NAPOMENA: Vežbe podrazumevaju da je student ovladao teorijom iz dela Uvod u objektno orijentisano programiranje: "Objektno orijentisano programiranje" (1_uvod.ppt i 2_uvod.ppt)

Objektno

Temin *object-oriented programming* (OOP) predstavlja način razmišljana za rešavanje programerskih problema. Srž ovog načina razmišljanja čini koncept objekta. Neki ovaj način razmišlanja nazivaju još i *class-oriented programming*, zato što se za opis objekta koriste klase. U pomenutom načinu razmišljanja posmatra se svet (domen) u kojem je potrebno rešiti nekakav problem. Analiziranjem sveta cilj nam je da uočimo i pokušamo da opišemo entitete od interesa.

U javi sve počinje sa klasama, sve se nalazi u klasama i sve se bazira na klasama. Sve što napišemo (metode, promenljive…) mora se naći u okviru neke klase (i *main* metoda se nalazi u nekoj klasi!!!). Klasu možemo posmatrati kao apstraktni opis osobina grupe entiteta. Klase se dizajniraju tako da sadrže sve najbitnije osobine entiteta. Instanciranjem klasa nastaju objekti.

Koja je razlika između klase i objeta?

Objekat je stvar, jedan primerak entiteta. Klasa je dizajn šema konstruisana za opis entiteta. Ako posmatramo studente (*Petar Petrović sa indeksom SW 11/2016, Marija Marijanović sa indeksom SW 22/2016...*) na fakultetu, svakog studenta možemo da opisemo sa njegovim imenom i prezimenom i brojem indeksa. Klasa *Student* sa atributima ime i prezime i indeks omogućuje opis svakog studenta na fakultetu, dok bi konkretna pojava studenta npr. Petar Petrović sa indeksom SW 11/2016 predstavljao objekt proizašao instanciranjem klase *Student* sa stvarnim vrednostima.

Dakle, klasa predstavlja model koji se sastoji od atributa i metoda. Atributi mogu biti primitivni tipovi ili reference na objekte drugih klasa. Objekat ne može postojati bez prethodno napisane klase. Kada se napiše klasa moguće je kreirati proizvoljan broj objekta te klase (odnos 1:N). Objekat nastaje isključivo instanciranjem samo jedne klase.

Primer 1. Definisanje klase sa atributom i metodom i testne klase koja instancira objekat prethodno definisane klase.

```
public class NazivKlase {
    // deklarisanje atributa tipa boolean
    boolean nazivAtributa;

    // definicija metode tj funkcije
    void nazivMetode() {
        // promena vrednosti atributa
        nazivAtributa = true;
    }
}

class TestKlasa {
    public static void main(String args[]) {
        // deklarisanje promenljive k tipa NazivKlase
        NazivKlase k;
        // instanciranje objekta tipa NazivKlase
        k = new NazivKlase();
        // pozivanje metode nazivMetode iz klase k
        k.nazivMetode();
    }
}
```

Atributi klasa prilikom deklaracije imaju podrazumevane vrednosti. Za razliku od atributa, lokalne promenljive u metodama nemaju podrazumevane vrednosti i izazivaju grešku prilikom kompajliranja. Podrazumevane vrednosti atributa za različite tipove su date u nastavku:

Tip	**Početna vrednost**
boolean	*false*
char	'\u0000'
byte	(*byte*)0
short	(*short*)0
int	0
long	0L
float	0.0f
double	0.0d
*Object**	*null*

* - pod tipom *Object* se podrazumevaju reference na sve tipove objekata.

Primer 2. Izmodelovati klasu *Student* (ime, indeks, grad) i napisati test klasu koja instancira niz objekata tipa *Student* i ispisuje njihove podatke. Takođe, napraviti dinamički niz objekata tipa *Student* korišćenjem kontejnerskog objekta ArrayList. Komentarisati prednosti i mane oba niza objekata.

Primer 3. Izmodelovati klasu Krug koja predstavlja geometrijsku figuru koja ima svoju površinu i obim. Takođe omogućiti proveru da li se proizvoljna tačka nalazi unutar prostora koji zauzima figura. Krug je u prostoru definisan sa tačkom T(x,y) koja predstavlja centrar kruga i dužinom radiusa. Upotreba ključne reči **this** i atributa kao referenca na drugi objekat.

Modifikatori pristupa definišu dostupnost podataka – klase, atributa, metoda i konstruktora. Time se postiže kontrolisan pristup (čitanje, pisanje) podataka. Postoje četiri tipa java modifikatora pristupa:

- **private** – dostupnost unutar klase
- **protected** – dostupnost unutar paketa i u klasama naslednicama. Primenljivo nad atributima, metodama i konstruktorima, a ne i nad klasama.
- **public** – dostupnost u svim klasama i paketima
- **nespecificirani** – dostupnost unutar paketa (***package-private***)

Radi kontrole pristupa, atributi klase se uglavnom deklarišu sa **private** vidljivošću (pristupom) a klasa obezbeđuje metode za pristup i modifikovanje vrednosti atributa *x*. *Get* metoda (*getX*) obezbeđuje pristup samo za čitanje vrednosti određenog atributa. Isto tako, *set* metoda (*setX*) menja vrednost određenog atributa.

Primer 4. Proširiti klasu *Student* (iz primera 2) atributima i metodama sa različitim pravima pristupa. Takođe obezbediti metode za kontrolisan pristup atributa.

Konstruktor je specijalna metoda klase čije ime je isto kao naziv klase i nema povratan tip. Služi za instanciranje objekta i inicijalizaciju njegovih atributa. Ukoliko se ne definiše ni jedan konstruktor, kompajler će izgenerisati podrazumevani konstruktor. Može se definisati i više konstruktora koji se moraju razlikovati po broju ili tipu parametara.

Primer 5. Unutar klase *Student* (iz primera 4) definisati konstruktore (bez parametara, parametri koji su atributi klase, referenca na objekat) i metode za manipulaciju sa atributima klase.

Klase omotači (*wrapper* klase) omogućavaju objektnu pretstavu primitivnih tipova. Za svaki primitivni tip postoji kalsa omotač koja obezbeđuje mehanizme za rukovanje objektima primitivnog tipa.

Primer 6. Primer korišćenja *Integer* omotač klase primitivnog *int* tipa. Na isti način se koriste i ostale *wrapper* klase.

```
//definicija i inicijalizacija
Integer ceoBrojObjekat = null;
int     ceoBrojPrimitivna = 5;

//kreiranje Wrapper objekta sa parametrom string
ceoBrojObjekat = new Integer("1");
System.out.println("1. Ispis:"+ceoBrojObjekat);

//kreiranje Wrapper objekta Integer sa primitivnim tipom int
ceoBrojObjekat = new Integer(ceoBrojPrimitivna);
System.out.println("2. Ispis:"+ceoBrojObjekat);

//pozivanje statickih metoda Wrapper objekta koje parsiraju String
int ceoBroj = Integer.parseInt("3");
boolean logicka = Boolean.parseBoolean("true");
float   realanBroj = Float.parseFloat("3.14");

//u slučaju pogrešno prosleđene vrednosti metoda izaziva izuzetak
// na ovaj način neće doći do prekida izvršavanja programa
try {
    ceoBrojPrimitivna = Integer.parseInt("3.14");
} catch (Exception e) {
    System.out.println("Greska prilikom parsiranja!");
}
```

Primer 7. Upotreba rezervisanih reči **static** i **final**.

```
public class Brojac {
    public static int vrednost = 1;
    public static final String NAZIV = "Brojac 1";
}

class Test {
    public static void main(String[] args) {
        Brojac br1 = new Brojac();
        Brojac br2 = new Brojac();

        // rad sa statickim atributima
        System.out.println(Brojac.vrednost);
        br1.vrednost = 2;
        System.out.println(br2.vrednost);
        System.out.println(Brojac.NAZIV);
        // greška - nije moguće menjati vrednost finalne promenljive
        Brojac.NAZIV = "Naziv 2";
    }
}
```

Zadaci

Zadatak 1. Po ugledu na klasu Krug (primer 3) definisati Klasu Pravougaonik i odgovarajuće metode potrebne za određivanje površine, obima i pripadnosti tačke pravougaoniku. Testirati navedene metode. Pravougaonik u prostoru je odredjen sa dve tacke A(xA, yA) i B(xB, yB) koje se nalaze na krajevima dijagonale.

Zadatak 2. Kreirati klasu Artikal prodaje i obezbediti da su svi atributi zaštićeni, da postoje više konstuktora (bez parametara, parametri koji su atributi klase, referenca na objekat), korisničke metode i set/get metode za atibute klase.
Klasa Artikal prodaje potrebno je da ima sledeće atribute: šifra (tipa *String*), naziv (tipa *String*), cena (tipa *double*), rasopolživa količina (tipa *int*), opis (tipa *String*), kategorija (tipa *String*). Testirati na primeru prodavnice koja sadrži više artikala i potrebno je omogućiti sledeće opcije:

- Unos novog artikla
- Izmena podataka postojećeg
- Ispis
    - svih podataka o određenom artiklu
    - svih artikala formatu (redni broj, naziv, cena, kolicina i šifra)
    - svih artila koji pripadaju određenoj kategoriji
- Sortiranje artikla po nazivu (opadajuće i rastuće)
- Sortiranje artikla po ceni (opadajuće i rastuće)
- Sortiranje artikla po kategoriji (opadajuće i rastuće)

Zadatak 3. Kreirati klasu Knjiga i obezbediti da su svi atributi zaštićeni, da postoje više konstuktora (bez parametara, parametri koji su atributi klase, referenca na objekat), korisničke metode i set/get metode za atibute klase.
Klasa Knjiga potrebno je da ima sledeće atribute: šifru (tipa *String*), naslov (tipa *String*), godinaIzdanja (tipa *int*), autori (tipa *String* - npr "Petar Petrović, Marko Marković, Filip Filipović"), cena (tipa *double*). Testirati klase na primeru biblioteke i omogućiti izbor 4 osnovne opcije - dodavanje, brisanje, izmena i prikaz.

Zadatak 4. Proširiti prethodni zadatak 3. Izmeniti atribut autori tako da se umesto Stringa koji bi sadržao imena svih autora u tu svrhu koristi listu objekata Autor (tip *ArrayList*<Autor>) pri čemu bi svaki objekat Autor sadržao atribute ime i prezime (tipa *String*), jmbg (tipa *String*) i autorska dela (tipa lista Knjiga). Dodatno omogućiti sledeće opcije:

- pretrage knjiga (po nazivu ili autoru)
- Sortiranje knjiga po godini izdanja (opadajuće i rastuće)
- Sortiranje knjiga po naslovu (opadajuće i rastuće)
- Sortiranje knjiga po jediničnoj ceni (opadajuće i rastuće)

Zadatak 5. Kreirati klase Ocena, Student, Predmet i obezbediti da su svi atributi zaštićeni, da postoje više konstuktora (bez parametara, parametri koji su atributi klase, referenca na objekat), korisničke metode i set/get metode za atibute klase.
Klasa Student potrebno je da ima sledeće atribute: indeks (tipa *String*), ime i prezime (tipa *String*), godina upisa (tipa *int*), ocene (tipa lista Ocena), prosek (tipa double). Klasa Predmet potrebno je da ima sledeće atribute: sifra predmeta(tipa *String*), naziv predmeta (tipa *String*), semestar (tipa *int*), profesor (tipa *String*). Klasa Ocena potrebno je da ima sledeće atribute: predmet (tipa *Predmet*) i ocena (tipa *int*). Testirati klase na primeru fakulteta i omogućiti izbor 4 osnovne opcije - dodavanje, brisanje, izmena i prikaz. Dodatno omogućiti opciju pretrage studenata (po indeksu, godini upisa, proseku i imenu i prezimenu) i predmeta (po šifri, semestru i profesoru). Omogućiti sledeće funkcionalnosti:

1. rad sa predmetima
    a. unos podataka o novom predmetu
    b. izmena podataka o predmetu (odabrati predmet na osnovu šifre)
    c. brisanje podataka o predmetu (logicko brisanje, odabrati predmet na osnovu šifre)
    d. ispis podataka svih predmeta u formi (naziv, šifra, profesor, semestar) u odnosu na status zapisa:
        i. svi predmeti
        ii. samo aktivni
        iii. samo obrisani
    e. ispis podataka o određenom predmetu u formi (naziv, šifra, profesor, semestar) (odabrati predmet na osnovu šifre)
2. rad sa studentima
    a. unos podataka o novom studentu
    b. izmena podataka o studentu (odabrati studenta na osnovu indeksa)
    c. brisanje podataka o studentu (logicko brisanje, odabrati studenta na osnovu indeksa)
    d. ispis podataka svih studenata u formi (broj indeksa, ime i prezime, godina upisa studija, prosek) u odnosu na status zapisa:
        i. svi studenti
        ii. samo aktivni
        iii. samo obrisani
    e. ispis podataka o određenom studentu u formi (broj indeksa, ime i prezime, godina upisa studija, prosek) i ispis položenih ispita za studenta u formi (ocena, predmet, student) (odabrati studenta na osnovu indeksa).
    f. unos ocena studenta (odabrati studenta na osnovu indeksa, odabrati predmet na osnovu šifre predmeta), računanje proseka studenata
    g. sotiranje studenata po
        i. upisanoj godini studija (rastuće)
        ii. imenu i prezimenu (rastuće)
        iii. broju položenih ispita (opadajuće)
        iv. prosecnoj oceni (opadajuće)