Statička implementacija polinoma

Apstraktni tipovi podataka

- Jedna od moćnih i ključnih tehnika rešavanja problema u programiranju (i uopšte) je **apstrakcija**
- Pod apstrakcijom podrazumevamo zanemarivanje nepotrebnih detalja i koncentrisanje na bitne elemente rešenja
- Na taj način o rešenju problema razmišljamo posredstvom globalnih i opštih operacija, čije detalje za početak zanemarujemo
- Posle, odnosno nezavisno od globalnog rešenja problema realizuju se i (prethodno “apstrahovane”) operacije
- Apstrakciju je moguće uspešno primeniti i na strukture podataka u programiranju

Apstraktni tipovi podataka

- Možemo reći da svaka struktura podataka ima dva aspekta:
  - **spoljašnji** i
  - **unutrašnji**

- Spoljašnji aspekt su one karakteristike strukture podataka koje opisuju šta struktura predstavlja - šta su elementi strukture podataka i koje su operacije sa njima dozvoljene

- Unutrašnji aspekt je način na koji su struktura podataka i operacije nad njom realizovane

- Pošto spoljašnji aspekt strukture podataka ne zavisi od realizacije, on se često naziva i **apstrakcijom** strukture podataka, dok se unutrašnji aspekt često naziva **implementacijom** apstraktne strukture podataka

Apstraktni tipovi podataka

- Tipovi podataka se karakterišu:
  - skupom vrednosti i
  - skupom operacija nad tim vrednostima
- Ako apstraktnu strukturu podataka “snabdemo” skupom vrednosti koje ta struktura može da predstavi i skupom operacija nad tim vrednostima, dobijamo **apstraktni tip podataka**
- Apstraktni tip podataka je konačan **skup vrednosti**, koji se naziva domen, i **skup operacija** i relacija definisanih nad tim skupom vrednosti
- Pri tom operacije i relacije treba shvatiti u širem smislu - dopuštaju se operacije sa argumentima koji mogu biti i iz nekog drugog skupa, a ni rezultat operacije ne mora biti u domenu
- Skup vrednosti mora biti konačan, jer će se implementacija apstraktnog tipa podataka izvršavati na računaru

Apstraktni tipovi podataka

- Za rešavanje problema apstraktni tip podataka je važniji od njegove realizacije jer se rešenje problema formuliše kao algoritam u terminima apstraktnog tipa podataka (*šta* a ne *kako* raditi sa vrednostima, odnosno *kojim* operacijama, a ne *kako* su one realizovane)

- Implementacija apstraktnog tipa podataka se može ostaviti za kasnije, a ako je apstraktni tip podataka već implementiran, tada se taj aspekt može trajno zanemariti

- Programski jezik Java pomoću koncepta referencijalnih tipova, odnosno klasa, omogućava kreiranje korisničkih apstraktnih tipova podataka (u daljem tekstu samo: apstraktnih tipova podataka)

Apstraktni tipovi podataka

- Za definisanje i implementaciju apstraktnih tipova podataka u Javi, glavni mehanizam su klase
- U klasama se definiše struktura podataka koju apstraktni tip ima, kao i operacije realizovane metodima
- Demonstriračemo dva pristupa ogranizaciji i implementaciji apstraktnog tipa podataka:
  - Proceduralni
  - Objektno-orijentisani (OO)
- Kod proceduralnog pristupa struktura podataka koja prestavlja tip je “odvojena” od operacija, koje u Javi implementiramo statičkim metodima
  - Metodi su “nezavisni” od objekata na koje se primenjuju, koji se metodima prosleđuju preko parametara
- Kod OO pristupa su podaci i operacije “spakovani” u jednu strukturu (klasu) kao polja i nestatički metodi
  - Metodi se odnose na objekat preko kog se pozivaju
- Akcenat stavljamo na proceduralni pristup

Apstraktni tipovi podataka

- Napomenimo ovde da nam apstraktni tipovi podataka omogućavaju da prevaziđemo i jedno ograničenje programskog jezika vezano za tipove podataka

- Ugrađeni prosti tipovi podataka imaju unapred definisani skup vrednosti i unapred definisani skup operacija nad tim vrednostima

- Tipovi koje smo do sad uglavnom uvodili sadržali su samo podatke, čime smo im zadavali skup mogućih vrednosti, ali ne i skup operacija

- Najzad, apstraktni tipovi podataka omogućavaju potpunu slobodu: programer može da odabere i skup vrednosti i skup operacija nad tim vrednostima - pod uslovom da ih sam i realizuje

Polinom - statička implementacija

- **Polinom** $p(x)$ je funkcija po promenljivoj x definisana sledećim (matematičkim) izrazom:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

za date:

  - ceo broj $n \geq 0$: **stepen** polinoma
  - realni brojevi $a_0, a_1, \ldots, a_n$: **koeficijenti** polinoma
    - ako je $n > 0$, tada mora da važi $a_n \neq 0$

- Izraze $a_i x^i$, za $i \in \{0, 1, \ldots, n\}$ nazivamo **monomi** polinoma

- Koeficijenti polinoma su obično brojevi. Mi ćemo raditi sa realnim koeficijentima, ali koeficijenti mogu da budu i celi brojevi, kompleksni brojevi, itd.

Polinom - statička implementacija

Jedan uobičajen skup osnovnih operacija sa polinomima je:

- Anulirati polinom, tj. staviti da je polinom jednak nuli
- Kopirati polinom
- Naći stepen polinoma
- Izračunati vrednost polinoma za neku vrednost promenljive *x*
- Učitati polinom
- Štampati polinom
- Sabrati dva polinoma
- Oduzeti dva polinoma
- Pomnožiti polinom konstantom
- Pomnožiti dva polinoma i
- Podeliti polinom polinomom

Polinom - statička implementacija

- Polinom ćemo predstaviti nizom koeficijenata tipa `double`, i to tako da je indeks elementa u nizu jednak indeksu koeficijenta u polinomu i jednak stepenu monoma čiji je to koeficijent
- Koristićemo nizove fiksne veličine sa unapred zadatom maksimalnom vrednošću stepena (maxSt) za sve polinome. Za takve nizove (i uopšte strukture podataka) kažemo da su **statički**
  - (Nisu dinamički jer im se veličina i struktura ne mogu menjati u toku izvršavanja programa)
  - Zato kažemo da je ova implementacija polinoma statička
- Ako se monom nekog stepena ne pojavljuje u polinomu, odgovarajući koeficijent je 0
- Pošto je za svaki polinom karakterističan stepen, odlučujemo se da ga u reprezentaciji polinoma posebno čuvamo
- Takođe želimo da napravimo razliku između nula-polinoma i konstantnog polinoma. Razlikovaćemo ih dogovorom da će nula polinom imati stepen -1, a konstantan polinom različit od nule imaće stepen 0

Polinom - statička implementacija

- Prema tome, pogodna struktura za predstavljanje polinoma je:

```
class Polinom {
  static final int maxSt = 100;
  double[] k = new double[maxSt+1];
  int st = -1;
}
```

- Polje `maxSt` je `static final`, što znači da predstavlja jedinstvenu konstantu za sve polinome
- Niz koeficijenata `k` sadrži redom a_0, a_1, ..., a_n, gde $0 \leq n \leq$ `maxSt`
- Polje `st` sadrži n, a na početku ga inicijalizujemo na `-1` da bi objekat tipa `Polinom` predstavljao nula-polinom
  - (elementi niza `k` se automatski inicijalizuju na `0.0`)

Polinom - statička implementacija

- Operacije nad polinomima realizovaćemo kao statičke metode u klasi `PolinomN`:

```
class PolinomN {
  /* Anulira polinom p */
  static void anuliraj(Polinom p) { ... }

  /* Kreira i vraca kopiju polinoma p */
  static Polinom kopiraj(Polinom p) { ... }

  /* Pronalazi stepen polinoma p i smesta ga u strukturu */
  static void nadjiStepen(Polinom p) { ... }

  /* Izracunava vrednost polinoma p za dato x */
  static double izracunaj(double x, Polinom p) { ... }

 /* Ucitava polinom */
  static Polinom ucitaj() { ... }

  /* Stampa polinom p */
  static void stampaj(Polinom p) { ... }
```

Polinom - statička implementacija

```
  /* Sabira polinome p1 i p2 vracajuci zbir */
  static Polinom saberi(Polinom p1, Polinom p2) { ... }

  /* Oduzima polinom p2 od polinoma p1 vracajuci razliku */
  static Polinom oduzmi(Polinom p1, Polinom p2) { ... }

  /* Mnozi broj c sa polinomom p vracajuci proizvod */
  static Polinom brojPuta(Polinom p, double c) { ... }

  /* Mnozi polinom p1 sa p2 vracajuci proizvod */
  static Polinom puta(Polinom p1, Polinom p2) { ... }

  /* Deli dva polinoma, vracajuci kolicnik i ostatak u nizu */
  static Polinom[] deli(Polinom p1, Polinom p2) { ... }
}
```

Polinom - statička implementacija

```
/* Anulira polinom p */
static void anuliraj(Polinom p) {
  if (p != null) {
    p.st = -1;
    for (int i = 0; i <= p.maxSt; i++)
      p.k[i] = 0.0;
  }
}
```

Polinom - statička implementacija

```
/* Kreira i vraca kopiju polinoma p */
static Polinom kopiraj(Polinom p) {
  if (p == null)
    return null;
  Polinom q = new Polinom();
  q.st = p.st;
  for (int i = 0; i <= p.maxSt; i++)
    q.k[i] = p.k[i];
  return q;
}
```