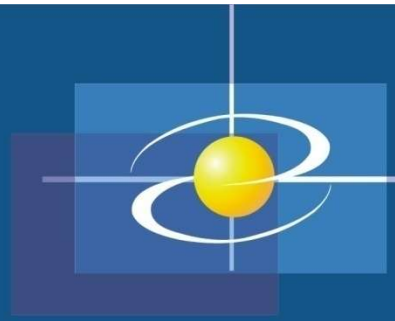


Programski jezik JAVA
PREDAVANJE 1
2024

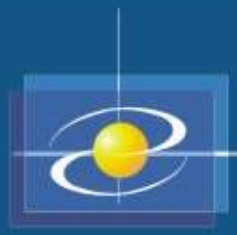
Literatura

- Dejan Živković: Osnove JAVA programiranja, knjiga ili dostupno na Webu
- Bruce Eckel: Misliti na Javi, izdanje Mikro knjiga, ili original dostupno na Webu.

- Radno okruženje
  - Java Development Kit (JDK 6) - http://java.sun.com/
  - JCreator-radno okruženje - http://www.jcreator.com/
  - NetBeans IDE - http://netbeans.org/
  - Eclipse - http://www.eclipse.org/

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)**.
- 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.

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

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

```
javac Zdravo.java
```

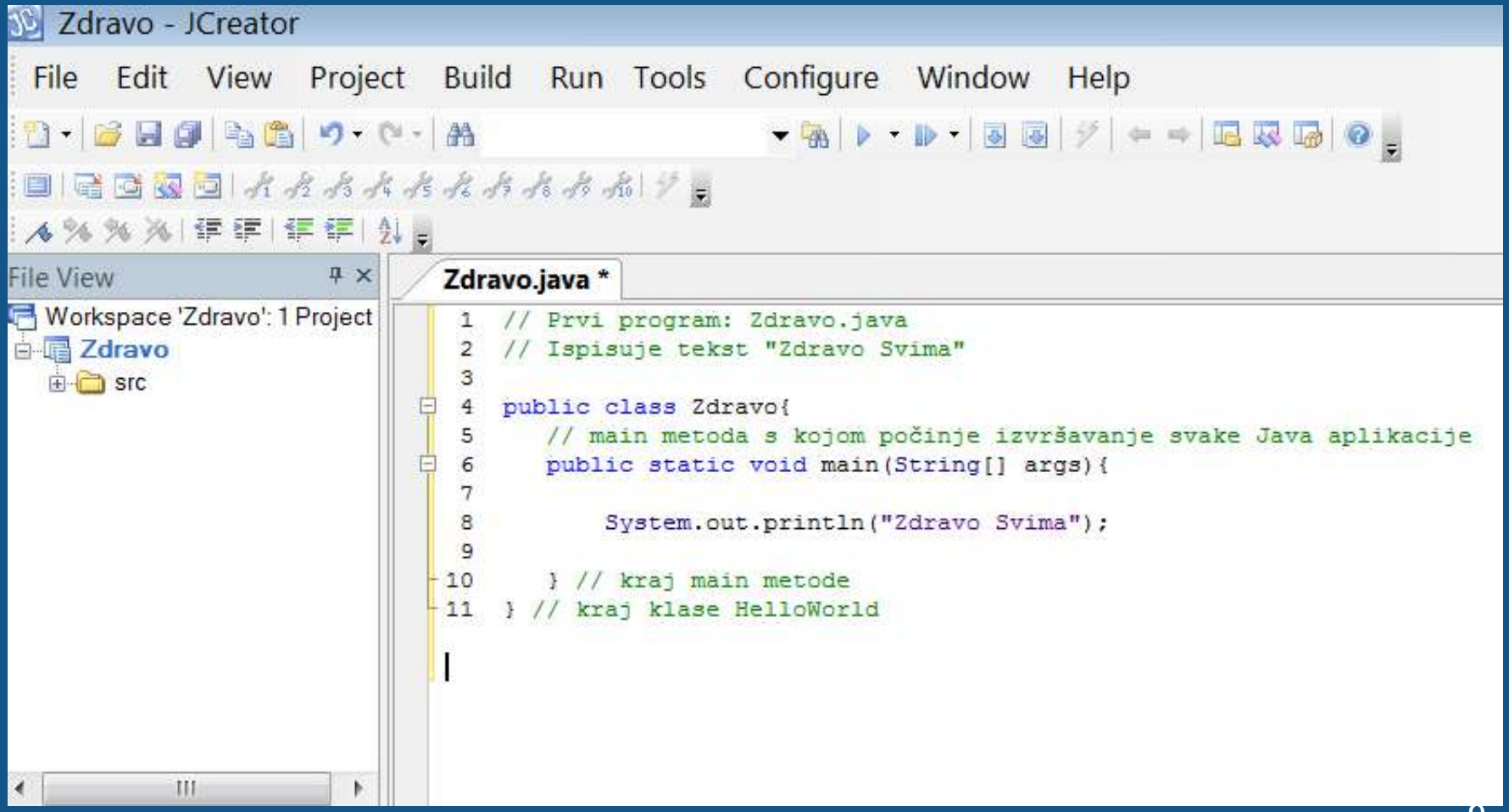
- nakon prevođenja kreira se novi fajl sa ekstenzijom .class

- Izvršavanje

```
java Zdravo
```

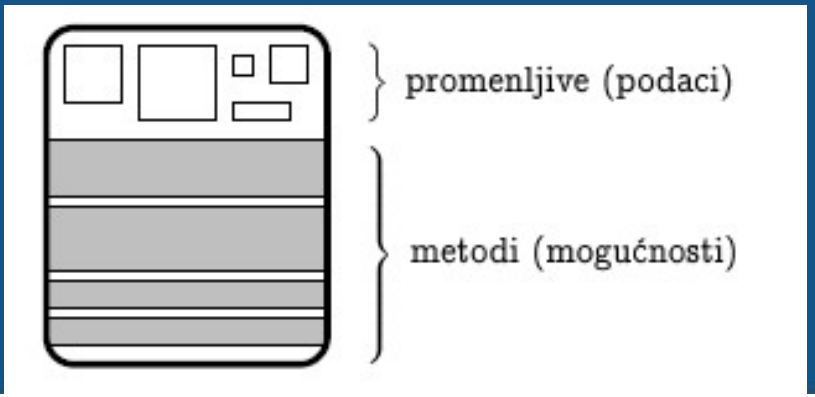
Uvod u programski jezik Java

Grafičko radno okruženje u JCreator-u



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

- Primjer definicije klase

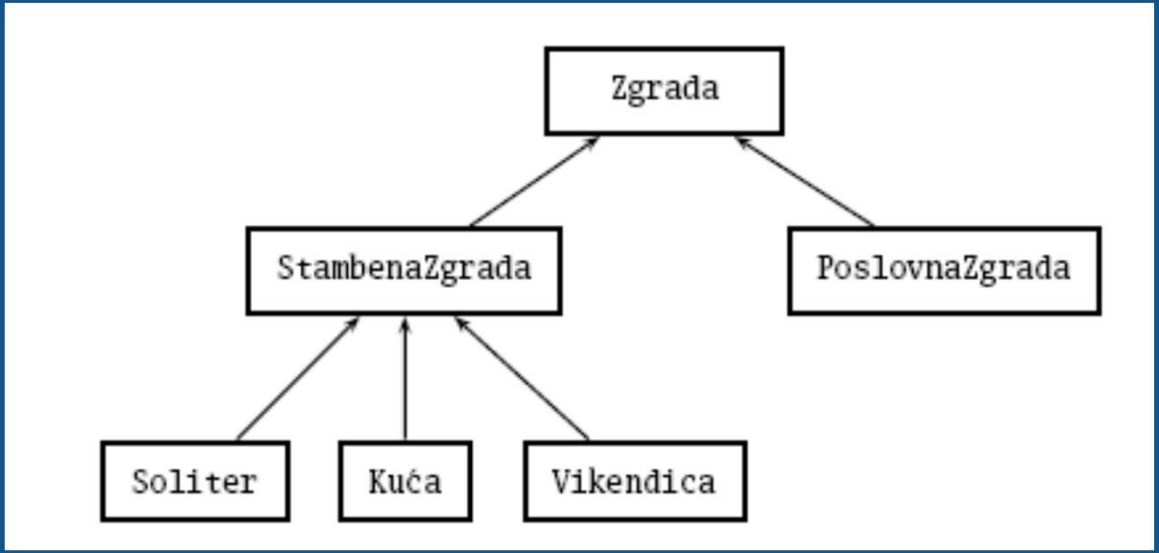
```
class Zgrada
{
int brojSpratova;
String boja;
String adresa;
...
public void okreči() { ... };
public void dozidaj() { ... };
}
```

- *Objekti*

Osnovne karakteristike OOP - Nasljeđivanje

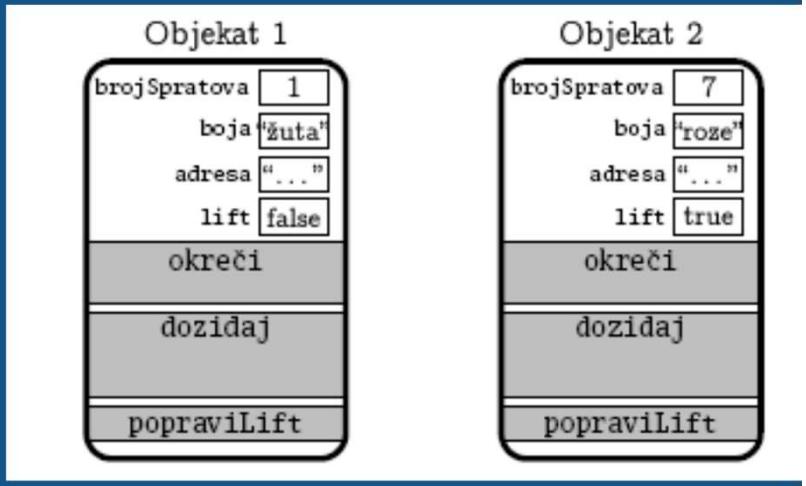
- Način za formiranje novih klasa od postojećih
- Nasljeđivanjem se uspostavlja hijerarhijska relacija između srodnih klasa
- Nova klasa proširuje postojeću klasu i nasleđuje sve atribute i ponašanja postojeće klase
- Terminologija:
  - Bazna klasa — klasa koja se proširuje
  - Izvedena (proširena) klasa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

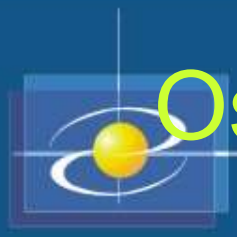
Osnovne karakteristike OOP - Nasljeđivanje



- ***Izvedena klasa***

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



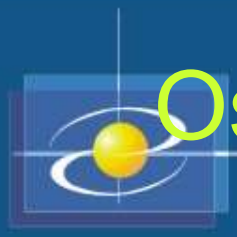


Osnovne karakteristike OOP - Paketi

- 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



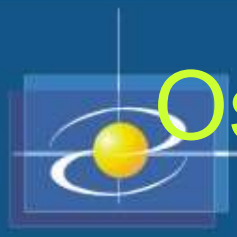
Osnovne karakteristike OOP - Paketi

- 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 podrazumijevanom (anonimnom) paketu

Osnovni elementi Java jezika

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

Osnovni elementi Java jezika - Imena

- Imena za razne elemente Java programa
  - Ime mora da počinje slovom ili _
  - 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 – Promjenljive

- 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;
```

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 ako
>	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.

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;
  ```
  ```

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 …");
    }
```

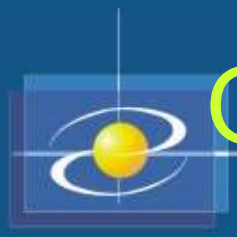
Rezervisana riječ **void** samo saopštava prevodiocu da metoda main() ne vraća nikakvu vrednost.

- 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!**

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)



Osnovne klase u Javi - System

- System.out.print(. . .)
- System.out.println(. . .)
- System.out.printf(. . .)

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

- System.in omogućava unos sa tastature

Osnovne klase u Javi - Math

- 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 - String

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

- s1.length()
- s1.toUpperCase(), s1.toLowerCase()
- s1.equals(s2)
- s1.equalsIgnoreCase(s2)
- s1.charAt (n)
- s1.substring(n,m)

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
  - Naredbe ponavljanja (petlje, ciklusi)

Naredba definisanja (deklarisanja) promenljivih

- Svaka promjenljiva 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;

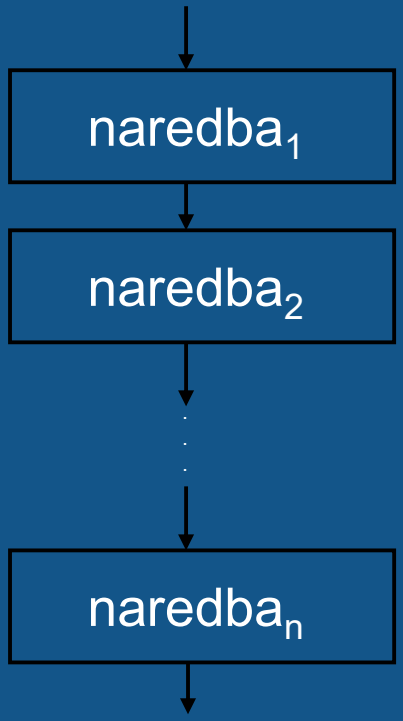
a /= b + c;      a = a / (b + c);

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!
}
```

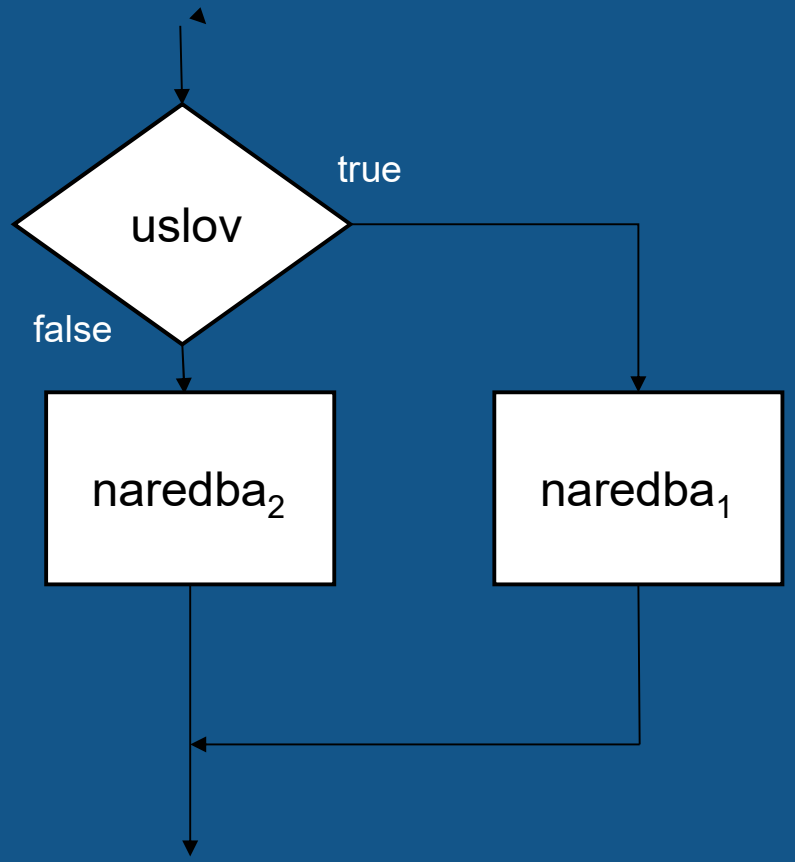
Naredbe kontrole toka programa - naredbe grananja

- **if** naredba
- **if-else** naredba
- složena **if-else** naredba
- **switch** naredba

- Naredba **if-else**

- Format:

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



Naredbe kontrole toka programa - naredbe 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 vrijednost1 : naredba1;
  case vrijednost2 : naredba2;
   ...
  case vrijednostn : naredban;
  default : naredba;     }
```

- Primjer:

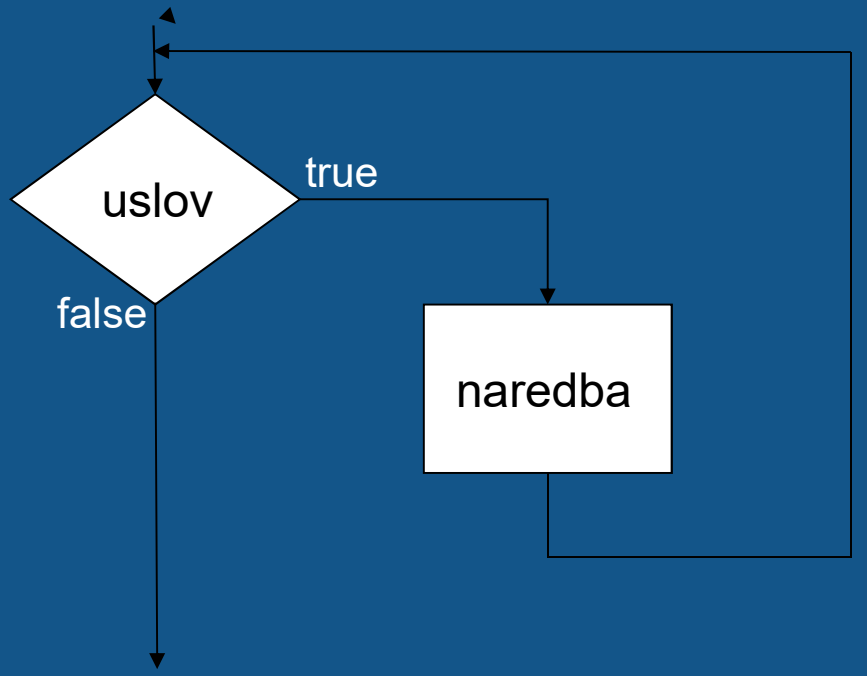
```
switch (brojač) {
  case 1:
    System.out.println("Jedan");   break;
  case 2:
    System.out.println("Dva");     break;
  case 3:
    System.out.println("Tri");     break;
  default:
    System.out.println("Ni jedan, ni dva, ni tri");   break;
}
```

Naredbe kontrole toka programa - naredbe ponavljanja

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

- **while** petlja - format:

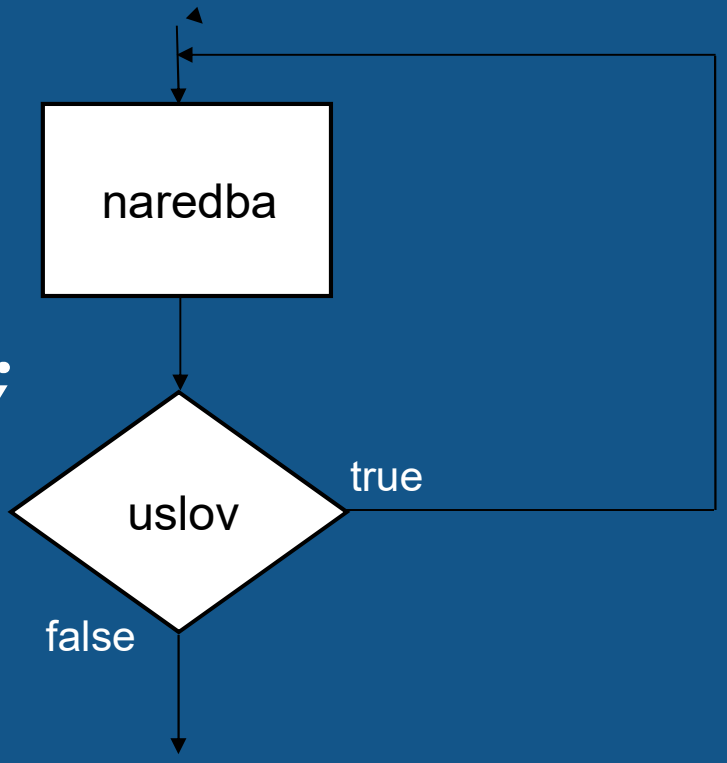
```
while (uslov) {
   naredbe;    }
```



Naredbe kontrole toka programa - naredbe ponavljanja

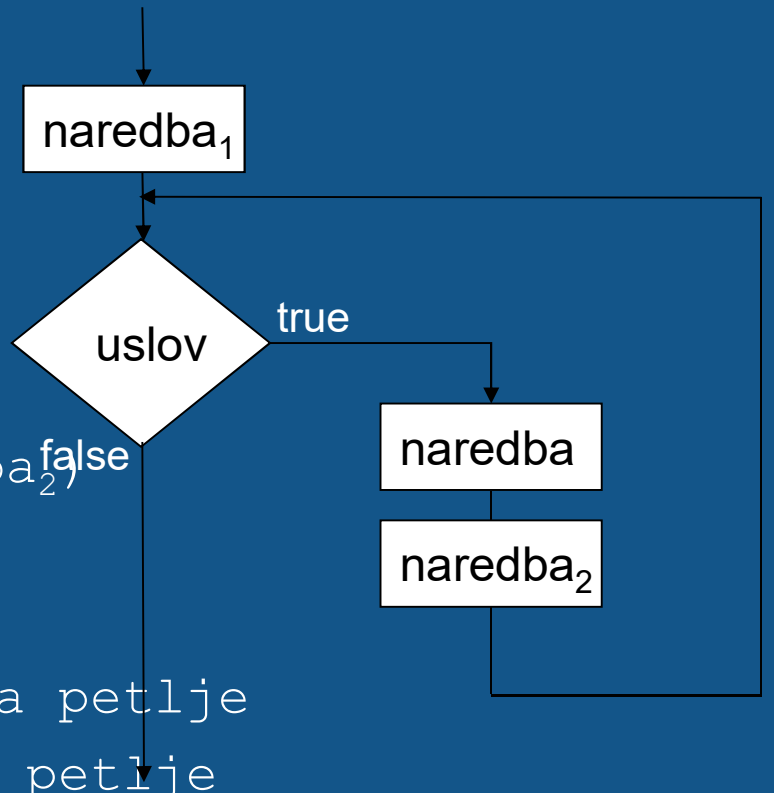
- **do-while** petlja - format:

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



Naredbe kontrole toka programa - naredbe ponavljanja

- **for** petlja - format:




```
for (naredba1; uslov; naredba2)
  naredba;
```

- $naredba_1$ ≈ inicijalizacija petlje
- $naredba_2$ ≈ završnica tela petlje

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 ugnježdenim 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 */
```