Java Core Programming

Čas 3: Sintaksa programskog jezika Java

ITAcademy

Sintaksa programskog jezika Java (1)

- Sintaksa bilo kog jezika predstavlja skup pravila koje korisnik jezika mora da ispoštuje kako bi ostali korisnici tog jezika mogli da ga razumeju.

- Programski jezik Java, kao i većina drugih modernih jezika, najveći deo sintakse preuzima iz jezika C/C++.
  - Ljudi koji su kreirali Javu izabrali su C++ kao uzor, zato što je ovaj jezik predstavljao jedan od najrasprostranjenijih u tom trenutku, i na taj način su mnogim programerima olakšali učenje ovog jezika.

Sintaksa programskog jezika Java (2)

- C++ i Java su sintaksno veoma slični.

- Ipak, Java omogućava nešto veću slobodu u pisanju. Tačnije, ne obavezuje nas da se bavimo stvarima koje nisu u kontekstu projekta, a moraju se ispoštovati da bi program funkcionisao, kao što je rukovanje memorijom i slično.

Sastavni delovi Java programa

- Svaki program u Javi sastoji se od sledećih elemenata:
  - naredbe
  - blokovi naredbi
  - identifikatori
  - komentari
  - prazni prostori

Naredbe (1)

- Naredba predstavlja najmanji samostalni deo jednog programa.
  - Svaki Java program sastoji se iz jedne ili više naredbi.

- Najčešće pravilo programske sintakse jeste da svaka naredba mora da se završi oznakom ;

- Primer:

```
int x = 1;
```

Naredbe (2)

- U posebnim slučajevima, oznaka ; ne mora da se pojavi na kraju naredbe.

- Ti slučajevi su specifične naredbe za kontrolu toka ili petlje.

- Primer:

```
if (x == 10)
    System.out.println("x je 10");
```

Blokovi naredbi

- Blok naredbi je najjednostavnija vrsta struktuiranog izraza.
- Koristi se da jednostavno okupi niz naredbi vitičastim zagradama.
- Blok naredbi može da se koristi svugde gde može da se javi izraz.
- Primer:

```
{
    // Ovaj blok ispisuje vrednost promenljive x!
    System.out.println("Vrednost promenljive x je " + x);
}
```

Komentarisanje koda

- Komentari koda omogućavaju da u kodu nešto napišemo, ali da to ne bude izvršeno niti tretirano kao deo programa.

- Na taj način vodimo beleške o delovima programa, olakšavamo rad kolegama, a možemo i da privremeno isključimo delove koda.

- Komentari u kodu mogu da se označe na više načina:
  - jednolinijski komentari počinju oznakom //
  - višelinijski komentari se označavaju sa /* */

Osetljivost na mala i velika slova

- Java je osetljiva na mala i velika slova.
  - To znači da identifikator koji je napisan malim slovima ne predstavlja isto što i identifikator napisan velikim slovima.

- Pored osetljivosti na mala i velika slova, moramo da pazimo i da u identifikatore ne smeštamo prazne karaktere, kao i da ih ne započinjemo brojevima.

Primer – površina pravougaonika

```
package cas3;

public class Pravougaonik {

    final static String MERA = "cm";

    public static void main(String[] args) {
        int a = 2;
        int b = 3;

        int povrsina = a * b;

        String poruka = "Povrsina pravougaonika sa stranicama od " + a + MERA
                + " i " + b + MERA + " je " + povrsina + MERA;

        System.out.println(poruka);
    }
}
```

NAZIV PAKETA

OPERATOR DODELE

KONSTANTA

PROMENLJIVA

ARITMETIČKI OPERATOR

OPERATOR KONKATENACIJE

ISPIS NA IZLAZ

Formatiranje izlaza

- Ukoliko nam je potrebno, možemo da formatiramo određeni tip podataka prilikom prikazivanja.
  - Ovo je veoma čest slučaj prilikom prikazivanja brojeva u pokretnom zarezu.

- Primer:

```
double x = 113.0 / 112.0;
System.out.println("Vrednost promenljive x je " + x);
System.out.format("Vrednost promenljive x je %.2f", x);
```

Promenljive

- Promenljive su privremeni kontejneri u koje se skladište neke vrednosti.
  - Nakon što je vrednost uskladištena u memoriju, ona biva reprezentovana kroz promenljivu.

- Sledećom linijom koda predstavljeno je deklarisanje jedne celobrojne promenljive:

```
int promenljiva;
```

- Deklaracija se sastoji iz dva dela.
  - Prvi označava tip podatka koji će promenljiva predstavljati, a drugi predstavlja naziv promenljive.

Konvencije za imenovanje promenljivih

- Iako pri samom imenovanju promenljivih ne moramo poštovati nikakva jezička pravila, poželjno je da se koristi jedna od nekoliko notacija za pisanje.
  - Jedna od najpopularnijih je Camel Case notacija, koja podrazumeva veliko slovo na početku svake reči u promenljivoj, kao na primer:

```
imeMojePromenljive
```

Inicijalizacija promenljivih

- Inicijalizacija promenljive predstavlja dodelu vrednosti.

- Promenljive se mogu prvo deklarisati, a zatim inicijalizovati, ili inicijalizovati prilikom deklaracije:

```
int imeMojePromenljive;
imeMojePromenljive = 1;

int imeMojePromenljive = 1;
```

Tipovi podataka i operatori

- Java je strogo tipiziran jezik jer zahteva da se za svaki podatak u svakom trenutku zna kom tipu pripada.
- Prilikom deklaracije promenljive obavezno je navesti i njen tip.
- U Javi postoje dva tipa podataka:
  - prosti tipovi podataka (int, char, byte, long, float, boolean)
  - složeni tipovi podataka (objekti i nizovi)
- Detaljnije karakteristike prostih tipova podatka prikazane su u tabeli.

Prosti tipovi podataka (1)

Prosti tipovi podataka u Javi				
Tip podataka	Opis	Default vred.	Veličina	Minimalna vrednost Maksimalna vrednost
boolean	Logički tip podatka sadrži true ili false	False	1 bit	- -
char	Karakter (Unicode)	\u0000	16 bita	\u0000 \uFFFF
byte	Ceo broj	0	8 bita	-128 127
short	Ceo broj	0	16 bita	-32768 32767
int	Ceo broj	0	32 bita	-2147483648 2147483647
long	Ceo broj	0	64 bita	-9223372036854775808 9223372036854775807

Prosti tipovi podataka (2)

float	Realan broj sa pokretnim zarezom	0.0	32 bita	+/-1.40239846E-45 +/- 3.40282347E+38
double	Realan broj u eksponencijalnom obliku	0.0	64 bita	+/-4.94065645841246544E-324 +/- 1.79769313486231570E+308

Operatori (1)

JAVA operatori		
Tip operatora	Operator	Opis
Aritmetički operatori	++	Inkrementiranje (unarni operator)
	--	Dekrementiranje (unarni operator)
	+	Sabiranje
	-	Oduzimanje
	*	Množenje
	/	Deljenje
	%	Deljenje po modulu
Operatori poredjenja	==	Jednakost
	!=	Nejednakost
	<	Manje od
	>	Veće od
	<=	Manje ili jednako
	>=	Veće ili jednako

Operatori (2)

Logički operatori	&&	Logičko I
	||	Logičko ukljucujuće ILI
	^	Logicko isključujuće ILI
	!	Negacija

Operatori (3)

Operatori dodele vrednosti	=	Dodela vrednosti
	+=, -=, *=, / =,%= . . .	Dodela vrednosi sa primenom aritmetičke operacije
Operator za String objekate	+	Konkatenacija stringova

Upravljačke strukture

- U Javi postoji šest upravljačkih struktura:
  - Blok naredbi
  - While petlja
  - Do-while petlja
  - For petlja
  - If izraz
  - Switch izraz

- Svaka od ovih struktura smatra se jednim "izrazom", iako se zapravo radi o struktuiranom izrazu koji u sebi može sadržati jednu ili više drugih naredbi.

IF struktura (1)

- Da bi se kod u if strukturi izvršio potrebno je da postavite uslov koji će imati vrednost *true.*

- Postoje 2 oblika if strukture:
  - sa vitičastim zagradama
  - bez vitičastih zagrada

- Oblik bez vitičastih zagrada koristi se kada je kod u u strukturi jednolinijski.

- Ovaj oblik izgleda ovako:

```
if (uslov)
     izraz
```

IF struktura (2)

- Oblik sa vitičastim zagradama se koristi kada u strukturi imamo više izraza za izvršavanje.

- Ovaj oblik izgleda ovako:

```
if (uslov) {
   izraz1;
   izraz2;
   izraz3;
}
```

IF struktura (3)

- Sintaksa strukture if/else:

```
if (izraz ) {
    uslovne_naredbe
} else { // grana nije obavezna
    uslovne_naredbe_2
}
```

- if, else - ključne reči
- izraz - može biti tačan ili netačan
- uslovne_naredbe - izvršavaju se ako je izraz tačan
- uslovne_naredbe_2 - izvršavaju se kada je izraz netačan

IF struktura (4)

- Else struktura se koristi isključivo uz if strukturu i ona sadrži kod koji će se izvršavati ukoliko uslov u if strukturi nije ispunjen.

- Else if struktura ima istu ulogu kao i if struktura, samo što se else if struktura koristi posle if strukture.

- Else if struktura ima isti oblik kao i if struktura, tj. ima zaseban uslov koji se postavlja ako želite da se drugi kod izvrši.

```
if (uslov)
     blok_naredbi_1;
else if (uslov2)
     blok_naredbi_2;
```

IF struktura (5)

- Pomoću posebnih operatora moguće je postaviti više uslova u jednu strukturu, tako da se kod izvrši ukoliko je jedan od tih uslova ima vrednost *true* ili ukoliko sve vrednosti daju *true*.

- To radimo pomoću operatora konjukcije (&&) i disjunkcije (||).

- Ukoliko želimo da postavimo dva uslova u jednu strukturu i da se kod izvršava samo ako su oba uslova tačna koristićemo konjukciju, a ukoliko želimo da postavimo dva uslova i da izvršavamo kod u strukturi samo ako je jedan od ta dva uslova istinit koristićemo disjunkciju (||).

IF struktura (6)

```
public static void main(String[] args) {
    int a = 15;
    int b = 11;

    if (a > 10 && b > 10) {
        System.out.println("Oba broja su veca od 10.");
    } else {
        System.out.println("Oba broja nisu veca od 10.");
    }
}
```

WHILE i FOR petlje

- *While* i *for* petlje se koriste za izvršavanje jednog te istog koda po više puta.
- Koliko puta će se izvršiti kod zavisi od uslova koji ste postavili.
- Razlika između *while* i *for* petlje je u tome što u *for* petlji direktno možete da definišete promenljivu koju ćete da koristite u postavljanju uslova.
- Kod *while* petlje promenljivu koju postavljate u uslovu morate prvo da definišete.

WHILE petlja

- Koliko puta će se izvršiti kod zavisi od uslova koji ste postavili.

- Sintaksa naredbe *while*:

```
while (izraz) {
   naredbe_petlje;
}
```

- izraz - testira se na početku svakog prolaza kroz petlju
- naredbe_petlje - izvršavaju se ako je izraz tačan

FOR petlja (1)

- Sintaksa *for* ciklusa:

```
for (inicijalizacija; izraz; inkrement) {
  naredbe_petlje
}
```

- inicijalizacija - postavlja početnu vrednost izraza
- izraz - testira se na početku svakog prolaza kroz petlju
- inkrement - izvršava se na kraju svakog prolaza kroz petlju.

FOR petlja (2)

- Kod *for* petlje definišemo promenljivu koju ćemo da koristimo za proveravanje uslova i dodeljujemo joj prvu vrednost koju će imati.
- Zatim na mesto uslova postavljamo uslovni izraz pomoću relacijskih operatora.
- Ukoliko je uslov ispunjen izvršiće se kod, a potom će se toj promenljivoj dodeliti nova vrednost koju definišete na mestu inkrementa.
- Nova vrednost je uglavnom operator kojim menjamo vrednost promenljive, a najčešće je to uvećavanje za jedan (i++).

WHILE i FOR petlja

```
public static void main(String[] args) {
    int a = 1;

    while (a <= 5) {
        System.out.println(a);
        a++;
    }

    for (int i = 6; i <= 10; i++) {
        System.out.println(i);
    }
}
```

Zadaci za vežbu

- Napisati program kojim se štampaju brojevi od 1 do 100, kao i njihovi kvadrati.

- Napisati program kojim se štampaju brojevi od 1 do 10, a pored svakog označava da li je paran ili neparan.