Programski jezik JAVA
PREDAVANJE 1
2020

Prezentacija kreirana na osnovu sljedeće literature :
Dejan Živković: Osnove Java programiranja; Bruce Eckel: Misliti na Javi

Savladavanje gradiva

- Teorijska nastava
- Praktična nastava u računarskim salama
- Preduslovi:
  - Poznavanje osnovnih principa i koncepata programiranja
  - Savladano gradivo predmeta Programiranje 1 i 2
  - **Samostalan, praktičan i kontinuiran rad!**

Literatura

- Dejan Živković: Osnove JAVA programiranja, knjiga ili dostupno na Webu
- Bruce Eckel: Misliti na Javi, izdanje Mikro knjiga, ili original dostupno na Webu
- Internet
- Radno okruženje
  - Java Development Kit (JDK 6) - http://java.sun.com/
  - JCreator-radno okruženje - http://www.jcreator.com/
  - NetBeans IDE 6.8 - http://netbeans.org/
  - Eclipse - http://www.eclipse.org/

Osnovne napomene o programskom jeziku JAVA

- U jesen 1995. godine "Sun Microsystems" prvi put je predstavio programski jezik Java -- kao inovativan alat za Web.
- James Gosling i drugi projektanti u kompaniji "Sun" su bili angažovani na projektu razvoja interaktivne televizije i “pametnih” kućnih uređaja
- Nezadovoljni što koriste C++, kreirali su novi programski jezik pogodan za njihov projekat.
- Ime Java programu je dato po nazivu jedne vrste kafe
- ***Java je u potpunosti objektno orjentisani programski jezik***
- Danas se Java primjenjuje u velikom broju slučajeva, između ostalog i za:
  - ***Web servere***.
  - ***Relacione baze podataka.***
  - ***Personalne digitalne asistente.***
  - ***Mobilne telefone.***
- Java podržava ***višenitnost***, tako da se mogu praviti program i koji paralelno izvršavaju više aktivnosti.
- Važna karakteristika Jave je i sigurnost, jer ima ugrađene mehanizme za zaštitu od virusa i drugih zloupotreba.

Osnovne napomene o programskom jeziku JAVA

- Vrste Java programa
  - Aplikacija
    - uobičajeni program koji rješava neki problem potpuno samostalno
  - Aplet
    - izvršava se u Web pretraživaču u okviru neke Web strane
    - automatska distribucija i instalacija
    - kako se appleti učitavaju sa Interneta, uvedena su neka ograničenja radi sprečavanja zloupotrebe:
      - appleti ne mogu da čitaju ili pišu u fajl sistemu korisnika
      - ne mogu da komuniciraju sa serverima, osim sa onim sa kog su učitani
      - ne mogu da pokreću druge programe
  - Servlet, JSP (Java Server Pages)
    - izvršava se na Web serveru (dinamičke strane)

Uvod u programski jezik Java

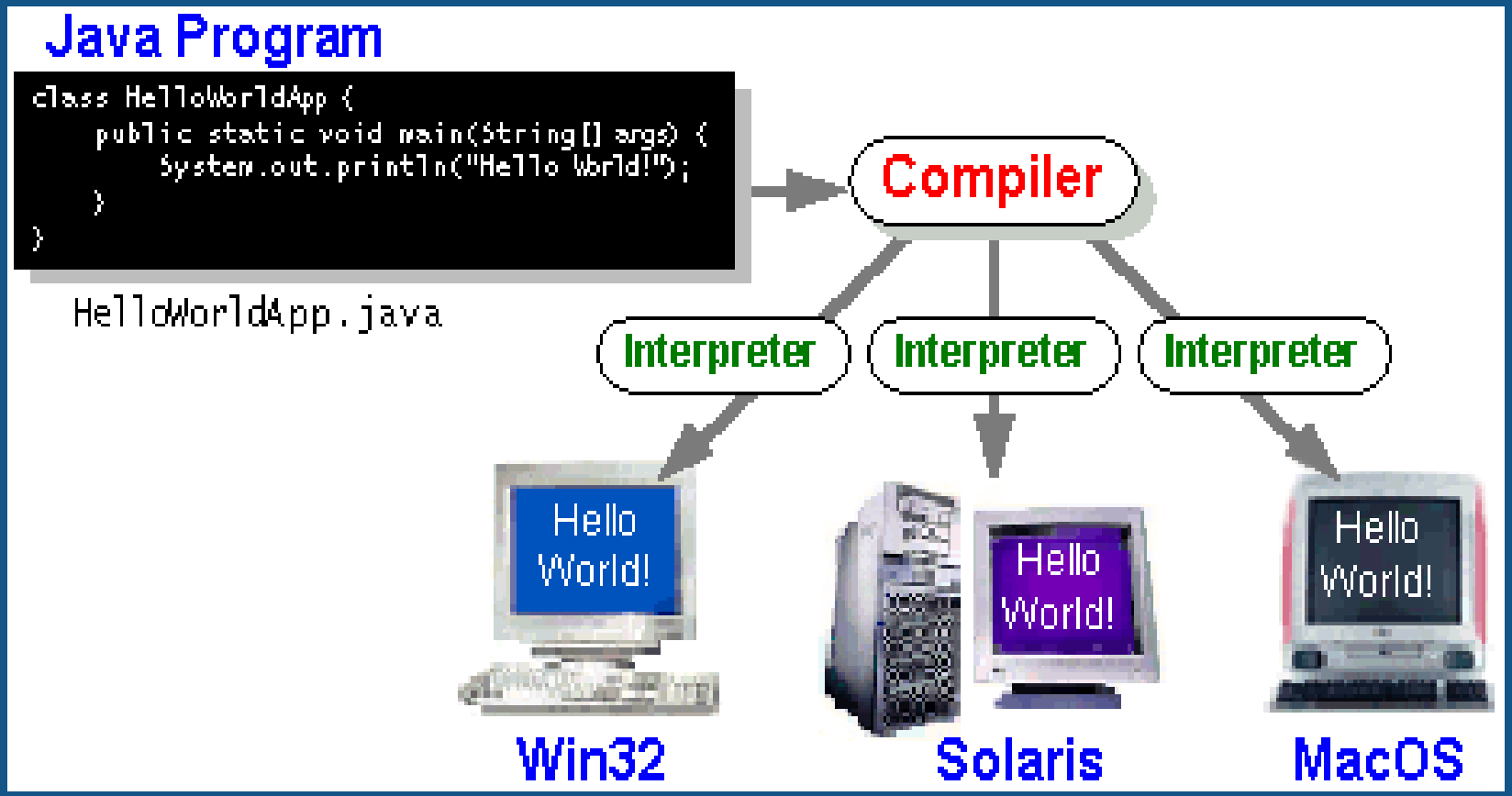
- Java je objektno-orijentisani, nezavistan od platforme, bezbjedan programski jezik, koji je projektovan tako da ga je jednostavnije naučiti od C++-a, a teže zloupotrijebiti od C-a i C++-a.
- **Objektno-orijentisano programiranje (OOP)** je metodologija razvoja softvera u kojoj se program:
  - sastoji od grupa objekata koji zajedno funkcionišu
  - objekti se kreiraju korišćenjem klasa
  - klase mogu biti korisnički definisane ili pripadaju nekom od postojećih paketa
- Java je projektovana tako da bude jednostavnija od programskog jezika C++, i to prije svega zbog sljedećeg:
  - u okviru programskog jezika Java automatski se vrši alokacija i dealokacija memorije.
  - Java ne sadrži pokazivače.

Uvod u programski jezik Java

- **Platformska nezavisnost** je mogućnost programa da se izvršava bez modifikacija u okviru različitih radnih okruženja. Dakle, Java programski jezik ne zavisi od operativnog sistema i tipa računara na kom se izvršava.
- Java programi se prevode u format koji se naziva bajtkod. Prevođenje se vrši pomoću Java Virtuelne Mašine (JVM).
- Java Virtuelna Mašina predstavlja “izmišljen računar” koji omogućava da Java kod bude interpretiran na bilo kom tipu procesora.
- Bajtkod u okviru bilo kog operativnog sistema može da izvrši bilo koji softver ili uređaj koji sadrži interpreter programskog jezika Java.
- Dakle, samo Java interpreteri zavise od procesora na kom se izvršavaju.
- ***Pisanje interpretatora Java bajtkoda za novi tip računara je mnogo jednostavnije od pisanja Java prevodioca za isti računar***

Uvod u programski jezik Java

- Prevođenje i interpretiranje programa pisanih u Javi



Uvod u programski jezik Java

- Primjer jednostavnog koda napisanog u Javi:

```
class Zdravo {
  public static void main (String args[]) {
   System.out.println (“Zdravo svima, “);
   System.out.println (“ovo je Java program…”);
  }
}
```

Uvod u programski jezik Java

- Struktura Java programa
  - Java program se sastoji od jedne ili više klasa
  - Izvorni kod svake klase se piše u posebnom fajlu čije ime mora biti isto kao ime klase
  - IME FAJLA MORA BITI ISTO KAO I IME OSNOVNE KLASE
  - Ekstenzija fajla Java izvornog koda mora biti **.java**

Uvod u programski jezik Java

- Za razvoj i izvršavanje Java koda mogu se koristiti tekstualno i grafičko radno okruženje
- Tekstualno okruženje može biti:
  - Notepad
  - WordPad
  - bilo koji drugi tekstualni editor (osim WORD-a i sličnih tekst procesora).
- Kao grafičko okruženje mogu se koristiti:
  - JCreator
  - NetBeans IDE
  - DrJava
  - Eclipse
  - Java Studio
  - JBuilder, ...

Uvod u programski jezik Java

- JDK bundle, takode poznat i kao Java SE (Standard Edition) sadrži:
- *java kompajler* (javac)
- *JRE* (Java okruženje) u cijem sastavu su:
  - *JVM* (Java virtuelna mašina)
  - *API* (Aplikacioni programski interfejs)

Uvod u programski jezik Java

- Prevođenje i izvršavanje u tekstualnom okruženju (DOS prozor)
- Prevođenje

```
javac Zdravo.java
```

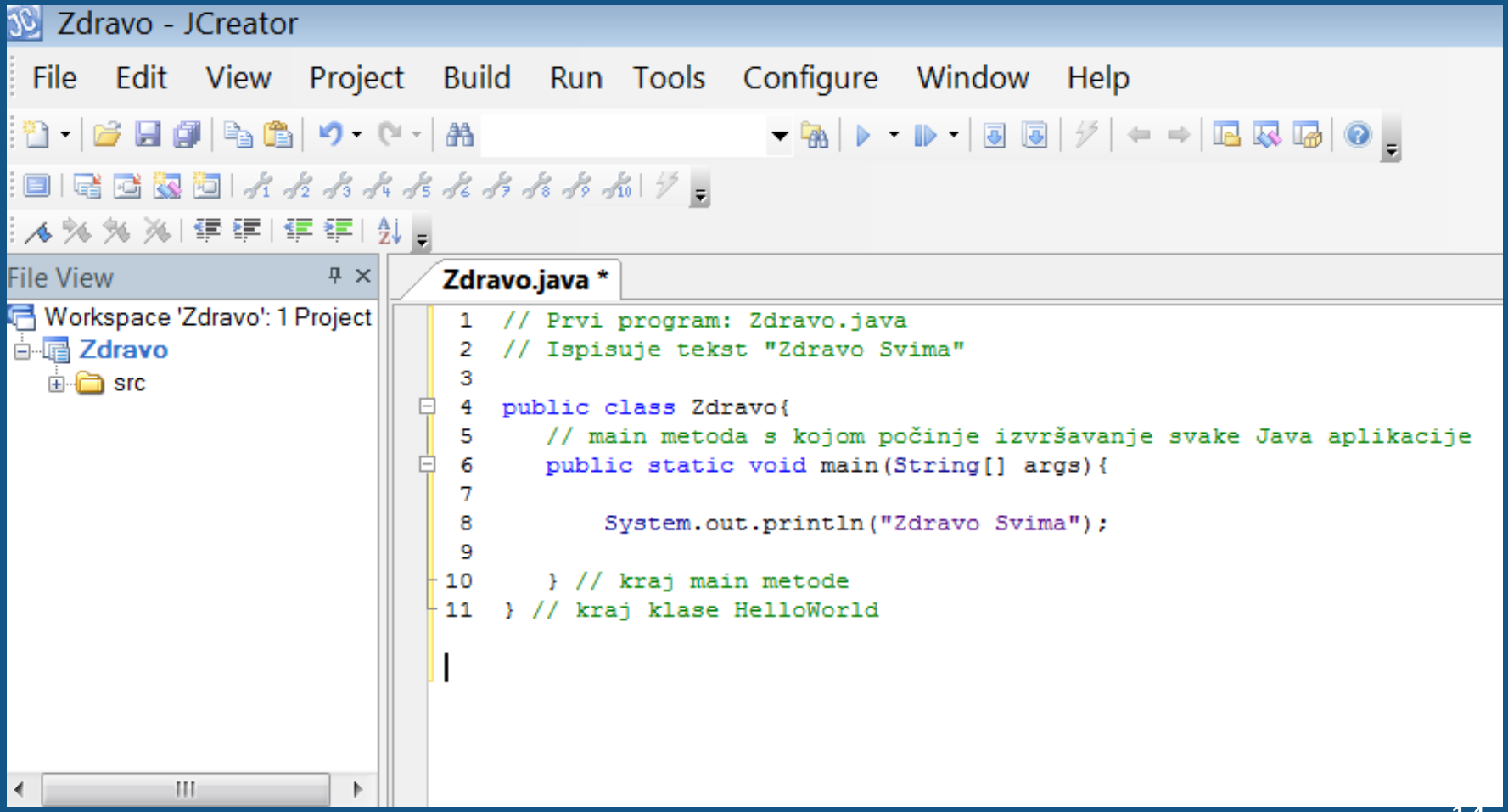
- nakon prevođenja kreira se novi fajl sa ekstenzijom .class

- Izvršavanje

```
java Zdravo
```

Uvod u programski jezik Java

Grafičko radno okruženje u JCreator-u

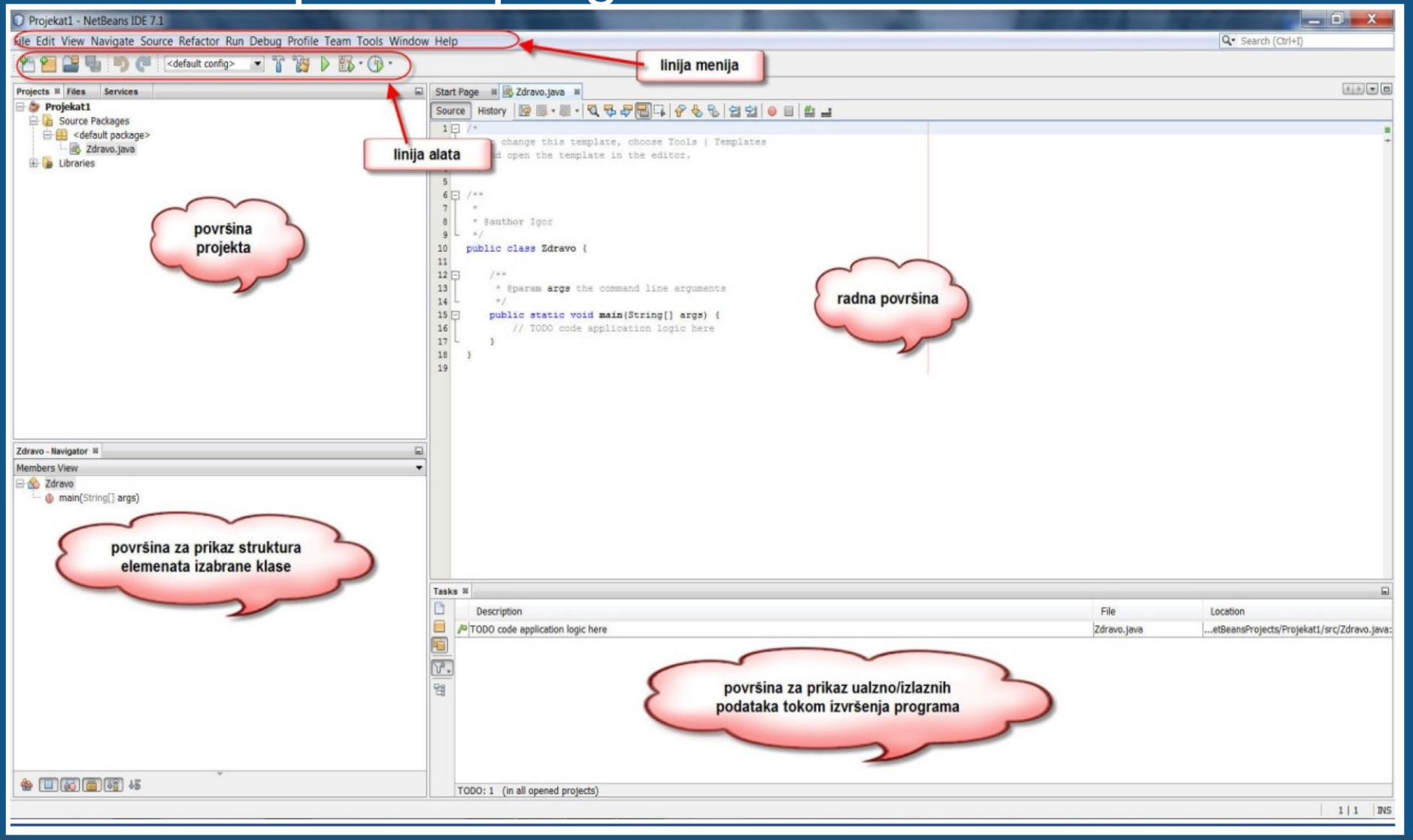


Grafičko okruženje NetBeans

- **NetBeans** (Integrated Development Environment) je integrisano grafičko razvojno okruženje koje je namijenjeno razvoju različitih vrsta programa
- Java SE Development Kit (JDK) mora biti instalirana na računaru prije pokretanja instalacije

Grafičko okruženje NetBeans

- radni prozor programa NetBeans:

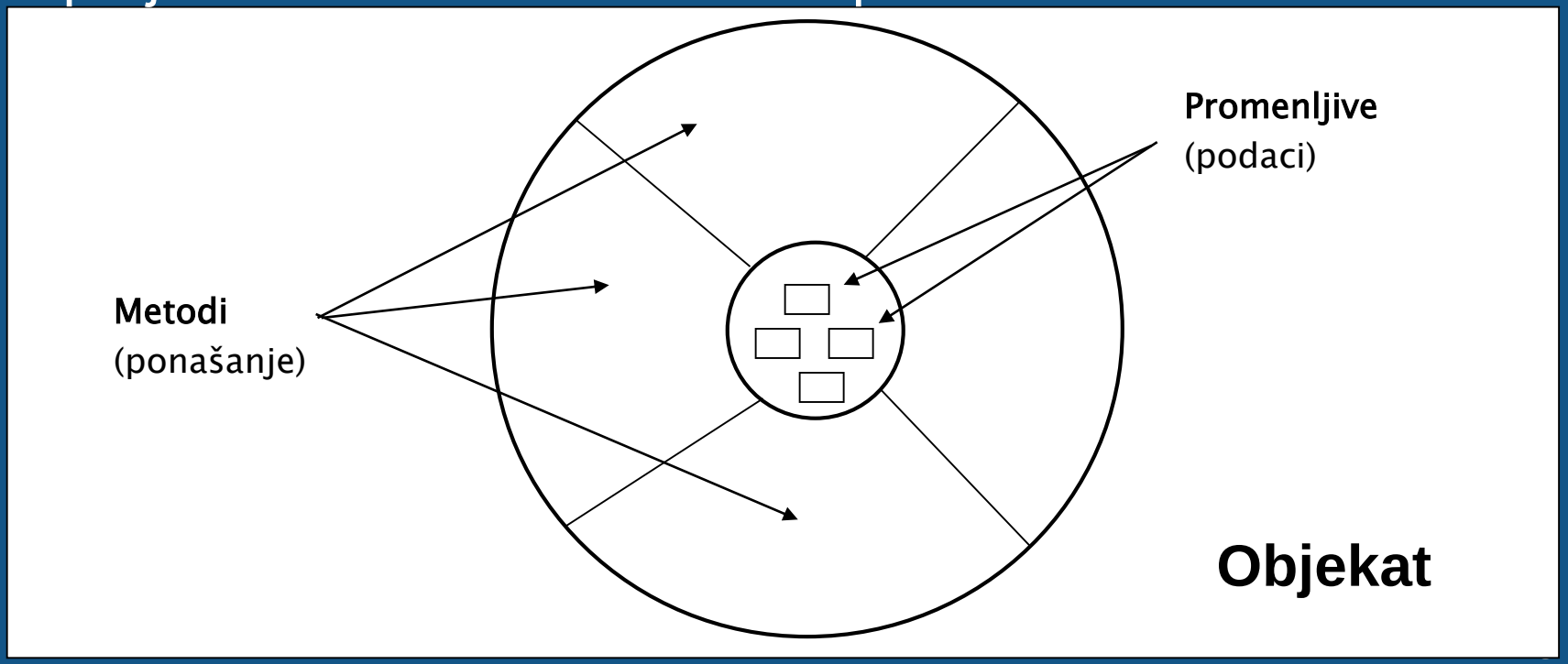


Objektno orijentisano programiranje - OOP

- Objektno-orijentisano programiranje je pristup razvoju računarskih programa koji imitira način na koji su objekti definisani u realnom svijetu.
- U programskom jeziku Java sve se realizuje korišćenjem klasa i objekata.
- Neophodno je prvo naučiti na koji način Java implementira principe objektno-orijentisanog programiranja - što znači da moramo naučiti:
  - organizovanje programa pomoću elemenata koji se nazivaju klase i postupak kojim se kreiraju objekti na osnovu klasa.
  - definisanje klase: način na koji treba da se ponaša i atribute koje treba da sadrži.
  - međusobno povezivanje klasa - nasljeđivanje i povezivanje klasa korišćenjem paketa i interfejsa.

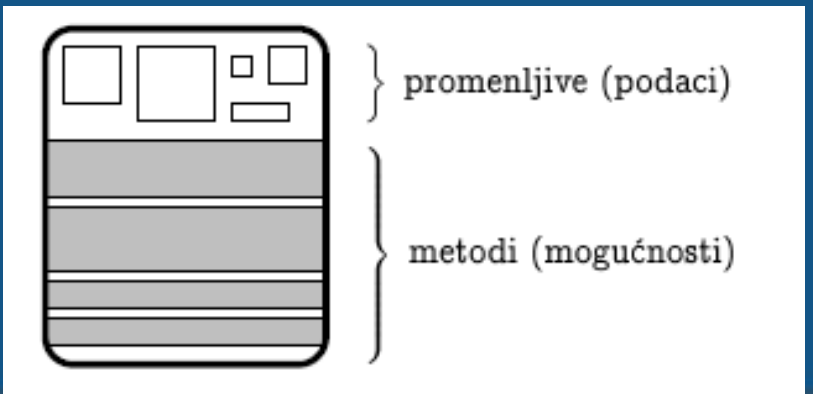
Osnovne karakteristike OOP - Objekti

- U slučaju OOP-a, svaki program se kreira kombinujući novokreirane objekte i postojeće objekte.
- Objekat je nezavistan element računarskog programa, koji predstavlja grupu povezanih funkcionalnosti i koji je projektovan tako da izvršava specifičan zadatak.



Osnovne karakteristike OOP - Objekti

- Sve je objekat.
  - Objekti su promjenljive koje sadrže podatke
- Objekat ima svoju memoriju koja je opet sastavljena od objekata.
- Novi objekti se kreiraju iz postojećih.
- Svaki objekat ima tip, odnosno svaki objekat je instanca neke klase.
- Program je skup objekata koji komuniciraju jedni sa drugima.



Osnovne karakteristike OOP - Klase

- Klasa se koristi za opis objekata sa zajedničkim svojstvima, odnosno ***klasa definiše šablon za stvaranje objekata za zajedničkim svojstvima***
- Klasa definiše atribute i ponašanja koje posjeduju svi objekti napravljeni na osnovu klase
- U OOP se pišu klase, a objekti se ne opisuju nego se stvaraju na osnovu klasa
- Instanca (primjerak) klase = objekat
- Stvaranje objekta = instanciranje odgovarajuće klase
- Java programi se sastoje isključivo od klasa, pa se sve mora nalaziti unutar klasa

Osnovne karakteristike OOP - Klase

- Primjer definicije klase

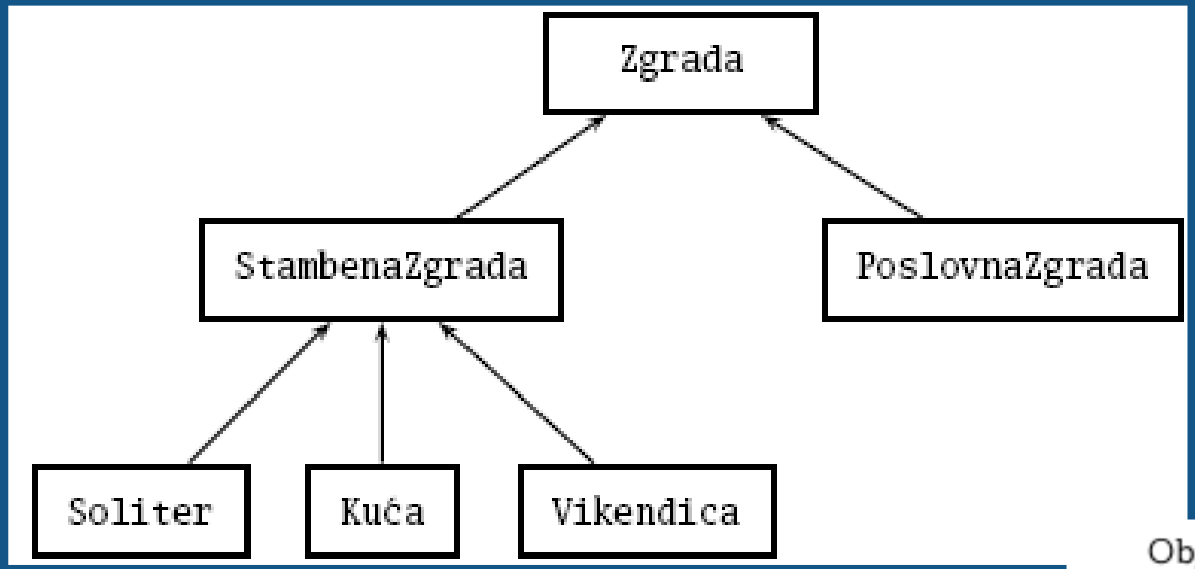
```
class Zgrada
{
int brojSpratova;
String boja;
String adresa;
...
public void okreči() { ... };
public void dozidaj() { ... };
}
```

- *Objekti*

Osnovne karakteristike OOP - Nasljeđivanje

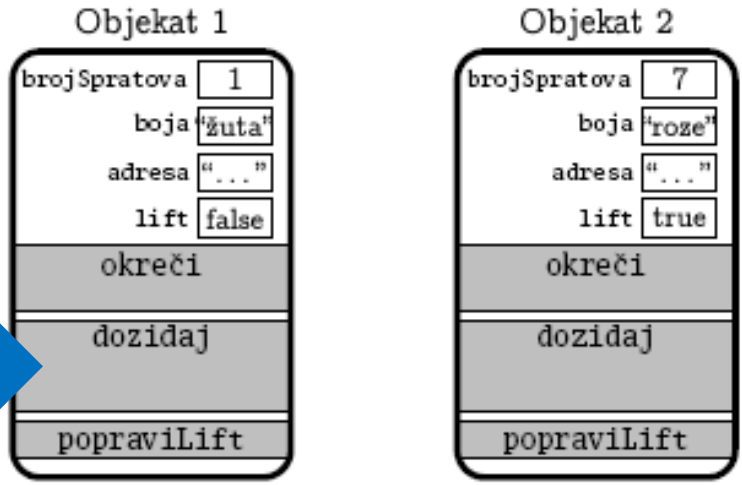
- Način za formiranje novih klasa od postojećih
- Nasljeđivanjem se uspostavlja hijerarhijska relacija između srodnih klasa
- Nova klasa proširuje postojeći klasu i nasleđuje sve atribute i ponašanja postojeće klase
- Terminologija
  - Bazna klasa — klasa koja se proširuje
  - Izvedena (proširena) klasa — nova klasa
- Bazna klasa = natklasa, klasa-roditelj
- Izvedena (proširena) klasa = potklasa, klasa-dijete
- Nasljeđivanje se vrši pomoću ključne riječi **extends**

Osnovne karakteristike OOP - Nasljeđivanje



- ***Izvedena klasa***

```
class StambenaZgrada extends Zgrada
{
boolean lift;
public void popraviLift(){…};
}
```



Osnovne karakteristike OOP - Paketi

- Klase su organizovane po paketima, analogno odnosu fajla i foldera u okviru fajl-sistema
- Paket je kolekcija klasa koje čine srodnu cjelinu (namijenjenih jednoj vrsti posla), odnosno **paketi čine biblioteke klasa**
- Osnovni paketi:

java.lang	java.util	java.io	java.net	java.awt	java.applet

- Paketi olakšavaju nalaženje i korišćenje klasa
- Paketi sprečavaju konflikte imena klasa, jer **različiti paketi mogu da sadrže klase sa istim imenom**
- Paketi omogućavaju kontrolu pristupa klasama

Osnovne karakteristike OOP - Paketi

- Java platforma sadrži veliki broj klasa koje su grupisane u pakete

- java.applet
- java.awt
- java.beans
- java.io
- java.lang
- java.math
- java.net
- java.nio
- java.rmi
- java.security
- java.sql
- java.text
- java.util
- javax.accessibility
- javax.crypto
- javax.imageio
- javax.naming
- javax.net
- javax.print
- javax.rmi
- javax.security
- javax.sound
- javax.sql
- javax.swing
- javax.transaction
- javax.xml
- org.ietf.jgss
- org.omg.CORBA
- org.omg.CosNaming
- org.omg.Dynamic
- org.omg.IOP
- org.omg.Messaging
- org.omg.PortableInterceptor
- org.omg.PortableServer
- org.omg.SendingContext
- org.omg.stub.java.rmi
- org.w3c.dom
- org.xml

Osnovne karakteristike OOP - Paketi

- Pri pisanju neke klase, mogu se jednostavno koristiti samo klase iz istog paketa
- Klase iz drugog paketa se mogu koristiti uz navođenje punog imena:

```
java.util.Date v = new java.util.Date();
```

- Deklaracija ***import*** "uvozi" pojedine klase iz nekog paketa
- Navodi se prije početka teksta klase

```
import java.util.Date;
   class MojaKlasa {
      ...
      Date v = new Date();
      ...
   }
```

Osnovne karakteristike OOP - Paketi

- Deklaracija ***import*** "uvozi" sve klase iz nekog paketa pomoću džoker-znaka *
- Navodi se prije početka teksta klase

```
import java.util.*;
class MojaKlasa {
   ...
   Date v = new Date();
   ...
}
```

- Paket ***java.lang se automatski uvozi u sve programe***
- Svaka klasa mora da pripada nekom paketu
- Ako se ništa ne navede, klasa pripada podrazumevanom (anonimnom) paketu

Osnovni elementi Java jezika

- Imena (identifikatori)
- Tipovi podataka
- Promenljive
- Izrazi

Osnovni elementi Java jezika - Imena

- Imena za razne elemente Java programa
  - Ime mora da počinje slovom ili _
  - Razmaci u imenu nijesu dozvoljeni
  - Ostali znaci: slova, cifre ili _
  - Razlikuju se mala i velika slova
  - Dužina nije ograničena
  - Ne mogu se koristiti rezervisane (službene, ključne) riječi

byte	else	instanceof	return	transient
case	extends	int	short	try
catch	final	interface	static	void
char	finally	long	strictfp	volatile
class	float	native	super	while
const	for	new	switch	
continue	goto	package	synchronized	assert

Osnovni elementi Java jezika - Imena

- Konvencije za imenovanje
  - Paketi: sva slova su mala
    - ***mojpaket***
  - Klase: početna slova svake riječi su velika slova
    - ***MojaKlasa***
  - Metod/promjenljiva: početno slovo je malo, a naredne riječi počinju sa velikim slovima
    - ***mojMetod***, ***mojaPromjenljiva***
  - Konstante: sva slova su velika
    - ***MOJA_KONSTANTA***

Osnovni elementi Java jezika - Tipovi Podataka

- ***Primitivni tipovi podataka i reference.***

Cjelobrojni tipovi podataka

Tip	Memorija	Opseg
int	4 bajta	-2,147,483,648 do 2,147,483, 647
short	2 bajta	-32,768 do 32,767
long	8 bajtova	-9,223,372,036,854,775,808L do 9,223,372,036,854,775,807L
byte	1 bajt	-128 do 127

Brojevi sa pokretnim zarezom

Tip	Memorija	Opseg
float	4 bajta	aproksimativno ±3.40282347E+38F
double	8 bajtova	aproksimativno ±1.79769313486231570E+308

Znakovi

Tip	Kodiranje
char	Unicode (vidi http://www.unicode.org/)

Logički

Tip	Vrijednosti
boolean	false, true

Osnovni elementi Java jezika - Tipovi Podataka

- Primitivni tipovi podataka:
  - ***byte, short, int, long, float, double***
- Klase (paket *java.lang*) koje enkapsuliraju primitivne tipove podataka:
  - ***Boolean, Byte, Character, Double, Float, Integer, Long i Number.***

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

        String stringIme = "125";
        int najInteger = Integer.MAX_VALUE;
        int brojI = 34;
        float minFloat = Float.MIN_VALUE;
        double brojF = 35.67;
        float c = Float.parseFloat(stringIme);

        System.out.println("Najveci integer = " + najInteger);
        System.out.println("Najmanji float  = " + minInteger);
        System.out.println("Primitivni tip int = " + brojI);
        System.out.println("Primitivni tip double " + brojF);
        System.out.println("Metod parse.Float() " + c);
    }
}
```

Najveci integer = 2147483647
Najmanji float = 1.4E-45
Primitivni tip int = 34
Primitivni tip double 35.67
Metod parse.Float() 125.0

Osnovni elementi Java jezika – Tipovi Podataka

- Tip *char* zauzima dva bajta umjesto uobičajenog jednog bajta
  - predstavlja Unicode znakove (Unicode standard definiše kodni raspored koji obuhvata praktično sve današnje jezike, uključujući indoevropske, dalekoistočne itd.)
- Tip *void* označava “prazan” tip

- Korisnički definisani tipovi podataka su predstavljeni klasama
- Vrijednosti klasnih tipova su reference (adrese) objekata odgovarajuće klase

Osnovni elementi Java jezika – Promjenljive

- Za formiranje imena promjenljivih važi isto pravilo kao u C-u.
- Promjenljiva može biti deklarisana unutar klase i tada se naziva promjenljiva članica (klase).
- Promjenljiva deklarisana unutar neke metode je lokalna promjenljiva.
- Lokalne promjenljive imaju oblast važenja bloka.
- Globalne promjenljive u JAVI NE POSTOJE:
- Nema razlike između definicije i deklaracije promjenljive.
- Svaka promjenljiva se mora definisati (deklarisati)
- Deklaracija promjenljive:

  tip + ime + (eventualno) početna vrijednost
- Format:

```
tip ime = vrijednost;
```

- Tip promjenljive je primitivni ili klasni tip.

Osnovni elementi Java jezika – Konstante

- Sintaksa konstanti je ista kao u C-u. Na primjer:

```
int     flag  = 0x23ff; // cijeli broj zadat heksadecimalno
double  xCoo  = 2.3E-11;// floating point konstante su tipa double
float   xx    = 11.2F;  // konstanta tipa float ima F na kraju
char    slovo = 'c';

char    znak  = '\u05D0'; // znak א predstavljen Unicodovim kodom
```

- U Javi se promjenljiva može učiniti **konstantnom** pomoću ključne riječi **`final`** (`const` u C-u).
- Na primjer: `final x = 3;` znači da promjenljivu x više ne možemo mijenjati.

Osnovni elementi Java jezika – Operatori

- Aritmetički operatori – binarni:

Operator	Upotreba	Opis
+	x + y	zbraja x i y
-	x - y	oduzima y od x
*	x*y	množi x i y
/	x/y	dijeli x sa y
%	x%y	ostatatak cjelobrojnog dijeljenja x sa y

- Aritmetički operatori – unarni:

Operator	Upotreba	Opis
++	x++	poveća x za 1 i to tako da u y=x++ najprije kopira staru vrijednost za x u y a onda poveća x za 1
++	++x	poveća x za 1 i to tako da ako u y=++x najprije poveća x a onda ga kopira x u y
--	x--	smanjuje x za 1 i to tako da u y=x-- najprije kopira staru vrijednost za x u y pa onda smanji x
--	--x	smanjuje x za 1 i to tako da u y=--x najprije smanji x pa onda kopira u y

Osnovni elementi Java jezika – Operatori

- Relacioni operatori

Operator	Upotreba	Rezulat je true ak
>	x > y	x je veće od y
>=	x>=y	x je veće ili jednako y
<	x< y	x je manje od y
<=	x <= y	x je manje ili jednako y
==	x==y	x i y su jednaki
!=	y != y	x i y su razliciti

Logički operatori

Operator	Upotreba	Rezultat je true ako:
&&	x && y	x i y oba imaju vrijednost true; ako je x false y se ne računa
||	x || y	x ili y imaju vrijednost true; y se računa samo ako je x false
!	!x	x je false
&	x & y	x i y oba imaju vrijednost true; evaluira uvijek i x i y
|	x | y	x ili y imaju vrijednost true; evaluira uvijek i x i y
^	op1 ^ op2	ako x i y imaju različite vrijednost

Osnovni elementi Java jezika – Operatori

- Operatori pridruživanja

Operator	Upotreba	Ekvivalent
+=	x+= y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
&=	x &= y	x = x& y
|=	x |= y	x = x | y
^=	x ^= y	x= x ^ y
>>=	x >>= y	x = x >> y
>>>=	x >>>= y	x = x >>> y
<<=	x <<= y	x = x << y

Osnovni elementi Java jezika – Operatori

- Operator **new** alocira memoriju za objekat ili polje.

```
int [] x = new int[3]
```

- Operator ?: (slično kao if-else naredba):

```
x ? y : z
Primjer: (2 > 0)? 5 : 7   Odgovor: 5
```

- Operator “ **.** “ se upotrebljava kod pristupa promjenljivim i metodama u klasi.
- **Instanceof** operator odredjuje da li je dati objekat instanca klase.

```
Na primjer: x instanceof y   // true, false
```

Konverzija tipova podataka

- Konverzija podataka (*casting*) nastaje prilikom dodjeljivanja vrijednosti jednog tipa promjenljivoj drugog tipa
- Automatska konverzija ukoliko:
  - tipovi su međusobno kompatibilni
  - ne može doći do gubitka tačnosti

byte → short → int → long → float → double

- Implicitne konverzije u kojima se GUBI informacija (`long` u `int` ili `double` u `float`) nijesu dozvoljene.

Konverzija tipova podataka

- Eksplicitna konverzija se mora koristiti ako postoji mogućnost gubitka tačnosti
- U tim slučajevima se koristi *cast* operator.
- Format:

  `(tip) izraz`
- Primjer:
- ```
  double x = 10.1;
  int    y = (int) x*x;
  ```
  ```

Prioritet operatora

Unarni operatori:	++, --, eksplicitna konvercija
Množenje i dijeljenje:	*, /, %
Sabiranje i oduzimanje:	+, -
Relacijski operatori:	<, >, <=, >=
Jednakost i nejednakost:	==, !=
Ligičko I:	&&
Logičko ILI:	||
Operator izbora:	? :
Operatori dodjele	=, +=, -=, *=, /=, %=

Prvi jednostavan Java program

```
class Zdravo {
   public static void main (String[] args) {
     System.out.println (“Zdravo svima, “);
     System.out.println (“ovo je Java program
   …”);
   }
}
```

Prvi jednostavan Java program

Metoda main mora biti deklarisana kao **public**, pošto pri pokretanju programa mora biti pozvana izvan klase.

Rezervisana riječ **static** dozvoljava da metoda main() bude pozvana bez pravljenja posebne instance klase. To je neophodno, jer Javin interpretator poziva metodu main() prije nego što je stvoren ijedan objekat.

Svaka klasa može da sadrži više metoda, ali samo jedna je glavna (engl. *main*). Sa metodom **main**, počinje izvršavanje svih Java aplikacija.

```
clas Zdravo {
    public static void main (String[] args) {
      System.out.println (“Zdravo svima, “);
      System.out.println (“ovo je moj prvi program …”);
    }
```

Rezervisana riječ **void** samo saopštava prevodiocu da metoda main() ne vraća nikakvu vrednost.

- Metodama mogu da se proslijede podaci preko promenljivih (tzv. Parametri) koje su navedene u zagradi iza imena metode.
- U metodi main() postoji samo jedan parametar, ali on nije jednostavan. **String[] args** deklariše parametar args, koji predstavlja niz instanci klase String.
- Objekti tipa String označavaju znakovne nizove. Znači, kada se program pokrene u niz *args* biće smješteni eventualni argumenti unijeti na komandnu liniju.

- **Vodite računa i main!**

Koraci kod pravljenja prvog Java programa

- Nakon unosa koda, sljedeći korak je da se sačuva i da ime datoteci.
- Datoteka sa izvornim kodom u Javi se zvanično zove kompilatorska jedinica koja predstavlja tekstualnu datoteku koja sadrži jednu ili više definicija klasa.
- S obzirom da Java sav kod smješta unutar klasa, ime određene klase treba da odgovara imenu datoteke koja sadrži program. Znači Java stavlja sve klase u poseban tekst fajl koji predstavlja Javin izvršni fajl. Dakle, prethodni primjer treba snimiti pod nazivom Zdravo.java.
- Sledeći korak je kompajliranje programa. Prevodilac jezika Java zahtijeva da datoteka sa izvornim kodom ima nastavak .java. Naziv Javinog prevodioca je *javac*. Ukoliko se koristi MSDos Prompt, na komandnoj liniji treba otkucati:

```
C:\>javac Zdravo.java
```

Koraci kod pravljenja prvog Java programa

- Prevodilac *javac* će napraviti datoteku Zdravo.class.
- Datoteka Zdravo.class će sadržati bajtkod programa. (bajtkod predstavlja poluproizvod sa instrukcijama koje treba da izvrši interpretator).
- Prevodilac javac ne generiše kod koji se odmah može izvršavati. Da bi se pokrenuo program mora se pozvati Javin interpretator nazvan java na sljedeći način:

```
C:\>java Zdravo
```

- Javina virtuelna mašina izvršava program tako što prvo traži istoimenu klasu, a potom unutar nje metodu main(). Kada se program izvrši, na ekranu će se dobiti sljedeća poruka:

```
Zdravo svima,
ovo je program u Javi ...
```

Osnovne klase u Javi

- Veliki broj klasa sa unaprijed definisanim metodima (procedurama) koji obavljaju specifični zadatak
  - System (java.lang)
  - Math (java.lang)
  - String (java.lang)
  - Scanner (java.util)

Klasa ***Paket***

Osnovne klase u Javi - System

- `System.out.print(. . .)`
- print() – metod
- out – statičko polje klase System
- System.out – objekat klase System koji predstavlja standardni izlaz
- Ukoliko se iza imena nalaze zagrade, onda predstavlja metod a ukoliko nema zagrada, ime predstavlja promjenljivu

Osnovne klase u Javi - System

- `System.out.print(. . .)`
- `System.out.println(. . .)` - pomjera kursor na početak sljedećeg reda
- `System.out.printf(. . .)` - printf podešava format izlaznih podataka
- `System.in` omogućava unos sa tastature

Primjeri:

```
System.out.print("Suma brojeva je: " + s);

System.out.println("Suma brojeva je: " + s);

System.out.printf("Suma brojeva je: %8d", s);
```

Osnovne klase u Javi - System

- `System.out.printf("Suma brojeva je: %8d", s);`
- Razlika između %d i %8d je u tome što se u prvom slučaju prikazuje cijeli broj sa onoliko cifara koliko ih ima, i pri tome se računa i znak „-“ za negativno brojeve
- U drugom slučaju se za prikaz broja koristi tačno 8 cifara.
- Ako broj ima manje od 8 cifara, dodaju se prazna mjesta ispred cifara broja

```
int a=30;
System.out.printf(„Broj je: %d\n", a);
System.out.printf(„Broj je: %8d\n", a);
```

Na ekranu će biti:

```
Broj je: 30
Broj je:       30
```

Osnovne klase u Javi - Math

- Statički metodi
- Metodi koji vraćaju vrijednost
  - `Math.sqrt(x)`
  - `Math.abs(x)`
  - `Math.sin(x), Math.cos(x), ...`
  - `Math.exp(x)`
  - `Math.log(x)`
  - `Math.pow(x, y)`
  - `Math.random()`

Osnovne klase u Javi - Math

- `Math.sqrt(x)`:
  - `System.out.println(Math.sqrt(x))` ili
  - `double a = Math.sqrt(x)`
- `Math.sin(x), Math.cos(x)` – ugao se navodi u radijanima
- `Math.pow(x,y)` - x^y
- `Math.random()` - slučajni brojevi u opsegu [0,1)
- Klasa Math sadrži i nekoliko statičkih polja kojima su predstavljene poznate matematičke konstante:
  - `Math.PI` –predstavlja broj pi=3.14159...
  - `Math.E` - predstavlja broj e=2.71828...

Osnovne klase u Javi - String

```
String imePrezime = "Marko Marković";
String s1, s2;
s1=imePrezime;
```

- s1.length()

`imePrezime.length()` – poziv metoda length za objekat stringa na koga ukazuje promjenljiva imePrezime

- s1.toUpperCase(), s1.toLowerCase()
- s1.equals(s2) – vraća true ili false
- s1.equalsIgnoreCase(s2)
- s1.charAt (n) – vraća znak sa n-te pozicije stringa s1
- s1.substring(n,m) – vraća podniz tipa String koji se sastoji od znakova s1 na pozicijama od n do m-1

Osnovne klase u Javi - Scanner

- Ova klasa olakšava učitavanje ulaznih i izlaznih podataka
- Metodi ove klase su objektni – za njihovu primjenu prvo se mora konstruisati objekat klase Scanner
- Objekat se konstruiše pomoću konstruktora
- Konstruktor klase je metod kojim se konstruiše objekat klase i memoriji i inicijalizuje se
- Kao rezultat dobija se pokazivač na konstruisani objekat

```
String tastatura = new Scanner(System.in);
```

Konstruktor klase Scanner

Osnovne klase u Javi - Scanner

- tastatura.next()
- tastatura.nextInt()
- tastatura.nextDouble()
- tastatura.nextLine()
- tastatura.hasNextInt()
- tastatura.hasNextDouble()
- tastatura.hasNextLine()

Klase omotači

- Ponekad je primitivne vrijednosti u Java-i potrebno tretirati kao da su objekti
- Vrijednost primitivnog tipa se može „umotati“ u objekat odgovarajuće ***klase omotača***
- Za svaki od 8 primitivnih tipova podataka postoji odovarajuća klasa omotač: Byte, Short, Integer, Long, Float, Double, Character i Boolean
- ***Ove klase konstruišu objekte koji predstavljaju vrijednosti primitivnih tipova***

Klase omotači

- Primjer:
- Objekat omotač za vrijednost 1.22, tipa double, može se konstruisati kao:

```
Double d = new Double (1.22)
```

- `d` je objektna promjenljiva i predstavlja iste informacije kao i primitivna promjenljiva 1.22, ali u formi objekta
- Moguća je automatska konverzija između primitivnih tipova i klasa omotača
- Npr. ako se vrijednost tipa int koristi u slučaju u kom je potreban objekat tipa Integer, int vrijednost će se automatski pretvoriti u integer objekat

Klase omotači

- Klase omotači sadrže i neke statičke metode za rad sa odgovarajućim tipovima podataka
- Npr. klasa Integer sadrži metod parseInt() koji string pretvara u vrijednost tipa int:
- Integer.parseInt(“10“) kao rezultat vraća broj 10 tipa int
- Integer.valueOf (“10“) kao rezultat daje objekat klase Integer koji sadrži cjelobrojnu vrijednost 10

Naredbe

- Naredbe su elementi programa koji se izvršavaju.
- Naredbe u Javi se pišu sa “;” na kraju.
- Vrste naredbi:
  - Naredba definisanja (deklarisanja) promjenljivih
  - Naredba dodjele vrijednosti promjenljivim
  - Blok naredba
  - Naredbe grananja (if, if-else, switch)
  - Naredbe ponavljanja (petlje, ciklusi – while, do-while i for naredbe)

Naredba definisanja (deklarisanja) promenljivih

- Svaka promenljiva se mora definisati (deklarisati) prije nego što se upotrijebi
- Format:

```
tip ime = vrijednost;
```

- Primjeri:
  - `int i=7;`
  - `float j=3.14;`
- Definicija promenljive u Javi *ne* mora se pisati na početku programa

Naredbe dodjele

- Format:

```
promenljiva ○= izraz;
```

- Ekvivalentno sa:

```
promenljiva = promenljiva ○ izraz;
```

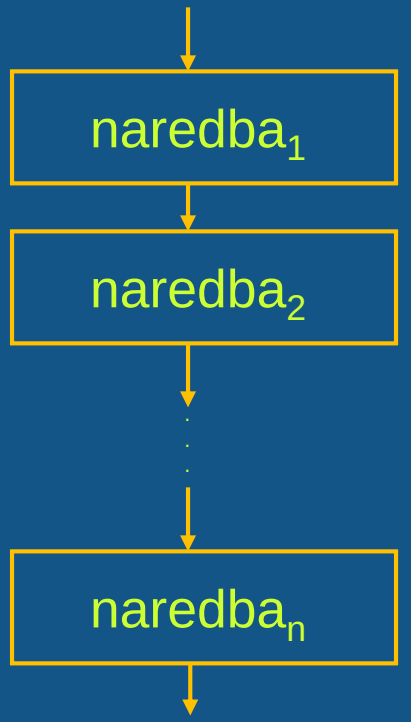
- Primjeri:

```
x += 2;        x = x + 2;
a /= b + c;    a = a / (b + c);
m %= n;        m = m % n;
```

Blok naredbe

- Niz naredbi između { i }:

```
{
   naredba1;
   naredba2;
      . . .
   naredban;
}
```



- Blok naredba se može pisati na svakom mjestu u programu gdje se može koristiti obična naredba

Blok naredbe

- Oblast važenja promjenljive definisane u bloku je od mjesta deklaracije do kraja bloka
- Lokalne promjenljive – ne mogu se koristiti u okolnim blokovima
- Naredbe u bloku mogu koristiti promjenljive iz okolnih blokova
- Primjer:

```
{
    int x, y;
    {
      int i=5;

      x = (i++) - 3;
      y = i + 4;
    }
    i = 0; // GREŠKA!
}
```

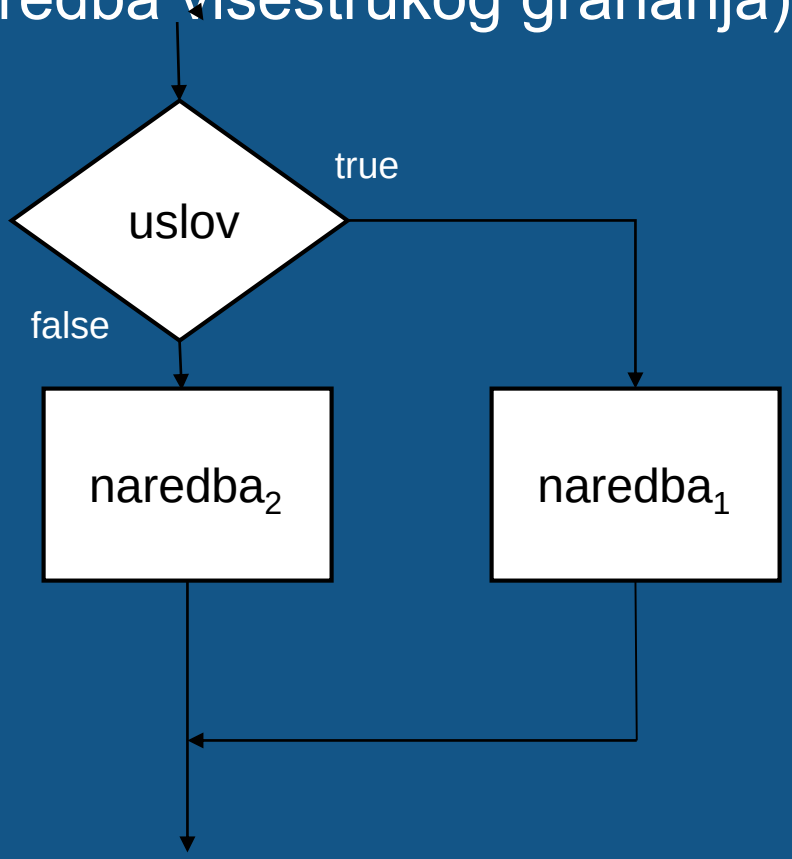
Naredbe kontrole toka programa - naredbe grananja

- **if** naredba
- **if-else** naredba
- složena **if-else** naredba (naredba višestrukog grananja)
- **switch** naredba

- Naredba **if-else**

- Format:

```
if (uslov)
  naredba1;
else
  naredba2;
```



Naredbe kontrole toka programa - naredbe grananja

- Naredba višestrukog grananja - složena **if-else** naredba
- Primjer:

```
if (p >= 90)
  ocjena = ‘A’;
else if (p >= 80)
  ocjena = ‘B’;
else if (p >= 70)
  ocjena = ‘C’;
else if (p >= 60)
  ocjena = ‘D’;
else if (p >= 50)
  ocjena = ‘E’;
else
  ocjena = ‘F’;
```

Naredbe kontrole toka programa - naredbe grananja

- Naredba **switch**

```
switch (izraz) {
  case konstanta1 : naredba1;
  case konstanta2 : naredba2;
    ...
  case konstantan : naredban;
  default : naredba;     }
```

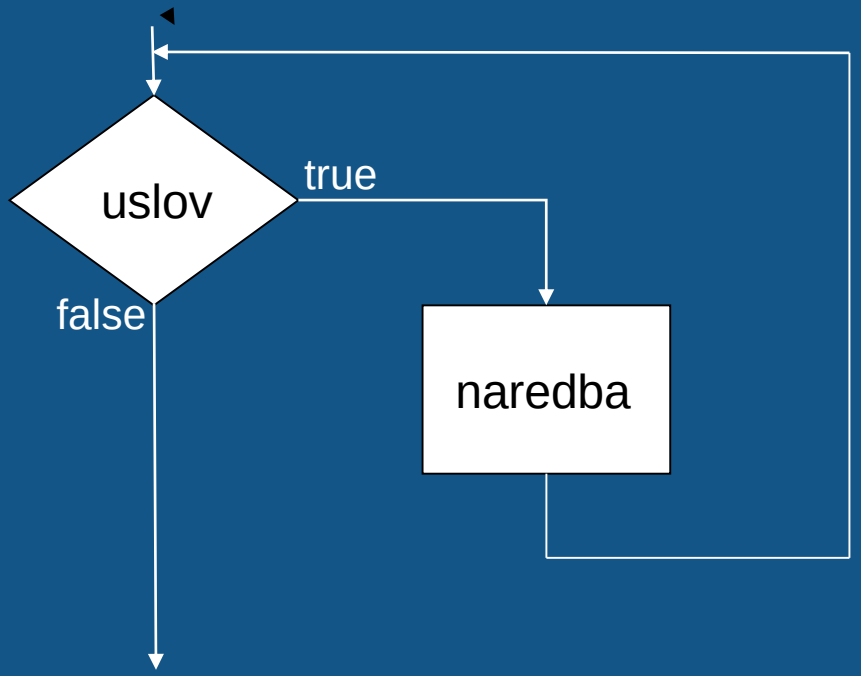
- Primjer:

```
switch (brojač) {
  case 1:
    System.out.println("Jedan");   break;
  case 2:
    System.out.println("Dva");     break;
  case 3:
    System.out.println("Tri");     break;
  default:
    System.out.println("Ni jedan, ni dva, ni tri");   break;
}
```

Naredbe kontrole toka programa - naredbe ponavljanja

- **while** petlja
- **do-while** petlja
- **for** petlja

- **while** petlja - format:




```
while (uslov) {
        naredba ili niz naredbi;
   }
```

Naredbe kontrole toka programa - naredbe ponavljanja

- Prvo se izračunava vrijednost uslova u zagradi
- Ako je ta vrijednost false, prekida se izvršavanje programa i program se normalno nastavlja od naredbe koja je iza while petlje
- Ako vrijednost uslova u zagradi daje true, izvršava se naredba ili niz naredbi u bloku a zatim se ponovo izračunava vrijednost uslova u zagradi
- Postupak se ponavlja sve dok je logički izraz u zagradi tačan

```
while (uslov) {
        naredba ili niz naredbi;
  }
```

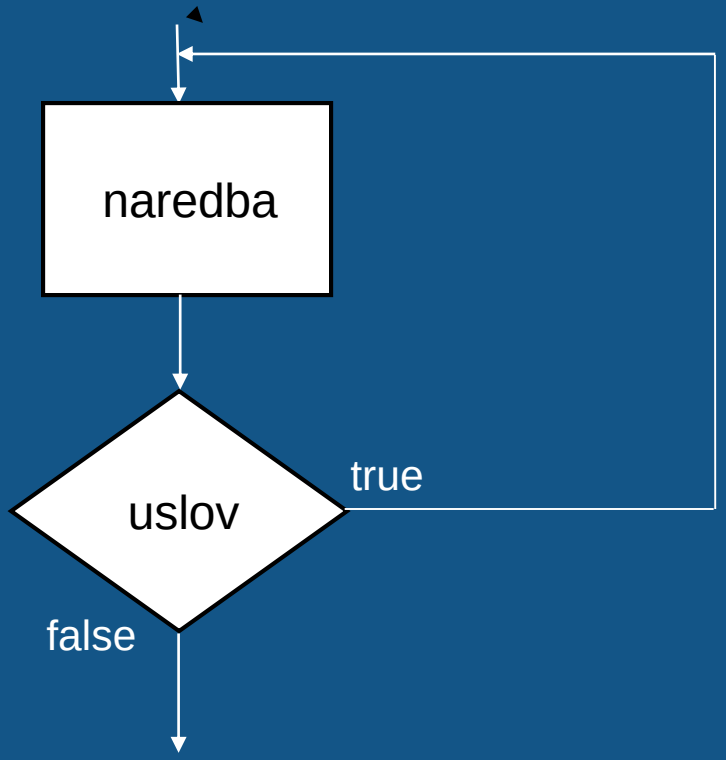
Naredbe kontrole toka programa - naredbe ponavljanja

- Kod **while** petlje uslov prekida se provjerava na početku svake iteracije
- **do-while** petlja uslov prekida se provjerava na kraju svakog izvršavanja tijela petlje
- Sintaksa:

```
do
naredba;
while (uslov);
```

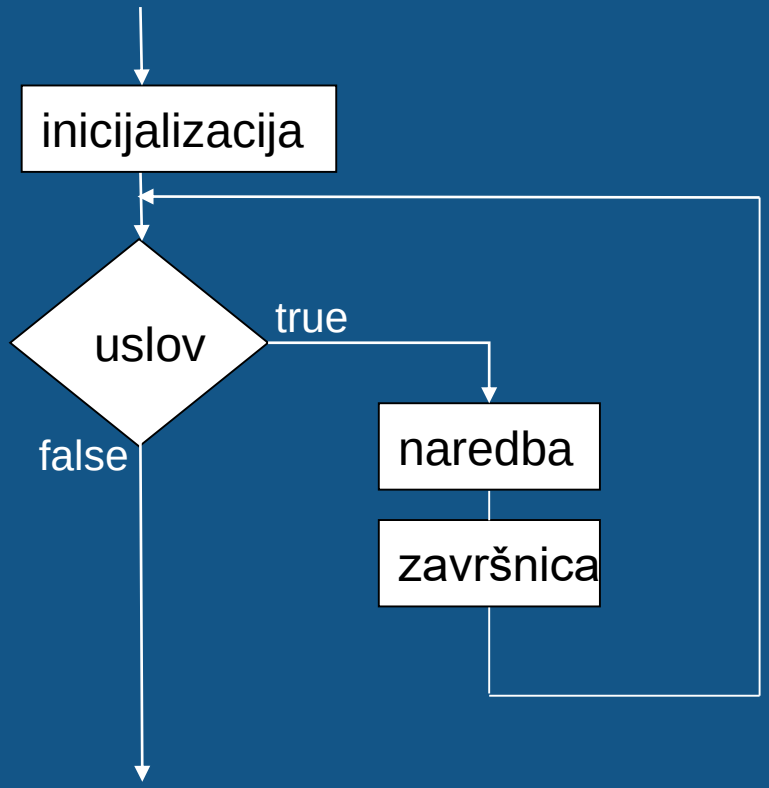
Ili

```
do {
    niz-naredbi;}
    while (uslov);
```



Naredbe kontrole toka programa - naredbe ponavljanja

- **for** petlja - format:




```
for (inicijalizacija; uslov;završnica)
        naredba;
```

Naredbe kontrole toka programa - naredbe prekida

- Naredbe **break** i **continue**
- **break** prevremeno prekida izvršavanje petlje (**while, do-while, for**), kao i naredbe **switch**
- **continue** prekida izvršavanje samo aktuelne iteracije petlje
- U ugnježđenim petljama se odnose samo na petlju u kojoj se nalaze
- Primjer:

```
for (int i = 1; i <= 10; i++) {
   if (i == 5) break;
   System.out.print(i + " ");
}
System.out.println();

/* 1 2 3 4 */
```

Naredbe kontrole toka programa - naredbe prekida

Primjer 2:

```
    for (int i = 1; i <= 10; i++) {
  if (i%2 != 0) continue;
  System.out.print(i + " ");
    }
 System.out.println();

   /* 2 4 6 8 10 */
```

Primjer 3:

```
        int k = 0;
        for (int i = 1; i <= 5; i++)
           for (int j = i; j <= 5; j++) {
                if (i == 3) break;
                   k++;
                      }
      System.out.println("k = " + k);

      /* k = 9 */
```

Naredbe kontrole toka programa - naredbe prekida

- Ugnježdene petlje