Objektno-orijentisano programiranje

Apstraktne klase i interfejsi

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

Polimorfizam

- Osim korišćenja za definisanje novih klasa na osnovu postojećih, nasleđivanje klasa obezbeđuje i tzv. polimorfizam.
- Polimorfizam omogućuje da se jedan isti poziv metoda 'ponaša' različito u zavisnosti od tipa objekta nad kojim se metod poziva.
- Različito 'ponašanje' jednog istog metoda se postiže predefinisanjem metoda nasleđenog iz bazne klase na različite načine u izvedenim klasama.
- Prilikom poziva tog metoda preko pomenljive tipa bazne klase, koji će metod biti izvršen zavisi od tipa objekta nad kojim se metod poziva, a ne od tipa promenljive koja čuva referencu na taj objekat.

```
Oblik o = new Trougao();
o.translate();
```

Polimorfizam

- Da bi metod imao polimorfno ponašanje mora biti:
  - deklarisan u baznoj klasi
  - definisan u izvedenim klasama
  - pozvan preko promenljive tipa bazne klase
  - potpis metoda u baznoj i izvedenoj klasi mora biti isti
  - modifikator vidljivosti u izvedenoj klasi ne sme biti restriktivniji nego u baznoj klasi
- Metod ne može biti deklarisan u baznoj klasi kao privatan iz dva razloga:
  - ne bi mogao da se pozva pomoću promenljive tipa bazne klase
  - privatni metodi se ne nasleđuju, pa samim tim ne mogu biti ni predefinisani u izvedenim klasama

Polimorfizam

- Mehanizam nasleđivanja omogućuje polimorfno izvršavanje metoda na sledeći način:
  - Referenca na objekat izvedene klase se može čuvati u promenljivoj tipa te klase ili u promenljivoj tipa njene direktne ili indirektne natklase.
  - Kako promenljiva baznog tipa može da čuva referencu na objekat proizvoljnog izvedenog tipa, tip smeštenog objekta je poznat tek u vreme izvršavanja programa.
  - Koji će se metod izvršiti na mestu poziva polimorfnog metoda određuje se dinamički (u vreme izvršavanja programa).
  - Prilikom različitih pokretanja programa na ovom mestu se može izvršiti svaki put različit metod (metod druge izvedene klase) čime se postiže polimorfno izvšravanje ovog metoda, odnosno čitavog programa.

Polimorfizam

- Polimorfizam je jedan od fundamentalnih koncepata objektno orijentisano paradigme.
- **NAPOMENA:** Polimorfizam se primenjuje samo na metode, ne i na atribute.
- Kada pristupamo atributima, tip promenljive koja čuva referencu na objekat uvek određuje tip objekata, odnosno (pod)skup atributa tog objekta kojima se može pristupiti pomoću te promenljive.
- To povlači da promenljiva tipa bazne klase može da se koristi samo za pristup atributima iz bazne klase. Čak i kada ona referiše na objekat neke njene potklase, možemo je koristiti samo za pristup atributima iz bazne klase koji su deo datog objekta potklase.
- Takođe, pomoću promenljive tipa bazne klase ne može se pristupiti metodima izvedene klase koji nemaju polimorfno ponašanje.

Apstraktni metodi i klase

- Metodi sa polimorfnim ponašanjem su u baznoj klasi deklarisani a nisu definisani. Pošto nemaju definiciju i ne mogu se izvršiti, ti metodi se zovu apstraktni metodi.
- Deklaraciji apstraktnog metoda prethodi ključna reč abstact.
- Svi metodi osim konstruktora mogu biti apstraktni.*
- Klasa koja sadrži jedan ili više apstraktnih metoda je apstraktna klasa. Klasa može biti apstraktna čak iako ne sadrži ni jedan apstraktni metod.
- Definiciji apstraktne klase prethodi ključna reč abstact.
- Klase koje nasleđuju apstraktnu klasu moraju da predefinišu sve nasleđene apstaktne metode ili da i one takođe budu apstraktne.

* Konstruktori se nikad ne nasleđuju, pa samim tim ne mogu biti predefinisani u potklasama.

Apstraktni metodi i klase

- Apstraktne klase se koriste u slučaju kada postoji zajednička natklasa za više klasa, ali je ona veoma opšta tako da se u pojedinim aspektima njeno ponašanje (metodi) uopšte ne može definisati.
- Tada se zahteva da svaka od njenih potklasa mora da ima svoju konkretnu realizaciju opšteg ponašanja (metoda) natklase ili da takođe bude apstraktna.
- Kako apstraktna klasa nema u potpunosti definisano ponašanje (metode), njene instance ne predstavljaju konkretne objekte koje bi imalo smisla kreirati.
- Stoga, nije dozvoljeno instancirati objekte apstraktne klase, ali je moguće deklarisati promenljivu tipa apstraktne klase i koristiti je za čuvanje reference na konkretne objekte izvedenih klasa.

Interfejsi

- Java dopušta definisanje u potpunosti apstraktnih klasa koje se nazivaju interfejsi.
- U okviru definicije interfejsa mogu se naći samo deklaracije apstraktnih metoda (što se ne mora posebno naglasiti jer se podrazumeva) i definicije konstatni.
- Za definisanje klase koristimo ključnu reč interface koju prati ime interfejsa i u okviru vitičastih zagrada sama definicija strukture interfejsa.

```
interface ImeInterfejsa {
        // članovi interfejsa (apstraktni metodi i konstante)
}
```

- Interfejsi opisuju ponašanje (metode), a klase koje ih implementiraju definišu kako se to ponašanje (metodi) realizuje.

Interfejsi

- Apstraktne klase takođe nameću uslov koji to skup metoda moraju da implementiraju sve njene potklase. Pored toga, apstraktne klase definišu atribute i metode koji su zajednički za sve njihove potklase - to je ono što apstraktnu klasu u opštem smislu razlikuje od interfejsa.
- Terminologija: klase se nasleđuju, a interfejsi se implementiraju.
- Klasa koja implementira interfejs mora da implementira sve (apstraktne) metode interfejsa.
- Java podržava nasleđivanje samo jedne klase, ali dopušta da klasa implementira više od jednog interfejsa. Objekat koji je primerak klase koja implementira više interfejsa možemo postatrati kao primerak svakog od tih interfejsa.
- Takođe je moguće da interfejs nasleđuje druge interfejse.

Određivanje klase objekta

- Određivanje klase objekta se može postići pozivom metoda getClass() koji vraća objekat tipa Class pomoću koga se mogu dobiti informacije o odgovarajućoj klasi.

  PRIMER:
```
public static void printMethodsAndFields(Object o){
        Class c = o.getClass();
        Method[] methods = c.getMethods();
        for (Method m : methods)
                System.out.println(m.getName());
        Field[] fields = c.getDeclaredFields();
        for (Field f : fields)
                System.out.println(f.getName());
        }
```
- Pripadnost objekta klasi proverava se pomoću operatora instanceof.