Programiranje I – RIN
Računalništvo I – MA

Podatkovne strukture in Generics

20. predavanje

Vsebina

- Generics
- Podatkovne strukture
  - Vector
  - Hashmap
  - TreeMap

Generics

- Hrošči so del programerjevega življenja,
- delna rešitev:
  - skrbno načrtovanje, programiranje in testiranje,
- ampak nekako, nekje se bodo vedno znašli v vaši kodi.

Generics

- Nekatere napake lažje odkrije kot druge:
  - napake, ki jih zaznamo pri prevajanju (compile-time bugs) najdemo samodejno,
- lahko uporabimo sporočila o napakah prevajalnika, da ugotovimo kje je napaka in jo odpravimo.
- Napake med izvajanjem (runtime errors, bugs) so veliko bolj problematične:
  - ne opazimo jih takoj,
  - ko jih opazimo, je lahko na točki v programu, ki je daleč od dejanskega vzroka problema.
- Generics dodajajo stabilnost v kodo, več napak je zaznavnih v času prevajanja.

Generics

- Generics omogočajo parametrizacijo tipov (razredov in vmesnikov) pri definiciji razredov, vmesnikov in metod,
- Parametri tipov omogočajo:
  - ponovno uporabo iste kode z različnimi vhodnimi podatki,
  - podobno formalnim parametrom, ki se uporabljajo pri deklaraciji metod.

Razlika je v tem, da so vhodi formalnih

Generics - benefits

- Močnejši pregled tipov v času prevajanja:
  - močno preverjanje tipov (strong type checking) se uporablja za generično kodo,
  - napake se pojavi, če koda krši varnost tipov,
  - reševanje napak, ki so odkrite pri prevajanju je lažje kot reševanje napak pri izvajanju.
- Omogoča programerjem implementacijo generičnih algoritmov:
  - programerji lahko izvajajo generične algoritmov, ki delujejo v zbirkah različnih tipov,
  - se lahko priredijo potrebam, so varni s stališča tipov in lažje berljivi.

Generics - benefits

- Odprava pretvorbe tipov (typecasting):
  - koda brez generics:

```
HashMap mapa;
mapa = new HashMap();
mapa.put("banana", new Integer(17));
String s = (String) mapa.get("hello"); //cast
```

  - Z uporabo generics, koda ne potrebuje pretvorbe tipov:

```
HashMap<String, Integer> mapa;
mapa = new HashMap<String, Integer>();
mapa.put("banana", new Integer(17));
Integer value = mapa.get("banana"); //no cast
```

Duck- typing

- Ko vidimo ptico, ki hodi kot raca in plava kot raca in se oglaša kot raca, to ptico imenujemo raca.
- Predmetno naravnani programski jeziki:
  - duck typing je stil tipizacije,
  - metode in lastnosti predmeta določajo veljavno semantiko,
  - namesto njegove dediščine iz posameznega razreda (extends) ali izvajanja vmesnika (implements).
- Java temelji na račjem tipiziranju (duck-typing).
- Python se tudi močno zanaša na duck-typing.

Podatkovne strukture

- **Vector**
- **Hashmap**
- **TreeMap**

Vector

`public class Vector<E>`

- implementira rastoče polje predmetov,
- vsebuje komponente, ki jih je mogoče dostopati z uporabo celoštevilskega indeksa.

Vector

`void add(int index, E element)`

`boolean add(E e)`

- Doda element v vektor (Vector)

`public E get(int index)`

- Vrne element, ki se nahaja na izbrani poziciji.

Vector

- **Primer:**
  - **`ExampleVector.java`**

HashMap

Class HashMap<K,V>

- HashMap uporablja razpršilno tabelo za izvajanje vmesnika Map,
- čas osnovnih operacij, kot so get () in put() je konstanten tudi pri velikih zbirkah.

HashMap

`V put(K key, V value)`

- Povezuje določeno vrednost z določenim ključem v tej podatkovni zbirki.

`V get(K key)`

- Vrne vrednost, ki je povezana s ključem (key).

HashMap

- **Primer:**
  - **`ExampleHashmap.java`**

Treemap

`Class TreeMap<K,V>`

- Implementacija vmesnika Map z Rdeče-črnim drevesom,
- Podatki so razvrščeni (sortirani) v skladu s svojimi ključi.

TreeMap

`V put(K key, V value)`

- Povezuje določeno vrednost z določenim ključem v tej podatkovni zbirki.

`V get(K key)`

- Vrne vrednost, ki je povezana s ključem (key).

TreeMap

- **Primer:**
  - **`ExampleTreemap.java`**