Objektno orijentisano programiranje
vežbe

Staša Vujičić Stanković

"Copy – konstruktor"

- Iznenađujuće često javlja se potreba za pravljenjem identične kopije objekta.
- Ako kreiramo objekat klase *Sphere* sa:

  Sphere eightBall =
  new Sphere(10.0, 10.0, 0.0);

- Ako kasnije u programu želimo da kreiramo objekat *newBall* identičan objektu *eightBall* i napišemo

  Sphere newBall = eightBall;

  to neće biti ono što želimo, već će samo obe promenljive *newBall* i *eightBall* referisati na isti objekat.
  Kreira se nova promenljiva *newBall,* ali ne poziva se konstruktor i ne kreira se novi objekat.

- To se može postići dodavanjem novog konstruktora koji kao argument prima postojeći objekat klase Sphere i kopira vrednosti instancnih promenljivih objekta koji je prosleđen kao argument u odgovarajuće instancne promenljive novog objekta.

"Copy – konstruktor"

- Primer:

```
Sphere(final Sphere oldSphere) {
    radius = oldSphere.radius;
    xCenter = oldSphere.xCenter;
    yCenter = oldSphere.yCenter;
    zCenter = oldSphere.zCenter;
    ++broj;
}
```

Paketi

- Paket predstavlja jedinstveno imenovanu kolekciju klasa.
- Imena klasa jednog paketa neće se mešati sa imenima klasa nekog drugog paketa jer se imena klasa paketa kvalifikuju imenom tog paketa. Npr. puno ime klase *String* iz paketa *java.lang* je *java.lang.String*.
- Ako bismo definisali svoju klasu sa istim imenom, *String*, korišćenje imena *String* odnosilo bi se na tu našu klasu, dok bismo se standardnoj klasi *String* morali obraćati sa *java.lang.String*.

Paketi

- Svaka klasa u Javi je sadržana u nekom paketu, a ako se ne navede eksplicitno, naše klase se smeštaju u podrazumevani (default) paket koji nema ime.
- Standardne Java klase su sadržane u paketima tako što se u jednom paketu nalaze klase koje su na neki način povezane.

Paketi

- Stavljanje klase u imenovani paket je jednostavno.
- Samo je potrebno dodati *package* deklaraciju kao **prvu** liniju fajla koji sadrži definiciju klase.
- Ispred ove linije mogu se naći jedino komentari ili prazne linije.
- package deklaracija je oblika:

  package ime_paketa;

Paketi

- Ako želimo da klasa paketa bude dostupna i izvan njega, potrebno je deklarisati klasu koristeći ključnu reč **public** u prvoj liniji definicije klase.
- Ukoliko ispred definicije klase ne postoji ključna reč *public*, definicija te klase je dostupna samo metodima klasa koje se nalaze u istom paketu.

```
package geometrija;
public class Sfera{
   // detalji klase
}
```

Paketi

- Svaka klasa koju hoćemo da uključimo u paket *geometrija* mora sadržati istu *package* deklaraciju na početku i
  svi fajlovi koji sadrže definicije klasa tog paketa moraju se sačuvati u direktorijumu koji ima isto ime kao i paket,
  dakle *geometrija*.
- Takođe, metode i konstruktore unutar **public** klase treba definisati kao **public** ako hoćemo da budu dostupni izvan paketa.

Paketi i struktura direktorijuma

- Paketi su usko povezani sa strukturom direktorijuma u kojoj se čuvaju!
- Ime paketa može biti složeno, npr.

  geometrija.geometrija3D,

  što znači da je *geometrija3D* poddirektorijum direktorijuma *geometrija*.
- Ime može biti proizvoljne složenosti, ali treba da odražava strukturu direktorijuma u koju je paket smešten.

Paketi i struktura direktorijuma

- Na primer, ako pravimo više kolekcija klasa koje se tiču jedne iste oblasti (*geometrija*), možemo da kreiramo više paketa: *geometrija2D*, *geometrija3D*, ...
- Podrazumeva se da se klase iz prvog paketa nalaze u direktorijumu *geometrija2D*, iz drugog u direktorijumu *geometrija3D* i da su *geometrija2D* i *geometrija3D* poddirektorijumi direktorijuma *geometrija*.

Paketi i struktura direktorijuma

- Ako želimo da uključimo neke klase iz paketa geometrija2D, pišemo:

  import geometrija.geometrija2D.*;

- Ne može se pisati:

  import geometrija.*;

  za uključivanje svih paketa iz direktorijuma *geometrija*.

- Znak * se odnosi na sve klase jednog paketa.

Dodavanje klasa iz paketa u program

- Ako su definisane sa ključnom rečju *public*, možemo dodati proizvoljnu ili sve klase iz imenovanog paketa kodu našeg programa korišćenjem **import deklaracija**.
- U programu zatim možemo referisati klase koje smo učinili dostupnim import deklaracijama samo navođenjem njihovih imena.
- import deklaracije se navode na početku fajla.

Primer

import geometrija.geometrija3D.*;

- uključuje sve klase paketa *geometrija.geometrija3D*
- Sada možemo referisati proizvoljnu *public* klasu iz paketa navođenjem samo njenog imena.
- Obično je bolje "importovati" samo klase paketa koje naš kôd koristi, npr. za klasu Sfera

import geometrija.geometrija3D.Sfera;

Paketi i imena u našim programima

- Unutar jednog paketa možemo davati imena klasama bez brige da li ta imena već postoje negde izvan paketa.
- Java tretira ime paketa kao deo imena klase, zapravo kao njegov prefiks.
- Npr. klasa *Sfera* iz paketa *geometija.geometija3D* ima puno ime *geometija.geometija3D.Sfera*. Ukoliko ne koristimo *import* deklaraciju, ovu klasu možemo koristiti navođenjem njenog punog imena.

Importovanje statičkih članova klase

- Mogu se importovati imena statičkih članova klase iz imenovanog paketa u naš program.
- Statički članovi se zatim mogu koristiti prosto navođenjem njihovih nekvalifikovanih imena.
- Npr.

```
import static java.lang.Math.PI;
...
return 4.0/3.0*PI*radius*radius*radius;
```

- importovanje svih statičkih članova klase *Math*:

```
import static java.lang.Math.*;
```

- **Napomena!**

  import statičkih članova klase nije moguć za klase koje se nalaze u default – bezimenom paketu. Paket mora imati ime!

Standardne klase koje enkapsuliraju primitivne tipove podataka

- Boolean, Byte, Character, Short, Integer, Long, Float, Double – nalaze se u paketu java.lang.
- Svaka od ovih klasa ima statički metod **toString()** za konvertovanje vrednosti odgovarajućeg primitivnog tipa u **String** objekat.
- Postoje i metodi za konvertovanje iz String objekta u primitivni tip.
- Npr. statički metod **parseInt()** klase Integer prihvata String reprezentaciju celog broja kao argument i vraća ekvivalentnu vrednost tipa int. Odgovarajući metodi postoje i u ostalim klasama.

Standardne klase koje enkapsuliraju primitivne tipove podataka

- Svaka klasa takođe definiše i metod **value()** koji vraća vrednost enkapsuliranu objektom kao vrednost odgovarajućeg primitivnog tipa.

  Double number;

  number.value() -> double
- Svaka numerička klasa ima *static final* konstante MAX_VALUE i MIN_VALUE koje definišu najveću i najmanju vrednost koje mogu biti predstavljene tim tipom.
- Floating-point klase imaju i definisane konstante: POSITIVE_INFINITY, NEGATIVE_INFINITY i NaN koje se mogu koristiti za poređenja prilikom izračunavanja.
- Postoje i statičke metode isInfinite() i isNan() za testiranje floating-point vrednosti.

AUTOBOXING
VREDNOSTI PRIMITIVNIH TIPOVA

- Ako se metodu, koji kao argument zahteva referencu na objekat, prosledi vrednost primitivnog tipa, ukoliko okolnosti dozvoljavaju, kompajler će izvršiti automatsku konverziju vrednosti primitivnog tipa u odgovarajući klasni tip.
- Konverzije primitivnog tipa u odgovarajući klasni tip zovu se ***boxing konverzije***, a automatske konverzije ove vrste zovu se ***autoboxing.***
- Kompajler takođe radi i inverzne konverzije – referencu na objekat klase primitvnog tipa konvertuje u vrednost koju on enkapsulira (***unboxing***).

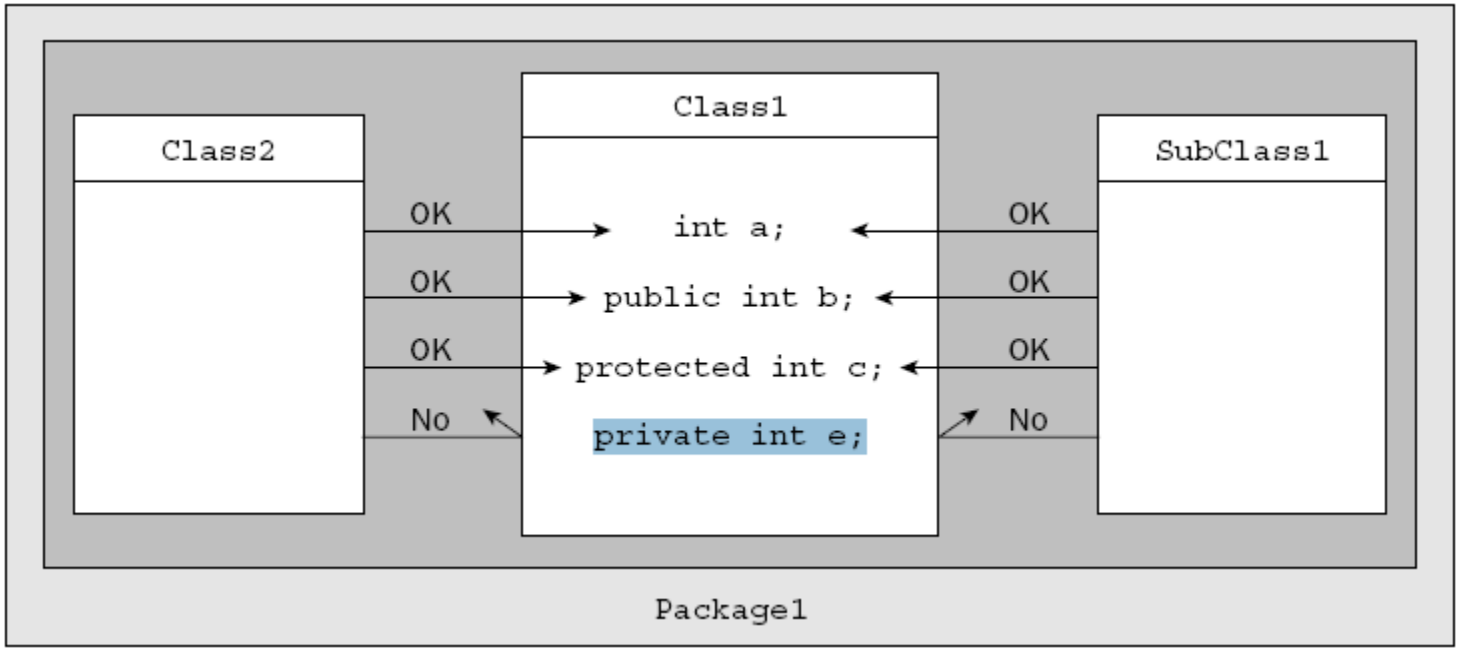
Pristup članovima klase (dostupnost)

- Svaka klasa ima direktan pristup svim ostalim klasama istog paketa, ali promenljive i metode koje su članovi tih klasa ne moraju nužno biti dostupne.
- Dostupnost se kontroliše **atributima pristupa (access atributi).**
- Klasama koje ne deklarišemo kao **public** može se pristupiti samo iz istog paketa.

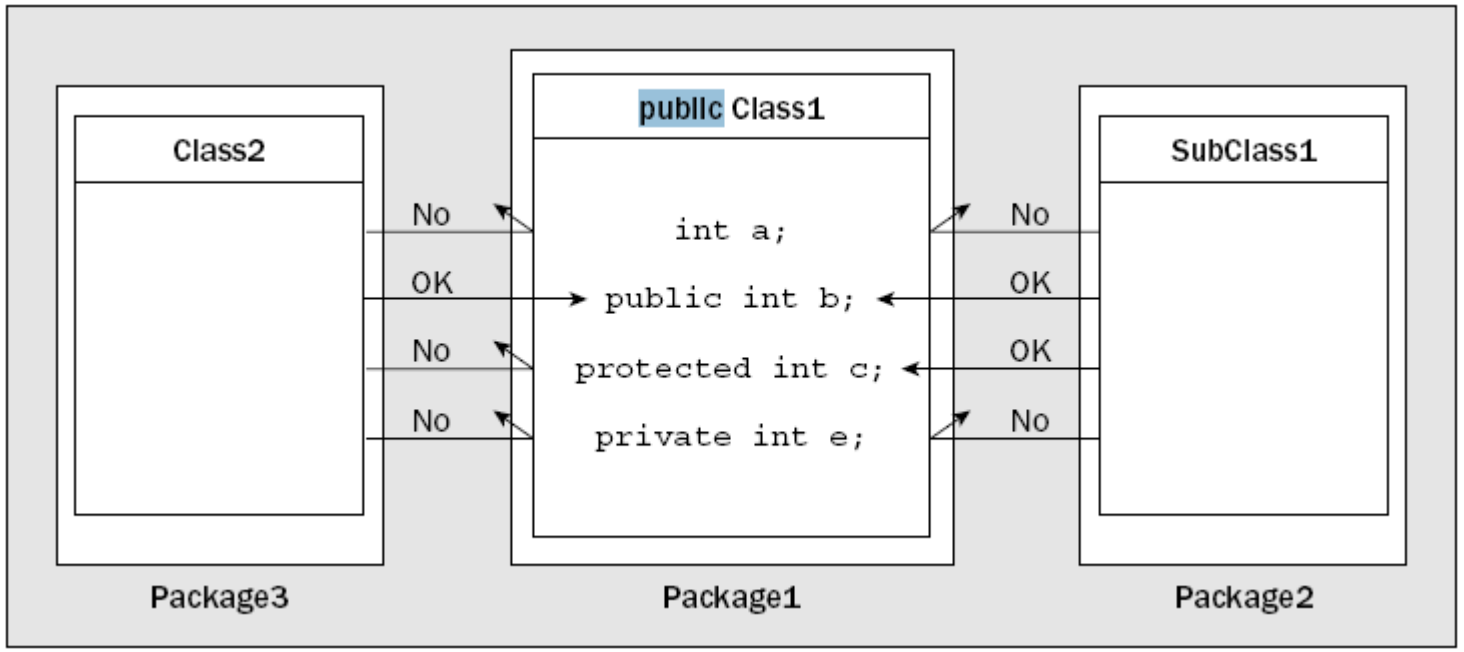
Pristupni atributi za članove klasa

- Postoje 4 mogućnosti za atribute pristupa članicama klase:
  - **bez pristupnog atributa** – dopušten pristup iz metoda proizvoljne klase iz istog paketa.
  - **public** – dopušten pristup iz metoda proizvoljne klase (ne nužno iz istog paketa),
    sve dok je klasa čiji je to član deklarisana kao *public*.
  - **private** – dostupan samo iz metoda unutar klase. Nikakav pristup izvan klase.
  - **protected** – dopušten pristup iz metoda proizvoljne klase istog paketa i iz proizvoljne potklase (ne nužno iz istog paketa).

Primer, klase u istom paketu



Primer, različiti paketi



Pristupni atributi

- Uobičajeno je da instancne promenljive (nestatički atributi) budu **private** tako da im se ne može direktno pristupati,
  niti se mogu direktno menjati izvan klase.
  Jedini način da im se pristupi ili
  da se njihove vrednosti promene jeste pomoću metoda iste klase.
- Ukoliko je potrebno pristupiti vrednostima **private** atributa izvan klase, to se postiže tzv. **pristupnim (accessor) metodom klase.**

Accessor metodi

- Npr.

```
public double getX(){
        return x;
}
```

- Na ovaj način vrednost atributa postaje dostupna svuda, ali se može menjati samo unutar klase.
- Accessor metodi se obično zovu **get*().**

Mutator metodi

- Metodi koji dopuštaju menjanje vrednosti **private** atributa nazivaju se **mutator metodi**.
- Njihova imena su obično oblika **set*().**
- Nova vrednost atributa se prosleđuje kao argument.
- Npr.

```
public void setX(double inputX){
        x = inputX;
}
```

Prednost mutator metoda

- Zašto koristiti **public** metod da bismo menjali vrednost **private** atributa kada možemo jednostavno da ga učinimo **public**?
- Glavna prednost korišćenja mutator metoda jeste što je moguće vršiti proveru nove vrednosti i sprečiti pridruživanje neodgovarajućih vrednosti atributima.

Izbor pristupnih metoda

- Najčešće,
  atributi u *public* klasi treba da budu *private*,
  a metodi koji će se pozivati izvan klase *public*.
- Izuzeci:
  - za klase paketa koje nisu *public*, a time nisu dostupne izvan paketa, konvencija je dozvoliti drugim klasama paketa direktan pristup atributima
  - ako imamo *final* atribute koji su verovatno korisni izvan klase, možemo ih deklarisati kao *public*
  - metode klase koje će interno koristiti samo drugi metodi iste klase treba definisati kao *private*
  - u klasama poput *Math* koja je samo kontejner za korisne funkcije i standardne vrednosti podataka, sve treba da bude *public*
- Nakon uvođenja potklasa, još neke stvari se moraju uzeti u obzir.

Zadatak 1. – dopuna Geometije Tacka.java

```
package Geometrija;
public class Tacka
{
    // Konstruktor na osnovu zadate x i y koordinate tacke
    public Tacka(double xVal, double yVal)
    {
            x = xVal;
            y = yVal;
    }

    /* Konstruktor na osnovu zadatog objekta Tacka
    * Kljucna rec final oznacava da se argument
    * nece menjati u okviru metoda */
    public Tacka(final Tacka tacka)
    {
            x = tacka.x;
            y = tacka.y;
    }
```

```
// Metod za pomeranje Tacke po x i y osi
public void pomeri(double x_pomeraj, double y_pomeraj)
{
        x += x_pomeraj;
        y += y_pomeraj;
}

// Metod za racunanje rastojanja do zadate tacke
public double rastojanje(final Tacka tacka)
{
        return Math.sqrt(
                (x - tacka.x)*(x - tacka.x) + (y - tacka.y)*(y - tacka.y) );
}

// Metod za konverziju u String
public String toString()
{
        return "(" + x + ", " + y + ")";
}
```

```
    // Metod za dobijanje vrednosti x koordinate
    public double getX()
    { return x;  }

    // Metod za dobijanje vrednosti y koordinate
    public double getY()
    { return y;  }

    // Metod za postavljanje vrednosti x koordinate
    public void setX(double inputX)
    { x = inputX;  }

    // Metod za postavljanje vrednosti y koordinate
    public void setY(double inputY)
    { y = inputY;  }

    // Koordinate tacke
    private double x;
    private double y;
}
```

DUZ.JAVA

```
package Geometrija;
public class Duz
{
    // Konstruktor na osnovu zadate pocetne i krajnje tacke duzi.
    public Duz(final Tacka pocetna, final Tacka krajnja)
    {
            pocetak = new Tacka(pocetna);
            kraj = new Tacka(krajnja);
    }

    /* Konstruktor na osnovu zadate x i y koordinate pocetne tacke,
     * i x i y koordinate krajnje tacke. */
    public Duz(double x_pocetak, double y_pocetak, double x_kraj,
    double y_kraj)
    {
            pocetak = new Tacka(x_pocetak, y_pocetak);
            kraj = new Tacka(x_kraj, y_kraj);
    }
```

```
    // Metod za racunanje duzine duzi.
    public double duzina()
    {
            return pocetak.rastojanje(kraj);
    }
    // Metod za konverziju u String.
    public String toString()
    {
            return pocetak + ":" + kraj;
    }

    // Instancne promenljive
    Tacka pocetak;
    Tacka kraj;
}
```

TestGeometrija.java

```
package Geometrija;

public class TestGeometrija
{
   public static void main(String[] args)
   {
         double[][] koordinate = { {1.0, 0.0}, {6.0, 0.0}, {6.0,
   10.0},
                     {10.0,10.0}, {10.0, -14.0}, {8.0, -14.0}};

   // Kreiramo niz tacaka sa datim koordinatama
   Tacka[] tacke = new Tacka[koordinate.length];
   for(int i = 0; i < koordinate.length; i++)
     tacke[i] = new Tacka(koordinate[i][0],koordinate[i][1]);
```

```
    // Kreiramo niz duzi koje su odredjene susednim tackama gornjeg niza
    Duz[] duzi = new Duz[tacke.length - 1];
    double ukupna_duzina = 0.0;
    for(int i = 0; i < tacke.length - 1; i++)
    {
      duzi[i] = new Duz(tacke[i], tacke[i+1]);

      // Dodajemo duzinu tekuce duzi na ukupnu duzinu
      ukupna_duzina += duzi[i].duzina();
      System.out.println("\nDuz "+(i+1)+": " +duzi[i] +
                         " - duzina joj je " + duzi[i].duzina());
    }

    // Stampamo ukupnu duzinu duzi
    System.out.println("\nUkupna duzina duzi je " + ukupna_duzina);

    }
}
```

Zadatak 2.

Napisati klasu Krug koja od instancnih promenljivih sadrži: x i y koordinatu centra i poluprečnik kruga i ima klasnu promenljivu brojač koja sadrži broj kreiranih objekata klase Krug.

Od metoda obezbediti konstruktor sa odgovarajuća tri parametra (x,y koordinata i poluprečnik), konstruktor bez parametara (postavlja krug na jedinični u koordinatnom početku), nestatičke metode za računanje obima i površine kruga i metod tackaUkrugu koji kao argument prima x i y koordinatu tačke i ispituje da li se tačka nalazi u datom krugu, i statički metod brojKrugova koji vraća vrednost klasne promenljive brojač. U zasebnoj klasi napisati main() funkciju koja demonstrira ponašanje ove klase.

Zadatak 3.

Napisati klasu Polinom za rad sa polinomima sa celobrojnim koeficijentima. Realizovati metode: konstruktor sa celobrojnim argumentom n (konstruiše se polinom x^n), konstruktor na osnovu niza datih koeficijenata (argument je: int[]), metod za sabiranje, metod za oduzimanje i metod za množenje dva polinoma. U posebnoj klasi kreirati main() funkciju koja testira ponašanje klase Polinom.