Objektno-orijentisano programiranje

Konstruktor kopije, statički atributi i metodi, nasleđivanje klasa

Nevena Ćirić
www.matf.bg.ac.rs/~nevena_ciric

Konstruktor kopije

- Objekat čije unutrašnje stanje, tj. vrednost nekog njegovog atributa može da se promeni naziva se mutirajući ili promenljivi objekat. U suprotnom radi se o nemutirajućem, tj. nepromenljivom objektu.
- Treba biti posebno pažljiv prilikom definisanja konstruktora koji kao argumente za inicijalizaciju atributa dobijaju reference na mutirajuće objekte.
- Naime, ukoliko se za odgovarajući argument novokreiranog objekta postavi vrednost argumenta konstruktora, tada:
  - argument konstruktora i odgovarajući atribut novokreiranog objekta referišu na isti objekat
  - svaka promena objekta koji je prosleđen kao argument konstruktoru dovodi do promene odgovarajućeg atributa novokreiranog objekta, i obrnuto

Konstruktor kopije

PRIMER:
```
class Oblik{
        private Tacka centar;
        public Oblik(Tacka centar) {
                this.centar = centar;
        }
}
```

- Umesto inicijalizacije atributa kopijom reference, potrebno je napraviti kopiju objekta čija je referenca prosleđena kao argument konstruktora i njime inicijalizovati atribut.

PRIMER:
```
class Oblik{
        private Tacka centar;
        public Oblik(Tacka centar){
                this.centar = new Tacka(centar.getX(), centar.getY());
        }
}
```

Konstruktor kopije

- Umesto kreiranje kopije objekta svaki put pomoću njegovog najopštijeg konstruktora (koji za svaki atribut uzima po jedan argument), definiše se nova vrsta konstruktora koja je posebno namenjena za kreiranje kopija objekata – konstruktor kopije.
- Konstruktor kopije kao argument uzima kao argument postojeći objekat date klase i vrši kopiranje vrednosti njegovih atributa u odgovarajuće atribute novokreiranog objekta kopije.

```
class Tacka{
    private double x, y;
    public Tacka(Tacka t){
        this.x = t.x;
        this.y = t.y;
    }
}
```

```
class Oblik{
    private Tacka centar;
    public Oblik(Tacka centar){
        this.centar = new Tacka(centar);
    }
}
```

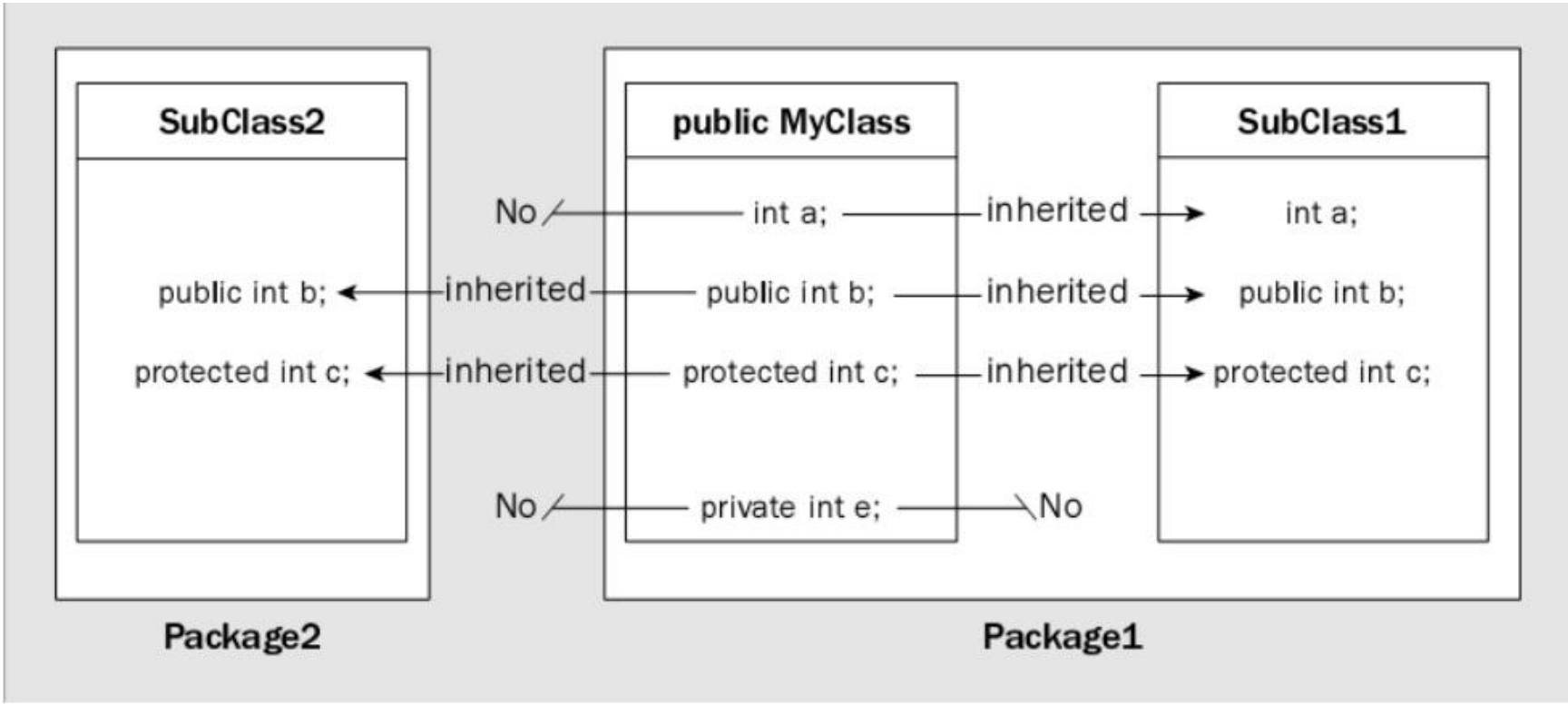
Statički atributi i metodi

- Statički (klasni) atributi su atributi koji opisuju neko svojstvo klase. Koriste za čuvanje podataka koji su zajednički za sve objekte te klase.
- Uvek postoji tačno jedan primerak statičkog atributa bez obzira na to koliko objekata te klase je kreirano, čak i u slučaju kada nije kreiran nijedan objekat te klase.
- Deklarišu se korišćenjem modifikatora static.
- Pristupa im se na osnovu imena klase ili posredstvom nekog od objekata te klase. Preferirani način je pristup na osnovu imena klase.
- Slično, statički (klasni) metodi opisuju ponašanje koje je zajedničko za sve objekte klase. Takođe se deklarišu korišćenjem modifikatora static.
- Statički metodi se pozivaju isključivo na osnovu imena klase. Mogu da se pozivaju i pre nego što se kreira ijedan objekat te klase.
- NAPOMENA: Kako nisu pridruženi konkretnom objektu klase, u telu statičkog metoda se ne može direktno referisati na nestatičke atribute i metode te klase.

Nasleđivanje klasa

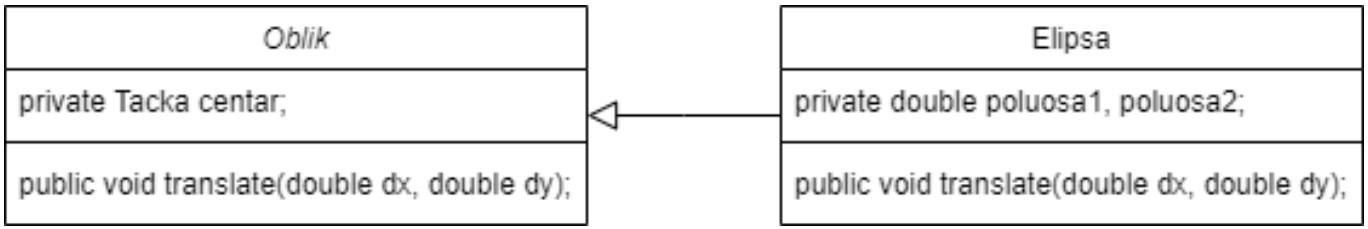
- Nasleđivanje je postupak kojim se na osnovu postojećih klasa definišu (izvode) nove.
- Nova klasa se naziva izvedena klasa ili potklasa klase iz koje je izvedena, dok se klasa iz koje se vrši izvođenje naziva bazna klasa ili natklasa.
- Izvedene klase predstavljaju dalju konkretizaciju bazne klase. One nastaju dodavanjem novih svojstava (atributa i metoda) ili modifikovanjem postojećih svojstava natklase.
- Objekat izvedene klase u sebi sadrži kompletan objekat bazne klase (sa svim atributima i metodima), kao i nove članove klase koji su definisani u izvedenoj klasi.
- Međutim, ne moraju svi članovi bazne klase biti dostupni u okviru definicije izvedene klase. Za atribute i metode bazne klase koji su dostupni unutar izvedene klase kažemo da su nasleđeni.

Nasleđivanje klasa



Nasleđivanje klasa

- Da li će neki atribut ili metod (sa izuzetkom konstruktora) bazne klase biti nasleđen zavisi od njegovog nivoa vidljivosti:
  - privatni atributi i metodi se nikada ne nasleđuju
  - atributi i metodi sa podrazumevanom (paketnom) vidljivosti se nasleđuju samo u izvedenim klasama koje su definisane u istom paketu kao i bazna klasa
  - javni i zaštićeni atributi i metodi se uvek nasleđuju
- Metodi bazne klase koji su nasleđeni u izvedenoj klasi mogu da pristupe svim članovima bazne klase unutar objekta izvedene klase, čak i onima koji nisu nasleđeni.
- PRIMER:



Nasleđivanje klasa

- Konstruktori bazne klase se nikada ne nasleđuju.
- U konstruktoru izvedene klase moguće je pozvati konstruktor bazne klase za inicijalizaciju dela objekta izvedene klase koji se sastoji od atributa bazne klase.
- Svaki nestatički metod izvedene klase pored promenljive this ima i promenljivu super koja predstavlja referencu na objekat bazne klase koji je sadržan u tekućem objektu izvedene klase za koji je metod pozvan. Konstruktori bazne klase se pozivaju sa super(...).
- Poziv konstruktora bazne klase je obavezno prva naredba u telu konstruktora izvedene klase. Ukoliko se ne pozove eksplicitno konstruktor bazne klase, kompajler sam umeće poziv podrazumevanog konstruktora super().
- **NAPOMENA**: Ovo može dovesti do greške, u slučaju da u baznoj klasi jesmo definisali neki konstruktor, ali ne i podrazumevani konstruktor.

Predefinisanje metoda

- Proces definisanja metoda u izvedenoj klasi koji ima isti potpis kao i metod u natklasi naziva se predefinisanje metoda (*eng*. overriding).
- Predefinisanjem metoda postiže se promena ponašanja ili dodavanje novog ponašanja nasleđenog metoda u izvedenim klasama.
- Modifikator vidljivosti predefinisanog metoda u izvedenoj klasi ne sme biti restriktivniji nego u natklasi.
- Prilikom definisanja metoda u izvedenoj klasi poželjno je koristiti anotaciju @Override kojom se kompajler informiše o nameri programera da predefiniše metod natklase. Kompajler proverava da li postoji metod sa istim potpisom u natklasi i prijavljuje grešku ukoliko ne pronađe isti.
- Iako je predefinisan, metod iz natklase postoji i u izvedenoj klasi i moguće je pozvati ga pomoću reference super.