**VEŽBA BR. 3**

## Klase, objekti, metode i konstruktori

**Cilj vežbe: Upoznavanje sa osnovnim konceptima OOP**

Kada kažemo klase i objekti tu već pričamo o osnovnim konceptima OOP-a. U ovoj vežbi ćemo objasniti pojam klase i atributa, nakon toga pojam objekta, čime sa postavlja razgraničenje između objekta i klase. Metode i konstruktori, kao elementi klase definišu ponašanje.

### KLASE

Klasa je opšti predstavnik nekog skupa objekata (predmeta ili pojava) koji imaju istu strukturu i ponašanje. Klasa se definiše kao uprošćena slika ovih realnih predmeta i pojava i obuhvata njihove:

- karakteristike (atribute)
- ponašanja (metode)
- odnose sa drugim klasama (relacije).

Atributi, metode i relacije se nazivaju elementi (članice) klase.

```
class NazivKlase {

	//definicije atributa…
	//definicija metoda…

}
```

Definicija klase počinje rezervisanom reči „**class**“, posle koje ide naziv klase. Ovaj prvi red nazivamo i **zaglavlje klase.** Sledeći element je **vitičasta zagrada,** a nakon toga ide **telo klase.** Definicija klase se završava **zatvorenom vitičastom zagradom.**

**Java** je Case Sensitive programski jezik u kome se pravi razlika između malih i velikih slova. To znači da se nazivi klasa, atributa, metoda mogu razlikovati u zavisnosti od toga da li se pišu malim ili velikim slovima.

NAPOMENA!

U Javi postoji nepisano pravilo da se nazivi klasa pišu sa prvim veliki slovom.
Ako se naziv klase sastoji iz više reči, onda se reči pišu spojeno, a prvo slovo svake reči je veliko.
Naziv klase ne sme da ime blanko znak (važi i za nazive atributa, metode).

### Atributi

Atributi predstavljaju neke karakteristike (osobine) klasa. Za klasu **Box**, atributi su:

- width

- height
- depth.

Ove osobine se najčešće mogu izraziti putem nekog broja, slova ili niza slova.

Definisanje atributa izgleda ovako:

```
tip_podataka nazivAtributa;
```

Tip podataka predstavlja skup mogućih vrednosti atributa, ceo broj, realan broj, slovo, niz slova ili nešto drugo.

NAPOMENA!

Prema nepisanom pravilu, nazivi atributa bi trebalo da počinju malim slovom.
Ako se naziv sastoji iz više reči, sve reči se pišu spojeno.
Prva reč se piše malim slovom, a sve ostale velikim.
Definicija atributa jedne klase se završava tačka-zarezom.

Uz atribute obavezno idu i njihovi tipovi podataka koji su dati u sledećoj tabeli:

| ***Naziv tipa podatka*** | ***Opis*** | ***Primeri*** |
|---|---|---|
| `int` | celi brojevi | 1, ’55, 0, 100000 |
| `double` | realni brojevi | 12.55, -234.77, 0.26 |
| `char` | znak (slovo, cifra ili neki drugi znak) | ‘a’, ‘A’, ‘e’, ‘!’, ‘;’, ‘’(blanko znak, ‘4’, ‘9’ |
| `boolean` | logička promenljiva | true, false |
| `String` | niz znakova | “REKA”, “Zarko”, “1234567”, “blu!” |
| `Calendar` | datum i vreme | 2020-01-15 14:44 |

Tipovi kao što su int, double, boolean i char su **prosti tipovi podataka**, dok su String i Calendar složeni tipovi podataka. Svi složeni tipovi podataka se u Javi predstavljaju korišćenjem klasa, te su String i Calendar zapravo dve predefinisane Java klase. Potrebno ke napomenuti da postoje još neki tipovi podataka u Javi, a koji se ne koriste tako često: long (skup celih brojeva čiji je raspon veći u odnosu na int tip podatka), short (skup celih brojeva čiji je raspon manji u odnosu na int tip podatka) i float (skup realnih brojeva, ali sa manje decimalnih mesta nego kod double tipa).

U Javi se karakterne vrednosti moraju pisati pod jednostrukim znacima navoda, dok se String vrednosti obavezno pišu pod dvostrukim znacima navoda.

***Primer 1.*** Napisati klasu **Racunar** koja sadrži sledeće atribute: *takt procesora (realan broj, npr. 3.7 GHz), radna memorija (relan broj, npr. 8.0 Gb), napajanje (ceo broj, npr. 600W).*

```
class Racunar{

	double taktProcesora;
	double radnaMemorija;
	int napajanje;

}
```

Redosled u kojem se definišu atributi u okviru klase nije bitan. Atributima je moguće dodeliti i neku podrazumevanu, početnu vrednost odmah pri definisanju klase.

***Zadatak 1.*** Ispraviti prethodni primer tako da atributi imaju početne vrednosti.

***Zadatak 2.*** Napisati klasu **SmartTV** koja sadrži sledeće atribute:

- Atribut ***ukljucen*** koji označava da li je televizor uključen ili nije (TRUE ili FALSE vrednost). Na početku se smatra da je televizor isključen.
- Atribut ***trenutniProgram*** čija je početna vrednost 1.
- Atribut ***jacinaTona*** koji je ceo broj i njegova početna vrednost je 0.
- Atribut ***jacinaKontrasta*** koji je ceo broj i njegova početna vrednost je 60.

***Zadatak 3.*** Napisati klasu **Student** koja ima atribute:

- Atribut **ime**. Početna vrednost je „nepoznato“.
- Atribut **prezime.** Početna vrednost je „nepoznato“.
- Atribut **pol**. Može imati vrednosti ‘M’ ili ‘Z’.
- Atribut **brojIndeksa** (niz slova).
- Atribut **prosecnaOcena.**

## Objekti

Obejkat predstavlja jedan konkretan primerak (instancu) ili pojavljivanje neke klase. Prema tome, klasa može da se definiše kao skup objekata koji imaju iste osobine.

Da bi objekat mogao da se koristi (da se pozivaju njegove metode, menjaju vrednosti atributa, itd.), potrebno ga je inicijalizovati. Ako se objektu pokuša pristupiti bez inicijalizacije, Java će prijaviti grešku.

Inicijalizacija se vrši korišćenjem naredbe „**new**“ na sledeći **način 1**:

```
NazivKlase nazivobjekta;

nazivobjekta = new NazivKlase();
```

**način 2**:

```
NazivKlase nazivobjekta = new NazivKlase();
```

Šta se zapravo dešava kada se inicijalizuje objekat?

Kada se deklariše neki objekat, samo se stvori pokazivač koji će da referencira objekat. Ali kada se izvrši inicijalizacija, tek onda se alocira deo memorije računara (deo RAM memorije) u kome će da bude objekat i poveže se sa pokazivačem – pokazivač u tom trenutku sadrži adresu memorijske lokacije objekta. Tek nakon ovoga se objekat može koristiti.

ZANIMLJIVOST!

Potpuno je normalna situacija da se u toku izvršavanja nekog ozbiljnijeg programa za kratko vreme stvori (inicijalizuje) na stotine ili hiljade objekata. Svaki od tih objekata zauzima neki deo memorije računara, a često se dešava da mnogi od njih više nisu potrebni, pa bi ih trebalo izbaciti iz memorije. Ako se ovo ne uradi, može doći do preopterećenja memorije.

Pričali smo od o **garabage collection** – u.
On služi za automatsko brisanje nepotrebnih objekata.
Objekat je nepotreban ako na njega ne pokazuje nijedna promenljiva ili pokazivač.
Dovoljno je promenljivoj ili pokazivaču dodeliti **null** vrednost ili inicijalizovati novi objekat preko iste promenljive pa da stari objekat bude automatski izbrisan posle nekog vremena.

Objekat je konkretno pojavljivanje klase koje ima konkretne vrednosti atributa. Da bi se vrednosti atributa (promenljivih) promenile ili pročitale, potrebno im je pristupiti na određen način. Pristup atributima objekta se vrši preko naziva objekta i naziva atributa međusobno odvojenih tačkom.

```
imeObjekta.nazivAtributa;
```

NAPOMENA!

U Javi se komande (kod) izvršava odozgo nadole, odnosno u redosledu u kome su napisane.
Ukoliko se u okviru main metode prvo napišu komande za pristup atributima, a tek posle sledi komanda za inicijalizaciju objekta, java će da prijavi grešku.

**println** – posle ispisa teksta prelazi u sledeći red!
**print** – posle ispisa teksta i dalje ispisuje vrednosti u postojećem redu!

***Zadatak 1.*** Napisati klasu Grad. Ova klasa bi trebvalo da sadrži:

- Atribut ***naziv***. Početna vrednost atributa je “nepoznat”.
- Atribut ***brojStanovnika***. Početna vrednost atributa je 0.

Napisati klasu ***TestGrad*** koja ima main metodu i u okviru nje kreira tri objekta klase ***Grad***: Beograd (1.374.000 stanovnika), Bratislava (424.428 stanovnika) i Njujork.

***Zadatak 2.*** Napisati klasu ***Osoba.*** Ova klasa sadrži:

- Atribut ***imePrezime.*** Početna vrednost atributa je „nepoznato“.
- Atribut ***tezina.*** Težina može imati vrednosti 56.1 kg, 121.2 kg, itd.

Napisati klasu ***TestOsoba*** koja u okviru main metode kreira dva objekta klase ***Osoba***: Pera Peric (92.2 kg) i Mika Mikic (72.9 kg). Ispisati vrednosti atributa na ekranu uz propratni tekst, npr. „Ime osobe je: “ i „Tezina osobe je: “. Ispisati vrednosti atributa u redovima jedan ispod drugog, a zatim ispisati vrednosti atributa u jednom redu.

## Metode

Metode su ponašanja jedne klase.

Definicija metode u Javi je sledeća:

```
tip_povratne_vrednosti nazivMetode(…parametri…){

	//telo metode

}
```

Metoda se sastoji od **zaglavlja**:

- tipa povratne vrednosti iste,
- naziva metode,
- parametara unutar zagrada metode.

i od **tela koje se nalazi unutar vitičastih zagrada!**

**Metodama** se definiše ponašanje klase. Pored toga što nešto radi, metoda može da vraća neku vrednost kao rezultat izvršavanja!

Da bi neki program mogao da se pokrene, potrebno je da ima tzv. **"main"** metodu. Zaglavlje ove metode je uvek isto, a njena definicija se piše u okviru tela klase na sledeći način:

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

	//naredbe

}
```

U slučajevima gde metoda **nema tip povratne vrednosti, odnosno tip povratne vrednosti je "void",** ova metoda vraća samo one vrednosti koje su predefinisane u njoj.

Metoda može da ima samo **jednu povratnu vrednost**, odnosno da vraća jedan tip podataka.

NAPOMENA!

Prema nepisanom pravilu, nazivi metoda se pišu na isti način kao nazivi atributa. Prva reč počinje malim slovom, a ako ima više reči ostale se pišu velikim.

### return

Postoje situacije gde je u metodi potrebno vratiti povratnu vrednost atributa nekog objekta ili vratiti vrednost neke računske operacije. U tom slučaju se kao tip povratne vrednosti ispred imena metode ne navodi **void** vec tip podataka vrednosti podatka koji će biti vraćen.

Pored toga na kraju samog tela metode se mora upisati komanda **return**. Kako bi se označila vrednost koju je potrebno vratiti.

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

### Pozivanje metoda

Da bi se metoda pozvala, prethodno je potrebno kreirati objekat. Razlika između poziva metode i atributa jeste što se u prvom slučaju iza imena metode stavljaju zagrade (čak iako metoda nema parametre).

```
nazivObjekta.nazivMetode();
```

Ako metoda ima povratnu vrednost, prvo je potrebno deklarisati promenljivu koja će da primi povratnu vrednost metode, pa tek onda pozvati metodu.

```
tip_promenljive naziv_promenljive;

naziv_promenljive = nazivObjekta.nazivMetode();
```

### Parametri metoda

Metode mogu da imaju i neke ulazne vrednosti odnosno **parametre.** Parametri predstavljaju one vrednosti koje je potrebno proslediti nekoj metodi da bi ona mogla pravilno da se izvrši. Ako metoda nema parametre prostor između zagrada ostavljamo prazan.

```
tip_povratne_vrednosti nazivMetode(tip_parametra parametar){
```

```
        //telo metode

    }
```

Pozivanje metoda koje imaju parametre se vrši na taj način što se između zagrada navedu neke konkretne vrednosti ili promenljive. Kada se pri pozivu metode kao parametri proslede ove konkretne vrednosti ili promenljive, one se nazivaju **argumenti (realni, tj. stvarni parametri). U okviru poziva ovakve metode, mora da se ispoštuje broj parametara, njihov tip i redosled.** Drugim rečima, ako metoda prima dva parametra nekog tipa, svaki poziv te metode mora da bude takav da se metodi zaista proslede tačno dva argumenta koji odgovaraju ovim parametrima po tipu i redosledu.

```
    nazivMetode (argument);
```

***Primer 2.*** Napraviti klasu ***AutomatNovca*** koja sadrži sledeće atribute i metode:

- Atribut ***stanje*** koji predstavlja trenutni iznos novca u automatu (realan broj) i čija je početna vrednost 7300.0 dinara.
- Metodu ***podigniIznos*** koja prima kao parametar iznos novca koji korisnik želi da podigne (realan broj, npr. 450.2) i smanjuje vrednost atributa stanje za taj iznos.
- Metodu ***uloziIznos*** koja prima kao parametar iznos novca koji korisnik želi da uloži (realan broj) i povećava vrednost atributa stanje za taj iznos.
- Metodu ***vratiStanje*** koja vraća trenutni iznos nova u automatu (vrednost atributa stanje).
- Metodu ***ispisiStanje*** koja na ekranu ispisuje koja je trenutna količina novca u automatu (vrednost atributa stanje).

Napraviti klasu ***ProveraAutomataNovca*** koja kreira dva objekta klse ***AutomatNovca.*** U prvi automat novca je potrebno uložiti 2004.32 dinara, i ispisati stanje pre i posle ulaganja. Potrebno je podići 555.23 dinara iz drugog automata i ispisati stanja pre o posle podizanja.

```
    class AutomatNovca{

        double stanje = 7300.0;

        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);
        }


    }
```

```
class ProveraAutomataNovca{

	public static void main(String args[]){

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

		a1.ispisiStanje();
		a1.uloziIznos(2004.32);
		a1.ispisiStanje();

		a2.ispisiStanje();
		a2.podigniIznos(555.23);
		a2.ispisiStanje();

	}

}
```

***Koja je razlika između parametara (formalnih parametara) i argumenata (realnih parametara) u prethodnom primeru?***

**Konstruktori**

Konstruktor predstavlja posebnu metodu pomoću koje se može izvrđiti dodeljivanje početnih vrednosti atributima objekta prilikom njegove inicijalizacije. Konstruktor klase se **automatski poziva svaki put kada se pozove komanda „new“**. Za razliku od metoda konstruktori imaju dva ograničenja:

- Naziv konstruktora je uvek isti kao i naziv klase (ako se nazove drugačije, Java tu metodu neće smatrati za konstruktor).
- Konstruktori nikada nemaju povratnu vrednost, pa se u okviru zaglavlja ne piše čak ni „void“.

Klasa može imati i više konstruktora, ali se onda moraju razlikovati po parametrima (broj i/ili tipovi parametara). Ovo je preklapanje konstruktora, i slično je preklapanju metoda.

Kada neka klasa nema eksplicitno naspisan konstruktor, Java joj automatski dodeljuje podrazumevani „default“ konstruktor koji ne menja vrednosti atributa niti vrši bilo kakve vidljive promene.

Postoje dve vrste konstruktora:

1. Paramtrizovani konstruktori,
2. Podrazumevani (default) konstruktori.

***Primer 1.*** Napraviti klasu ***PrehrambeniArtikal*** koja ima dva atributa: ***naziv*** i ***kalorijskaVrednost.*** Prvom atributu dodeliti početnu vrednost „nepoznat“ a drugom 0.0.

Napraviti klasu ***PrehrambeniArtikal2*** koja ima dva atributa: ***naziv*** i ***kalorijskaVrednost.*** Napraviti konstruktor koji prvom atributu dodeljuje početnu vrednost „nepoznat“ a drugom 0.0.

```
class PrehrambeniArtikal{

	String naziv = “nepoznat”;
	double kalorijskaVrednost = 0.0;

}

class PrehrambeniArtikal2{

	String naziv;
	double kalorijskaVrednost;

	PrehrambeniArtikal2(){
		naziv = “nepoznat”;
		kalorijskaVrednost = 0.0;
	}

}
```

***Zadatak 1.*** Napraviti klasu ***ZdravstvenaUstanova*** koja ima:

- Atribut ***naziv.***
- Atribut ***adresa.***
- Konstruktor koji nema nijedan parametar i koji predstavlja vrednosti oba atributa na „nepoznato“.
- Konstruktor koji kao parametar prima naziv ustanove i postavlja vrednost atributa naziv.
- Konstruktor koji kao parametre prima naziv i adresu ustanove i postavlja vrednosti oba atributa.
- Metodu ***ispisi***koja na ekranu ispisuje vrednosti oba atributa u jednom redu u formatu: „Naziv: _____ Adresa: _____“.

Napraviti i klasu ***TestZdravstvenaUstanova*** koja kreira četiri objekta klase ZdravstvenaUstanova. Prvi objekat je potrebno kreirati korišćenjem podrazumevanog konstruktora i onda mu, pozivanjem atributa, dodeliti naziv „KCN“ i adresu „Bulevar dr. Zorana Djindjica 48, 18108 Nis“. Drugi objekat kreirati koriscenjem konstruktora koji ima jedan parametar i proslediti mu naziv „Dom zdravlja Nis“. Kreirati treći objekat korišćenjem parametrizovanog konstruktora koji ima dva parametra i dodeliti mu sledeći naziv i adresu: „Institut Niska Banja“ i „Srpskih junaka 2, 18205 Niska Banja“. Posle kreiranja svakog objekta, ispisati vrednosti njegovih atributa na ekranu.

***Zadatak 2.*** Napraviti klasu proizvod koja ima:

- Atribut ***sifra***m npr. 12322299.
- Atribut ***naziv***.
- Konstruktor koji kao parametar prima naziv proizvoda i postavlja vrednost atributa naziv. Naziv parametra konstruktora bi trebalo da bude „naziv“ (isto kao ime atributa).
- Konstruktor koji kao parametre prima naziv i šifru proizvoda i postavlja vrednosti oba atributa. Nazivi parametara bi trebalo da budu isti kao nazivi atributa „naziv“ i „sifra“.
- Metodu ***ipsisi*** koja na ekranu ispisuje vrednosti oba atributa u jednom redu u formatu „Sifra: _____ Naziv: _____“.

Napraviti i klasu ***TestProizvod*** koja kreira dva objekta klase ***Proizvod***. Prvi objekat kreirati korišćenjem konstruktora koji ima jedan parametar i proslediti mu naziv „Stolica“. Kreirati drugi objekat korišćenjem parametrizovanog konstruktora koji ima dva parametra i dodeliti mu sledeći naziv i šifru: „Drveni sto“, 1234. Posle kreiranja svakog objekta, ispisati vrednosti njegovih atributa na ekranu.

**Zaključak**

U Nišu

Potpis