mogu biti ugnežđeni, što znači da se jedan blok može staviti u drugi, kao što je prikazano u sledećem kodu:

```
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



Slika 2.3 Spoljni blok (klase) i unutražnji blok (metoda)

Otvarajuća zagrada mora biti usklađena sa zagradom koja zatvara blok. Kad god otkucate početnu zagradu, odmah otkucajte zagradu da biste sprečili grešku koja nedostaje. Većina Java IDE automatski umeće završnu zagradu za svaku otvornu zagradu. Java izvorni programi razlikuju velika i mala slova. Bilo bi pogrešno, na primer, zameniti main u programu sa Main.

ZAGRADE I DRUGI ZNACI

Velike yagrade oynačavaju programski blok, male, argumente metoda, a srednje - nizove podataka.

Znaci	Naziv	Opis
{ }	Otvorena i zatvorena velika zagrada	Oznaka jednog bloka
()	Otvorena i zatvorena mala zagrada	Upotreba u metodima
[]	Otvorena i zatvorena srednja zagrada	Označava niz
//	Dvostruka kosa crta	Prethodi komentaru
" "	Otvareni i zatvarani znaci navoda	Određuje jedan string
;	Tačka i zapeta	Označava kraj naredbe

Slika 2.4 Značenje zagrada i drugih znakova u Java programu

PRIMER SA TRI PORUKE

Svaka programska linija se mora da završi sa znakom "tačka-zarez", tj sa ;

Kao i svaki programski jezik, Java ima sopstvenu sintaksu i potrebno je da napišete kod koji je u skladu sa pravilima sintakse. Ako vaš program krši pravilo — na primer, ako nedostaje tačka-zarez, nedostaje zagrada, nedostaje navodnik ili je reč pogrešno napisana — Java kompajler će izvesti sintaksičke greške. Pokušajte da kompajlirate program sa ovim greškama i vidite šta prevodilac izveštava.

Evo primera koji prikazuje tri poruke:

```
1 public class WelcomeWithThreeMessages {  
2     public static void main(String[] args) {  
3         System.out.println("Programming is fun!");  
4         System.out.println("Fundamentals First");  
5         System.out.println("Problem Driven");  
6     }  
7 }
```

```
Programming is fun!  
Fundamentals First  
Problem Driven
```

Slika 2.5 Primer sa tri poruke

PRIMER SA MATEMATIČKIM IZRAZOM

Metod println pomera na početak sledećeg reda posle prikaza stringa, ali print ne prelazi na sledeći red kada se završi.

Evo programa koji obračunava vrednost određena sledećim matematičkim iskazom:

$$\frac{10.5 + 2 \times 3}{45 - 3.5}$$

Slika 2.6 Matematički izraz

```
1 public class ComputeExpression {  
2     public static void main(String[] args) {  
3         System.out.print("(10.5 + 2 * 3) / (45 - 3.5) = ");  
4         System.out.println((10.5 + 2 * 3) / (45 - 3.5));  
5     }  
6 }
```

```
(10.5 + 2 * 3) / (45 - 3.5) = 0.39759036144578314
```

Slika 2.7 Primer sa matematičkim izrazom

Metod štampanja u redu 3 je identičan metodi **println** osim što se *println* pomera na početak sledećeg reda nakon prikaza stringa, ali *print* ne prelazi na sledeći red kada se završi. Operator množenja u Javi je *. Kao što vidite, prevođenje aritmetičkog izraza u Java izraz je jednostavan proces. Dalje ćemo razgovarati o Java izrazima u lekciji 2..

Ako kliknete na ovaj link, moćićete ovaj program da isvršite, a i da ga animaciom, bolje razumete: <https://liveexample.pearsoncmg.com/liang/intro12e/html/ComputeExpression.html>

▼ Poglavlje 3

Test za samotestiranje 1

PRVI OD DVA TESTA ZA SAMOTESTIRANJE STUDENTA

Testom proveravate vaše razumevanje do sada izloženog gradiva.

Cilj ovog testa je da proverite sebe, da li ste dobro shvatili izloženo gradivo lekcije do ovog momenta. Test se ne ocenjuje i ne donosi dodatne poene studentu. Koristimo originalni test koji je autor udžbenika pripremio za svako poglavlje. Ako imate problem sa engeskim jezikom, možete se koristiti i sa Google Translate aplikacijom. U ovom testu, odgovorite na pitanja od broja 1.17 do 1.28.

Ovo će vam pomoći da bolje savladate pređeno gradivo do ovog momenta, jer vam test posle svakog pitanja i vašeg odgovora, daje informaciju da li ste dali tačan odgovor i mogućnost da vam prikaže tačan odgovor.

Da bi aktivirali test, kliknite na sledeći link:

[https://liveexample-ppe.pearsoncmg.com/selftest/
selftest12e?chapter=1&username=liang12e](https://liveexample-ppe.pearsoncmg.com/selftest/selftest12e?chapter=1&username=liang12e)

▼ Poglavlje 4

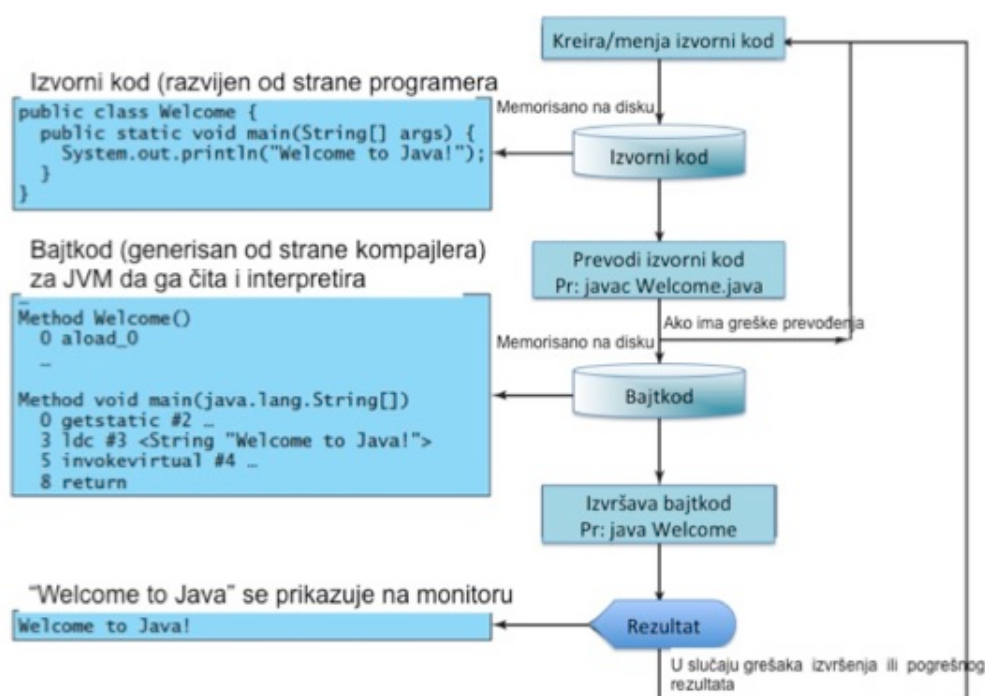
Kreiranje, kompilacija i izvršavanje programa

PROCES PRIPREME, KOMPILACIJE I IZVRŠENJA JAVA PROGRAMA

Java program čuvate u .java fajlu i kompajlirate ga u .class fajl. Datoteku .class izvršava Java virtuelna mašina (JVM).

Morate kreirati svoj program i kompajlirati ga pre nego što se može izvršiti. Ovaj proces se ponavlja, kao što je prikazano na slici 4.1. Ako vaš program ima greške pri kompajliranju, morate da izmenite program da biste ih popravili, a zatim ga ponovo kompajlirajte. Ako vaš program ima greške tokom izvođenja ili ne daje tačan rezultat, morate da izmenite program, ponovo ga kompajlirate i ponovo izvršite. Možete koristiti bilo koji uređivač teksta ili IDE za kreiranje i uređivanje Java datoteke izvornog koda.

Izvršavanje Java programa znači pokretanje bajt koda programa. **Bajtkod** možete izvršiti na bilo kojoj platformi sa **JVM**-om, koji je tumač. On prevodi pojedinačne instrukcije u bajtkodu u kod ciljnog mašinskog jezika jednu po jednu, a ne ceo program kao jednu celinu. Svaki korak se izvršava odmah nakon što je preveden.



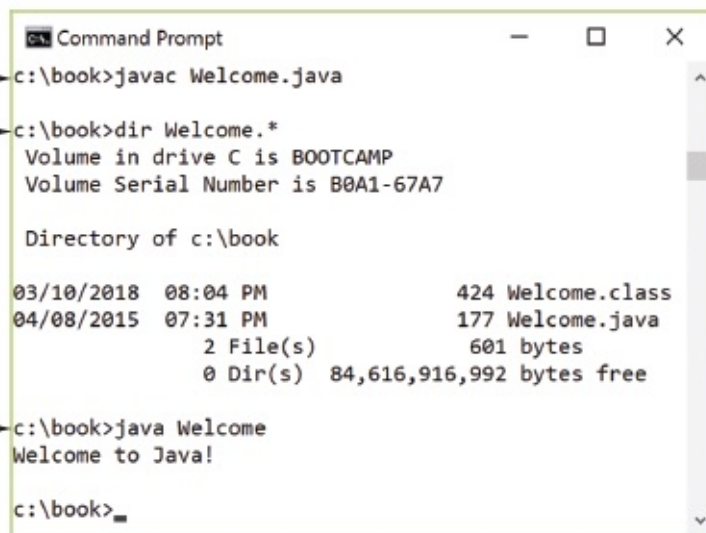
Slika 4.1 Proces pripreme, kompilacije i izvršenja Java programa

KOMPILACIJA WELCOME.JAVA I IZVRŠAVANJE JAVA.CLASS BAJKODA

Kompilacija java programa tkriva sintaktičke greške, ako ih ima, i kreira bajtkod.

Sledeća komanda pokreće bajt kod za **listing 2.1:**
java Welcome

Slika 4.2 prikazuje komandu **javac** za kompajliranje **Welcome.java**. Kompajler generiše datoteku **Welcome.class**, a ovaj fajl se izvršava pomoću java komande.



```
Command Prompt
c:\book>javac Welcome.java
c:\book>dir Welcome.*
Volume in drive C is BOOTCAMP
Volume Serial Number is B0A1-67A7

Directory of c:\book

03/10/2018  08:04 PM                424 Welcome.class
04/08/2015  07:31 PM                177 Welcome.java
               2 File(s)                601 bytes
               0 Dir(s)  84,616,916,992 bytes free

c:\book>java Welcome
Welcome to Java!

c:\book>
```

Slika 4.2 Kompilacija Welcome.java i izvršavanje Welcome.class fajlova

IZVRŠAVANJE BAJKOD FAJLA

U JDK 11, možete koristiti `java ClassName.java` za kompajliranje i pokretanje programa koda sa jednom datotekom

Nemojte koristiti ekstenziju **.class** u komandnoj liniji kada izvršavate program. Koristite **java ClassName** za pokretanje programa. Ako koristite `java ClassName.class` u komandnoj liniji, sistem će pokušati da preuzme `ClassName.class.class`.

U JDK 11, možete koristiti **java ClassName.java** za kompajliranje i pokretanje programa koda sa jednom datotekom. Ova komanda kombinuje kompajliranje i pokretanje u jednoj komandi. Program izvornog koda sa jednom datotekom sadrži samo jednu klasu u datoteci. To je slučaj sa svim našim programima u prvih osam lekcija.

Ako izvršite datoteku klase koja ne postoji, pojaviće se greška **NoClassDefFoundError**. Ako izvršite datoteku klase koja nema glavni metod ili pogrešno otkucate glavni metod (npr. otkucavanjem `Main` umesto `main`), pojaviće se greška **NoSuchMethodError**.

Prilikom izvršavanja Java programa, JVM prvo učitava bajt-kod klase u memoriju koristeći program koji se naziva učitavač klase. Ako vaš program koristi druge klase, učitavač klase ih dinamički učitava neposredno pre nego što su potrebne. Nakon što se klasa učitava, JVM koristi program koji se zove verifikator bajtkoda da proveri validnost bajtkoda i da osigura da bajtkod ne krši Javina bezbednosna ograničenja. Java sprovodi strogu bezbednost kako bi se osiguralo da datoteke Java klase nisu neovlašćene i da ne štete vašem računaru.

▼ Poglavlje 5

Stil programiranja i dokumentacija

KOMENTARI I STILOVI KOMENTARA

Dobar stil programiranja i odgovarajuća dokumentacija čine program lakim za čitanje i pomažu programerima da spreče greške.

Stil programiranja se bavi time kako programi izgledaju. Program može da kompajlira i radi ispravno čak i ako je napisan u samo jednom redu, ali pisanje svega u jednom redu bi bio loš stil programiranja jer bi ga bilo teško čitati. Dokumentacija je skup napomena i komentara koji se odnose na program.

Uključite rezime na početak programa koji objašnjava šta program radi, njegove ključne karakteristike i sve jedinstvene tehnike koje koristi. U dugom programu takođe treba da uključite komentare koji uvode svaki glavni korak i objašnjavaju sve što je teško pročitati. Važno je da komentari budu sažeti kako ne bi stvarali gužvu u programu ili otežali čitanje.

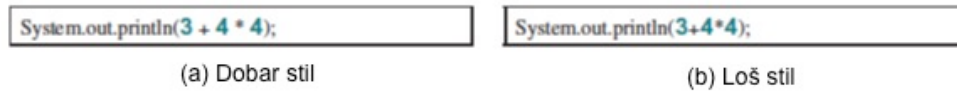
Pored komentara na red (koji počinju sa //) i blok komentara (koji počinju sa /*), Java podržava komentare posebnog tipa, koji se nazivaju javadoc komentari. javadoc komentari počinju sa /** i završavaju se sa */. Mogu se izdvojiti u HTML datoteku pomoću JDK-ove javadoc komande. Za više informacija pogledajte dodatni **javadocscomments.pdf** koje nalazi na LAMS-u, u okviru ove, prve lekcije. Koristite javadoc komentare (/** . . . */) za komentarisanje cele klase ili cele metode.

RAZMACI OKO OPERATORA I STILOVI PROGRAMSKIH BLOKOVA

Ovaj predmet koristi stil kraja reda (linije) kako bi bio u skladu sa izvornim kodom Java API-ja.

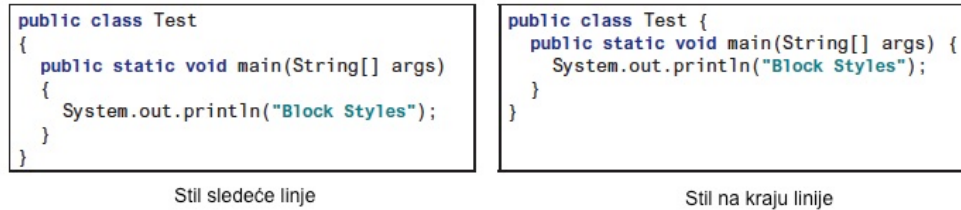
Dosledan stil uvlačenja čini programe jasnim i lakim za čitanje, otklanjanje grešaka i održavanje. **Uvlačenjelinija** se koristi da ilustruje strukturne odnose između komponenti ili izjava programa. Java može da čita program čak i ako su sve izjave na istoj dugačkoj liniji, ali ljudima je lakše da čitaju i održavaju kod koji je ispravno poravnat. Uvucite svaku podkomponentu ili izraz najmanje dva razmaka više od konstrukcije u kojoj je ugnežđena.

Treba dodati jedan razmak sa obe strane binarnog operatora, kao što je prikazano u (a), umesto u (b).



Slika 5.1 Dodavanje raymaka pre i posle operatora

Blok je grupa izjava okruženih zagradama. Postoje dva popularna stila, stil sledeće linije i stil kraja reda, kao što je prikazano u nastavku.



Slika 5.2

Stil sledećeg reda poravnava zagrade vertikalno i čini programe lakim za čitanje, dok **stil kraja reda** štedi prostor i može pomoći da se izbegnu neke suptilne programske greške. Oba su prihvatljiva blok stilova. Izbor zavisi od ličnih ili organizacionih preferencija. **Trebalo bi da dosledno koristite blok stil — ne preporučuje se mešanje stilova.** *Ovajpredmet koristi stil kraja reda kako bi bio u skladu sa izvornim kodom Java API-ja.*

▼ Poglavlje 6

Greške programiranja

SINTAKSIČKE GREŠKE

Greške u sintaksi su rezultat grešaka u konstrukciji koda (pisanju programa)

Greške koje detektuje kompajler nazivaju se sintaksičke greške ili greške kompajliranja. Greške u sintaksi su rezultat grešaka u konstrukciji koda, kao što je pogrešno upisivanje ključne reči, izostavljanje neke neophodne interpunkcije ili korišćenje početne zagrade bez odgovarajuće zagrade.

Prijavljene su četiri greške, ali program zapravo ima dve greške:

- Ključna reč **void** nedostaje ispred main u redu 2.
- String "Welcome to Java" treba da bude zatvoren završnim navodnikom u redu 3.

Pošto jedna greška često prikazuje mnogo redova grešaka pri kompajliranju, dobra je praksa da se greške ispravljaju od gornjeg reda i rade nadole. Ispravljanje grešaka koje se javljaju ranije u programu takođe može da ispravi dodatne greške koje se javljaju kasnije.

```
1 public class ShowSyntaxErrors {  
2     public static main(String[] args) {  
3         System.out.println("Welcome to Java);  
4     }  
5 }
```

Slika 6.1 Demonstracija prisustva sintaksičkih grešaka

GREŠKE TOKOM IZVRŠAVANJA PROGRAMA

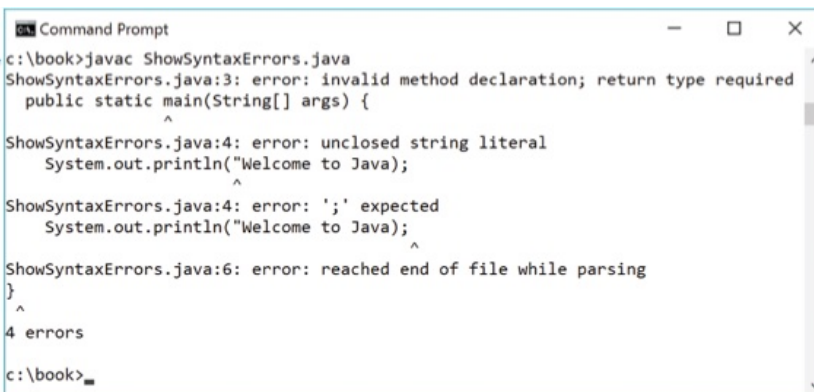
One se javljaju dok je program pokrenut ako okruženje otkrije operaciju koju je nemoguće izvršiti.

Greške tokom izvršavanja su greške koje uzrokuju nenormalan završetak programa. One se javljaju dok je program pokrenut ako okruženje otkrije operaciju koju je nemoguće izvršiti.

Greške u unosu obično uzrokuju greške u izvršavanju. Greška u unosu se javlja kada program čeka da korisnik unese vrednost, ali korisnik unese vrednost koju program ne može da obradi. Na primer, ako program očekuje da će pročitati broj, ali umesto toga korisnik unese string, to dovodi do pojave grešaka tipa podataka u programu.

Još jedan primer grešaka tokom izvršavanja je deljenje sa nulom. Ovo se dešava kada je delilac jednak nuli za celobrojne podele. Na primer, program u Listingu 1.5 bi prouzrokovao a

Kompilacija
(prevođenje) →



```
c:\book>javac ShowSyntaxErrors.java
ShowSyntaxErrors.java:3: error: invalid method declaration; return type required
public static main(String[] args) {
      ^
ShowSyntaxErrors.java:4: error: unclosed string literal
    System.out.println("Welcome to Java);
                        ^
ShowSyntaxErrors.java:4: error: ';' expected
    System.out.println("Welcome to Java);
                        ^
ShowSyntaxErrors.java:6: error: reached end of file while parsing
}
^
4 errors
c:\book>
```

Slika 6.2 Izveštaj o sintaksicčkim greškama dobijenij kompilacijom ShowSyntaxErrors.java programa

GREŠKE PRILIKOM IZVRŠAVANJA PROGRAMA

Oni se javljaju dok je program pokrenut ako okruženje otkrije operaciju koju je nemoguće izvršiti

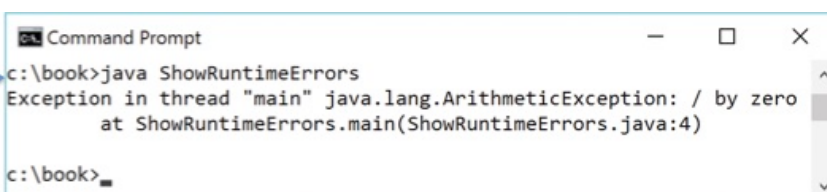
Greške tokom izvršavanja su greške koje uzrokuju nenormalan završetak programa. Oni se javljaju dok je program pokrenut ako okruženje otkrije operaciju koju je nemoguće izvršiti. Greške u unosu obično uzrokuju greške u izvršavanju.

Greška u unosu se javlja kada program čeka da korisnik unese vrednost, ali korisnik unese vrednost koju program ne može da obradi. Na primer, ako program očekuje da će pročitati broj, ali umesto toga korisnik unese string, to dovodi do pojave grešaka tipa podataka u programu.

Još jedan primer grešaka tokom izvršavanja je deljenje sa nulom. Ovo se dešava kada je delilac jednak nuli za celobrojne podele, kao što je to slučaj u sledećem listingu:

```
public class ShowRuntimeErrors {
    public static void main(String[] args) {
        System.out.println(1 / 0);
    }
}
```

Izvršenje
programa →



```
c:\book>java ShowRuntimeErrors
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at ShowRuntimeErrors.main(ShowRuntimeErrors.java:4)
c:\book>
```

Slika 6.3 Prilikom izvršenja klase ShowRunTimeErrors dobija se izveštaj o grešci izvršenja zbog deljenja sa 0

Ako kliknete na ovaj link, moćićete ovaj program da isvršite, a i da ga animaciom, bolje razumete:

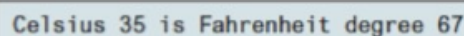
<https://liveexample.pearsoncmg.com/liang/intro12e/html/ShowRuntimeErrors.html>

LOGIČKE GREŠKE

Logičke greške nastaju kada program ne radi onako kako je trebalo.

Logičke greške nastaju kada program ne radi onako kako je trebalo. Greške ove vrste nastaju iz mnogo različitih razloga. Na primer, pretpostavimo da ste napisali program u Listingu 1.6 za konverziju Celzijusa od 35 stepeni u stepen Farenhajta:

```
1 public class ShowLogicErrors {  
2     public static void main(String[] args) {  
3         System.out.print("Celsius 35 is Fahrenheit degree ");  
4         System.out.println((9 / 5) * 35 + 32);  
5     }  
6 }
```



Slika 6.4 Demonstracija javljanja logičke greške

Dobićete 67 stepeni Farenhajta, što je pogrešno. Trebalo bi da bude **95.0**. U Javi, deljenje celih brojeva je količnik—razlomak je skraćen—tako da je u Javi $9 / 5 = 1$. Da biste dobili tačan rezultat, potrebno je da koristite $9.0 / 5$, što rezultira **1.8**.

Uopšteno govoreći, sintaksičke greške je lako pronaći i lako ih je ispraviti jer kompajler daje indikacije o tome odakle su greške došle i zašto su pogrešne. Takođe nije teško pronaći greške u toku rada, jer se razlozi i lokacije grešaka prikazuju na konzoli kada se program prekine. Pronalaženje logičkih grešaka, s druge strane, može biti veoma izazovno. U narednim lekcijama naučićete tehnike praćenja programa i pronalaženja logičkih grešaka.

ČESTE GREŠKE

Nedostaje zagrada, nedostaje tačka i zarez, nedostaju navodnici za stringove i pogrešno napisana imena su uobičajene greške za nove programere.

Česta greška 1: Nedostaje zagrada

Zagrade se koriste za označavanje bloka u programu. Svaka otvorena zagrada mora biti usklađena sa zatvaračem. Česta greška je nedostatak zagrade. Da biste izbegli ovu grešku, otkucajte zagradu kad god se otkuca početna zagrada, kao što je prikazano u sledećem primeru:

```
public class Welcome {
```

← Desna zagrada zatvaranja bloka je isuviše udaljena o zagrade otvaranja bloka.

Slika 6.5 Desna zagrada zatvaranja bloka je isuviše udaljena od leve zagrade otvaranja bloka

Ako koristite IDE kao što su NetBeans i Eclipse, IDE automatski ubacuje završnu zgradu za svaku otkucanu početnu zgradu.

Česta greška 2: nedostaje tačka-zapeta (;)

Svaka izjava se završava terminatorom izraza (;). Često, novi programer zaboravlja da stavi terminator izraza za poslednju naredbu u blok, kao što je prikazano u sledećem primeru:

```
public static void main(String[] args) {  
    System.out.println("Programming is fun!");  
    System.out.println("Fundamentals First");  
    System.out.println("Problem Driven")  
}
```

↑
Nedostaje tačka sa zapetom

Slika 6.6 Greška- nedostaje na kraju linije tačka sa zapetom

ČESTE GREŠKE (NASTAVAK)

Nedostaje zagrada, nedostaje tačka i zarez, nedostaju navodnici za stringove i pogrešno napisana imena su uobičajene greške za nove programere

Uobičajena greška 3: nedostaju navodnici

Niz se mora staviti unutar navodnika. Često, novi programer zaboravlja da stavi navodnik na kraj stringa, kao što je prikazano u sledećem primeru:

```
System.out.println("Problem Driven ");
```

↑
Nedostaje znak navoda

Slika 6.7 Nedostaj znak navoda na kraju

Ako koristite IDE kao što su NetBeans i Eclipse, IDE automatski ubacuje završni navodnik za svaki otkucani početni navodnik.

Česta greška 4: pogrešno napisana imena

Java je osetljiva na velika i mala slova. Pogreška u pisanju imena je uobičajena greška za nove programere. Na primer, reč main je pogrešno napisana kao Main, a String je pogrešno napisana kao string u sledećem kodu:

```
public class Test {  
    public static void Main(string[] args) {  
        System.out.println((10.5 + 2 * 3) / (45 - 3.5));  
    }  
}
```

Slika 6.8 Pogrešno napisan naziv

▼ Poglavlje 7

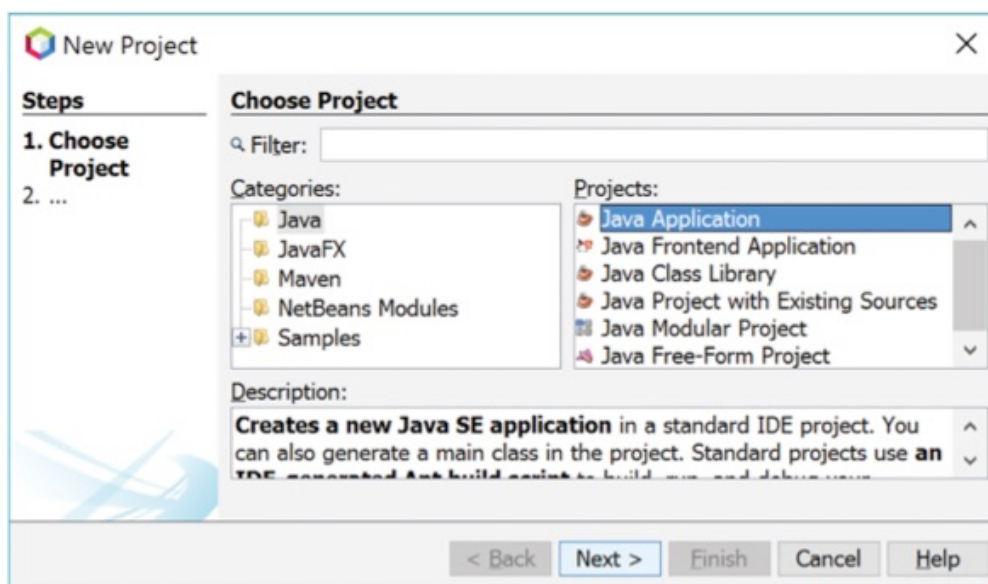
Razvijanje Java programa pomoću NetBeans-a

KREIRANJE JAVA PROJEKTA

Dijalog New Project kreira folder Project u kome se nalaze Java programi i prateći fajlovi.

Pre nego što budete mogli da kreirate Java programe, prvo morate da kreirate projekat. Projekat je poput fascikle u kojoj se nalaze Java programi i svi prateći fajlovi. Potrebno je da kreirate projekat samo jednom. Evo koraka za kreiranje Java projekta:

1. Izaberite **File, New Project** da biste prikazali okvir za dijalog New Project, kao što je prikazano na slici 7.1.



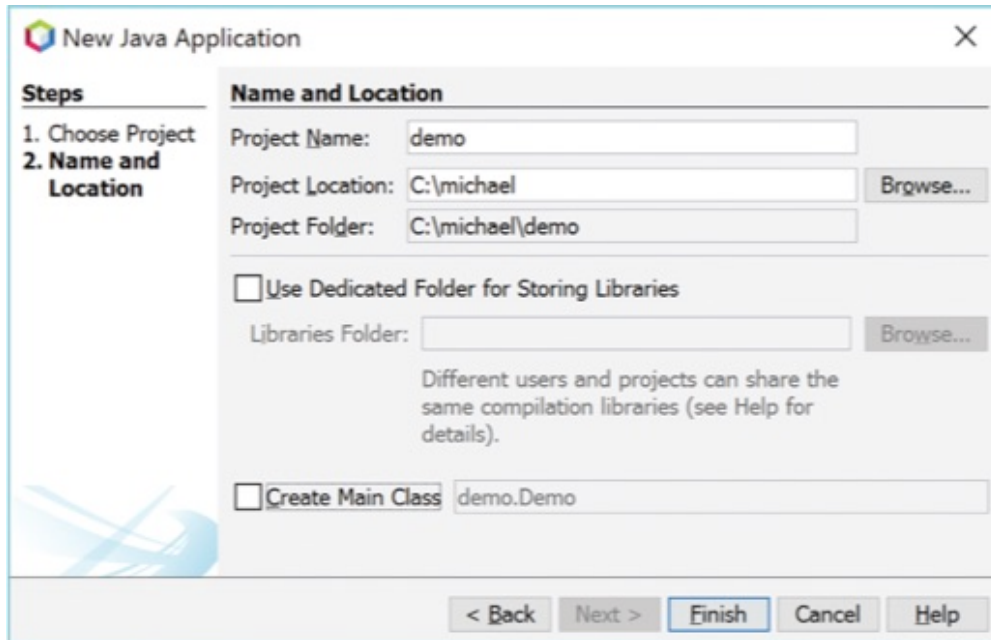
Slika 7.1 Dijalog New Projekt se koristi za kreiranje novog projekta i određivanje tipa projekta

KREIRANJE NOVE JAVA APLIKACIJE

Dijalog New Java Application služi za određivanje naziva projekta i lokacije

2. Izaberite Java u odeljku Categories i **Java Application** u odeljku Projects, a zatim kliknite na Next da biste prikazali dijalog New Java Application, kao što je prikazano na slici 7.2

3. Unesite demo u polje New Project i c:\michael u polje Project Location. Opozovite izbor opcije Use Dedicated Folder for Storing Libraries i opozovite izbor opcije Create Main Class

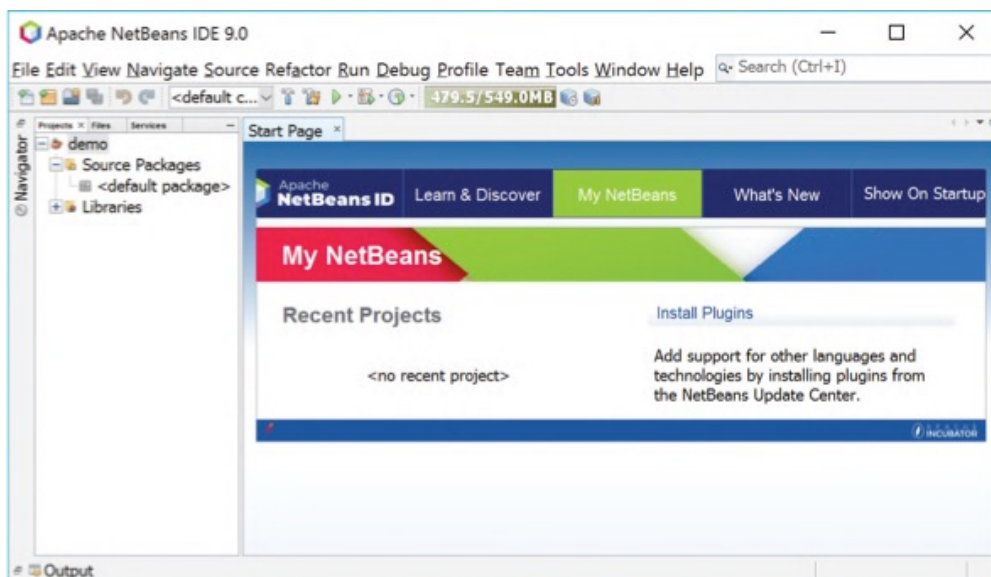


Slika 7.2 Dijalog New Java Application služi za određivanje naziva projekta i lokacije

KRAJ KREIRANJA NOVOG PROJEKTA

Klikom na Finish, kreiran je novi projekat pod nazivom demo.

4. Kliknite na **Finish** da završite kreiranje projekta, kao što je prikazano na slici 7.3.



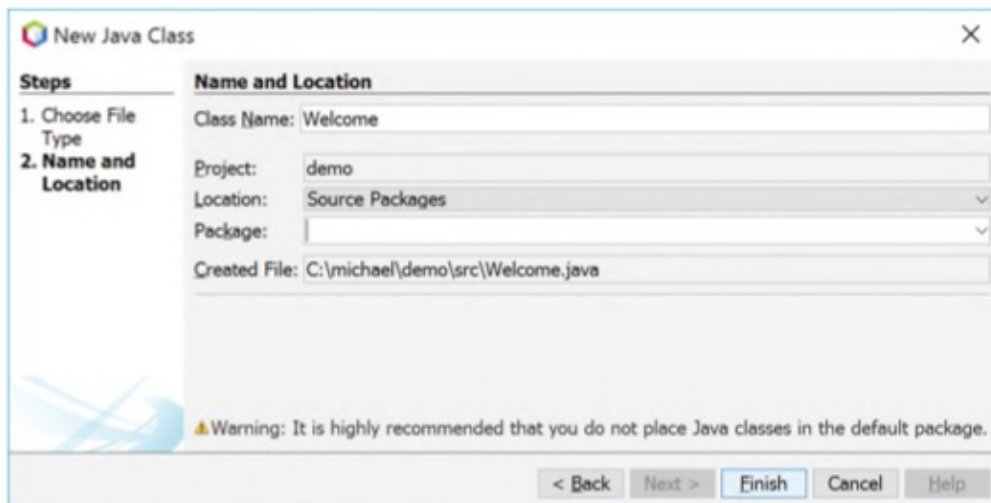
Slika 7.3

KREIRANJE NOVE JAVA KLASE

Kreiranje i lociranje nove klase u Javi.

Posle kreiranja novog projekta, možete kreirati Java programe u projektu koristeći sledeće korake:

1. Kliknite desnim tasterom miša na demo čvor u oknu projekta da biste prikazali kontekstni meni. Izaberite **New, Java Class** da biste prikazali dijalog **New Java Class**, kao što je prikazano na slici 7.4.
2. Otkucajte Welcomeu polje **Class Name** i izaberite Izvorni paketi u polju Location. Ostavite polje **Package** praznim. Ovo će kreirati klasu u podrazumevanom paketu
3. Kliknite na **Finish** da biste kreirali klasu dobrodošlice. Datoteka izvornog koda **Welcome.java** smeštena je u čvor **default package**

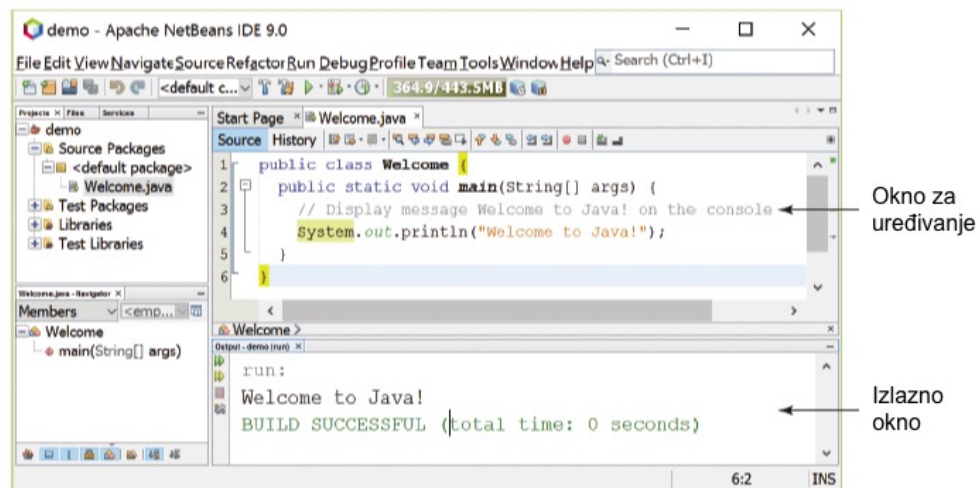


Slika 7.4 Okvir za dijalog New Java Class se koristi za kreiranje nove Java klase

IZMENE U PROGRAMU I IZVRŠENJE

Uređivanje izvornog koda se vrši u oknu za uređivanje, a prikaz rezultata sa daje u izlaznom oknu.

4. Izmenite kod u klasi **Welcome** tako da odgovara Listingu 1.1 u tekstu, kao što je prikazano na slici 7.5.



Slika 7.5 Iymene u programue se vrše u Oknu za uređivanje

PREPORUČENI LINKOVI SA VIDEO ZAPISIMA - PRVI VIDEO ZAPIS

Prvi video zapis povezan sa delovima lekcije 1

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

PREPORUČENI LINKOVI SA VIDEO ZAPISIMA - DRUGI VIDEO ZAPIS

Drugi video zapis povezan sa delovima lekcije 1

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

PREPORUČENI LINKOVI SA VIDEO ZAPISIMA - TREĆI VIDEO ZAPIS

Treći video zapis povezan sa delovima lekcije 1

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

✓ Poglavlje 8

Test za samotestiranje 2

TEST PITANJA OD 1.29 DO 1.44

Proverite svoje razumevanje pređenog gradiva ovim samotestiranjem.

U ovoj lekciji, odgovorite na pitanja od pitanja 1.29 do pitanja 1.45.

Da bi se aktivirao test, potrebno je da kliknete na sledeći link:

[https://liveexample-ppe.pearsoncmg.com/selftest/
selftest12e?chapter=1&username=liang12e](https://liveexample-ppe.pearsoncmg.com/selftest/selftest12e?chapter=1&username=liang12e)

▼ Poglavlje 9

Pokazna vežba

PRIMER 1C

Primer sa ispisivanjem pet puta poruke "Welcome to Java"

Pokazne primere, a kasnije i zadatke, klasifikovaćemo po težni, tako da pored broja primera ili zadatka, sadrže i malo slovo a, b ili c, sa sledećim značenjima:

a – težak zadatak

b - umereno težak zadatak

c – lak zadatak

Ovde ćemo definisati zadatak, i dati i rešenje, uz eventualno kratko objašnjenje. a na vežbama će se opisati i sicutovati postupak rešavanja

Zadatak: (Prikaži pet poruka) Napišite program koji pet puta prikazuje Welcome to Java.

Rešenje:

```
public class Exercise01_02 {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java");  
        System.out.println("Welcome to Java");  
        System.out.println("Welcome to Java");  
        System.out.println("Welcome to Java");  
        System.out.println("Welcome to Java");  
    }  
}
```

PRIMER 2C

Daje se zadatak i rešenje.

Primer 1c:

Zadatak 1: (Štampanje tabele) Napišite program koji prikazuje sledeću tabelu:

a a² a³

1 1 1

2 4 8

3 9 27

4 16 64

Rešenje:

```
public class Exercise01_04 {  
    public static void main(String[] args) {  
        System.out.println("a      a^2      a^3");  
        System.out.println("1      1      1");  
        System.out.println("2      4      8");  
        System.out.println("3      9      27");  
        System.out.println("4      16     64");  
    }  
}
```

PRIMERI 3C I 4C

Za dati zadatak treba napisati program u Javi - ovde se prikazuje rešenje zadatka

Primer 3c:

Zadatak: (Sabiranje članova reda) Napišite program koji prikazuje rezultat sledećeg izraza:

$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9$

Rešenje:

```
public class Exercise01_06 {  
    public static void main(String[] args) {  
        System.out.println(1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9);  
    }  
}
```

Primer 4c:

Zadatak: (Površina i obim kruga) Napišite program koji prikazuje površinu (area) i obim kruga (perimeter) čiji je poluprečnik (radius) 5.5 upotrebom sledeće formule:

$\text{perimeter} = 2 \times \text{radius} \times \pi$

$\text{area} = \text{radius} \times \pi$

Rešenje:

```
public class Exercise01_08 {  
    public static void main(String[] args) {  
        // Display area  
        System.out.println(5.5 * 5.5 * 3.14159);  
        // Display perimeter  
        System.out.println(2 * 5.5 * 3.14159);  
    }  
}
```

PRIMERI 5C I 6C

Za date zadatke treba napisati program u Javi - ovde su prikazana rešenja

Primer 5c:

Zadatak: (proračun prosečne brzine u miljama) Pretpostavimo neki trkač pretrči 14 kilometra za 45 minuta i 30 sekundi. Napišite program koji prikazuje prosečnu brzinu u miljama na sat (napomena: 1 milja = 1,6 kilometara)

Rešenje:

```
public class Exercise01_10 {  
    public static void main(String[] args) {  
        System.out.println((14 / 45.5) * 60 / 1.6);  
    }  
}
```

Primer 6c:

Zadatak: (Prosečna brzina u kilometima) Pretpostavimo neki trkač pretrč 24 milja za 1 sat, 40 minuta i 35 sekundi. Napišite program koji prikazuje prosečnu brzinu u kilometrima na sat (napomena: 1 milja = 1,6 kilometara)

Rešenje:

```
public class Exercise01_12 {  
    public static void main(String[] args) {  
        System.out.println(24 / (1 + (40 + 35.0 / 60) / 60) * 1.6);  
    }  
}
```

✓ Poglavlje 10

Individualna vežba: Zadaci za vežbu

ZADACI ZA INDIVIDUALNI RAD 1C, 2C, 3C I 4C

Ove zadatke student treba da samostalno uradi za vreme individualnih vežbi

: Zadatak 1c:

(Prikaži tri poruke) Napišite program koji prikazuje "Welcome to Java", "Welcome to Computer Science" i "Programiranje je zabavno."

Zadatak 2c:

(Prikaži šablon) Napišite program koji prikazuje sledeći obrazac (ovde ukucan sa fontom Lucida Sans Typewriter) :

```
      J      A      V      V      A
      J      A A      V      V      A A
J      J      AAAAA      V V      AAAAA
J J      A      A      V      A      A
```

Slika 10.1 Željeni rezultat koji program treba da prikaže

Zadatak 3c:

(Izračunavanje izraza) Napišite program koji prikazuje rezultat sledećeg matematičkog izraza:

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

Slika 10.2 Zadatak 3c

ZADACI 4C I 5C

Ove zadatke samostalno radite za vreme individualnih vežbi

Zadatak 4c:

(Približno π) π se može izračunati korišćenjem sledeće formule:

$$\pi = 4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \dots \right)$$

Slika 10.3

Napišite program koji će izračunati sledeće izraze:

$$4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} \right)$$

$$4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} \right)$$

Slika 10.4

Koristite 1.0 umesto 1 u svom programu

Zadatak 5c:

Površina i obim pravougaonika) Napišite program koji prikazuje površinu i obim pravougaonika širine 4,5 i visine 7,9 koristeći sledeću formulu:

area = width * height

ZADATAK 6B

Ovo je umereno težak zadatak. Ako ga ne uradite na vežbi, pokušajte to kod kuće.

Zadatak 6b:

Projekcija stanovništva) Biro za popis stanovništva SAD projektuje stanovništvo na osnovu sledećih pretpostavki:

- Jedno rođenje svakih 7 sekundi
- Jedna smrt svakih 13 sekundi
- Jedan novi imigrant svakih 45 sekundi

Napišite program za prikaz stanovništva za svaku od narednih pet godina. Pretpostavimo da trenutno stanovništvo ima 312.032.486, a jedna godina ima 365 dana.

Savet: U Javi, ako dva cela broja izvrše deljenje, rezultat je ceo broj. Razlomni deo je skraćen. Na primer, 5/4 je 1 (a ne 1,25), a 10/4 je 2 (a ne 2,5). Da biste dobili tačan rezultat sa razlomkom, jedna od vrednosti uključenih u deljenje mora biti broj sa decimalnom zapetom. Na primer, 5,0 / 4 je 1,25, a 10 / 4,0 je 2,5.

ZADATAK 7B

Ovo je umereno težak zadatak. Ako ga ne uradite na vežbi, uradite ga kod kuće

Zadatak 7b:

(Algebra: reši linearne jednačine 2×2) Možete koristiti Kramerovo pravilo da rešite sledeći sistem linearnih jednačina 2×2 pod uslovom da $ad - bc$ nije 0:

$$\begin{array}{l} ax + by = e \\ cx + dy = f \end{array} \quad x = \frac{ed - bf}{ad - bc} \quad y = \frac{af - ec}{ad - bc}$$

Slika 10.5

Napišite program koji rešava sledeću jednačinu i prikazuje vrednost za k i i : (Savet: zamenite simbole u formuli brojevima da biste izračunali k i i . Ova vežba se može uraditi u lekciji 1 bez upotrebe materijala u kasnijim lekcijama.)

$$\begin{array}{l} 3.4x + 50.2y = 44.5 \\ 2.1x + .55y = 5.9 \end{array}$$

Slika 10.6

▼ Poglavlje 11

Zaključak

ZAKLJUČAK

Ovo je rezime pređenog gradiva lekcije 1

1. Ime Java izvorne datoteke mora da odgovara imenu javne klase u programu. Datoteke Java izvornog koda moraju se završavati ekstenzijom .java.
2. Svaka klasa se kompajlira u zasebnu datoteku bajtkoda koja ima isto ime kao klasa i završava se ekstenzijom .class.
3. Da biste kompajlirali Java datoteku izvornog koda iz komandne linije, koristite komandu javac.
4. Da biste pokrenuli Java klasu iz komandne linije, koristite komandu java.
5. Svaki Java program je skup definicija klasa. Ključna reč class uvodi definiciju klase. Sadržaj klase je uključen u blok.
6. Blok počinje otvorom ({) i završava se zagradom (}).
7. Metode su sadržane u klasi. Da bi pokrenuo Java program, program mora imati glavni metod. Glavni metod je ulazna tačka na kojoj program počinje kada se izvrši.
8. Svaka izjava u Javi završava se tačkom i zarezom (;), poznatom kao terminator iskaza.
9. Ključne reči imaju specifično značenje za kompajler i ne mogu se koristiti u druge svrhe u programu.
10. U Javi, komentarima prethode dve kose crte (//) na liniji, koje se nazivaju linijskim komentarom, ili su zatvorene između /* i */ na jednom ili više redova, što se naziva blok komentar ili komentar pasusa. Kompajler ignoriše komentare.
11. Java izvorni programi razlikuju velika i mala slova.
12. Greške u programiranju mogu se kategorisati u tri tipa: sintaksičke greške, greške u izvršavanju i logičke greške. Greške koje je izvestio kompajler nazivaju se sintaksičkim greškama ili greškama kompajliranja. Greške u toku izvršavanja su greške koje uzrokuju nenormalan završetak programa. Logičke greške se javljaju kada program ne radi onako kako je trebalo.

LITERATURA

Pored ovog nastavnog materijala za e-učenje, studentima se preporučuje i sledeće reference

Osnovna literatura: Ovaj nastavni materijal je pripremljen u najvećoj meri u skladu sa osnovnim udžbenikom, koji se preporučuje studentima:

1. Y. Daniel Liang, Introduction to Java Programming and Data Structures, **Chapter 1**, Comprehensive Version, 12th edition, Global Edition, Pierson, 2019

Dopunska literatura:

1. Herbert Schildt. 2018. Java: A Beginner's Guide, Eighth Edition, Oracle
2. Joshua Bloch. 2017. Effective Java, Addison Wesley

Dopunski materijali koji se nalaze u LAMS-u u okviru lekcije 1:

- Supplement1aInstallingJDK11.pdf
- InstallNetBeans.pdf
- Supplement2dNetBeans.pdf
- Supplement2eNetBeansEffectiveTeaching.pdf
- Expanded Guidelines on Programming Style and Documentation

Preporučeni onlajn Java kursevi:

1. <https://docs.oracle.com/javase/tutorial/>
2. <https://www.ntu.edu.sg/home/ehchua/programming/index.html>
3. <http://www.javatpoint.com/java-tutorial>