Objektno-Orijentisano programiranje

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 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.

Razlika između klase i objekta

Objekat je stvar, jedan primerak entiteta. Klasa je dizajn šema konstruisana za opis entiteta. Ako posmatramo studente na fakultetu, svakog studenta možemo da opisemo sa njegovim imenom i prezimenom i brojem indeksa (Konstantin Srdanov sa indeksom RS 1/2011, Strahinja Baki sa indeksom RS 2/2011...). Klasa Student sa atributima ime i prezime i indeks omogućuje opis svakog studenta na fakultetu, dok bi konkretna pojava studenta npr. Konstantin Srdanov sa indeksom RS 1/2011 predstavljao objekt proizašao instanciranjem klase Student sa stvarnim vrednostima. 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 jedne klase, nemoguće ja da objekat nastane kombinacijom više klasa.

Svaka klasa se sastoji od atributa i metoda. Atributi su promenljive unutar klase, mogu biti primitivni tipovi ili reference na objekte drugih klasa. Metode su funkcije definisane unutar klase. One pripadaju klasi u kojoj su napisane.

Primer klase Student koja modeluje studenta na fakultetu:

```
public class Student {

	// Atributi klase Student
	String index;
	String ime;
	String grad;
	int[] ocene;

	// Metode klase
	double izracunajProsek() {
		int zbirOcena = 0;
		for (int i = 0; i < ocene.length; i++) {
			zbirOcena += ocene[i];
		}
		return zbirOcena / (double)ocene.length;
	}
}
```

Rukovanje objektima

U Objektno-orijentisanom projektu, klase pišemo kako bi smo modelovali entitete iz sistema koji želimo da predstavimo ili kako bismo modularizovali skupove funkcionalnosti. U projektu postoji samo jedna klasa koja sadrži **`main`** metodu, i nju ćemo koristiti prilikom pokretanja projekta.

Za potrebe testiranja nove klase *Student*, kreiraćemo klasu *StudentiMain*, u kojoj ćemo instancirati nekoliko objekata klase *Student*. Objekti se kreiraju upotrebom operatora **`new`**.

```
public class StudentiMain {

	public static void main(String[] args) {

		// Kreiranje objekta klase Student
		Student studentA = new Student();
		// Dodela vrednosti atributa
		studentA.index	= "RS 01/2016";
		studentA.ime	= "Marko Markovic";
		studentA.grad	= "Novi Sad";
		studentA.ocene	=  new int[] {9, 10, 7, 10};


		// Pozivanje metode
		System.out.println("Prosek studenta " + studentA.ime + " je: " +
				studentA.izracunajProsek());

		// Kreiracemo jos jedan objekat klase Student
		Student studentB = new Student();
		studentB.index  = "RS 02/2016";
		studentB.ime    = "Jovana Jovanovic";
		studentB.grad   = "Novi Sad";
		studentB.ocene  =  new int[] {10, 7, 8, 9};

		System.out.println("Prosek studenta " + studentB.ime + " je: " +
				studentB.izracunajProsek());
	}
}
```

Nakon pokretanja, na konzoli dobijamo sledeći ispis:

```
Prosek studenta Marko Markovic je: 9.0
Prosek studenta Jovana Jovanovic je: 8.0
```

Kreirani objekti su, u suštini, promenljive u našem programu (u ovom slučaju, lokalne promenljive **`main`** metode klase *StudentiMain*), tako da je, prilikom instanciranja, svakom objektu potrebno dodeliti jedinstveno ime. Pristup metodama i atributima svakog od objekata se vrši pomoću operatora pristupa "**.**" (eng. *dot operator*, *member operator*).

Konstruktori

Kao što se vidi iz primera sa klasom *Student*, da bi smo kreirali jedan objekat klase, potrebno je prvo da napišemo naredbu za instanciranje, potom moramo pojedinačno da postavljamo vrednosti atributima. Ovo može da predstavlja problem za klase sa velikim brojem atributa. Mehanizam konstruktora u objektnom programiranju nam omogućava da prilikom kreiranja objekta navedemo vrednosti za njegove atribute, pri čemu se potpuno instanciranje obavlja sa jednom naredbom.

Konstruktori su posebne funkcije unutar klase koje se pozivaju kada se kreiraju objekti te klase (upotrebom operatora **`new`**). Prilikom pisanja konstruktora, moramo se pridržavati sledećih pravila:

1. Konstruktori se zovu isto kao i klasa
2. Konstruktori nemaju povratnu vrednost

Klasa može da ima proizvoljan broj konstruktora koji se razlikuju po broju parametara. Parametri konstruktora nam daju slobodu da kontrolisemo kreiranje objekata prema našim potrebama.

Konstruktor bez parametara

Tzv. *podrazumevani konstruktor* (eng. *default constructor*).

Ukoliko ne napišemo ni jedan konstruktor za našu klasu, Java obezbeđuje da možemo da koristimo ugrađeni konstruktor bez parametara (kao što je bio slučaj sa klasom *Student*), koji postavlja vrednosti svih atributa na **`null`**. Preporuka je da se za svaku od klasa napiše konstruktor bez parametara u kojem se vrednosti atributa postave na neke inicijale vrednosti (“” za **`String`** atribute, 0 za brojne vrednosti, **`false`** za **`boolean`**, …) kako bi se izbegle nepostojeće vrednosti. Primer konstruktora bez parametara za klasu *`Student`*.

```
public Student() {
	this.brojIndeksa = "";
	this.ime = "";
	this.grad = "";
	this.ocene = new int[4];
}
```

Ključna reč **`this`** unutar konstruktora se odnosi na trenutni objekat klase (odnosno na objekat koji se kreira) i služi za razlikovanje atributa klase od parametara konstruktora.

Nakon poziva ovog konstruktora:

```
Student student = new Student();
```

dobijamo novi objekat klase Student, čiji atributi imaju vrednost navedenu u telu konstruktora (`brojIndeksa = ””, ime = “”, grad = “”, …`).

Konstruktori sa parametrima

Konstruktori sa parametrima nam omogućavaju da prilikom kreiranja objekta klase navedemo vrednosti za odabrane (ili sve) atribute novog objekta. Ovo u velikoj meri ubrzava kreiranje novih objekata.

Primer konstruktora sa parametrima za klasu Student:

```
public Student(String brojIndeksa, String ime, String grad, int[] ocene) {
	this.brojIndeksa = brojIndeksa;
	this.ime = ime;
	this.grad = grad;
	this.ocene = ocene;
}
```

Sada, objekte klase Student možemo kreirati na sledeći način:

```
Student student = new Student("RS 02/2016", "Jovana Jovanovic", "Novi Sad", new int[]
{10, 7, 8, 9});
```

Obično je praksa da se za klasu navede barem jedan konstruktor sa parametrima koji, ukoliko je moguće pokriva sve atribute klase, međutim, klase mogu da imaju proizvoljan broj konstruktora sa parametrima.

Napomena: Imena parametara konstruktora ne moraju da se poklapaju sa imenima atributa klase.

Osim postavljanja vrednosti atributa, unutar konstrktora možemo da napišemo proizvoljnu Java logiku koja će se izvršiti prilikom kreiranja objekta.

```
public Student(String brojIndeksa, String ime, String grad, int[] ocene) {
	this.brojIndeksa = brojIndeksa;
	this.ime = ime;
	this.grad = grad;

	for (int i = 0; i < ocene.length; i++) {
		if(ocene[i] < 5 || ocene[i] > 10) {
			System.out.println("Ocena " + ocene[i] + " nije validna.");
		}
	}
	this.ocene = ocene;
}
```

Konstruktor kopije

Posebna vrsta konstruktora sa parametrima. Služe za kreiranje novih objekata na osnovu postojećeg objekta iste klase.

Primer konstruktora kopije za klasu Student:

```
public Student(Student original) {
        this.brojIndeksa = original.brojIndeksa;
        this.ime = original.ime;
        this.grad = original.grad;
        this.ocene = original.ocene;
}
```

Zadaci

1. Kreirati projekat *UvodUObjektno* i unutar paketa *objektno.geometrija* napisati klasu *Tacka* koja predstavlja tačku u postoru. Svaka tačka je određena x i y koordinatom. Za klasu napisati konstruktor bez parametara koji vrednosti atributa postavlja na 0 kao i konstruktor sa parametrima koji će služiti za postavljanje vrednosti sva tri atributa.

2. Unutar paketa *objektno.geometrija* napisati klasu *Krug* koja predstavlja geometrijski lik kruga. Krug je određen tačkom koja je centar kruga i poluprečnikom. Za klasu napisati konstruktor bez parametara koji vrednosti atributa postavlja na 0 kao i konstruktor sa parametrima koji će služiti za postavljanje vrednosti sva tri atributa. U klasi napisati metode za računanje obima i površine kruga.

3. Klasu *Krug* proširiti metodom **`tackaPripadaKrugu`** koja proverava da li prosleđena tačka pripada krugu. Metoda treba da vrati logičku vrednost **`true`** ukoliko se tačka nalazi unutar kruga, u suprotnom **`false`**. Određivanje pripadnosti tačke krugu se vrši tako što se odredi udaljenost tačke od centra kruga po formuli:

   $$udaljenost = \sqrt{(tackaX - centarX)^2 + (tackaY - centarY)^2}$$

   Ukoliko je ova udaljenost veća od poluprečnika kruga, tačka ne pripada krugu. Ukoliko je manja ili jednaka, smatra se da tačka pripada krugu.

4. U sklopu paketa *objektno.geometrija k*reirati klasu *GeometrijaTest* sa **`main`** metodom u kojoj treba konstruisati dva objekata klase *Krug* i dva objekta klase *Tacka*. Ispisati vrednosti atributa ovih objekata i proveriti da li tačke pripadaju krugovima. Primer ispisa:

```
Krug 1: centar = (10.0, 5.4), poluprecnik = 45.0
      Krug 2: centar = (0.2, 15.0), poluprecnik = 3.0
      Tacka 1: 2.0, 4.0
      Tacka 2: 6.0, 10.0
      Tacka 1 pripada krugu 1: true
      Tacka 2 pripada krugu 1: true
      Tacka 1 pripada krugu 2: false
      Tacka 2 pripada krugu 2: false
```

Statičke metode

Statičke metode se definišu navođenjem ključne reči **`static`** pre imena metode. Statičke metode se pozivaju samostalno, bez prethodno kreiranog objekta. Ukoliko se statička metoda poziva iz neke druge klase, potrebno je u pozivu navesti ime klase.

Statičke metode nemaju pristup atributima i metodama klase koji nisu statički. Obično se koriste za smeštanje konstanti na nivou projekta ili za pružanje određenih funkcionalnosti nad već predefinisanim podacima.

Primer statičke metode u klasi *Krug*:

```
static void identifikujSe() {
	System.out.println("Ovo je krug!");
}
```

Primer poziva iz druge klase:

```
Krug.identifikujSe();
```

Osim metoda, i atribute klase možemo proglasiti statičkima.

Zadaci za domaći

1. Kreirati klasu *Pravouganik*. Pravougaonik je određen dužinama stranica a i b, kao i x i y koordinatama gornjeg levog ugla. Za klasu napisati konstruktor bez parametara koji vrednosti atributa postavlja na 0 kao i konstruktor sa parametrima koji će služiti za postavljanje vrednosti svih atributa. U klasi napisati metode za računanje obima i površine.

2. Kreirati klase *Ocena*, *Student*, *Predmet.*

   - Klasa *Student* potrebno je da ima sledeće atribute: *indeks* (tipa **`String`**), *imeIPrezime* (tipa **`String`**), *godinaUpisa* (tipa **`int`**), *ocene* (niz tipa **`int`**).

   - Klasa *Predmet* potrebno je da ima sledeće atribute: *sifraPredmeta*(tipa **`String`**), *nazivPredmeta* (tipa **`String`**), *semestar* (tipa **`int`**), *profesor* (tipa **`String`**).

   - Klasa *Ocena* potrebno je da ima sledeće atribute: *student* (tipa **`Student`**), *predmet* (tipa **`Predmet`**) i ocena (tipa **`int`**). Za sve klase kreirati konstruktore bez parametara, sa parametrima i konstruktor kopije.

3. Kreirati klasu *Fakultet* koja ima atribute *naziv* (tipa **`String`**), *adresa* (tipa **`String`**), *studenti* (niz tipa **`Student`**), *predmeti* (niz tipa **`Predmet`**) i ocene (niz tipa **`Ocena`**). Za klasu napisati konstruktore bez parametara, sa parametrima i konstruktor kopije. U klasi *FakultetMain*, u **`main`** metodi, kreirati nekoliko objekata klasa *Student, Predmet* i *Ocena* i jedan objekat klase *Fakultet*. Ubaciti objekte klase *Student* u niz *studenti* na fakultetu, objekte klase *Predmet* u niz *predmeti*, a objekte klase Ocena u niz *ocene* kreiranog fakulteta. Ispisati sve podatke o kreiranom fakultetu. Primer ispisa:

```
-----------------------------------------------------
Fakultet Tehnickih Nauka
Trg Dositeja Obradovica 6, Novi Sad
-----------------------------------------------------
Studenti na fakultetu:
       RS 1/2016 Marko Markovic
       RS 2/2016 Vesna Simic
       RS 3/2016 Jelena Markovic
       RS 4/2016 Marko Simic

Predmeti na fakultetu:
       P1 Telekomunikacije(Nikola Tesla)
       P2 Uvod u psihoanalizu(Slavoj Zizek)
       P3 Osnove kosarke(Magic Johnson)
       P4 Primenjena matematika(Stevan Mokranjac)

Student Marko Markovic iz predmeta Osnove kosarke ima ocenu: 10
Student Marko Simic iz predmeta Primenjena matematika ima ocenu: 7
Student Vesna Simic iz predmeta Osnove kosarke ima ocenu: 8
Student Jelena Markovic iz predmeta Telekomunikacije ima ocenu: 9
Student Jelena Markovic iz predmeta Primenjena matematika ima ocenu: 6
Student Marko Markovic iz predmeta Uvod u psihoanalizu ima ocenu: 10
```