

Funkcije

Funkcija (metoda) je izdvojeni skup programskog koda koji se može pozvati (izvršiti) u bilo kom trenutku u programu. Dekompozicijom programskog koda u manje izdvojene celine (funkcije) eliminiše se potreba za njegovim ponavljanjem što doprinosi da sam program bude organizovaniji i čitljiviji. Funkcije se mogu najlakše objasniti ako se posmatraju kao podprogrami (podalgoritmi) specifične namene.

Definicija funkcije:

```
povratni_tip ime_funkcije(parametri) {  
    programski kod  
}
```

Za svaku metodu koju definišemo moramo navesti njeno ime, skup ulaznih parametara i tip povratne vrednosti - rezultata funkcije. Broj i tip ulaznih parametara je proizvoljan, što implicira da je moguće kreirati funkcije koje nemaju ulazne parametre (samo otvorena i zatvorena zagrada se navode u tom slučaju). Svaki od ulaznih parametara definiše se preko njegovog tipa i imena. Svaka funkcija vraća najviše jednu povratnu vrednost čiji tip mora da se poklopi sa navedenim povratnim tipom u definiciji funkcije.

Ulazni parametri mogu biti primitivni tip ili referenca na objekat (klasa, niz, mapa...).

Povratna vrednost može biti primitivni tip, referenca na objekat (klasa, niz, mapa...) ili bez povratne vrednosti – **void**.

Ukoliko funkcija vraća povratnu vrednost, u telu funkcije mora postojati naredba **return** iza koje sledi povratna vrednost. Funkcije se mogu pozivati (izvršiti) u bilo kom trenutku u programu, potrebno je samo da se navede ime funkcije i njene parametre (ukoliko postoje).

Primer

Funkcija koja na konzoli ispisuje tekst "Hello World". Funkcija nema ulazne parametre i nema povratnu vrednost (**void**). Za sada smatrati da su ključne reči **public** i **static** obavezne.

Mesto gde je funkcija napisana u programu se zove **definicija funkcije**:

```
public static void pozdrav() {  
    System.out.println("Hello world");  
}
```

Kako bi smo izvršili kod napisan u funkciji, moramo je pozvati. Funkcije se pozivaju navođenjem imena funkcije i prosleđivanjem ulaznih parametara ukoliko su potrebni (u ovom primeru, funkcija ne prima parametre, pa su zagrade posle imena funkcije prazne).


```
public static void main(String[] args) {  
    pozdrav();  
}
```


Zadatak 1

Kreirati Java projekat *JavaFunkcije* i u sklopu paketa *funkcije.zadatak1* kreirati klasu *Zadatak01*. Napisati funkciju koja na osnovu dužina kateta računa hpotenuzu pravouglog trougla (funkcija sa dva ulazna parametra i povratnom vrednošću tipa **double**). Dužina hipotenuze se računa po formuli:

$$c = \sqrt{a^2 + b^2}$$

Ukoliko se tip povratne vrednost iz tela funkcije ne poklopi sa tipom povratne vrednosti navedenim u naslovu funkcije ili ako implicitna konverzija tipova nije moguća, alat Eclipse će prijaviti grešku. Takođe, do greške će doći i ako se zaboravi naredba **return**.

Promenite tip povratne vrednosti u naslovu funkcije sa tipa **double** na tip **int**. Ukoliko se klikne mišem na ikonicu: , ispisaće se greška: **Type mismatch: cannot convert from double to int**, koja ukazuje da se zahtevani i navedeni tip ne poklapaju.

Zakomentarišite programski kod naredbe **return**. Ukoliko se klikne mišem na , ispisaće se greška: **This method must return a result of type int**, koja ukazuje da naredba za vraćanje vrednosti nije navedena u telu funkcije.

Prenošenje parametara po vrednosti

Sve vrednosti parametara koje se navode u trenutku poziva funkcije prenose se po vrednosti u samu funkciju. Java na stek kopira vrednosti parametara iz poziva funkcije kao vrednosti ulaznih parametrara. Nemoguće je promeniti prosleđenu promenljivu iz poziva funkcije.

Primer

Napisati **void** funkciju **promenaVrednosti** koja prima jedan celobrojni parametar. U telu funkcije, vrednost parametra promeniti na 5 i ispisati je u kozoli.

Unutar **main** funkcije, kreirati jednu celobrojnu promenljivu i dodeliti joj vrednost različitu od 5. Ispisati vrednost promenljive. Pozvati funkciju i kao ulazni parametar proslediti joj kreiranu promenljivu. Posle poziva funkcije ponovo ispisati vrednost promenljive. Prodiskutovati šta se dešava.

Za razliku od ovog pristupa, neki programski jezici koriste prenos parametara po referenci, pri čemu i funkcija koja je pozvana i funkcija iz koje je promenljiva prosleđena pristupaju konkretnoj promenljivoj, a ne njenoj vrednosti.

Method overloading

Method overloading je pojava više funkcija sa istim imenom. U Java programskom jeziku, više funkcija može da ima isto ime ali se moraju razlikovati po broju ili tipovima ulaznih parametara.

Primer

Kreirati funkcije za računanje dužine hipotenuze pravouglog trougla. Jedna funkcija služi za slučaj kada su katete različite i prima 2 celobrojna parametra, dok druga uzima u obzir slučaj kada su dužine kateta iste, i prima jedan celobrojni parametar. Obe funkcije nazvati isto i definisati im **double** kao povratni tip.

Napomena: Funkcija koja računa dužinu hipotenuze za iste vrednosti kateta ne treba da ponovo računa formulu, već da za to iskoristi prvu funkciju.

Zadatak 2

U sklopu projekta *JavaFunkcije* kreirati paket *funkcije.zadatak2* i u njemu klasu *Zadatak02*. U klasi definisati funkcije **odrediX1** i **odrediX2** koje računaju pojedinačne korene kvadratne jednačine. Funkcije pozvati unutar main metode i rešenja ispisati na ekran. Formule za računanje korena kvadratne jednačine su:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{i} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Ispisati rešenja sledeće kvadratne jednačine: $x^2 - 4x - 320$

Petlje i ciklusi

for petlja

Primer **for** petlje koja ispisuje prvih 10 prirodnih brojeva:

```
public static void prirodniBrojevi() {  
    for (int i = 1; i <= 10; i++) {  
        System.out.println(i);  
    }  
}
```

Zadatak 1

U eclipse okruženju kreirati projekat *JavaPetlje* i u okviru paketa *petlje.zadatak01* kreirati klasu *Zadatak01*. U klasi napisati prikazanu metodu **prirodniBrojevi**, ali je izmeniti tako da se ne ispisuju brojevi do 10, već da se granica prosleđuje kao parametar funkcije. Funkciju pozvati iz **main** funkcije i pokrenuti program.

Zadatak 2

U projektu *JavaPetlje* kreirati paket *petlje.zadatak02* i u klasi *Zadatak02* funkciju **forPetljaUnazad** koja ispisuje prvih 10 prirodnih brojeva unazad, od broja 10 do broja 1. Kreiranu funkciju pozvati u main metodi.

while petlja

Služi za ponavljajuće akcije u kojima unapred ne znamo broj iteracija. Ključne reči za kontrolu **while** ciklusa:

break – prekida izvršavanje tekuće iteracije i izlazi iz petlje

continue – prekida izvršavanje tekuće iteracije i prelazi na sledeću

Primer: Funkcija koja ispisuje parne brojeve od 1 do 10 čiji zbir je manji od 15.

```
public static void whilePetlja() {  
    int i=1, suma=0;  
    while(i <= 10) {  
        if(i % 2 != 0) {  
            i += 1;  
            continue;  
        }  
        if(suma+i > 15) {  
            break;  
        }  
        System.out.println(i);  
        suma += i;  
        i += 1;  
    }  
}
```

Nizovi

Niz je kontejnerski objekat koji sadrži fiksni broj elemenata istog tipa. Dužina niza se ustanovljava kada se kreira niz, i od tog trenutka na dalje njegova dužina je nepromenljiva (niz je statička struktura). Svakom elementu niza pristupa se preko indeksa koji određuje njegovu poziciju u nizu. Indeks prvog elementa niza je 0, a svaki sledeći element poseduje indeks uvećan za jedan.

Definisanje

Definisanje promenljive tipa niz se vrši navođenjem tipa i uglastih zagrada nakon čega sledi ime promenljive. Inicijalizacija novog niza se vrši pomoću operatora **new**.

```
// Deklaracija praznog niza čiji su elementi celi brojevi
// Niz sadrži 5 elemenata
int[] niz = new int[5];

// Dodela vrednosti elementima niza
niz[0] = 1;
niz[1] = 2;
niz[2] = 3;
niz[3] = 4;
niz[4] = 5;

// Prilikom definicije niza moguće je odmah odrediti vrednosti elemenata
int[] niz2 = { 1, 2, 3, 4, 5 };

// Elementi niza mogu biti konstante (3), promenljive (broj) ili
// rezultati funkcija

// Niz celih brojeva (int[])
int broj = 2;
int[] celi = { 7, 5, broj, 11 / 3, -13, 1 };

// Niz realnih brojeva (double[])
double[] realni = { Math.PI / 2, 3.0, 2 + 3 / 5, Math.E, Math.sqrt(3) };

/*
 * Niz karaktera (tip char) char vrednosti se u Javi definišu unutar
 * jednostrukih navodnika. Ukoliko se umesto karaktera navede broj, na
 * tom mestu će biti prikazan karakter sa tim brojem u ASCII tabeli.
 * ASCII tabela: http://www.asciitable.com/
 * ASCII kod za slovo 'c' je 99.
 */
char[] karakteri = { 'a', 'b', 99 };
```

Pristup elementima

Pristup elementima niza se vrši navođenjem imena niza i zatim indeksa željenog elementa unutar uglastih zagrada.

```
// Pristup elementu niza
System.out.println("Drugi element niza 'niz' je: " + niz[1]);
System.out.println("Poslednji element niza 'niz' je: " + niz[niz.length-1]);

// Ispis svih članova niza se vrši pomoću for petlje
System.out.println("Elementi niza 'karakter':");
for (int i = 0; i < karakteri.length; i++) {
    System.out.println(karakter[i]);
}
```

Zadaci

1. U eclipse okruženju kreirati projekat *JavaNizovi* i u okviru paketa *nizovi.zadatak01* kreirati klasu *Zadatak01*. U kreiranoj klasi napravite metodu **inicijalizacijaNiza** u kojoj treba da deklarirate niz od 5 celobrojnih elemenata sa sledećim vrednostima: 2, 63, 3, 15 i -4.

Uz pomoć **for** petlje izračunati i ispisati sumu elemenata niza.

2. U projektu *JavaNizovi* kreirati *paket.zadatak02* u njemu klasu *Zadatak02*. U sklopu klase kreirati funkciju **definicijaNiza** u kojoj treba da deklarirate prazan niz od 5 celobrojnih elemenata. Korišćenjem **for** petlje dodelite vrednosti elementima niza tako što će vrednost prvog elementa biti 1, dok je svaki sledeći element uvećan za 10 u odnosu na prethodni. U istoj funkciji ispisati sve članove niza.

Matrice – Dvodimenzionalni nizovi

Matrice predstavljaju dvodimenzionalne statičke strukture. U programskim jezicima, matrice se najčešće definišu kao nizovi čiji su elementi nizovi. Zbog ovoga, sintaksa i pravila rukovanja matricama su ista kao i za nizove, uz dodatak da svaki element opet ima svoje elemente.

Osim očigledne matematičke primene, matrice su vrlo korisne za modelovanje tabelarnih struktura u rukovanju podacima, kao i u prostornom modelovanju.

Definisanje

```
// Definisanje prazne matrice celih brojeva sa 5 vrsta i 4 kolone
int[][] matricaA = new int[5][4];

// Kao i kod nizova, moguće je odmah dodeliti vrednosti elementima
int[][] matricaB = {
    { 8, 45, -1, 0},
    { 58, 2, 5, 9},
    {-4, 34, 2, 78}
};
```

Pristup elementima

Pozicija elementa unutar matrice se određuje pomoću dva indeksa. Jedan određuje broj reda, drugi broj kolone.

```
System.out.println("Element u 1. redu i 2. je: " + matricaB[0][1]);

// Ispis elemenata matrice se vrši pomoću dve for petlje
// Ovo ce ispisati svaki element matrice u posebnom redu
for (int i = 0; i < matricaB.length; i++) {
    for (int j = 0; j < matricaB[i].length; j++) {
        System.out.print(matricaB[i][j]);
    }
}
```

Zadaci

1. U sklopu projekta *JavaNizovi*, u paketu *matrice.zadatak01* kreirati klasu *Zadatak01*. Kreirati funkciju **ispisMatrice** koja će vršiti ispis matrice u konzoli pri čemu će svaki red matrice biti ispisan u posebnom redu, a elementi u kolonama poravnati vertikalno. U **main** metodi pozvati ovu funkciju i proslediti joj proizvoljno kreiranu matricu.

Primer ispisa matrice *matricaB* iz ovog teksta:

8	45	-1	0
58	2	5	9
-4	34	2	78

2. U sklopu projekta *JavaNizovi*, u paketu *matrice.zadatak02* kreirati klasu *Zadatak02*. Kreirati funkciju **ispisGlavneDijagonale** koja će za zadanu matricu ispisati elemente na glavnoj dijagonali. Elementi na glavnoj dijagonali se prepoznaju po tome što su im indeksi vrste i kolone isti ($i == j$).

Primer ispisa glavne dijagonale za matricu *matricaB*:

Elementi na glavnoj dijagonali:
8 2 2

Zadaci za domaći

Kreirati Java projekat *Domaci02* i u njemu za svaki zadatak kreirati poseban paket i klasu.

1. Kreirati funkciju **racunajProizvod** koja korišćenjem for petlje računa proizvod parnih brojeva do broja 10 do broja 20. Dobijeni rezultat vratiti kao povratnu vrednost funkcije i ispisati u funkciji **main**.
2. Napisati program koji za niz { -10, 3, 16, 1, 4, -2} određuje:
 1. Najveći i najmanji element niza, kod napisati u funkciji **najveciNajmanji**, koja treba da prima jedan parametar koji je niz i ispiše najmanju i najveću vrednost, funkcija nema povratnu vrednost.
 2. Napisati funkciju **srednjaVrednost** koja računa srednju vrednost prosleđenog niza.
 3. Program koji ispisuje sve pozitivne elemente niza koji su manji od njegove srednje vrednosti, kod napisati u funkciji **pozitivniElementi**, koja treba da prima dva parametra, prvi parametar je niz, a drugi parametar je srednja vrednost izračunata u funkciji **srednjaVrednost** iz prošlog zadatka.
3. Napisati metodu **racunanjeZbira** u kojoj ce se izračunati zbir elemenata na glavnoj dijagonali ($i=j$) matrice. Funkciji se prosledjuje kreirana matrica.
4. Proširiti prethodni zadatak dodavanjem dodatne metode **proizvodElementa** u kojoj će se izračunati proizvod elemenata iznad glavne dijagonale ($i < j$). Funkciji se prosledjuje kreirana matrica iz prethodnog zadatka.