Objektno-orijentisano programiranje

Enumeracije, stek i red

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

Enumeracije

- Enumeracije se koriste u slučajevima kada je u vreme pisanja programa poznato koji konačan skup vrednosti neka promenljiva može da uzima.
- Na primer, veličina odeće ima četiri vrednosti: small, medium, large и extra large. Ove vrednosti se mogu predstaviti celim brojevima 1, 2, 3, 4 ili stringovima “S”, “M”, “L”, “XL”, ali takav pristup je sklon greškama. Može se dogoditi da promenljiva koja čuva ovu vrednost dobije pogrešnu vrednost (poput 0 ili “m”).
- Enumeracije su klase koje opisuju konačan (nabrojiv) skup objekata. Ovakve klase se definišu korišćenjem ključne reči enum.
- Za prethodni primer može se definisati enumeracija Velicina na sledeći način:

```
enum Velicina {
        SMALL, MEDIUM, LARGE, EXTRA_LARGE
}
```

Enumeracije

- Prva linija definicije enumeracije je uvek (konačna) lista imenovanih objekata koje ta enumeracija opisuje. Pored toga, deficija enumeracije može da sadrži i deklaracije atributa i definicije metoda.
- NAPOMENA: Liste imenovanih objekata se mora završiti sa ; ukoliko nakon nje slede deklaracije atributa i/ili definicije metoda.
- Objektima koji su definisani enumeracijom se pristupa sa njihovim punim kvalifikovanim imenom koje se sastoji od imena enumerisacije i imena objekta (u prethodnom primeru: Velicina.SMALL, Velicina.MEDIUM, Velicina. LARGE, Velicina.EXTRA_LARGE).
- Enumeracije se interno implementiraju kao klase koje za svaki od objekata enumeracije definiše po jedan public static final* atribut.

* Promenljive sa final modifikatorom su zapravo konstante. Njima se vrednost dodeljuje samo jednom, prilikom njenog kreiranja i ne može im se nakon toga menjati vrednost.

Enumeracije

- Na primer, enumeracija iz prethodnog primera se interno implementira na sledeći način:

```
class Velicina extends Enum {
        public static final Velicina SMALL= new Velicina();
        public static final Velicina MEDIUM = new Velicina();
        public static final Velicina LARGE = new Velicina();
        public static final Velicina EXTRA_LARGE = new Velicina();

        private Velicina() {};
}
```

- Kako su u pitanju statički atributi, pristupamo im sa Velicina.SMALL, Velicina.MEDIUM, itd.
- Prema konvenciji, konstante se imenuju svim velikim slovima.
- Objekti* enumeracije se kreiraju podrazumevanim konstruktorom prilikom pokretanja programa (u tzv. fazi učitavanja klasa).

* još se nazivaju konstantama enumeracije

Enumeracije

- Moguće je dodati atribute i metode (uključujući i konstruktore) definiciji enumeracije.
- Enumeracijom je definisan konačan skup objekata (konstanti enumeracije) i nije moguće konstruisati nove objekte. Iz tog razloga svi konstruktori moraju biti privatni i mogu se pozivati samo pri kreiranju konstanti enumeracije.

```
enum Velicina {
        SMALL("S"), MEDIUM("M"), LARGE("L"), EXTRA_LARGE("XL");
        private String skracenica;
        private Velicina(String skracenica) {
                this.skracenica = skracenica;
        }
        public String getSkracenica() {
                return skracenica;
        }
}
```

Enumeracije

- Svi tipovi enumeracije su potklase klase Enum. Neki od metoda koji se nasleđuju od klase Enum:
  - toString() – vraća ime konstante enumeracije
    PRIMER: Velicina.SMALL.toString() vraća string "SMALL"
  - values() – statički metod koji niz konstanti enumeracije
    PRIMER: Velicina.values() vraća niz sa elementima Velicina.SMALL, Velicina.MEDIUM, Velicina.LARGE, Velicina.EXTRA_LARGE
  - ordinal() - vraća poziciju konstante enumeracije u definiciji enumeracije, računajući od 0
    PRIMER: Velicina.MEDIUM.ordinal() vraća 1.
  - compareTo() – poredi pozicije konstanti enumeracije, pri čemu je poredak konstanti određen definicijom enumeracije
    PRIMER: Velicina.SMALL.compareTo(Velicina.MEDIUM) vraća -1

Enumeracije

- Enumeracije se mogu koristiti prilikom višestrukog grananja pomoću switch naredbe:

```
Velicina v = getVelicina();
switch(v){
        case SMALL: ... break;
        case MEDIUM: ... break;
        case LARGE: ... break;
        case EXTRA_LARGE: ... break;
}
```

- Sintaksa switch naredbe za enumeracije je pojednostavljena. U case klauzama se ne mora navoditi puno kvalifikovano ime konstanti enumeracije, već samo ime konstante.
- Kako nije moguće kreirati nove objekte (duplikate) tipa enumeracije, za poređenje konstanti enumeracije ne mora se koristiti metod equals(), već je dovoljno koristiti operator ==.

Stek

- Stek je apstraktna struktura podataka koja funkcioniše po principu LIFO (eng. Last-In-First-Out), tj. kod koje se elementi dodaju i uklanjaju sa istog kraja.
- Osnove operacije sa stekom su:
  - PUSH – dodavanje elementa na vrh steka
  - POP – uklanjanje elementa sa vrha steka
  - TOP/PEEK – provera koji se element nalazi na vrhu steka
- Još neke od često podržanih operacija su :
  - SIZE – broj elemenata na steku
  - SHOW – prikaz sadržaja steka
- Stek se može implementirati pomoću raznih struktura podataka (npr. niza, liste, stabla). Jedini zahtev je da osnovne operacije budu implementirane tako da imaju vremensku složenost O(1).

Red

- Stek je apstraktna struktura podataka koja funkcioniše po principu FIFO (eng. First-In-First-Out), tj. kod koje se elementi dodaju sa jednog kraja, a uklanjaju sa drugog.
- Osnove operacije sa redom su:
  - ADD – dodavanje elementa na početak reda
  - REMOVE – uklanjanje elementa sa kraja reda
  - FRONT – provera koji se element nalazi na početku reda
  - BACK – provera koji se element nalazi na kraju reda
- Još neke od često podržanih operacija su :
  - SIZE – broj elemenata na steku
  - SHOW – prikaz sadržaja steka
- Red se, kao i stek, može implementirati pomoću raznih struktura podataka (npr. niza, liste, stabla). Takođe, zahtev je da osnovne operacije budu implementirane tako da imaju vremensku složenost O(1).