Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 1. PREDAVANJA (4.10.2007)

Študijsko leto 2007/08

Kratka ponovitev Jave

Objektno usmerjeno programiranje

- A PIE:

- **Abstraction – Abstrakcija**
- **Polimorfism – Polimorfizem**
- **Inheritance – Dedovanje**
- **Encaptulation – Enkapsulacija**

- Ključni pojmi:

- razred : objekt
- metoda : atribut

Kratka ponovitev Jave 2

Programski jezik Java

- Java 2 Platform, Standard Edition (J2SE), version 6
- zadošča J2SE, version (1.)5
- Sun Microsystems - `http:// java.sun.com`

Značilnosti:

- objektna usmerjenost
- neodvisnost od platforme
- primernost za razvoj spletnih aplikacij
- varnost
- relativna enostavnost (v primerjavi s C++)

Vrste programov v Javi :

- izvršljivi programi
- apleti
- servleti

Kratka ponovitev Jave 3

Primer: prvi program v Javi ...

Osnovni podatkovni tipi

- numerični tipi:

  cela števila: `byte` (8) -128 -- 127
  `short` (16) -32768 do -- 32767
  `int` (32) -2147483 648 -- 2147483 647
  `long` (64) $-9.22 * 10^{18}$ -- $9.22*10^{18}$

  realna števila: `float` (32) $-3.4*10^{34}$ -- $3.4*10^{34}$ (6–7 mest)
  `double` (64) $-1.7*10^{308}$ -- $1.7*10^{308}$ (14-15 mest)

- za logične vrednosti: `boolean`
- za znake: `char` (16)

Kontrola toka:

- **zaporedje** (ang. sequence)
- **izbira** (ang. selection)
  - `if` stavek
  - `switch` stavek
  - pogojni operator `?:`
- **ponavljanje** (ang. iteration)
  - `do ... while` zanka
  - `while` zanka
  - `for` zanka

Deklaracija razreda

- glava razreda (ang. class header)
  - način dostopa: public / (final) / (abstract)
- deklaracije atributov
  - način dostopa: public / private / protected
- deklaracije metod
  - konstruktor (ang. constructor)
  - "setty"metode (ang. mutator)
  - "getty" metode (ang. accessor)
  - pomožne metode (ang. utility)
  - način dostopa: public / private

Primer: Razred Point ...

Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 2. PREDAVANJA (11.10.2007)

Študijsko leto 2007/08

Kratka ponovitev Jave 6

Kreiranje objektov

Dva koraka:

- deklaracija objekta: `Point p;`
- generiranje objekta: `p = new Point (3,6);`

Primer: razred TestPoint ...

Dopolnitev razreda Point:

- več konstruktorjev
- utility metode
- metoda equals

Primer: razred Point(1), TestPoint(1) ...

Dedovanje: razred podeduje atribute in metode drugega razreda

- osnovni razred (ang. base class) : izpeljan razred (ang. derived class)

ali

- nadrazred (ang. superclass) – podrazred (ang. subclass)

ali

- starš (ang. parent class) – otrok (ang. child class)

Pogoji:

- osnovni razred obstaja
- ključna beseda **`extends`**
- deklariramo samo dodatne atribute in metode
- lahko redefiniramo obstoječe metode
- konstruktor podrazreda mora klicati konstruktor nadrazreda - **`super`**

Primer: razredi ColoredPoint, TestColoredPoint ...

Abstraktni razred

Splošen nadrazred, ki je osnova za izpeljavo različnih podrazredov

- ena ali več abstraktnih metod
- ne moremo generirati objektov tega razreda, le objekte podrazredov

Smisel: izpeljava različnih podrazredov na tej osnovi

Primer: razred Element ...

Razširitev razreda Element

Primer: razred Student

Pojem sortiranja podatkov

Cilj: določiti splošno metodo za sortiranje kakršnekoli tabele objektov

Dogovor:

- algoritmi za sortiranje delujejo nad tabelo objektov tipa Element
- dejansko sortiramo tabelo objektov razširjenega tipa (recimo Student)
- (pripravljena) tabela Element[] a je metodi podana kot parameter
- podatke urejamo v naraščajoče urejenem vrstnem redu (če ni rečeno drugače)

Delitev metod glede na zapis podatkov:

- sortiranje tabel: notranje sortiranje
- sortiranje datotek: zunanje sortiranje*

Principi sortiranja podatkov tabel:

- vstavljanje
- izbiranje
- zamenjava

Princip sortiranja datotek:

- zlivanje podatkov s trakov*

** - ni predmet obravnave pri OAPS I*

Delitev metod glede na izvedčasovno zahtevnost:

- navadne metode: $O(n^2)$
- izboljšane metode: manj kot $O(n^2)$
  - najbolje $O(n*\log n)$

Izvedba metod je lahko:

- iterativna
- rekurzivna

Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 3. PREDAVANJA (18.10.2007)

Študijsko leto 2007/08

Sortiranje tabel – Navadno vstavljanje 12

Grob opis algoritma:

```
for (int i=1; i<a.length;++i)
{
   x=a[i];
   vstavi x na pravo mesto med elemente od a[0] do a[i];
}
```

Prikaz delovanja algoritma ...

Realizacija metode v Javi: metoda StraightInsertion ...

Primer: razred SortiranjeObjektov (z metodo StraightInsertion)...

Sortiranje tabel – Navadno vstavljanje 13

Prikaz rešitve:

- razred Element
- razred Student
- razred SortiranjeObjektov
- **razred GlavniProgram ...**

Nadgradnja razredov za lepši izpis:

- razred Element: nova abstraktna metoda,
- razred Student: metoda za izpis,
- SortiranjeObjektov: klic metode za izpis
- GlavniProgram ...

Analiza časovne kompleksnosti: $O(n^2)$

- število primerjav C - $O(n^2)$
- število premikov M (zamenjav) – $O(n^2)$

Sortiranje tabel – Dvojiško vstavljanje 14

Izboljšava postopka navadnega vstavljanja:

- mesto za vstavljanje elementa poiščemo z bisekcijo

Prikaz delovanja algoritma ...

Ralizacija metode v Javi: metoda BinaryInsertion ...

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo BinaryInsertion),
- sprememba razreda GlavniProgram ...

Analiza časovne kompleksnosti: $O(n^2)$

- število primerjav C : $O(n*\ln n)$
- število premikov M : $O(n^2)$

Grob opis algoritma:

```
for (int i=0; i<=a.length-2;++i)
{
   a[k] naj bo min objekt med a[i] do a[a.length-1];
   zamenjaj a[i] in a[k]
}
```

Prikaz delovanja algoritma ...

Realizacija metode v Javi: metoda StraightSelection ...

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo StraightSelection),
- sprememba razreda GlavniProgram ...

Analiza časovne kompleksnosti: C = $O(n^2)$, M = $O(n^2)$ ali O(n)

Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 4. PREDAVANJA (25.10.2007)

Študijsko leto 2007/08

Sortiranje tabel – Bubblesort 16

Grob opis algoritma:

```
for (int i=1; i<a.length;++i)
{
   primerjaj dva sosednja objekta in ju po potrebi zamenjaj;
}
```

Prikaz delovanja algoritma ...

Ralizacija metode v Javi: metoda Bubblesort ...

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo Bubblesort),
- sprememba razreda GlavniProgram ...

Analiza časovne kompleksnosti: $C = M = O(n^2)$

Sortiranje tabel – izboljšava Bubblesort1 17

Sprememba postopka:

- preverimo, če je v prehodu prišlo do zamenjave
- če ni bilo zamenjave, postopek prekinemo

Prikaz delovanja algoritma ...

Ralizacija metode v Javi: metoda Bubblesort1 ...

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo Bubblesort1),
- sprememba razreda GlavniProgram ...

Analiza časovne kompleksnosti: $C = M = O(n^2)$

Sortiranje tabel – izboljšava Bubblesort2 18

Sprememba postopka:

- zapomnimo si mesto zadnje zamenjave v prehodu
- naslednji prehod ustavimo na tem mestu

Prikaz delovanja algoritma ...

Ralizacija metode v Javi: metoda Bubblesort2 ...

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo Bubblesort2),
- sprememba razreda GlavniProgram ...

Analiza časovne kompleksnosti: $C = M = O(n^2)$

Sortiranje tabel – Shakersort 19

Sprememba postopka - Menjavamo smer prehodov:

- pregledovanje od leve proti desni prestavi na pravo mesto **min** element
- pregledovanje od desne proti levi pripelje na pravo mesto **max** element
- neurejen del na sredini tabele se oži z obeh strani

Prikaz delovanja algoritma ...

Ralizacija metode v Javi: metoda Shakersort ...

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo Shakersort),
- sprememba razreda GlavniProgram ...

Analiza časovne kompleksnosti: $C = M = O(n^2)$

Sortiranje tabel – Shellsort 20

Izboljšava navadnega vstavljanja:

- sortiramo v več etapah z različnimi koraki
- korak se postopoma zmanjšuje do vrednosti 1

Grob opis algoritma:

```
for (int m=0; m<T;++m)
{
   določi korak k za to etapo;
   for (int i=k;i<a.length;++i)
   {
      x=a[i];
      upoštevajoč korak k vstavi x na pravo mesto;
   }
}
```

Sortiranje tabel – Shellsort 21

Prikaz delovanja algoritma ...

Ralizacija metode v Javi: metoda Shellsort ...

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo Shellsort),
- sprememba razreda GlavniProgram ...

Analiza časovne kompleksnosti

$T = O(n^{1.2})$ – Wirth **$T = O(n^{1.5})$ – Hubbard**

Obnašanje algoritma je odvisno od pravilne izbire korakov:

- koraki naj zagotavljajo prepletanje verig
- primer slabe izbire: 16, 8, 4, 2, 1
- dve priporočeni formuli za izbiro korakov ...

Sortiranje tabel – Shellsort 22

Koraki: $h_1, h_2, h_3, \ldots, h_t$

$h_t = 1$

$h_{i-1} > h_i$

1. možnost:

$t = [\log_3 n] - 1$

$h_t = 1$

$h_{i-1} = 3 * h_i + 1$

Koraki: 1, 4, 13, 40, 121, 364, ...

2. možnost:

$t = [\log_2 n] - 1$

$h_t = 1$

$h_{i-1} = 2 * h_i + 1$

Koraki: 1, 3, 7, 15, 31, 63, 127, 255, ...

Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 6. PREDAVANJA (8.11.2007)

Študijsko leto 2007/08

Sortiranje tabel – Sortiranje s kopico

Lastnosti kopice:

- binarno drevo
- vozlišče drevesa ustreza objektu v tabeli
- za vsako vozlišče velja, da je objekt večji ali enak od vseh naslednikov
- dolžini dveh poljubnih vej v kopici se razlikujeta kvečjemu za ena, daljše veje so skrajno levo

Ponazoritev kopice v računalniku:

- še vedno sortiramo tabelo `Element[] a`
- koren je v `a[0]`
- levi sin vozlišča `a[i]` je v `a[2*i+1]`
- desni sin vozlišča `a[i]` je v `a[2*i+2]`

Sortiranje tabel – Sortiranje s kopico 24

Opis algoritma: 4 koraki

1. vhodno zaporedje uredimo v kopico
2. zamenjamo prvi in zadnji element v kopici
3. popravimo kopico
4. ponavljamo koraka 2 in 3, dokler ne zmanjka podatkov

Popravljanje kopice (del koraka 1 in 3)

- kopico z neustreznim korenom popravimo tako, da koren pogreznemo po veji z večjim naslednikom
- postopek ponavljamo, dokler ne popravimo kopice (ali pridemo do lista)
- **Podprogram za popravljanje kopice: `heapify`**

Sortiranje tabel – Sortiranje s kopico 25

Gradnja kopice

- uporabimo postopek pogrezanja
- objekti v desni polovici tabele (`i>=a.length/2`) nimajo naslednikov
- objekte pogrezamo v zanki od sredine tabele proti začetku
- **del metode za ta del ...**

Zanka, v kateri ponavljamo koraka 2 in 3

- **Del programa za ta del ...**

Celoten algoritem ...

Prikaz delovanja algoritma ...

Ralizacija metode v Javi: metoda Heapsort ...

Sortiranje tabel – Sortiranje s kopico 26

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo Heapsort),
- sprememba razreda GlavniProgram ...

Izboljšava navadnega izbiranja

Analiza časovne kompleksnosti:

- gradnja kopice na začetku: $O(\log_2 n)$
- popravljanje kopice: $O(\log_2 n)$
- zamenjava prvega in zadnjega objekta: $O(n)$
- **Skupaj:** $T_w = O(n*\log_2 n)$

Sortiranje tabel – Quicksort 27

Osnova: porazdelitveni algoritem, ki tabelo razdeli na 2 dela:

1. izberemo poljubni element tabele **`x`** (npr. srednji)
2. pregledujemo tabelo od leve proti desni, dokler ne najdemo **`a[i] > x`**
3. pregledujemo tabelo od desne proti levi, dokler ne najdemo **`a[j] < x`**
4. zamenjamo **`a[i]`** in **`a[j]`**
5. ponavljamo korake 2 – 4, dokler se pregledovanji ne srečata

   Porazdelitveni algoritem ...

   Prikaz delovanja algoritma ...
6. postopek **rekurzivno** ponavljamo na levem in desnem delu tabele

Ralizacija metode v Javi: metodi Quicksort in Sort ...

Sortiranje tabel – Quicksort 28

Primer:

- dopolnitev razreda Sortiranje objektov (z metodama Quicksort in Sort),
- sprememba razreda GlavniProgram ...

Izboljšava principa navadne zamenjave

Analiza časovne kompleksnosti:

- ena porazdelitev: O(n)
- število rekurzivnih klicev:
  - ugoden (povprečen) primer: $O(\log_2 n)$
  - najslabši primer: O(n)
- **Skupaj:** $\mathbf{T_a = O(n*\log_2 n)}$, vendar $\mathbf{T_w = O(n^2)}$
- **Iskanje najslabše permutacije ...**

Sortiranje tabel – Mergesort 29

Ideja iz sortiranja datotek

Še ena implementacija metode Deli in vladaj:

1. Razdeli tabelo na dve (približno) enako veliki podtabeli
2. Rekurzivno uredi levo podtabelo
3. Rekurzivno uredi desno podtabelo
4. Zlij levo in desno podtabelo tako, da ohraniš urejenost

Prikaz delovanja algoritma ...

Problem zlivanja: pomožna tabela `temp`

Ralizacija metode v Javi: metode Mergesort, Msort in Merge ...

Primer:

- dopolnitev razreda Sortiranje objektov
  (z metodami Mergesort, Msort in Merge),
- sprememba razreda GlavniProgram ...

Izboljšava principa navadne zamenjave

Analiza časovne kompleksnosti:

- porazdelitev ni
- število rekurzivnih klicev Msort: $O(\log_2 n)$
- zlivanje: $O(n)$
- **Skupaj:** $\mathbf{T = O(n*\log_2 n)}$, vendar slabše kot Quicksort

Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I (OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 7. PREDAVANJA (15.11.2007)

Študijsko leto 2007/08

Sortiranje tabel – Quicksort 31

Iterativna rešitev:

- izdelamo seznam zahtev po porazdelitvah, ki še niso bile opravljene
- vsakič nastaneta 2 zahtevi:
  - eno (levo) obdelamo takoj,
  - drugo (desno) umestimo na seznam
- zahtevke obravnavamo v obratnem vrstnem redu: **utripajoč sklad**

Ponazoritev zahtevka: leva in desna meja dela tabele

Ponazoritev sklada: tabela objektov tipa `ElementSklada`

Razred ElementSklada ...

Prikaz delovanja algoritma ...

Sortiranje tabel – Quicksort 32

Ralizacija metode v Javi: metoda QuicksortI ...

Problem velikosti sklada

Ralizacija metode v Javi: popravljena metoda QuicksortI ...

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo QuicksortI),
- sprememba razreda GlavniProgram ...

Primer alternativne rešitve:

- dopolnitev razreda Sortiranje objektov (z metodo QuicksortI1),
- sprememba razreda GlavniProgram ...

Dodatna optimizacija metode:

- izbira "pravega" srednjega elementa
- kombinacija z navadno metodo

Sortiranje tabel – Bucketsort 33

Posplošitev ideje deli in vladaj do skrajnosti: problem delimo na toliko delov/košev, kolikor je elementov ...

Primeren, ko imamo veliko zalogo vrednosti (recimo n-mestna števila)

1. Pripravi (pomožne) tabele (zbirke) za n košev
2. Določi najmanjši in največji element (**`min`** in **`max`**)
3. Porazdeli elemente po koših:

   `a[i]` sodi v koš j, kjer je **`j= n*(a[i]-min)/(max-min+1)`**
4. Sortiraj vsak koš posebej (s kakršnokoli metodo)
5. Sestavi koše (po zaporedju) zopet v tabelo **`a`**

Prikaz delovanja algoritma ...

Težave: - iskanje najmanjšega in največjega elementa

- izvedba n zbirk s tabelami

Ralizacija metode v Javi: metoda Bucketsort ...

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo Bucketsort),
- dopolnitev razreda Student (z metodo Vrednost)
- sprememba razreda GlavniProgram ...

Analiza časovne kompleksnosti:

- število košev n: O(n)
- koši imajo v povprečju enega (ali le nekaj elementov)
- **Skupaj: T = O(n),** vendar slabše kot Quicksort
- potrebujemo veliko dodatnega prostora !!!

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 8. PREDAVANJA (22.11.2007)

Študijsko leto 2007/08

Sortiranje tabel – BucketsortK

Izboljšava:

1. Pripravi (pomožne) tabele (zbirke) za fiksno število **K** košev

...

4. Sortiraj vsak koš posebej (**rekurzivno** z metodo `BucketsortK`)

...

Ralizacija metode v Javi: metoda BucketsortK ...

Primer:

- dopolnitev razreda Sortiranje objektov (z metodo BucketsortK),
- sprememba razreda GlavniProgram ...

Analiza časovne kompleksnosti:

- število košev K
- število rekurzivnih klicev: $O(\log_K n)$

Omejitev:

- deli atributa (ključa) za urejanje imajo pomen
- uporabno za sortiranje **števil** in **nizov** (znakov)

Postopek:

1. elemente tabele razvrstimo v več razredov glede na vrednost dela atributa (mesto v ključu)
2. postopek ponavljamo za vse dele (mesta) od najmanj do najbolj pomembnega
3. vsaka iteracija elemente najprej porazdeli in nato spet prepiše v tabelo `a`

Prikaz delovanja algoritma za sortiranje trimestnih števil ...

Ralizacija v Javi: razreda LeksiSort in TestLeksiSort ...

Leksikografsko sortiranje 37

Primer za sortiranje števil:

- razreda LeksiSort in TestLeksi Sort...

Spremembe (dodatki) za sortiranje nizov...

Prikaz delovanja algoritma za sortiranje nizov ...

Ralizacija v Javi: razreda LeksiSort1 in TestLeksiSort1 ...

Primer za sortiranje nizov:

- dopolnitev razreda LeksiSort in TestLeksi Sort...

Analiza časovne kompleksnosti:

- upoštevamo število premikov, primerjave niso primerljive
- navidez zelo dobro: O(n)

Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 9. PREDAVANJA (29.11.2007)

Študijsko leto 2007/08

Vmesnik (ang. interface)

Nekateri programski jeziki omogočajo **večkratno dedovanje:** podrazred podeduje atribute in metode več kot enega nadrazreda

Problemi:

- atributi in metode v nadrazredih imajo enaka imena
- `super()` – klic konstruktorja iz katerega nadrazreda?

Java ne omogoča večkratnega dedovanja

Namesto tega ponuja koncept vmesnika

Razlika med vmesnikom in razredom:

- vse metode v vmesniku morajo biti `abstract`
- vsi atributi (če jih ima) morajo biti `static final`

Vmesnik predpiše metode, ki jih mora implementirati nek podrazred

=> predpiše obnašanje podrazreda

Vmesnik (ang. interface) 39

Podrazred lahko deduje samo od enega nadrazreda, implementira pa lahko več različnih vmesnikov.

```
public class Podrazred extends Nadrazred implements
                                        Vmesnik1, Vmesnik2
```

S stališča dedovanja `Podrazred`:

- podeduje atribute in metode razreda **`Nadrazred`**
- dodatno deklarira nove atribute in metode
- če mu katera od podedovanih metod ne ustreza, jo lahko redefinira

S stališča implementacije vmesnika `Podrazred`:

- deklarira vse metode, ki so specificirane v vmesnikih **`Vmesnik1`**, **`Vmesnik2`**
- lahko uporblja atribute (statične spremenljivke), ki so definirani v vmesnikih

Podrazred je razširitev treh tipov: Nadrazred, Vmesnik1 in Vmesnik2

Vmesnik (ang. interface) 40

Primerjava: abstraktni razred - vmesnik

Enako:

- ne moremo generirati objektov abstraktnega tipa ali vmesnika

Različno:

- v abs.razredih so lahko samo nekatere metode abstraktne, v vmesniku so vse
- atributi abs.razreda se obnašajo kot spremenljivke objektov, atributi vmesnika pa so statične konstante.

Uporaba abstraktnega razreda:

- vnaprej deklariramo znane atribute in metode, ki jih podeduje podrazred: recimo pri igri s kartami metodo **`mesaj()`**

Uporaba vmesnika:

- vmesnik določa le del značilnosti, ki jih vsak podrazred implementira na svoj način: recimo pri glasbenih instrumentih metodo **`zaigrajTon()`**

Zbirka (Collection) 41

Namen:

- podatkovne strukture za shranjevanje objektov določenega tipa
- zbirka objektov tipa `Object` ali drugega iz `Object` izpeljanega tipa
- varna in učinkovita uporaba tipičnih operacij za izbrano vrsto zbirke

Primer: seznam, sklad (LIFO), vrsta (FIFO), množica, razpršena tabela, ...

Realizacija v javi: Collections Framework – množica vmesnikov in razredov za različne vrste zbirk

Zbirke po različicah jave:

- do jave 1.2 : **`Vector`, `Stack`, `Hashtable`, `Bitset`, `Enumeration`**
- do jave 1.5: razredi in vmesniki po zgledu C++ (*Standard Template Library*)
- od jave 1.5: generični razredi in vmesniki

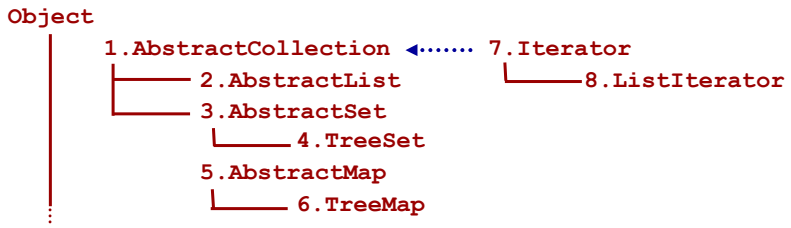
Zbirka (Collection) 42

Vnaprej predpisani vmesniki (v `java.util`):

`1.Collection`	osnovni vmesnik za neko splošno zbirko objektov
`2.List`	zaporedje elementov (seznam)
`3.Queue`	zaporedje elementov – FIFO (vrsta)
`4.Set`	zbirka elementov brez duplikatov (množica)
`5.SortedSet`	urejena zbirka elementov brez duplikatov
`6.Map`	zbirka parov (key, value); ključi morajo biti enolični
`7.SortedMap`	urejena zbirka parov (key, value); ključi morajo biti enolični
`8.Iterator`	objekt, s katerim se sprehajamo po zbirki
`9.ListIterator`	objekt, s katerim se sprehajamo po zaporedno urejeni zbirki
`10.Comparator`	objekt za primerjavo dveh elementov zbirke

Zbirka (Collection) 43

Obstajajo razredi, ki implementirajo te vmesnike:



Vsi vmesniki in razredi so definirani **generično** - recimo **`TreeSet<E>`**

V nadaljevanju:

- pregled posameznih podatkovnih struktur in njihove izvedbe v javi
- poudarek na naboru operacij (metod) in njihovi uporabi

Zbirka (Collection) 44

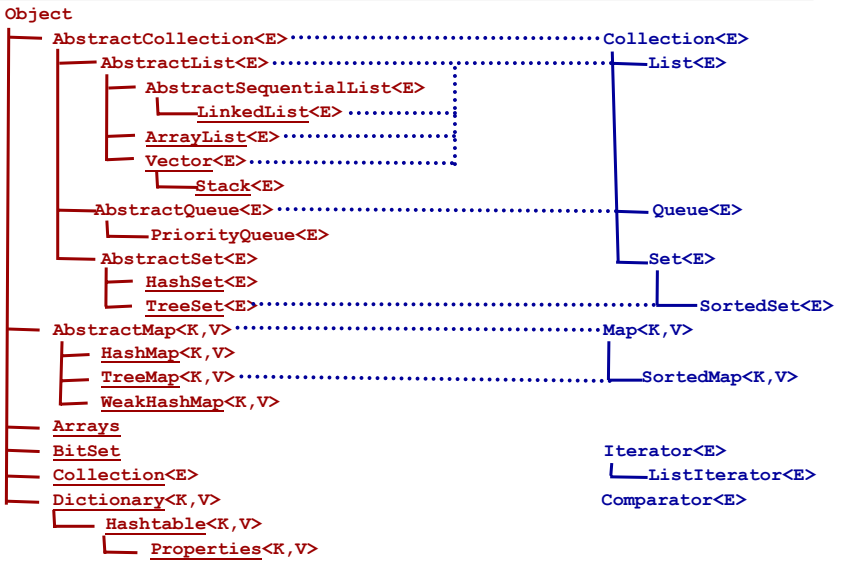


Tabela (razred `Arrays`) 45

Statična podatkovna struktura ...

Lastnosti:

- Tabele so objekti.
- Tabele se tvorijo (kreirajo) dinamično.
- Tabele so lahko prirejene objektom tipa `Object`.
- Vsaka metoda na objektom tipa `Object` se lahko uporabi nad tabelo.
- Tabela vsebuje zaporedje spremenljivk določene vrste.
- Spremenljivke imenujemo elementi tabele.
- Če je tip elementa `Type`, potem je tip tabele `Type[]`.
- Vsebina spremenljivke tipa tabela je naslov (referenca) na tabelo.
- Element tabele je lahko tudi tabela.

Tabela (razred `Arrays`) 46

- Element tabele je lahko osnovnega tipa ali objekt (naslov).
- Dolžina tabele je enaka številu elementov tabele.
- Dolžina tabele je določena ob kreiranju in je nespremenljiva.
- Dolžino tabele lahko preberemo s pomočjo spremenljivke `length`.
- Indeksi tabele tečejo med `0` in `length-1`.
- Prekoračitev sproži izjemo `ArrayIndexOutOfBoundsException`.
- Indeks je lahko tipa `int`, `short`, `byte` ali `char`.
- Za (popolno) kopiranje tabel uporabljamo metodo `Object.clone()`.
- Za preverjanje enakosti uporabljamo metodo `Arrays.equals()`.
- Tabele implementirajo lastnosti (vmesnik) `Clonable` in `Serializable`.

Tabela (razred `Arrays`) 47

Razred `Arrays` (`java.util.Arrays`)

Metode:

- `public static List asList (Object[])`
- `public static int binarySearch (...)`
- `public static boolean equals (...)`
- `public static void fill (...)`
- `public static void sort (...)`

`...` – tabela kateregakoli osnovnega tipa (razen `boolean`) ali `Object`

- Vse metode so statične
- Konstruktor je `private`

Primer: Tabele.java

Generični razredi in vmesniki (Generics) 48

Generični razred (vmesnik) predstavlja podatkovno strukturo, kateri je mogoče predpisati tip objektov, ki jih vsebuje

Velja v Javi od verzije (1.)5.0 dalje

Vsi vmesniki in razredi Collection Framework-a so generični

Primer: `Vector<String> v = new Vector<String>();`

- Tak pristop omogoča nadzor tipa objektov, ki jih vnašamo v strukturo - ni treba podrazredov, redefinicije metod in `instanceof` preverjanj
- V praksi se izognemo prilagajanju (casting), ker vse metode razreda vračajo predpisan tip `<E>`
- Dejansko se tip določi med samim izvajanjem, zato poskus kršenja pravil povzroči napako med izvajanjem
- Tip mora biti obvezno tip objektov, ne katerega izmed primitivnih tipov

`Vector<int> s = new Vector<int>(); NAROBE!!!`

Generični razredi in vmesniki (Generics) 49

Uporaba generičnega razreda brez določanja tipa:

```
Vector vektor = new Vector();

Note: C:\Vektor.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Process completed.
```

Izogibanje opozorilom:

```
Vector<?> seznam = new Vector<?>();
```

Uporaba pri argumentih metod:

```
public static Vector<String> zlij (Vector<String> n1,
                                   Vector<String> n2)
  {...}
```

Uporaba pri iteratorju:

```
for(Iterator<String> it=n.iterator(); it.hasNext(); )
    {...}
```


Generični razredi in vmesniki (Generics) 50

Pozor pri avtomatski pretvorbi tipov:

```
Vector<Short> lls = new Vector<Short>();

Vector<Short> l = lLs;          // V REDU !!

Vector<Number> lli = lls; // NAPAKA !!
```

Pisanje generičnih razredov:

```
public class Box<T>
{
    protected Vector<T> seznam;

    public Box()
    {
            seznam = new Vector<T>();
    }
    public T vrni() { ... };
    public void sprejmi(T element) { ... };
}
```

Uporaba pri dedovanju:

```
public class NumberBox<T extends Number> extends Box<T>
{
    public NumberBox()
    {
            super();
    }
    ...
}
```


Definira zbirko elementov tipa **`E`**, ki se obnaša podobno kot tabela.

Razlike v primerjavi s tabelo:

- velikost vektorja se po potrebi avtomatsko povečuje
- vektor lahko hrani objekte različnih tipov E

Definiran v paketu **`java.util...`**

Spremenljivka tipa **`Vector`** vsebuje naslov lokacije v pomnilniku

Kreiranje vektorjev - 4 konstruktorji:

- `Vector<E> v=new Vector<E>();`
- `Vector<E> v=new Vector<E>(int);`
- `Vector<E> v=new Vector<E>(int, int);`
- `Vector<E> v=new Vector<E>(Collection)`

Vektor (razred `Vector<E>`) 53

Kapaciteta in velikost vektorja:

- **capacity**: število objektov, ki jih lahko spravimo v vektor
- **size**: dejansko število shranjenih objektov
- pozicije v vektorju: od `0` do `size-1`

V razredu `Vector` obstaja več metod:

- `int capacity();`
- `int size();`
- `ensureCapacity(int);`
- `setSize(int);`
- `trimToSize();`

Vektor (razred `Vector<E>`) 54

Shranjevanje objektov v vektor:

- `boolean add(E);`
- `add(int, E);`
- `addElement(E);`
- `E set (int, E);`
- `setElementAt(E, int);`
- `boolean addAll(Collection);`
- `boolean addAll(int, Collection);`

Branje podatkov iz vektorja:

- `E get(int);`
- `E elementAt(int);`
- `E firstElement();`
- `E lastElement();`

Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 10. PREDAVANJA (6.12.2007)

Študijsko leto 2007/08

Vektor (razred `Vector<E>`)

Konverzija tipa **ni potrebna**, ker je razred generičen

Primer branja, če vektor hrani objekte tipa `Delavec`:

```
Delavec d= (Delavec) v.get(4);
...
Delavec d= (Delavec) v.firstElement();
```

Primer obdelave vseh elementov vektorja objektov tipa `Delavec`:

```
Delavec d;
for (int poz=0; poz< v.size(); ++poz)
{
    d= (Delavec) v.get(poz);
    // nadaljna obdelava objekta d
{
```

Vektor (razred `Vector<E>`) 56

Prepis elementov iz vektorja v tabelo

- uporaba vektorja zahteva dodatno režijo, zato včasih smiselno

Metodi:

- `E[] toArray();`
- `E[] toArray(E[]);`

Odstranjevanje elementov iz vektorja:

Pri odstranjevanju se kapaciteta ne zmanjšuje, zmanjšuje se samo velikost.

- `E remove(int);`
- `boolean remove (E);`
- `removeElementAt(int);`
- `clear();`
- `boolean removeAll(Collection);`

Vektor (razred `Vector<E>`) 57

Iskanje elementov v vektorju:

- `int indexOf(E);`
- `int indexOf(E, int);`
- `int lastIndexOf(E);`
- `int lastIndexOf(E, int);`

Primer: vsa nastopanja objekta `d` v vektorju:

```
...
int poz=0;
while (poz<v.size() && poz>=0)
{
   poz = v.indexOf(d,poz);
   if (poz != -1)
   {
      // obdelava objekta na poziciji poz
      ++poz;
   }
}
```

Vektor (razred `Vector<E>`) 58

Iterator

V vseh razredih, ki implementirajo vmesnik **`Collection<E>`**, obstaja objekt **`Iterator<E>`**, ki omogoča sprehajanje po elementih zbirke.

Razred **`Vector<E>`** implementira vmesnik **`Collection<E>`**.

`Iterator` ima konstruktor in 3 metode:

- `Iterator<E>();`
- `E next();`
- `boolean hasNext();`
- `remove();`

Vektor (razred `Vector<E>`) 59

Primer: Obhod s pomočjo iteratorja:

```
Delavec d;
Iterator<Delavec> it=v.iterator();
while(it.hasNext())
{
   d=it.next();
   //se naprej stavki za obdelavo tega elementa
}
```

Primer: Vektor.java

VektorG.java

Sklad (razred `Stack<E>`) 60

Podatkovna struktura, ki deluje po principu LIFO

Primer: zlaganje kovancev ...

Razred `Stack<E>` (razširitev razreda `Vector<E>`)

Operacije za delo s skladom:

- `empty()`
- `peek()`
- `pop()`
- `push (E)`
- `search (E)`

Deklaracija razreda `Stack` ...

Sklad (razred `Stack<E>`) 61

Dva primera:

a) Računanje aritmetičnih izrazov v postfiksni obliki (brez oklepajev)

- najprej zapišemo operanda, nato operator: `3 4 +` pomeni `3 + 4`
- **Postopek:**
  - beremo od leve proti desni
  - če naletimo na operand, ga postavimo na sklad
  - če naletimo na operator, vzamemo dva operanda s sklada, izvedemo operacijo in rezultat zapišemo na sklad
- **Prikaz izračuna ...**
- **Realizacija metode v Javi: razred `Kalkulator`**
- **Primer: Kalkulator.java**

Sklad (razred `Stack<E>`) 62

b) Odprava rekurzije s pomočjo sklada (hanojski stolpiči)

- **Opis problema:** n obročev različnih velikosti in 3 palice (A, B, C)
  - **Začetek:** - vsi obroči so na palici A
    - zloženi so od največjega do najmanjšega
  - **Cilj:** prestaviti obroče na palico C s pomočjo pomožne palice B
  - **Omejitvi:** - naenkrat lahko prestavimo en obroč
    - nikoli ne smemo prestaviti večji obroč na manjšega
- **Narava problema je rekurzivna:**
  - prestavi (n-1) obročev s palice A na palico B
  - prestavi en obroč s palice A na palico C
  - prestavi (n-1) obročev s palice B na palico C

Sklad (razred `Stack<E>`) 63

Rekurzivna rešitev:

- **Prikaz postopka ...**
- **Rekurzivna rešitev v Javi: razred `HanoiRek`**
- **Primer: HanoiRek.java**

Iterativna rešitev s skladom:

- **Postopek:**
  - na sklad postavimo začetni zahtevek
  - s sklada jemljemo zahtevke in jih obdelujemo (pri tem se na sklad dodajajo novi zahtevki)
  - postopek ponavljamo, dokler sklad ni prazen

Sklad (razred `Stack<E>`)

- **Grob opis postopka:**

```
postavi začetni zahtevek na sklad;
dokler sklad ni prazen
{   odvzemi zahtevek s sklada;
    obdelaj zahtevek; }
```

- **Prikaz postopka ...**
- **Predstavitev zahtevka:** razred **`Zahtevek`**

  Štiri atributi:

  n – število obročev, ki jih je treba prestaviti

  a – izvor

  b – pomožna palica

  c - ponor
- **Iterativna rešitev v Javi: razred `HanoiIte`**
- **Primer: HanoiIte.java**

Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 11. PREDAVANJA (13.12.2007)

Študijsko leto 2007/08

Ostale značilnosti Jave (1.)5.0

Avtomatska pretvorba ovojnih tipov - "autoboxing" in "unboxing":

```
Integer I1 = new Integer(5); // pred Javo 5.0
Integer I2 = 5;    // autoboxing
Number N = 2.2f;
int i1 = I1;       // unboxing
Integer I3 = null;
int i2 = I3;       // NAPAKA!!
```

Krajši zapis zanke for (za tabele in zbirke):

```
int[] t = {1,2,3,4,5,6,7,8,9,10};
for (int i : t) {          // zadaj se skriva iterator!!
    System.out.println(i);}
List sez = Arrays.asList(t);
for (Object i : sez) {...}
```

V svojem razredu moramo definirati ustrezen generični iterator ...

Ostale značilnosti Jave (1.)5.0 66

Naštevni tipi:

```
public enum Ocena = {5,6,7,8,9,10,BREZ};
class Student
{   ...
    Ocena oc1=Ocena.BREZ;
    Ocena oc2=Ocena.8;
    ... }
```

Nedoločeno število argumentov - “varargs”:

```
public void izpis(String s1, String...ostali);
{
    System.out.println(s1);
    for (String i:ostali)
            System.out.println(i);
    ... }
```

Oblikovanje izpisa – `printf(format,args)`:

```
System.out.printf("%4d %5.3f %-6s",15,3.14159,"Haha");
```


Ostale značilnosti Jave (1.)5.0 67

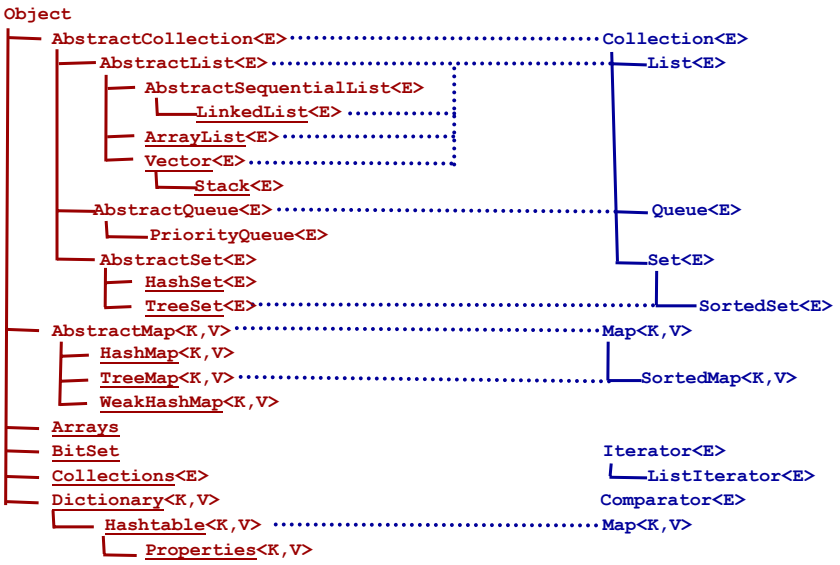
Statični import stavek:

```
import static java.lang.System.out;
public class Test {
    public static void main(String[] args) {
            println("Dober dan!");
    }
}
```

Dopolnjen razred `Arrays`:

```
int[][] t1={{1,2,3},{4,5,6},{7,8,9}};
int[][] t2={{1,2,3},{4,5,6},{7,8,9}};
System.out.println(Arrays.deepToString(t1));
if(Arrays.deepEquals(t1,t2))
    {...}
System.out.println(Arrays.deepHashCode(t1));
```

Collections Framework



Zbirka (vmesnik `Collection<E>`)

Definiran v paketu `java.util`

Zahteva implementacijo naslednjih metod:

- `public boolean add(E);`
- `public boolean addAll(Collection<? extends E>);`
- `public void clear();`
- `public boolean contains(Object);`
- `public boolean containsAll(Collection<?>);`
- `public boolean equals(Object);`
- `public int hashCode();`
- `public boolean isEmpty();`

Zbirka (vmesnik `Collection<E>`) 70

- `public abstract Iterator<E> iterator();`
- `public boolean remove(Object);`
- `public boolean removeAll(Collection<?>);`
- `public boolean retainAll(Collection<?>);`
- `public abstract int size();`
- `public Object[] toArray();`
- `public E[] toArray(E[]);`
- `public String toString();`

Razred `Collections<E>`: knjižnica statičnih metod za delo z zbirkami (kot `Arrays<E>` za tabele)

Zbirka (razred `AbstractCollection<E>`) 71

Delna implementacija vmesnika `Collection<E>`

- Implementira vse, kar se da implementirati, ne da bi poznali pomnilniško strukturo za predstavitev zbirke
- Manjka implementacija metod **`equals()`** in **`hashCode()`**
  - metode se podedujejo iz razreda **`Object`**
  - zanašamo se, da te metode redefinirajo podrazredi
- Po drugi strani razred redefinira metodo **`toString()`**
- Nekatere metode uporabljajo metodi **`iterator()`** in **`size()`**, ki sta še vedno abstraktni

Primer 1: metoda `toString()`

Primer 2: metoda `isEmpty()`

Torba (razred `Bag`) 72

Poljubna zbirka objektov, ki lahko vsebuje duplikate

bag = "torba" = multiset = množica z duplikati

Prikaz razširitve razreda **`AbstractCollection`**

`Bag` bo sprva razvit na negeneričen način

Postopen prikaz vsebine razreda **`Bag`**

Za predstavitev objektov bomo uporabili tabelo:

```
import java.util.*;
public class Bag extends AbstractCollection
{
    private Object[] objects;
    private int size = 0;
    private static final int CAPACITY = 16;
```


Torba (razred `Bag`) 73

Metode:

- `private void resize()`
- `public Bag()`
- `public Bag(int)`
- `public Bag(Object[])`
- `public Bag(Collection)`
- `public boolean add(Object)`
- `public boolean addAll(Collection)`
- `public void clear()`
- `public boolean contains(Object)`
- `public boolean containsAll(Collection)`

Torba (razred Bag) 74

- `private static int frequency(Collection, Object)`
- `public boolean equals(Object)`
- `public int hashCode()`
- **`public boolean isEmpty()`**

Realizacija iteratorja za razred `Bag`:

- potrebujemo metodo public **`Iterator iterator();`**
- vmesnik predpisuje 3 metode: **`hasNext()`, `next()`, `remove()`**

1. možnost:

- deklariramo razred, ki implementira vmesnik **`Iterator`**
- na podlagi tega lahko generiramo objekt tipa **`Iterator`**

Torba (razred Bag) 75

```
private class BagIterator implements Iterator
{   public boolean hasNext()
    public Object next()
    public void remove()
}
public Iterator iterator()
{   return new BagIterator();  }
```

2. možnost:

- anonimni notranji razred (anonymous inner class)

```
public Iterator iterator()
{   return new Iterator()
    {       public boolean hasNext()
            public Object next()
            public void remove()
    };
}
```

Torba (razred `Bag`) 76

Preostale metode:

- `public boolean remove(Object)`
- `public boolean removeAll(Object)`
- `public boolean removeAll(Collection)`
- `public boolean retainAll(Collection)`
- `public int size()`
- `public Object[] toArray()`
- `public Object[] toArray(Object[])`
- `public String toString()`

Primer:

- razred `Bag.java`
- razred `TestBag.java`

Torba (razred `Bag`) 77

Sprememba v generični razred `Bag<E>`:

- glava razreda
- notranja podatkovna struktura – še vedno tabela objektov:

  `E[] contents = (E[]) new Object[CAPACITY];`
- uporaba `AbstractCollection<E>`
- spremembe argumentov metod (ne povsod!)
- spremembe iteratorja

Primer:

- razred `BagG.java`
- razred `TestBagG.java`

Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 12. PREDAVANJA (20.12.2007)

Študijsko leto 2007/08

Seznam (`List<E>`)

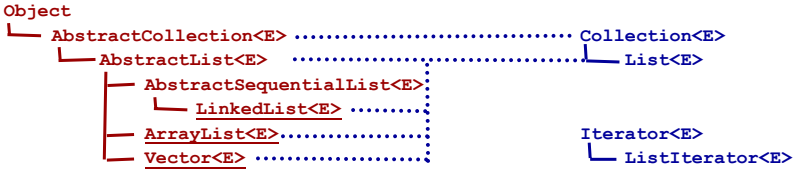
Podatkovna struktura predstavlja splošno zaporedje objektov:

- pojem pozicije
- dodajanje in izločanje objektov v predvidenem času

Primer: kompozicija vagonov ...

`Collection Framework` vsebuje:

- vmesnik **`List<E>`**
- abstraktna razreda: **`AbstractList<E>`**, **`AbstractSequentialList<E>`**
- več realiziranih razredov: **`LinkedList<E>`**, **`ArrayList<E>`**, **`Vector<E>`**
- iterator **`ListIterator<E>`**



Seznam (vmesnik `List<E>`) 79

```
public interface List<E> extends Collection<E>
{
    boolean add(E);
    add (int, E);
    boolean addAll(Collection);
    boolean addAll(int, Collection);
    clear();
    boolean contains(E);
    boolean containsAll(Collection);
    boolean equals(E);
    E get(int);
    int hashCode();
    int indexOf(E);
    boolean isEmpty();
```


Seznam (vmesnik `List<E>`) 80

```
    Iterator<E> iterator();
    int lastIndexOf(E);
    ListIterator listIterator<E>();
    ListIterator listIterator<E>(int);
    boolean remove(E);
    E remove(int);
    boolean removeAll(Collection);
    boolean retainAll(Collection);
    E set(int, E);
    int size();
    List subList(int, int);
    Object[] toArray();
    E[] toArray(E[]);
}
```

Seznam (abstraktni razred `AbstractList<E>`) 81

Dve realizaciji:

			get() set()	add() remove()
AbstractList	ArrayList	Tabela objektov	0(1)	0(n)
AbstractSequentialList	LinkedList	Povezan seznam objektov	0(n)	0(1)

```
public abstract class AbstractList<E> extends AbstractCollection<
      implements List<E>
{  (protected) AbstractList<E>()
   boolean add(E); //opcijsko
   add (int, E); // opcijsko
   boolean addAll(Collection);
   boolean addAll(int, Collection);
   clear();
   //boolean contains(E);
   //boolean containsAll(Collection);
   boolean equals(E);
   (abstract) E get(int);
```


Seznam (abstraktni razred `AbstractList<E>`) 82

```
   int hashCode();
   int indexOf(E);
   //boolean isEmpty();
   Iterator<E> iterator();
   int lastIndexOf(E);
   ListIterator<E> listIterator();
   ListIterator<E> listIterator(int);
   //boolean remove(E);
   E remove(int); // opcijsko
   removeRange(int, int);
   //boolean removeAll(Collection);
   //boolean retainAll(Collection);
   E set(int, E); // opcijsko
   //int size();
   List subList(int, int);
   //Object[] toArray();
   //E[] toArray(E[]);
}
```

Seznam (ab.razred AbstractSequentialList<E>) 83

```
public abstract class AbstractSequentialList<E> extends
    AbstractList<E>
{ (protected) AbstractSequentialList<E>()
  boolean add(E);
  boolean addAll(Collection);
  boolean addAll(int, Collection);
  E get(int);
  Iterator<E> iterator();
  int lastIndexOf(E);
  (abstract) ListIterator<E> listIterator();
  E remove(int);
  E set(int, E);
  (abstract) int size();
}
```


Seznam (razred ListIterator<E>) 84

Dvosmerni iterator

```
public interface ListIterator<E> extends Iterator<E>
{
   boolean hasNext();
   E next();
   remove();
   boolean hasPrevious();
   E previous();
   int nextIndex();
   int previousIndex();
   add(E);
   set(E);
}
```

Seznam (razred ArrayList<E>)


```
public class ArrayList<E> extends AbstractList<E>
      implements List<E>
{
   //vse metode iz vmesnika List ter dodatno:
   ArrayList<E>();
   ArrayList<E>(int);
   ArrayList<E>(Collection);

   ensureCapacity(int);
   trimToSize();
}
```


Seznam (razred LinkedList<E>)


```
public class LinkedList<E> extends AbstractSequentialList<E>
      implements List<E>
{  //vse metode iz vmesnika List ter dodatno:
   addFirst(E);
   addLast(E);
   E getFirst();
   E getLast();
   // manjka metoda get()
   LinkedList<E>();
   LinkedList<E>(Collection);
   boolean remove(E);
   E removeFirst();
   E removeLast();
}
```

Primer: Seznam.java

Vrsta (`Queue<E>`) 87

Podatkovna struktura, ki deluje po principu FIFO

Primer: vrsta pred blagajno v kinu ...

`Collection Framework` vsebuje nekoliko drugačen vmesnik in razred za prioritetno vrsto, zato zastavimo **svojo hierarhijo**

Sprva razvijemo **negenerične vmesnik in razrede**:

Izhajamo iz vmesnika **`Collection`**

- **3 deli:**
  - vmesnik **`Queue`**
  - abstraktni razred **`AbstractQueue`**
  - dve izvedbi: razreda **`ArrayQueue`** in **`LinkedQueue`**

Vrsta (vmesnik `Queue` in razred `AbstractQueue`) 88

- od vmesnika **`Collection`** podeduje 15 metod
- doda 4 operacije za delo z vrsto:
  - **`Object dequeue()`**
  - **`Object enqueue(Object)`**
  - **`Object getBack()`**
  - **`Object getFront()`**

Prikaz delovanja vrste ...

Deklaracija vmesnika `Queue` ...

Abstraktni razred `AbstractQueue`

- razširitev razreda **`AbstractCollection`**
- implementira vmesnik **`Queue`**

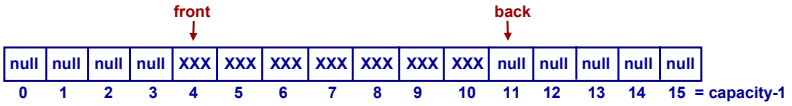
Deklaracija razreda `AbstractCollection`

Vrsta (razreda `ArrayQueue`) 89

Realizacija s tabelo objektov

Razširitev razreda **`AbstractQueue`**

Predstavitev vrste:



Deklaracija razreda `ArrayQueue`

Primer:

- **vmesnik `Queue.java`**
- **abstraktni razred `AbstractQueue.java`**
- **razred `ArrayQueue.java`**
- **razred `TestArrayQueue.java`**

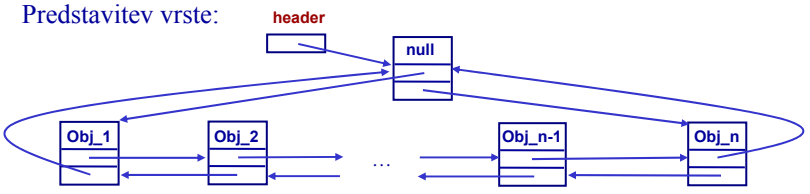
Vrsta (razred `LinkedQueue`) 90

Razširitev razreda **`AbstractQueue`**

Realizacija z **dvosmernimi seznami**: zaporedje elementov, ki so med seboj povezani v obeh smereh

Pojem slepega elementa ...

Predstavitev vrste:



Deklaracija razreda `LinkedQueue`

Primer:

- **razred `LinkedQueue.java`**
- **razred `TestLinkedQueue.java`**

Primer: QueueG.java

AbstractQueueG.java

ArrayQueueG.java

TestArrayQueueG.java

Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PREDVIDENE PROSOJNICE ZA NASLEDNJA PREDAVANJA

Študijsko leto 2007/08

Vrsta v Collection Framework (Java 1.5.0)

`Queue<E>`, `PriorityQueue<E>` in `Comparator<E>`:

- Metode vmesnika `Queue<E>` in razreda `PriorityQueue<E>`: `element()`, `offer(E)`, `peek()`, `pool()`, `remove()`
- `PriorityQueue<E>` upošteva naravni red ali redefinirani `Comparator<E>`

```
PriorityQueue<Integer> pv1 = new PriorityQueue<Integer>(10);
PriorityQueue<Integer> pv2 = new PriorityQueue<Integer>(10,
    new Comparator<Integer>(){
        public int compare(Integer i, Integer j)
        { int rez = i%2-j%2;
                  if (rez==0) rez=i-j;
                  return rez;
        }
    } );
```

Primer: razred `TestPriorityQueue.java`

Drevesa (Trees) 93

Drevo sestavlja množica elementov – ***vozlišč***. Vozlišča so povezana na podlagi relacije, ki določa hierarhično strukturo.

Primer: kazalo v knjigi (relacija: vsebovanost) ...

Formalna definicija:

Drevesna struktura z osnovnim tipom T je:

- **prazna struktura** ali
- **vozlišče tipa T, kateremu je prirejeno končno število tujih drevesnih struktur z osnovnim tipom T (*poddreves*)**

Ponazoritev dreves: graf

Primer grafa za odločitveno drevo:

- pet zlatnikov, eden je ponaredek (je lažji)
- iščemo ga s tehtanjem

Drevesa (Trees) 94

Pojmi, ki so povezani z drevesi:

- koren, vozlišče
- poddrevo, naddrevo
- naslednik, prednik
- neposredni naslednik (sin), neposredni predhodnik (oče)
- list (končni element), notranji element
- nivo, globina vozlišča
- višina drevesa
- stopnja vozlišča, stopnja drevesa (dvojiška, trojiška drevesa)
- pot do vozlišča (od korena)
- dolžina poti vozlišča, dolžina poti drevesa
- **Lastnosti (vrste) dreves:** - polnost
  - uravnoteženost
  - urejenost

Lastnosti dreves - POLNOST:

- Za **izpolnjeno (complete) drevo** velja, da so stopnje vseh njegovih notranjih vozlišč enake (stopnji drevesa)
  - ugodno za prepis v tabelo, ker ni praznih delov tabele
- **Polno (full) drevo PD** je izpolnjeno drevo, ki ima vse liste na istem nivoju
  - PD (stopnja `d`, višina `h`) ima število vozlišč `n:` $(d^{h+1}-1)/(d-1)$
  - PD (stopnja `d`, število vozlišč `n`) ima višino `h:` $\log_d(n*d-n+1)-1$

Zgled: **polno dvojiško drevo (d=2)** in **polno trojiško drevo (d=3)** :

- h=1: n= $(2^{1+1}-1)/(2-1)$=**3** | h=1: n=$(3^{1+1}-1)/(3-1)$ = **4**
- h=2: n = **7** | h=2: n = **13**
- h=10: n = **2047** | h=10: n = **88573**
- n=1000: h = $\log_2(1001)-1$= 9.97-1 = **9** | n=1000: h=$\log_3(2001)-1$= 6.92-1 = **6**
- 10^6: h=19.93-1 = **19** | h=10^6: h=13.21-1 = **13**

Vrste dreves - URAVNOTEŽENOST:

- Za **uravnoteženo drevo** velja, da se pri vsakem vozlišču višina vseh njegovih poddreves čimbolj enaka.
  - posredno zagotavlja majhno višino drevesa (približna polnost)
  - uravnoteženost zagotovimo z ustreznimi operacijami za vstavljanje in izločanje elementov
- **Popolnoma izravnano drevo** je dvojiško uravnoteženo drevo - pri vsakem vozlišču se število vozlišč njegovega levega in desnega poddrevesa razlikuje kvečjemu za 1
  - **Rekurzivna gradnja:** prvi element postane koren
    - iz prve polovice elementov (od drugega do srednjega elementa) rekurzivno zgradimo levo poddrevo
    - preostanek (od srednjega elementa do konca) rekurzivno zgradimo desno poddrevo

Zgled: dvojiško popolnoma izravnano drevo z 1, 2, 3, 4, 5, 6, 7, 8,...,15 vozlišči

- **AVL drevo** je primer uravnoteženega dvojiškega iskalnega drevesa

Vrste dreves - UREJENOST:

- Dodajanje in izločanje pri iskalnih drevesih upošteva urejenost in zagotavlja tako sestavo dreves, ki omogočajo učinkovito iskanje elementov
  - vloga vozlišč – usmerjanje pri iskanju
- Tipičen primer: **Dvojiško iskalno drevo (Binary Search Tree - BST)**
  - Za vsako vozlišče **iskalnega drevesa** velja:
    - (če je x vrednost trenutnega vozlišča)
    - vrednost vsakega vozlišča v levem poddrevesu je manjša od x
    - vrednost vsakega vozlišča v desnem poddrevesu je večja od x
    - (vsaka vrednost se v drevesu nahaja samo v enem vozlišču)
  - Najboljši, najslabši, povprečen primer

Zgled: postopna izgradnja BST za vozlišča: **{ 5, 2, 7, 1, 3, 4, 10, 8, 6, 5 }**
postopno brisanje vozlišč z vrednostmi: {**9, 5, 8, 3, 5, 4** }

- Drugi primeri: **Iskalna drevesa višjih stopenj (B drevesa)**
  AVL drevesa

Obhodi dreves:

- postopek, ki sistematično obišče vsa vozlišča drevesa (oziroma izvede določeno operacijo na vseh vozliščih – tipično izpis)
- **Poznamo naslednje obhode:**
  - Obhod po nivojih (ang. level order)
  - Premi vrsti red (ang. preorder)
  - Vmesni vrstni red (ang. inorder)
  - Obratni vrstni red (ang. postorder)

Zgled: vsi obhodi dvojiškega drevesa za aritmetični izraz

Postopki za vse vrste obhodov ...

Tipična ponazoritev dreves v računalniku:

- s pomočjo tabele, kjer vsak element tabele predstavlja eno vozlišče (kot pri sortiranju s kopico)
- z (ustrezno) povezanimi objekti, ki predstavljajo posamezna vozlišča

Prikaz: ponazoritev drevesa s tabelo:

- primer: neizpolnjeno dvojiško drevo
- vrednosti v vozliščih so znaki
- preslikovalna funkcija: levi sin i = 2*i+1, desni sin i = 2*i+2

Prikaz: ponazoritev s povezanimi objekti

- primer: isto drevo
- vsako vozlišče ima povezavo na (prednika,) levega in desnega sina

Izvedba: razreda `BinaryTree()` in `TestBinaryTree()`

Primer: `BinaryTree.java, TestBinaryTree.java`

Lastnosti dreves - POLNOST:

- Za **izpolnjeno (complete) drevo** velja, da so stopnje vseh njegovih notranjih vozlišč enake (stopnji drevesa)
  - ugodno za prepis v tabelo, ker ni praznih delov tabele
- **Polno (full) drevo PD** je izpolnjeno drevo, ki ima vse liste na istem nivoju
  - PD (stopnja `d`, višina `h`) ima število vozlišč `n:` $(d^{h+1}-1)/(d-1)$
  - PD (stopnja `d`, število vozlišč `n`) ima višino `h:` $\log_d(n*d-n+1)-1$

Zgled: **polno dvojiško drevo (d=2)** in **polno trojiško drevo (d=3)** :

- h=1: n= $(2^{1+1}-1)/(2-1)$=**3** | h=1: n=$(3^{1+1}-1)/(3-1)$ = **4**
- h=2: n = **7** | h=2: n = **13**
- h=10: n = **2047** | h=10: n = **88573**
- n=1000: h = $\log_2(1001)-1$= 9.97-1 = **9** | n=1000: h=$\log_3(2001)-1$= 6.92-1 = **6**
- 10^6: h=19.93-1 = **19** | h=10^6: h=13.21-1 = **13**

Vrste dreves - URAVNOTEŽENOST:

- Za **uravnoteženo drevo** velja, da se pri vsakem vozlišču višina vseh njegovih poddreves čimbolj enaka.
  - posredno zagotavlja majhno višino drevesa (približna polnost)
  - uravnoteženost zagotovimo z ustreznimi operacijami za vstavljanje in izločanje elementov
- **Popolnoma izravnano drevo** je dvojiško uravnoteženo drevo - pri vsakem vozlišču se število vozlišč njegovega levega in desnega poddrevesa razlikuje kvečjemu za 1
  - **Rekurzivna gradnja:** prvi element postane koren
    - iz prve polovice elementov (od drugega do srednjega elementa) rekurzivno zgradimo levo poddrevo
    - preostanek (od srednjega elementa do konca) rekurzivno zgradimo desno poddrevo

Zgled: dvojiško popolnoma izravnano drevo z 1, 2, 3, 4, 5, 6, 7, 8,...,15 vozlišči

- **AVL drevo** je primer uravnoteženega dvojiškega iskalnega drevesa

Vrste dreves - UREJENOST:

- Dodajanje in izločanje pri iskalnih drevesih upošteva urejenost in zagotavlja tako sestavo dreves, ki omogočajo učinkovito iskanje elementov
  - vloga vozlišč – usmerjanje pri iskanju
- Tipičen primer: **Dvojiško iskalno drevo (Binary Search Tree - BST)**
  - Za vsako vozlišče **iskalnega drevesa** velja:
    - (če je x vrednost trenutnega vozlišča)
    - vrednost vsakega vozlišča v levem poddrevesu je manjša od x
    - vrednost vsakega vozlišča v desnem poddrevesu je večja od x
    - (vsaka vrednost se v drevesu nahaja samo v enem vozlišču)
  - Najboljši, najslabši, povprečen primer

Zgled: postopna izgradnja BST za vozlišča: **{ 5, 2, 7, 1, 3, 4, 10, 8, 6, 5 }**
postopno brisanje vozlišč z vrednostmi: {**9, 5, 8, 3, 5, 4** }

- Drugi primeri: **Iskalna drevesa višjih stopenj (B drevesa)**
  AVL drevesa

Obhodi dreves:

- postopek, ki sistematično obišče vsa vozlišča drevesa (oziroma izvede določeno operacijo na vseh vozliščih – tipično izpis)
- **Poznamo naslednje obhode:**
  - Obhod po nivojih (ang. level order)
  - Premi vrsti red (ang. preorder)
  - Vmesni vrstni red (ang. inorder)
  - Obratni vrstni red (ang. postorder)

Zgled: vsi obhodi dvojiškega drevesa za aritmetični izraz

Postopki za vse vrste obhodov ...

Tipična ponazoritev dreves v računalniku:

- s pomočjo tabele, kjer vsak element tabele predstavlja eno vozlišče (kot pri sortiranju s kopico)
- z (ustrezno) povezanimi objekti, ki predstavljajo posamezna vozlišča

Prikaz: ponazoritev drevesa s tabelo:

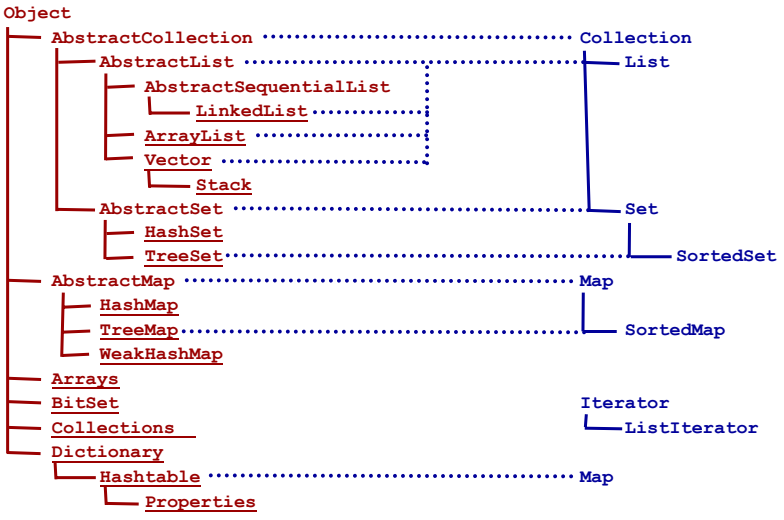
- primer: neizpolnjeno dvojiško drevo
- vrednosti v vozliščih so znaki
- preslikovalna funkcija: levi sin i = 2*i+1, desni sin i = 2*i+2

Prikaz: ponazoritev s povezanimi objekti

- primer: isto drevo
- vsako vozlišče ima povezavo na (prednika,) levega in desnega sina

Izvedba: razreda `BinaryTree()` in `TestBinaryTree()`

Primer: `BinaryTree.java, TestBinaryTree.java`



Iskalna tabela (Map) 101

Predstavlja zaporedje parov <ključ,vrednost>, ki omogoča učinkovito iskanje vrednosti po ključih

- angleški izrazi: map, lookup table, associative array, dictionary

Ključi: enolični, lahko so poljubnega tipa - ne nujno celoštevilski,

Vrednosti: vsebujejo podatke

Primer: slovar, legenda, kazalo, seznam lokacij ...

`Collection Framework` vsebuje:

- generični vmesnika **`Map`** in **`SortedMap`**
- generični abstraktni razred **`AbstractMap`**
- več realiziranih generičnih razredov: **`HashMap, TreeMap, WeakHashMap`**

Iskalna tabela (Map) 102

```
public interface Map
{  int size();
   boolean isEmpty();
   boolean containsKey(Object k);
   boolean containsValue(Object v);
   Object get(Object k);
   Object put (Object k, Object v);
   Object remove(Object k);
   putAll(Map);
   clear();
   Set keySet();
   Set entrySet();
```

```
    ...
    Collection values();
    public interface Entry
    {  Object getKey();
       Object getvalue();
       Object setValue(Object v);
       boolean equals(Object o);
       int hashCode();
    }
    boolean equals(Object o);
    int hashCode();
}
```


Razred `HashMap`

- realizacija z (zaprto) razpršeno tabelo (hash table)
- v ozadju tabela določene kapacitete
- razporejanje s pomočjo razpršilne funkcije:
  - zagotavlja hitro vstavljanje in iskanje
  - problem kolizij:
    - linerno ponovno razprševanje
    - kvadratično ponovno razprševanje
    - odprte razpršene tabele
  - zagotavljanje primerne kapacitete in zasedenosti

Prikaz razprševanja: slovensko – angleški slovar

Primer: `Slovar.java`

Razred `TreeMap`

- realizacija z dvojiškim iskalnim drevesom
- v ozadju ustrezno povezano drevo objektov
- upošteva urejenost po ključih – izpis

Primer uporabe: kazalo

Razred `WeakHashMap`

- realizacija s šibkimi razpršenimi tabelami
- ko ključ (ali vrednost) ni več dosegljiv, se element odstani iz tabele

Primer uporabe: seznam datotek na disku ...

KONEC

Veliko uspeha pri učenju!