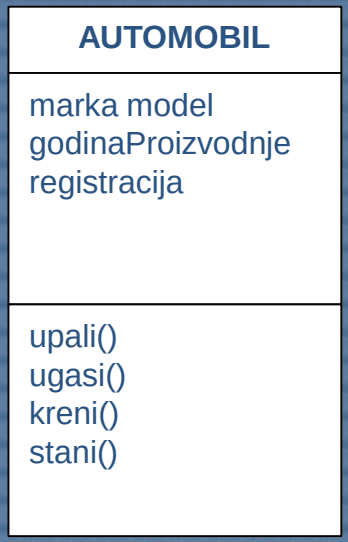
Objektno orjentisano programiranje:

Metode

[Prilagođeno od prof. dr Gordana Đorđević]

Metode



Svaka klasa može imati određena ponašanja koja se mogu izraziti u obliku metode definicija metoda u Javi vrši se u okviru tela klase:

```
tip_povratne_vrijednosti nazivMetode (...parametri...) {
            // telo metoda
    }
```

Tip povratne vrijednosti, njen naziv i parametri čine zaglavlje metode, a kod između vitičastih zagrada telo metode

Prema nepisanom pravilu, nazivi metoda se pišu na isti način kao nazivi atributa (prva reč malim slovom, ostala velikim)

Metode mogu imati ulazne vrijednosti – parametre, kao vrijednosti koje je potrebno poslati metodi da bi ona mogla da se pravilno izvrši.

Ako metoda nema parametre prostor između zagrada se ostavlja prazan

Metode

Pored toga što nešto radi, metoda može i da vraća neku vrijednost kao rezultat izračunavanja (povratna vrijednost metode)

Neke metode nemaju povratnu vrijednost

Kada metoda ne vraća nikakvu povratnu vrijednost, `tip_povratne_vrijednosti` je `void`, a kada vraća vrijednost, `tip_povratne_vrijednosti` može biti bilo koji prost ili složen tip podatka (`int`, `double`, `char`, `boolean`, `String`, ...)

Metoda može da ima samo jednu povratnu vrijednost

Zadatak

primjer kada metoda ne vraća vrijednosti

Napraviti klasu **Televizor**. Ova klasa treba da ima:

- Atribut **jacinaTona** koji je ceo broj i označava trenutnu jačinu tona na televizoru. Početna vrijednost ovog atributa je `0` (ton je utišan do kraja)
- Atribut **trenutniProgram** koji označava broj programa koji je trenutno na televizoru (npr. uključen je program `5`). Početna vrijednost ovog atributa je `1`.
- Atribut **uključen** koji označava da li je televizor uključen ili nije (ako je uključen ima vrijednost `TRUE`, inače ima vrijednost `FALSE`). Smatra se da je na početku televizor isključen.

Kreirajte:

- Metodu **ukljuci** koja uključuje televizor (postavlja vrijednost atributa `ukljucen` na `TRUE`)
- Metodu **iskljuci** koja isključuje televizor (postavlja vrijednost atributa `ukljucen` na `FALSE`)

```
class Televizor {
	int jacinaTona = 0;
	int trenutniProgram = 1;
	boolean ukljucen = false;

	void ukljuci () {
		ukljucen = true;
	}

	void iskljuci () {
		ukljucen = false;
	}

}
```

Dodaj u klasu **Televizor** sledeće metode:

Metodu **pojacajTon** koja povećava vrijednost atributa **jacinaTona** za jedan
Metodu **smanjiTon** koja smanjuje vrijednost atributa **jacinaTona** za jedan
Metodu **iskljuciTon** koja potpuno utišava ton (smanjuje vrijednost **jacinaTona** na 0)

```
Class Televizor {
        int  jacinaTona = 0;
        int  trenutniProgram = 1;
        boolean  ukljucen = false;
        void ukljuci () {
               ukljucen = true;
        }
        void iskljuci () {
               ukljucen = false;
        }
        void pojacajTon () {
               jacinaTona = jacinaTona + 1;
        }
        void smanjiTon () {
               jacinaTona = jacinaTona - 1
        }
        void iskljuciTon () {
               jacinaTona = 0;
        }
}
```

Metode koje vraćaju vrijednosti

Nekada je neophodno da metode vrate trenutnu vrijednost atributa objekta ili vrati rezultat neke računske operacije

U tom slučaju se kao tip povratne vrijednosti piše tip vrijednosti podatka koji će biti vraćen

U okviru tela metode koja vraća vrijednost mora se napisati komanda `return` koja će označiti vrijednost koju je potrebno vratiti

U trenutku kada se izvrši komanda `return`, izvršavanje metode se prekida

Zadatak

primjer kada metoda vraća vrijednosti

Dodati u klasu **Televizor** sledeće metode:

- Metodu **promijeniProgramNavise** koja povećava vrijednost atributa **trenutniProgram** za jedan
- Metodu **promijeniProgramNanize** koja smanjuje vrijednost atributa **trenutniProgram** za jedan
- Metodu **vratiTrenutniProgram** koja vraća vrijednost atributa **trenutniProgram**
- Metodu **vratiJacinuTona** koja vraća trenutnu vrijednost atributa **jacinaTona**
- Metodu **daLiJeUkljucen** koja vraća trenutnu vrijednost atributa **ukljucen**
- Metodu **ispisiParametre** koja ispisuje na ekranu trenutne vrijednosti svih atributa **Televizora** uz odgovarajuću poruku

```
class Televizor {
       int  jacinaTona = 0;
      int  trenutniProgram = 1;
      boolean  ukljucen = false;

          void ukljuci () {
          ukljucen = true;
          }

          void iskljuci () {
           ukljucen = false;
          }

          void pojacajTon () {
           jacinaTona = jacinaTona+ 1;
          }

          void smanjiTon () {
          jacinaTona = jacinaTona - 1;
          }

          void iskljuciTon () {
          jacinaTona = 0;
          }

          void promijeniProgramNavise () {
          trenutniProgram = trenutniProgram + 1;
          }

          void promijeniProgramNanize () {
             trenutniProgram = trenutniProgram -
          1;
          }

          int vratiProgram () {
                  return trenutniProgram;
          }

           int vratiJacinuTona () {
                   return jacinaTona;
           }

           boolean daLiJeUkljucen () {
                   return ukljucen;
           }

                Void ispisiParametre () {
                  System.out.println ("Jacina tona je: "+jacinaTona);
                  System.out.println ("Trenutni program je: "+
                  trenutniProgram);
                  System.out.println ("Televizor je ukljucen: "+ ukljucen);
                 }

                                                  }
```

Pozivanje metode

Da bi metode mogle da se pozivaju, potrebno je napraviti objekat, pa tek onda pozivati neku metodu preko naziva objekta i naziva metode.
Nakon ispisivanja naziva metode koja se poziva, moraju se obavezno pisati zagrade čak i ako nema parametara koji se predaju metodi

```
nazivobjekta.nazivMetode ()
```

Ako metoda ima povratnu vrijednost prvo je potrebno deklarisati promjenljivu koja će primiti povratnu vrijednost metode, pa tek onda pozivati metodu. Tip promjenljive mora biti isti kao i tip povratne vrijednosti metode

```
tip_promjenljive naziv_promjenljive
naziv_promjenljive = nazivobjekta.nazivMetode ()
```

Promjenljive

Promjenljive predstavljaju neki identifikator (ime, slovo, izraz i sl) koji je povezan sa nekom vrijednošću, pri čemu se ta vrijednost može mijenjati

Tip promjenljive odrediće vrijednost koja se može unijeti u promjenljivu

U zavisnosti od toga gdje se deklarišu u kodu i kako se mogu pozvati, postoje sledeće vrste promjenljivih:

- Lokalne promjenljive (ili samo promjenljive) – deklarišu se u okviru tela metode
- Atributi – promjenljive koje direktno pripadaju objektima neke klase; definišu se u okviru tijela te klase
- Globalne promjenljive – promjenljive koje ne pripadaju pojedinačnim objektima neke klase već se koriste na nivou cijelog programa; deklarišu se na sličan način kao i atributi
- Parametri – suštinski parametri metoda predstavljaju promjenljive; deklarišu se u zaglavlju metoda

Zadatak

pozivanje metoda

Iskoristiti klasu Televizor iz prethodnog primjera. Napraviti klasu TestTelevizor koji kreira jedan objekat klase Televizor i poziva neke od njegovih metoda.

Najpre pozvati metodu **ispisiParametre** i uoči početno stanje objekta. Nakon toga pozivaj redom sljedeće metode: **ukljuci, pojacajTon, promijeniProgramNavise, daLiJeUkljucen** i **vratiTrenutniProgram.**

Posle svakog poziva metode, pozvati metodu **ispisiParametre** i uoči promjene u vrijednosti atributa

```
class Televizor {
    int jacinaTona = 0;
    int trenutniProgram = 1;
    boolean ukljucen = false;
    void ukljuci() {
        ukljucen = true;
    }
    void iskljuci() {
        ukljucen = false;
    }
    void pojacajTon() {
        jacinaTona = jacinaTona + 1;
    }
    void stisajTon() {
        jacinaTona = jacinaTona - 1;
    }
    void iskljuciTon() {
        jacinaTona = 0;
    }
    void promeniProgramNavise() {
        trenutniProgram = trenutniProgram + 1;
    }
    void promeniProgramNanize() {
        trenutniProgram = trenutniProgram - 1;
    }
    int vratiTrenutniProgram() {
        return trenutniProgram;
    }
    int vratiJacinuTona() {
        return jacinaTona;
    }
    boolean daLiJeUkljucen() {
        return ukljucen;
    }
    void ispisiParametre() {
        System.out.println ("Jacina tona je: " + jacinaTona);
        System.out.println ("Trenutni program je: " + trenutniProgram);
        System.out.println ("Televizor je ukljucen: " + ukljucen);
        System.out.println ("==========================================");
    }
}
```

```
public   class TestTelevizor {
    public static void main (String [ ] args) {
                Televizor t1 = new Televizor();
                boolean uklj;
                int prog;

                t1.ispisiParametre();

                t1.ukljuci();
                t1.ispisiParametre();

                t1.pojacajTon();
                t1.ispisiParametre();

                t1.stisajTon();
                t1.ispisiParametre();

                uklj = t1.daLiJeUkljucen();
                System.out.println ("Televizor je ukljucen: " + uklj);

                prog = t1.vratiTrenutniProgram();
                System.out.println ("Na televiziji ide program: "+ prog);

                 t1.iskljuci();
                uklj = t1.daLiJeUkljucen();
                System.out.println ("Televizor je ukljucen: " + uklj);
        }
}
```

```
Jacina tona je: 0
Trenutni program je: 1
Televizor je ukljucen: false
==========================================
Jacina tona je: 0
Trenutni program je: 1
Televizor je ukljucen: true
==========================================
Jacina tona je: 1
Trenutni program je: 1
Televizor je ukljucen: true
==========================================
Jacina tona je: 1
Trenutni program je: 2
Televizor je ukljucen: true
==========================================
Jacina tona je: 0
Trenutni program je: 2
Televizor je ukljucen: true
==========================================
Televizor je ukljucen: true
Na televiziji ide program: 2
Televizor je ukljucen: false
```

Metode koje imaju parametre

Parametri metode predstavljaju vrijednosti koje se prosleđuju metodama da bi one mogle pravilno da se izvrše

Metoda može imati nijedan, jedan ili više parametara

Parametri se deklarišu u zaglavlju metode na isti način kao i promjenjljive (prvo tip vrijednosti, pa naziv parametra); ako metoda ima više parametara oni se odvajaju zarezom

```
tip_povratne_vrijednosti nazivMetode (tip_parametra parametar) {
                // telo metoda
                }
```

Pozivanje metoda koje imaju parametre vrši se tako što se u zagradi navode konkretne vrijednosti ili promjenljive. Kada se prilikom pozivanja metode proslede ove konkretne vrijednosti one se nazivaju argumenti.

U okviru poziva metode mora da se ispoštuje broj parametara, njihov tip i redoslijed.

Tako, ako metoda prima dva parametra nekog tipa, svaki poziv te metode mora da bude takav da se metodi prosleđuju tačno dva argumenta koji odgovaraju parametrima po tipu i redoslijedu

```
nazivMetode(argument)
```

Zadatak
primjer metoda sa parametrima

Napravi klasu AutomatNovca. Ova klasa bi trebalo da ima:

Atribut stanje koji predstavlja trenutni iznos novca u automatu (realan broj). Početna vrijednost ovog atributa je 5200.0 KM

Metodu podigniIznos koja prima kao parametar iznos novca koji korisnik želi da podigne (realan broj) i smanjuje vrijednost atributa stanje za taj iznos.

Metodu uloziIznos koja prima kao parametar iznos novca koji korisnik želi da uloži (realan broj) i povećava vrijednost atributa stanje za taj iznos.

Metodu vratiStanje koja vraća trenutni iznos stanja u automatu (vrijednost atributa stanje)

Metodu ispisiStanje koja na ekranu ispisuje koja je trenutna količina novca u automatu (vrijednost atributa stanje)

Napravi klasu ProveraAutomataNovca koja kreira dva objekta klase AutomatNovca. U prvi automat novca je potrebno uložiti 1002.03KM i ispisati stanje prije i poslije ulaganja. Potrebno je i podići 234.55KM iz drugog automata i ispisati stanje automata prije i poslije ulaganja.

```
class AutomatNovca {
    double stanje = 5100.0;

    void podigniIznos(double iznos) {
          stanje = stanje - iznos;
    }
    void uloziIznos(double iznos) {
          stanje = stanje + iznos;
    }
    double vratiStanje() {
          return stanje;
    }
    void ispisiStanje() {
          System.out.println ("Trenutni iznos u automatu je: " + stanje);
    }
}

public class ProveraAutomataNovca {
          public static void main (String [ ] args) {
                    AutomatNovca a1 = new AutomatNovca();
                    AutomatNovca a2 = new AutomatNovca();

                    a1. ispisiStanje();
                    a2. ispisiStanje();
                    System.out.println ("----------------------------------------");

                    a1. uloziIznos(1023.45);

                    a1. ispisiStanje();
                    a2. ispisiStanje();
                    System.out.println ("----------------------------------------");

                    a2. podigniIznos(611.68);

                    a1. ispisiStanje();
                    a2. ispisiStanje();
                    System.out.println ("----------------------------------------");
          }
}
```

```
        AutomatNovca a1 = new AutomatNovca();
        AutomatNovca a2 = new AutomatNovca();

        a1. uloziIznos(1023.45);

        a2. podigniIznos(611.68);
```

Rezultat izvršavanja

```
Trenutni iznos u automatu je: 5100.0
Trenutni iznos u automatu je: 5100.0
---------------------------------------
Trenutni iznos u automatu je: 6123.45
Trenutni iznos u automatu je: 5100.0
---------------------------------------
Trenutni iznos u automatu je: 6123.45
Trenutni iznos u automatu je: 4488.32
---------------------------------------
```

Vidljivost promjenljivih i način prenosa parametara metode

Svi parametri metode i promjenljive koje se deklarišu u okviru tijela metode (lokalne promjenljive) su vidljivi isključivo u okviru tijela te metode i nigde drugo.

Svi atributi klase su uvijek vidljivi u okviru tijela svih metoda te klase.

Konflikt vidljivosti se može javiti ako se u okviru tijela metode koristi parametar ili promjenljiva koji imaju isti naziv kao i neki atribut. U tom slučaju se koristi rezervisana riječ this da izdvoji atribut u odnosu na druge promjenljive.

Primjer vidljivosti promjenljivih i načina prenosa parametara metode

Neka klasa **Kalkulator** ima dvije metode – **saberi** i **obimKruga**. Prva metoda dobija dva cijela broja kao parametre i vraća rezultat sabiranja, dok druga prima poluprečnik kruga kao parametar i vraća obim kruga kao rezultat. Klasa ima i atribut **pi** koji predstavlja vrijednost odgovarajuće matematičke konstante.

```
class Kalkulator {
        double pi = 3.14159;

        int saberi(int a, int b) {
                int rezultat;
                rezultat = a + b;
                return rezultat;
        }

        double obimKruga(double poluprecnik) {
                double rezultat;
                rezultat = 2 * poluprecnik * pi;
                return rezultat;
        }
}
```

Primjer za konflikt vidljivosti

Neka klasa **Osoba** ima atribut ime i metodu **unesiIme** koja kao parametar prima neko ime i atributu dodeljuje tu vrijednost. Neka se parametar ove metode takođe zove ime.

```
class Osoba {
        String ime;

        void unesiIme(String ime) {
        ime = ime;
        }
}
```

```
void unesiIme(String ime) {
        this.ime = ime;
        }
```

this.ime se odnosi na atribut ime

Preklapanje metoda

Nazivi atributa u okviru jedne klase moraju da budu jedinstveni – ne mogu se pojaviti dva atributa sa istim nazivom

Metode u jednoj klasi mogu imati iste nazive, ali moraju imati različite parametre da bi ih Java razlikovala (drugačiju listu parametara – broj parametara i/ili njihov tip)

Primjer za preklapanje metoda

Napraviti klasu Kalkulator koja ima dvije metode sa istim nazivom – saberi. Prva metoda prima dva ceijla broja kao parametre i vraća njihov zbir (ceo broj). Druga metoda prima dva realna broja kao parametre i vraća njihov zbir (realan broj). Napisati i klasu TestKalkulator koja poziva ove metode.

```
class Kalkulator {
        int saberi(int x, int y) {
                int rezultat;
                rezultat = x + y;
                return rezultat;
        }
        double saberi (double x, double y) {
                double rezultat;
                rezultat = x + y;
                return rezultat;
        }
}
public class TestKalkulator {
        public static void main (   String [ ] args) {
        Kalkulator k = new Kalkulator();

        double r1;
        r1 = k.saberi(12.0, 15.0);
        System.out.println (r1);

        int r2;
        r2 = k.saberi(12, 15);
        System.out.println (r2);
        }
}
```

drugačija lista parametara:
broj parametara i/ili
tip parametara

```
        r1 = k.saberi(12.0, 15.0);
        r2 = k.saberi(12, 15);
```

Rezultat izvršavanja

```
27.0
27
```

Globalne promjenljive i globalne metode

Objekti imaju atribute i metode koji se mogu pozivati i koristiti tek kada se objekat inicijalizuje
Objekat ima i svoje zasebne vrijednosti atributa
Ponekad je neophodno da više objekata (iz isith ili različitih klasa) dijeli jednu promjenljivu
Takve promjenljive su vidljive na nivou cijelog programa i zovu se globalne promjenljive
U Javi se globalne promjenljive označavaju rezervisanom riječi **`static`** i pišu se u okviru tijela klase kao atributi klase

```
static tip_vrijednosti nazivPromjenljive
```

Pristup ovoj promjenljivoj realizuje se direktno preko klase

```
NazivKlase.nazivPromjenljive
```

Ponekad postoji i potreba za metodama koje pružaju neku opštu funkcioanlanost koja nije striktno vezana za neku klasu. Ovakve metode u Javi često postaju globalne metode
Globalne metode se u Javi označavaju rezervisanom reči **static** i često se nazivaju statičke metode

```
static tip_vrijednosti nazivMetode (... parametri...)
{
        // telo metode
}
```

Razlika u odnosu na običnu metodu je ta što se **ne mora** inicijalizovati objekat te klase da bi se metoda koristila, već se poziv vrši preko naziva klase

```
NazivKlase.nazivMetode (argumenti);
```

Primer

Napraviti klasu Kalkulator i u njoj definisati dvije globalne metode: saberi i oduzmi. Obe metode primaju dva cijela broja kao parametre, a vraćaju rezultat operacije sabiranja odnosno oduzimanja. Napravi klasu testKalkulator koja poziva ove dvije metode.

```
class Kalkulator {
        static int saberi(int x, int y) {
                int rezultat;
                rezultat = x + y;
                return rezultat;
        }
        static int oduzmi (int x, int y) {
                int rezultat;
                rezultat = x - y;
                return rezultat;
        }
}
public class TestKalkulator {
        public static void main (String [ ] args) {
                int x = 10;
                int y = 5;

                int r1;
                r1 = Kalkulator.saberi(x, y);
                System.out.println (r1);

                int r2;
                r2 = Kalkulator.oduzmi(x, y);
                System.out.println (r2);
        }
}
```

Metode se pozivaju bez kreiranja objekta!!!

```
int x = 10;
int y = 5;

r1 = Kalkulator.saberi(x, y);
r2 = Kalkulator.oduzmi(x, y);
```

Rezultat izvršavanja

```
15
5
```

Konstante

Za definisanje konstanti u okviru neke klase koristi se rezervisana riječ **`final`**

```
final tip_vrijednosti NAZIV_KONSTANTE = vrijednost;
```

Prema nepisanom pravilu nazivi konstante se pišu svim velikim slovima; ako se naziv sastoji od više riječi one se odvajaju znakom donje crte

Konstantama se moraju dodjeljivati vrijednosti odmah prilikom definisanja jer je to kasnije nemoguće

Pozivanje konstante se vrši na isti način kao pozivanje atributa klase; jedina razlika je u tome što konstanti nikada ne može da se dodijeli nova vrijednost

Za neke konstante je neophodno da budu globalno vidljive pa se pored riječi **final** stavlja i reč **static**

```
static final tip_vrijednosti NAZIV_KONSTANTE = vrijednost;
```

Primjer

Napravi klasu MatematickeKonstante. Ova klasa trebalo bi da ima:

- Statičku konstantu PI koja iznosi 3.141592
- Statičku konstantu E koja iznosi 2.71

Napraviti klasu TestMatematickihKonstanti koja na ekranu ispisuje vrijednosti dvije konstante iz klase MatematickeKonstante uz odgovarajuću poruku

```
class MatematickeKonstante {
        static final double PI = 3.141592;
        static final double E = 2.71;
}

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

        System.out.println ("PI = " + MatematickeKonstante.PI);
        System.out.println ("E =   " + MatematickeKonstante.E);
  }
}
```

```
        System.out.println ("PI = " + MatematickeKonstante.PI);
        System.out.println ("E  = " + MatematickeKonstante.E);
```

Rezultat izvršavanja

PI = 3.141592
E = 2.71