Podatkovne zbirke

Programiranje 2, Tomaž Dobravec

Splošno o podatkovnih strukturah

- Za učinkovito shranjevanje podatkov uporabimo primerno **podatkovno strukturo.**

- Za različne namene uporabljamo različne podatkovne strukture.

Primer:

- ocene predmeta Programiranje II

10, 9, 10, 8, 5, 0, 3, 9, 7, 10, 8, 1, 0

lahko shranimo v tabeli

- kratice držav udeleženk OI:

SI, CRO, A, IT, D, CZ, USA, ...

lahko shranimo v množici

Splošno o podatkovnih strukturah

- Pogoste podatkovne strukture:
  - sklad
  - vrsta
  - množica
  - ...

- Operacije nad podatkovnimi strukturami:
  - dodaj element
  - briši element
  - poišči element
  - jeElement?
  - jePrazna?
  - ...

Splošno o podatkovnih strukturah

- Možne izvedbe (implementacije) podatkovnih struktur:
  - tabela,
  - linearni seznam,
  - drevo,
  - zgoščena tabela,
  - ....

- Isto podatkovno strukturo lahko implementiramo na več načinov.

Primer: sklad lahko predstavimo s tabelo ali s seznamom

Hitrost operacij v pod. strukturah

- Pomembno: hitrost posamezne operacije

- Pomen zapisa
  - O(1) ... neodvisno od števila elementov
  - O(log n) ... logaritmično (1000x več podatkov, 3x več operacij)
  - O(n) ... linearno (2x več podatkov, 2x več operacij)
  - $O(n^2)$ ... kvadratično (100x več podatkov, 10000x več oper.)
  - .....

Primer: za shranjevanje elementa na zadnje mesto tabele potrebujemo O(1) časa, minimum tabele poiščemo v O(n) časa, tabelo s slabim algoritmom uredimo v $O(n^2)$,

z boljšim pa v O(n log n) časa.

Hitrost naraščanja funkcij

n	n^2	n^3	$logn$	$nlogn$	2^n	$n!$
10	100	1000	1	10	1024	$3628800 \approx 10^6$
20	400	4000	1,3	26	1048576=1ms	$\approx 10^{18}$
100	10000	10^6	2	200	$\approx 10^{30} = 380000A$	$\approx 10^{157}$
1000	10^6=1ms	10^9	3	3000	$2^{1000} = 10^{300}$	
10^6=1ms	10^{12}=16m	10^{18}	6	$6\ 10^6$	uf!	
10^9=1s	10^{18}=31l	$10^{27} = 380A$	9	$9\ 10^9$	uf!	

Opomba: pri računanju časa (s=sukund, m=minut, u=ur, d=dni, l=let, A=starost Zemlje) smo vzeli računalnik s procesorjem 1GHz in zelo grobo predpostavko, da se v enem ciklu izvede 1G=10^9 korakov (dejansko število korakov je namreč precej manjše).

Hitrost operacij v pod. strukturah

Teoretična hitrost operacij

	dodaj	išči	briši	i-ti element
tabela	O(1) ali O(n)	O(n)	O(n)	O(1)
seznam	O(n) ali O(1)	O(n)	O(n)	O(n)
urejeno drevo	O(log n)	O(log n)	O(log n)	O(log n)
zgoščena tab.	O(1)	O(1)	O(1)	O(1)

Opomba: dejanska hitrost je odvisna od implementacije

VPRAŠANJE: Katero podatkovno strukturo bom izbral?

ODGOVOR: Izberem tako podatkovno strukturo, ki mi najbolje pomaga rešiti dani problem.

Uporaba tabele

- Podatke najlažje hranimo v tabeli, vendar to ni vedno najboljša možna izbira.

- Slabosti tabele:
  - fiksna velikost,
  - počasno dodajanje elementov na začetek.

- Prednosti tabele:
  - enostavna uporaba,
  - hiter dostop do poljubnega elementa.

Podatkovne strukture v Javi

- Java pod skupnim imenov Java Collection Framework ponuja veliko podatkovnih struktur in orodij za delo z njimi

- JCF se v grobem deli na dva dela:

  - podatkovne strukture tipa

    java.util.Collection

    (vrednost1, vrednost2, ...)

    Primer: (38, 42, 46, 36, 39, ...)

    java.util.Map

  - podatkovne strukture tipa

    ((ključ1,vrednost1), (ključ2,vrednost2), ...)

    Primer: ((“Jan”, 31), (“Feb”, 28), (“Mar”, 31), ...)

Nekateri vmesniki in razredi JCF

vmesniki	razredi
`Collection`	
`Set`	`<- HashSet`
`SortedSet`	`<- TreeSet`
`List`	`<- ArrayList`
	`<- LinkedList`
`Map`	`<- HashMap`
`SortedMap`	`<- TreeMap`

http://java.sun.com/docs/books/tutorial/collections/

java.util.Collection

Nekatere metode vmesnika `Collection`

```
boolean add(Object o)
boolean remove(Object o)
int size()
boolean contains(Object o)
boolean isEmpty()
void clear()
Iterator iterator()
```

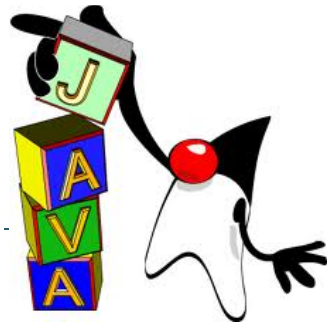
Iterator

- Iterator uporabimo za “sprehod” čez podatke
- Metode iteratorja:

```
public boolean hasNext()
public Object next()
public void remove()
```

Naloga

Množica dni



zbirke/Dnevi.java

Napiši program, v katerem boš imena dni v tednu shranil v strukturi “množica”. Vsa imena nato izpiši s pomočjo iteratorja.

Zanka foreach

- "Klasična" zanka `for` ... zaporedje indeksov
- Zanka `foreach` ... za sprehod po podatkovni zbirki

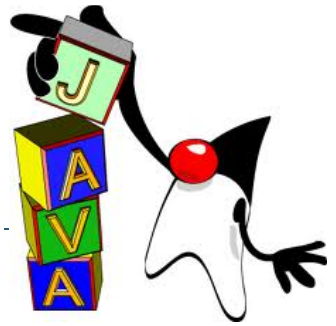
Primer:

```
Set<Integer> stevila = new TreeSet();
```


```
for (Integer x : stevila) {
    System.out.println(x);
}
```

Naloga

Množica dni

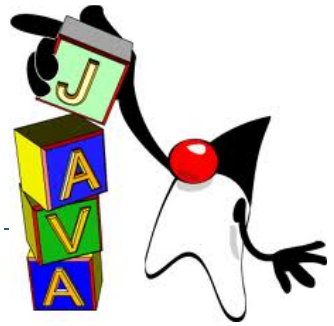


`zbirke/Dnevi.java`

Izpiši vse dni tedna (elementi množice dnevi) s pomočjo zanke `foreach`.

Naloga

Delitelji števil



zbirke/Delitelji.java

Napiši program, ki za dani števili `a` in `b` izpiše

a) vsa števila, ki delijo bodisi `a` bodisi `b`
b) vsa števila, ki delijo `a` in `b` (skupni delitelj)

Slovarji (java.util.Map)

- Za zbirke, katerih podatki so tipa (vrednost, ključ)
- Primerjava tabele in slovarja
  - pri **tabeli** se na elemente sklicuješ po številskem indeksu:

```
String a[] = new String[5];
a[1] = "Triglav";
...
System.out.println(a[1]);
```

  - pri **slovarju** za sklic navedeš poljuben objekt:

```
HashMap m = new HashMap();
m.put("Januar", 31);
...
System.out.println(m.get("Januar"));
```

Slovarji (java.util.Map)

- ključ (indeks v zbirki) mora biti enoličen

  Primer: Kaj izpiše spodnji program?

```
m.put("Januar",31);
m.put("Januar",30);
System.out.println(m.get("Januar"));
```

- ključi se primerjajo z metodo `equals`

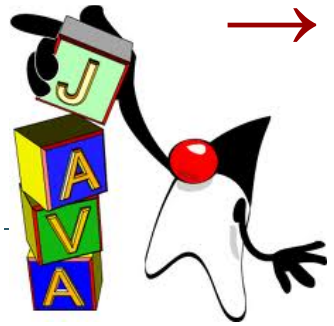
Slovarji (java.util.Map)

Nekatere metode vmesnika `Map`:

```
public int size()
public boolean isEmpty()
public boolean containsKey(Object key)
public boolean containsValue(Object value)
public Object get(Object key)
public Object put(Object key, Object value)
public Object remove(Object key)
public Set keySet()
public Collection values()
```

Naloga

Slovar mesecev

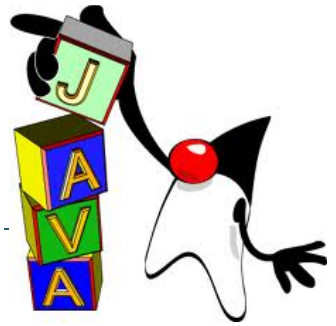


zbirke/Meseci.java

Napiši program, ki v slovarju hrani število dni posameznega meseca. Vse podatke iz slovarja nato izpiši s pomočjo iteratorja.

Naloga

Države na OI



`oi/Drzava.java, oi/Drzave.java`

V datoteki drzave.txt so zapisani podatki o državah, ki sodelujejo na olimpijskih igrah (glej spodaj). Izdelaj razred `Drzava` za hranjenje podatkov o eni državi (kratica, glavno mesto, število prebivalcev), nato preberi podatke iz datoteke `drzave.txt` v slovar (ključ: kratica države, vrednost: objekt razreda `Drzava`).

`drzave.txt`:

```
SLO:Ljubljana:2009245
ITA:Rome:58147733
AVT:Vienna:8199783
FIN:Helsinki:5238460
ZDA:Washington, DC:301139947
NEM:Berlin:82400996
FRA:Paris:64057790
SPA:Madrid:40448191
CRO:Zagreb:4493312
```

- test1: iz tipkovnice preberemo kratico, izpišemo vse podatke o državi,
- test2: izpis vseh držav (uporaba `iterator`ja po vrednostih),
- test3: izpis vseh držav (uporaba iteratorja po ključih).

Urejanje podatkov

Java zna urejati podatke, ki so shranjeni v

tabeli

- `Arrays.sort()`

seznamu

- `Collections.sort()`

Urejanje podatkov

Kakšnega tipa morajo biti podatki, da jih Java zna urediti z vgrajeno metodo `sort()`?

Poleg primitivnih podatkovnih tipov (`int, double`, ...) in nizov (`String`) lahko urejamo vse objekte razredov, ki imajo *implementiran vmesnik* `Comparable`.

Urejanje podatkov

Če želimo primerjati objekte razreda `Oseba` glede na njihovo višino, razred `Oseba` napišemo takole:

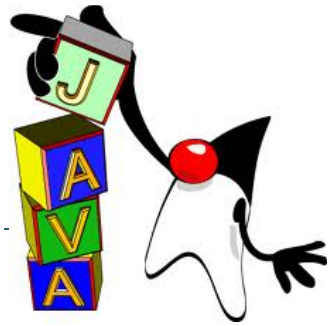
```
class Oseba implements Comparable<Oseba> {
  String ime;
  int visina;

  public int compareTo(Oseba o) {
    return new Integer(this.visina).compareTo(o.visina);
  }
}

// … uporaba (na primer v main metodi):
ArrayList<Oseba> osebe = new ArrayList<Oseba>();
osebe.add(. . .);
Collections.sort(osebe);
```

Naloga

Tekmovalci na OI



`oi/Tekmovalec.java, oi/Tekmovalci.java`

Štartna lista za finale teka na 100m je napisana v datoteki `tek.txt`. Ustvari razred `Tekmovalec`, ki hrani podatke o enem tekmovalcu, nato preberi podatke iz datoteke v pomnilnik. Tekmovalce izpiši urejene po abecednem vrstnem redu po priimkih.

`tek.txt:`

```
Usain Bolt:JAM
Justin Gatlin:USA
Trayvon Bromell:USA
…
```

Opombe:

- podatke preberemo v `ArrayList` (ker ne vemo, koliko jih je, ne moremo uporabiti navadne tabele),
- za urejanje izpisa bomo uporabili vgrajeno metodo `sort` (zato moramo implementirati vmesnik `Comparable`).

Urejanje podatkov

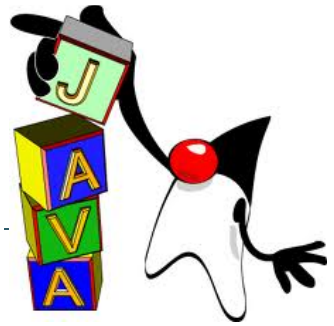
Urejamo lahko tudi tako, da ob klicu metode `sort()` podamo objekt razreda `Comparator`:

```
ArrayList<Oseba> osebe = new ArrayList<Oseba>();
osebe.add(. . . );

Collections.sort(osebe, new Comparator<Oseba>() {
  public int compare(Oseba o1, Oseba o2) {
    return o1.ime.compareTo(o2.ime);
  }
});
```

Naloga

Države tekačev



oi/Tek.java

Izpiši države, iz katerih prihajajo tekači iz datoteke `tek.txt`; države naj bodo urejene po abecednem redu.

Opomba:

- uporabili bomo urejeno množico (da preprečimo podvajanje in da zagotovimo urejenost)

Orodja za delo z zbirkami

- Metode iz razreda `java.util.Collections`
  - `sort(List l)`
  - `max(Collection c)`
  - `min(Collection c)`
  - `replaceAll(List l, Object o, Object no)`
  - `swap(List l, int i, int j)`

- `java.util.Arrays.sort(Object[] o)`