6

Objektno orijentisano programiranje
vežbe

Staša Vujičić Stanković

Zadaci :

- Prilikom čitanja slajdova OBAVEZNO uporedo čitati zadatke sledećim redom:

1. nasledjivanje_zivotinje
2. nasledjivanje_vise_nivoa

NASLEĐIVANJE KLASA

- Koncept nasleđivanja je vrlo bitan deo OO programiranja.
- **Nasleđivanje** – kreiranje nove klase koja je bazirana na već definisanoj klasi.
- **Bazna klasa (natklasa)** je klasa iz koje se vrši izvođenje.
- **Izvedena klasa** naziva se još i **direktna potklasa** bazne klase.
- Moguće je izvoditi klasu iz izvedene klase.

NASLEĐIVANJE KLASA

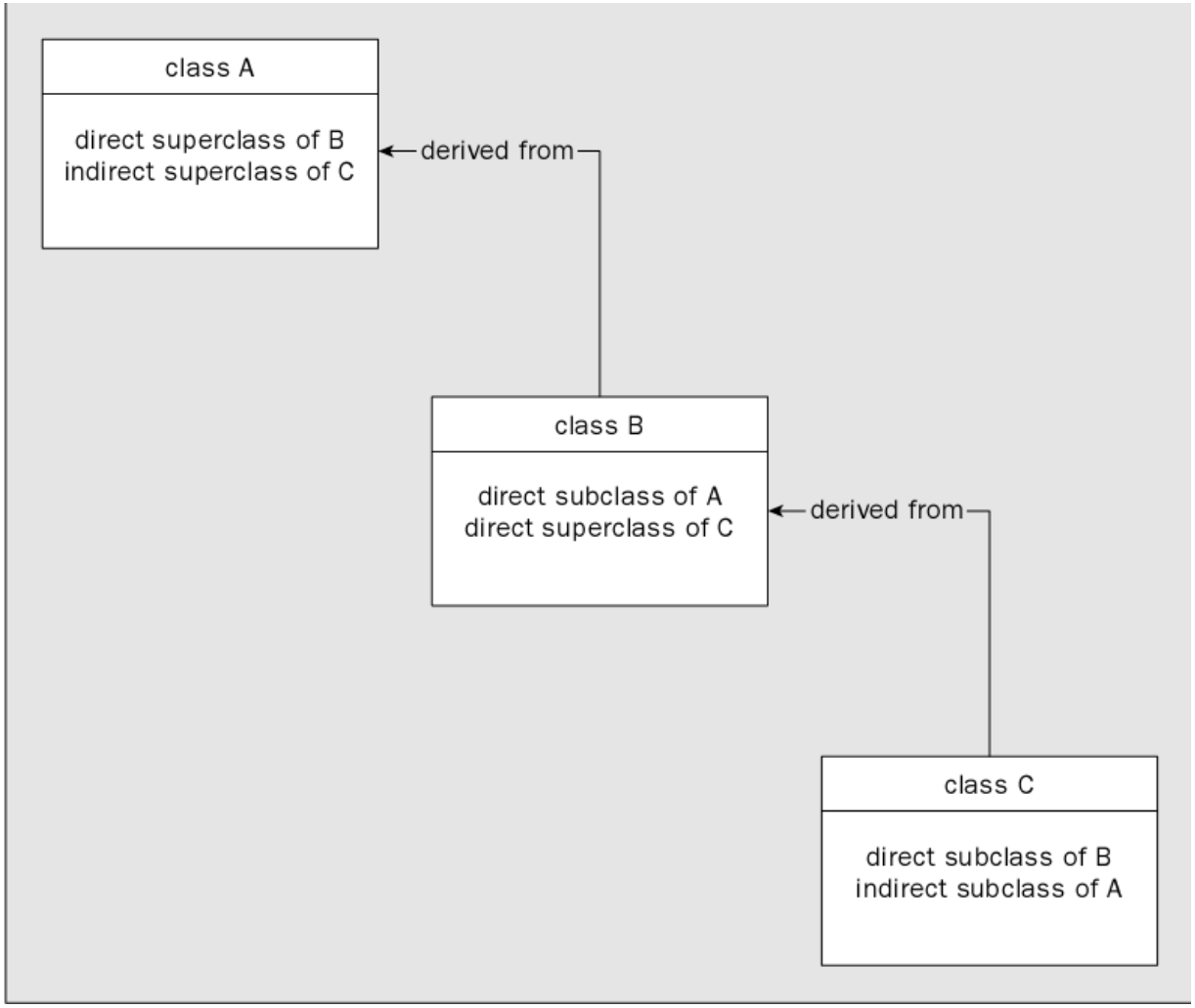
```
class Zivotinja {
        …
}
class Pas extends Zivotinja {
        …
}
```

- Ključna reč **extends** označava da je klasa *Zivotinja* bazna klasa za klasu *Pas*,

  tj.

  klasa *Pas* nasleđuje klasu *Zivotinja*.

NASLEĐIVANJE KLASA



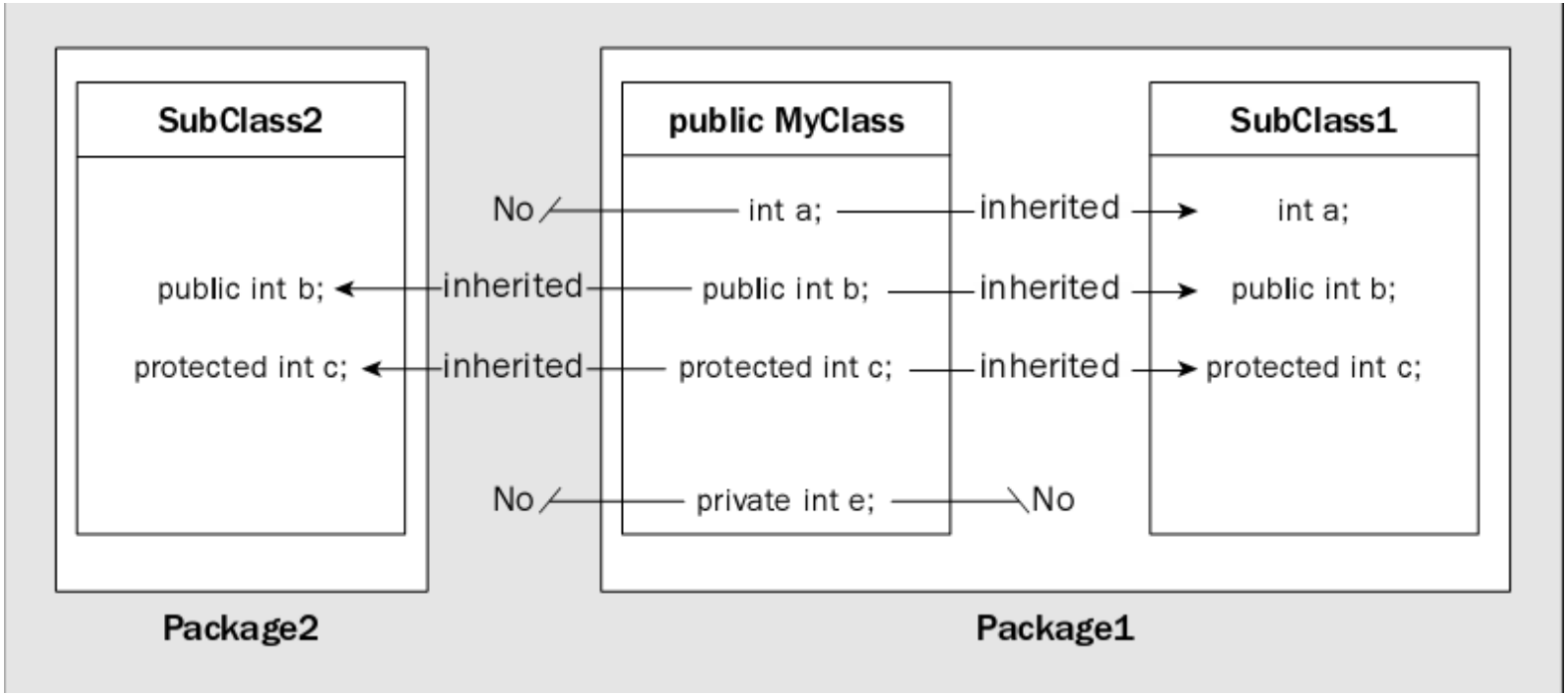
NASLEĐIVANJE KLASA

- Objekat izvedene klase u sebi sadrži kompletan objekat bazne klase (sa svim članicama - podacima i metodima), kao i nove članove koji su definisani u izvedenoj klasi.
- Ipak, članice definisane u baznoj klasi ne moraju uvek biti dostupne metodima specifičnim za izvedenu klasu!

NASLEĐIVANJE KLASA

- Za članicu bazne klase koja je **dostupna** unutar izvedene klase kažemo da je **nasleđena članica**.

NASLEĐIVANJE ČLANICA KLASE



NASLEĐIVANJE ČLANICA KLASE

- Potklasa definisana u istom paketu kao i bazna klasa nasleđuje sve osim private članica bazne klase.
- Potklasa definisana van paketa bazne klase ne nasleđuje private članice niti članice bez atributa.

Skrivene članice klase

- Članica izvedene klase može da ima isto ime kao i članica bazne klase, čime članica bazne klase biva skrivena.
  Tipovi članica i atributi pristupa mogu da se razlikuju.
- Svaka upotreba tog imena odnosi se na članicu izvedene klase.
- Članici bazne klase se pristupa pomoću **super.imeČlanice**.
- Ovo nije poželjan pristup pri definisanju izvedene klase!

NASLEĐIVANJE METODA

- Metodi bazne klase (sa izuzetkom konstruktora) nasleđuju se po istom principu kao i članice podaci.
- **Konstruktori bazne klase se nikad ne nasleđuju!!!**
- U izvedenoj klasi moguće je pozvati konstruktor natklase i to za inicijalizaciju onog dela koji je nasleđen. Razlog je logičan, jer objekat izvedene klase u sebi sadrži objekat bazne klase, pa je najbolji način za inicijalizaciju nasleđenog dela upravo poziv konstruktora natklase.

NASLEĐIVANJE METODA

- Metodi bazne klase koji su nasleđeni u izvedenoj klasi mogu da pristupe svim članovima bazne klase unutar izvedene klase, čak i onima koji nisu nasleđeni.

PRIMER: ZIVOTINJA

```
package nasledjivanje_zivotinje;
public class Zivotinja
{
        private String vrsta; // Private clanica. Ne nasledjuje se!

        public Zivotinja(String vrsta)
        {
                this.vrsta = new String(vrsta);
        }

        // Default konstruktor za slucaj da nam je potreban
        public Zivotinja()
        {}

        public String toString()
        {
                return "Ovo je " + vrsta;
        }
}
```

Primer: Zivotinja - Pas

```
package nasledjivanje_zivotinje;
public class Pas extends Zivotinja
{
        private String ime;
        private String rasa;

        public Pas(String ime)
        {
            super("Pas");      // Poziv konstruktora bazne klase Zivotinja
            this.ime = ime;    // Postavlja se ime na datu vrednost
            this.rasa = "Nepoznata";  // Podrazumevana vrednost za rasu je "Nepoznata"
        }

        public Pas(String ime, String rasa)
        {
            super("Pas");      // Poziv konstruktora bazne klase Zivotinja
            this.ime = ime;    // Postavlja se ime na datu vrednost
            this.rasa = rasa; // Postavlja se rasa na datu vrednost
        }
```

```
/*
public String toString()
{
        return "Ovo je " + ime + ", a rasa je  " + rasa;
}
*/

// Ispisujemo podatke o psu
public String toString()
{
        return super.toString() + "\nZove se " + ime + ", a rasa je " + rasa;
}

}
```

PRIMER: TEST ZIVOTINJE

```
package nasledjivanje_zivotinje;

public class TestZivotinje
{
        public static void main(String[] args)
        {
                // Kreira se objekat klase Pas sa datim imenom i rasom
                Pas pas1 = new Pas("Fido", "Bernardinac");

                // Kreira se objekat klase Pas sa datim imenom
                Pas pas2 = new Pas("Lesi");

                // Prikaz objekata - implicitno se poziva metod toString() klase Pas
                System.out.println(pas1);
                System.out.println(pas2);
        }
}
```

Primer: Zivotinja - Pas

- Klasa Pas nasleđuje samo metod toString() iz klase Zivotinja, jer se konstruktor i private članice ne nasleđuju.
- Članica vrsta se ne nasleđuje, tj. ne može joj se pristupiti iz metoda klase Pas.

  Međutim, ona mora biti pravilno inicijalizovana, što se vrši pozivom konstruktora bazne klase iz konstruktora izvedene klase – **super(...)** sa odgovarajućim brojem argumenata.

Primer: Zivotinja - Pas

- Poziv konstruktora bazne klase mora biti **prva naredba** u telu konstruktora izvedene klase!
  Ako je nema, kompajler sam umeće poziv default konstruktora **super().**

  Napomena!
  Ovo može dovesti do greške, u slučaju da u baznoj klasi jesmo definisali konstruktor,
  a nismo podrazumevani konstruktor.

Overriding (predefinisanje) metoda bazne klase

- Moguće je definisati metod izvedene klase koji ima isti potpis kao i metod bazne klase.
- Atribut pristupa predefinisanom metodu u izvedenoj klasi ne sme biti restriktivniji od atributa pristupa u baznoj klasi!
- Ovim se postiže da metod u izvedenoj klasi preklapa metod bazne klase.
- Metod bazne klase još uvek postoji u izvedenoj klasi i moguće je pozvati ga u izvedenoj klasi.
- Primer: metod *toString()* klase *Pas* preklapa metod *toString()* klase *Zivotinja*

Izbor pristupnih atributa za bazne članove

- Metode koji čine neki spoljni interfejs klase treba deklarisati kao public. Sve dok nisu predefinisani u izvedenoj klasi, biće u njoj dostupni. Obično ne treba deklarisati podatke članice kao public, osim ako nisu konstante namenjene opštoj upotrebi.
- Ako će se klasa koristiti kao bazna klasa za neke druge klase, članice je poželjno deklaristi kao private, a obezbediti public metode za pristup i manipulaciju njima kada je potrebno.
- Protected članovi se najčešće koriste u paketima klasa sa neograničenim pristupom svim članicama iz istog paketa, pri čemu je pristup van paketa ograničen na potklase.

Zadatak 1. - Kompleksni

Napisati klasu KompleksniBroj za rad sa kompleksnim brojevima. Definisati: konstruktor sa jednim double argumentom (postavlja vrednost kompleksnog broja na taj realan broj), konstruktor sa dva double argumenta (realni i kompleksni deo kompleksnog broja), konstruktor kopije, metode za sabiranje, oduzimanje, množenje i deljenje kompleksnih brojeva (rezultat je kompleksni broj) i za izračunavanje konjugovano-kompleksnog broja. Takođe implementirati i metode za izračunavanje vrednosti polarnih koordinata kompleksnog broja i '.

U drugoj klasi implementirati main() funkciju koja sa standardnog ulaza učitava dva kompleksna broja i ispisuje njihov trigonometrijski oblik; zatim sa standardnog ulaza učitava oznake operacija koje treba izvršiti nad unešenim kompleksnim brojevima (unose se sve dok se ne unese ”x”) i ispisuje rezultujuće kompleksne brojeve na standardni izlaz.

Zadatak 2. - Razlomak

Napisati klasu Razlomak za rad sa celobrojnim razlomcima. Definisati: konstruktor sa jednim celobrojnim argumentom (postavlja vrednost razlomka na taj ceo broj), konstruktor sa dva celobrojna argumenta (vrednosti brojioca i imenioca), konstruktor kopije, metode za sabiranje, oduzimanje, množenje i deljenje razlomaka (rezultat je razlomak) i za izračunavanje recipročnog razlomka. Takođe implementirati i metod za svođenje razlomka na formu gde su imenilac i brojilac uzajamno-prosti.

U drugoj klasi implementirati main() funkciju koja sa standardnog ulaza učitava dva razlomka i oznake operacija koje treba izvršiti nad unešenim razlomcima (unose se sa standardnog ulaza sve dok se ne unese ”q”) i ispisuje unešene i rezultujuće razlomke na standardni izlaz.