

Uvod u objektno programiranje

Uvod u programski jezik
Java

Programski jezik Java

- Java: platforma za izvršavanje programa
- Java: programski jezik

Java kao platforma

- Dizajniran da što manje zavisi od specifičnih karakteristika konkretnog računarskog sistema
- Jednom napisan i *preveden* program se izvršava na bilo kojoj platformi koja podržava Javu

Java kao platforma

- Interpretirani jezik
 - just in time compiler
- Bajt-kod
 - specifikacija je dostupna – više implementacija kompajlera
- Java virtuelna mašina (JVM)
 - specifikacija je dostupna – više implementacija JVM
 - Oracle Java - <https://www.oracle.com/technetwork/java/javase/overview/index.html>
 - Open JDK Java <http://openjdk.java.net/>
 - AdoptOpenJDK <https://adoptopenjdk.net/>

Java kao programski jezik

- Jezik opšte namene
- Objektno-orijentisano programiranje
- Literatura
 - Referentna dokumentacija: <https://docs.oracle.com/en/java/>
<https://openjdk.java.net/guide/>
 - Preporučene knjige:
 - Milosavljević, Vidaković: *Java i Internet programiranje*
 - Bruce Eckel: *Thinking in Java*, <http://www.bruceeckel.com> – besplatna
 - Osnove Java programiranja - Eckel, B., 2006. *Thinking in Java*, 4th ed. Prentice Hall, Upper Saddle River, NJ.
 - Java i structure podataka, sveobuhvatno izdanje - Liang Y., D., 2019. *Introduction to Java Programming and Data Structures, Comprehensive Version, Loose Leaf Edition (12th Edition)*, 12th Edition, ed. Pearson

Sintaksa

- Sve ključne reči programskog jezika Java se pišu malim slovima.
- Svaki Java program se sastoji iz deklaracije paketa, import sekcije, deklaracije atributa i definicije metoda. Sva četiri dela su opcionalna.
- Izrazi u programskom jeziku Java se obično završavaju simbolom tačka-zarez ';'.
- Programski blok čine dve ili više naredbi ili izraza i on je ograđen vitičastim zagradama. Ako imamo samo jednu naredbu ili izraz, ne moramo da je pišemo unutar vitičastih zagrada.
- Postoje dve vrste komentara: jednolinijski i višelinijski. Jednolinijski komentari staju u jedan red i navode se nakon dvostruke kose crte:
`// ovo je komentar`
- Višelinijski komentari mogu da se pružaju u više redova i nalaze se između `/*` i `*/` simbola.
- Postoje i tzv. JavaDoc komentari, koji će biti opisani kasnije, a koji se navode između `/**` i `*/` simbola.

Izvršavanje programa

- Metoda `main()`

Hello.java

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

Prevođenje i pokretanje

- prevođenje:

```
javac Hello.java
```

- pokretanje:

```
java Hello
```


Osnovni koncepti

- Sintaksa: podseća na C++
- Programski blok je ograđen vitičastim zagradama:
{ ... }
- Tipovi podataka
 - primitivni tipovi
 - kao lokalne promenljive i parametri metoda, čuvaju se na steku
 - kao parametri metoda, uvek se prenose po vrednosti!
 - klase
 - instance se zovu objekti i čuvaju se na heap-u
 - postoje samo **reference na objekte**, nikada se ne može pristupiti samom objektu
 - kao lokalne promenljive i parametri metoda, reference se čuvaju na steku
 - kao parametri metoda, uvek se prenose po vrednosti!
- Metode:
 - funkcije i procedure
 - sintaksa: **povratna_vrednost naziv(parametri) { ... }**

Osnovni koncepti

- Primitivni tipovi podataka

Primitivni tip	Veličina	Minimum	Maksimum
boolean	1-bit	–	–
char	16-bit	Unicode 0	Unicode $2^{16} - 1$
byte	8-bit	–128	+127
short	16-bit	-2^{15}	$+2^{15} - 1$
int	32-bit	-2^{31}	$+2^{31} - 1$
long	64-bit	-2^{63}	$+2^{63} - 1$
float	32-bit	IEEE754	IEEE754
double	64-bit	IEEE754	IEEE754
void	–	–	–

Literali

- Celobrojni: `2`, `2000000L`
- Razlomljeni: `3.14f`, `3.14`
- Heksadecimalni: `0xF`, `0xFF`
- Oktalni: `0123`
- Binarni: `0b1001101`
- Boolean: `true`, `false`
- Znakovni:
 - `'A'`
 - `'\n'`
 - `'\xxx'`, gde je `xxx` oktalni ASCII kod karaktera
- String: `"ovo je tekst"`

Deklaracija promenljive primitivnog tipa

- Promenljiva se može deklarirati u bilo kom bloku
– ne mora na početku metode.

int a;

ili

int a = 0;

ili

int a, b;

ili

int a = 0, b = 3;

Implicitna konverzija tipova

- Sa “užeg” ili “manjeg” tipa na “širi” ili “veći” tip.
- Nema gubitka informacije jer “uži” tip podatka staje u “širi” tip podatka.
- Primer:


```
double a;  
int i = 5;  
a = i;
```

Eksplicitna konverzija tipova

- Sa “šireg” na “uži” tip podatka – posledica je gubljenje informacije.
- Primer:

```
double a = 3.14;
```

```
int b = a;
```



Greška pri kompajliranju!

Eksplicitna konverzija tipova

- Pravilna eksplicitna konverzija – upotreba **cast** operatora:
- Primer:

```
double a = 3.14;
```

```
int b = (int)a;
```

Konstante

- Promenljive čija vrednost se ne može menjati
 - zadaje se prilikom deklaracije
- Primer:

```
final int a = 55;
```


Operatori

- Aritmetički operatori
- Relacioni i logički
- Bit-operatori
- Operatori pomeranja
- Operator dodele
- Cast operator

Aritmetički operatori

- Osnovne operacije:

$+$, $-$, $*$, $/$, $\%$

- Umesto $x = x + 1$

$x += 1$

- Automatski inkrement: $++x$ odn. $x++$
 - prefiksni – uvećaj i vrati uvećanu vrednost x

```
int x = 0;  
int y = ++x;  
System.out.println(" x = " + x + ", y= " + y);
```

– Ispis na konzoli $x = 1$, $y = 1$

Aritmetički operatori

- Automatski inkrement: `++x` odn. `x++`
 - sufiksni oblik – vrati vrednost `x` pa teko onda uvećaj `x`

```
int x = 0;  
int y = x++;  
System.out.println(" x = " + x + ", y= " + y);
```

- Ispis na konzoli `x = 1, y= 0`

Aritmetički operatori

$y = 5;$

Operator	Rezultat
$x=y+2$	$X \leftarrow 7$
$x=y-2$	$X \leftarrow 3$
$x=y\%2$	$X \leftarrow 1$
$x=++y$	$X \leftarrow 6, y \leftarrow 6$
$x=y++$	$X \leftarrow 5, y \leftarrow 6$
$x=--y$	$X \leftarrow 4, y \leftarrow 4$

Aritmetički operatori

$x = 10;$

$y = 5;$

Operator	Isto kao	Rezultat
$x=y$		$x \leftarrow 5$
$x+=y$	$x=x+y$	$x \leftarrow 15$
$x-=y$	$x=x-y$	$x \leftarrow 5$
$x*=y$	$x=x*y$	$x \leftarrow 50$
$x/=y$	$x=x/y$	$x \leftarrow 2$
$x\%=y$	$x=x\%y$	$x \leftarrow 0$

Relacioni i logički operatori

- Relacioni: < > <= >= == !=
- Logički: && (I), || (ILI), ! (NE)

Relazioni operatori

`x = 5;`

Operator	Rezultat
<code>==</code>	<code>x == 8</code> je netačno (false)
<code>!=</code>	<code>x != 8</code> je tačno (true)
<code>></code>	<code>x > 8</code> je netačno (false)
<code><</code>	<code>x < 8</code> je tačno (true)
<code>>=</code>	<code>x >= 8</code> je netačno (false)
<code><=</code>	<code>x <= 8</code> je tačno (true)

Logički operatori

- Logički: `&&` `||` `!`
- Rezultat logičkih operatora je tačno (true) ili netačno (false)
- Operandi logičkih operatora su logički izrazi

<code>&&</code>	false	true
false	false	false
true	false	true

<code> </code>	false	true
false	false	true
true	true	true

<code>!</code>	
false	true
true	false

Logički operatori

x = 6;

y = 3;

Operator	Objašnjenje	Primer
&&	konjukcija (and, i)	((x < 10) && (y > 1)) tačno (true)
	disjunkcija (or, ili)	((x==5) (y==5)) netačno (false)
!	negacija (not, ne)	!(x==y) tačno (true)

Bit operatori

- Logičko I nad bitovima: &
- Logičko ILI nad bitovima: |
- Ekskluzivno ILI (XOR) nad bitovima: ^
- Logička negacija nad bitovima -unarni operator: ~
- Kombinacija sa =:

&= |= ^=

Bit operatori

- `a = 3; // a=011 binarno`
- `b = 6; // b=110 binarno`
- `c = a & b;`

011

& 110

010

- `c ← 2;`

Bit operatori

- Ako su operandi bit operatora boolean vrednosti, tada ovi operatori postaju slično kao logički operatori:

```
int a=4, b=8;
```

```
boolean rez11 = (a > 3) & (b > 7); //true
```

```
boolean rez12 = (a > 3) && (b > 7); //true
```

```
boolean rez21 = (a > 3) | (b > 7); //true
```

```
boolean rez22 = (a > 3) || (b > 7); //true
```

```
boolean rez3 = (a > 3) ^ (b > 7); //false
```

Operatori pomeranja

- Shift-ovanje (pomeranje):
 - $a \gg b$ – pomera bitove u a za b mesta
 - ako je a pozitivan, ubacuje 0
 - ako je a negativan, ubacuje 1
 - $a \ll b$ – pomera bitove a u levo za b mesta i ubacuje 0 sa desne strane i ne čuva znak
 - $a \ggg b$ – pomera bitove u a u desno za b mesta i ubacuje 0 bez obzira na znak a.
- Rezultat pomeranja je 32-bitan, osim ako promenljiva koja prihvata rezultat nije long (tada je 64-bitan)!

Pomeranje

- `a = 3; // a = 0000011 binarno`
- `b = a<<2; // b = 0001100 binarno`

- `a = 7; // a = 0000111 binarno`
- `b = a>>2; // b = 0000001 binarno`

Operator dodele

- Osim dodeljivanja literala, promenljivoj se može dodeliti vrednost druge promenljive, ili neki izraz:

```
a = b;
```

```
c = (d * 10) / e;
```

- Ako su operandi primitivni tipovi, kopira se sadržaj:

```
int i = 3, j = 6;
```

```
i = j; // u i ubačeno 6
```

Cast operator

- Ako probamo da smestimo "širi" tip u "uži", to bi proizvelo grešku pri kompajliranju:

```
long a = 44;
```

```
int b = a;
```

- Kompajler ne može da smesti 64 bita u 32 bita i zbog toga prijavljuje grešku. To ćemo postići upotrebom kast (cast) operatora:

```
long a = 44;
```

```
int b = (int) a;
```


Kontrola toka

- `if else`
- `switch`
- `for`
- `while`
- `do while`
- `break`
- `continue`

if

//osnovni oblik

```
if (uslov)  
    akcija
```

//obavezan deo

- ili

```
if (uslov)  
    akcija
```

//obavezan deo

```
else  
    druga_akcija
```

//opcioni deo 0 ili 1

if else

```
int result = 0;
if(testval > target)           //obavezan deo
    result = -1;
else if(testval < target)      //opcionii deo 0 ili N
    result = +1;
else                           //opcionii deo 0 ili 1
    result = 0;
```

if else

```
if (bodovi >= 95)
    ocena = 10;
else if (bodovi >= 85)
    ocena = 9;
else if (bodovi >= 75)
    ocena = 8;
else if (bodovi >= 65)
    ocena = 7;
else if (bodovi >= 55)
    ocena = 6;
else
    ocena = 5;
```

Uslovni operator

```
a = i < 10 ? i * 100 : i * 10;
```

- isto kao:

```
if (i < 10)
```

```
    a = i * 100;
```

```
else
```

```
    a = i * 10;
```

switch

- Izraz u `switch()` izrazu mora da proizvede celobrojnu vrednost.
 - Od Java 1.7 može biti i `String`.
- Ako ne proizvodi celobrojnu vrednost, ne može da se koristi `switch,()` već `if()`!
- Ako se izostavi `break;` propašće u sledeći `case`.
- Kod default izraza ne mora `break;` - to se podrazumeva.

Primer

```
switch (ocena) {  
    case 5: System.out.println("odlican");  
        break;  
    case 4: System.out.println("vrlo dobar");  
        break;  
    case 3: System.out.println("dobar");  
        break;  
    case 2: System.out.println("dovoljan");  
        break;  
    case 1: System.out.println("nedovoljan");  
        break;  
    default: System.out.println("nepostojeca ocena");  
}
```

Primer, bez switch

```
if (ocena == 5)
    System.out.println("odlican");
else if (ocena == 4)
    System.out.println("vrlo dobar");
else if (ocena == 3)
    System.out.println("dobar");
else if (ocena == 2)
    System.out.println("dovoljan");
else if (ocena == 1)
    System.out.println("nedovoljan");
else
    System.out.println("nepostojeca ocena");
```


Primer

```
switch (c) {  
    case 'a':  
    case 'e':  
    case 'i':  
    case 'o':  
    case 'u':  
        System.out.println("samoglasnik");  
        break;  
    default:  
        System.out.println("suglasnik");  
}
```

for

- Za organizaciju petlji kod kojih se unapred zna koliko puta će se izvršiti telo ciklusa.
- Petlja sa početnom vrednošću, uslovom za kraj i blokom za korekciju.

- Opšta sintaksa:

```
for (inicijalizacija; uslov; korekcija)  
    telo
```

for

```
for (int i = 0; i < 10; i++)
```

```
    System.out.println(i);
```

```
for (int i = 10; i >= 1; i--)
```

```
    System.out.println(i);
```

- može i višestruka inicijalizacija i step-statement:

```
for (int i = 0, j = 1;
```

```
i < 10 && j != 11; i++, j++)
```

- oprez (može da se ne završi):

```
for (double x = 0; x != 10; x+=0.1) ...
```

for each petlja

- Za iteriranje kroz nizove (i kolekcije, o čemu će biti više reči kasnije) koristi se for petlja za iteriranje (for each petlja). Opšta sintaksa je sledeća:

```
for (promenljiva : niz)  
    telo
```

- Primer:

```
double[] niz = {1.0, 2.78, 3.14};  
for (double el : niz)  
    System.out.println(el);
```

while

- Za cikličnu strukturu kod koje se samo zna uslov za prekid.
- Telo ciklusa ne mora ni jednom da se izvrši
- Opšta sintaksa:
`while (uslov)`
 telo
- Važno: izlaz iz petlje na false!

while

```
int i = 0;
while (i <= 10) {
    System.out.println("Trenutno
je " + i);
    i=i+1;
}
```

- Važno: izlaz iz petlje na false!

do while

- Za cikličnu strukturu kod koje se samo zna uslov za prekid
- Razlika u odnosu na while petlju je u tome što se telo ciklusa izvršava makar jednom.
- Opšta sintaksa:

do

telo

while (uslov) ;

- Važno: izlaz iz petlje na false!

do while

```
int i = 0;  
do {  
    System.out.println(i++);  
} while (i < 10);
```

- Važno: izlaz iz petlje na false!

break i continue

- break – prekida telo tekuće ciklične strukture (ili case: dela) i izlazi iz nje.
- continue – prekida telo tekuce ciklične strukture i otpočinje sledeću iteraciju petlje.

break i continue

```
for (int i = 0; i < 10; i++) {  
    if (i==7) {  
        break;  
    }  
    if (i == 2)  
        continue;
```

```
    System.out.println("Broj je:" + i);  
}
```

break i continue

```
int i = 0;
System.out.println("Broj je:" + i);
while (i++ <= 9) {
    if (i == 7) {
        break;
    }
    if (i == 2)
        continue;
    System.out.println("Broj je:" + i);
}
```

break i continue

```
int i = 0;
do {
    if (i == 7) {
        break;
    }
    if (i == 2)
        continue;
    System.out.println("Broj je:" + i);
} while (i++ <= 9);
```

Izlaz iz ugnježdene petlje

```
for (...)
{
    for (...)
    {
        ...
        if (uslov)
            break;
    }
}
```