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 (5.10.2006)

Študijsko leto 2006/07

Kratka ponovitev Jave

Objektno usmerjeno programiranje

- A PIE:

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

- Ključni pojmi:

- razred : objekt
- metoda : atribut

Programski jezik Java

- Java 2 Platform, Standard Edition (J2SE), version (1.)5.0
- J2SE, version (1.)6 Beta 2 – še ne uporabljamo
- 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

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 ...

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

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 : $O(n^2)$

Sortiranje tabel – Navadno izbiranje 15

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 = $O(n * \ln n)$

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

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, ...

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:** $T_a = O(n*\log_2 n)$, vendar $T_w = O(n^2)$
- **Iskanje najslabše permutacije ...**

Sortiranje tabel – Quicksort 29

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 30

Ralizacija metode v Javi: metoda QuicksortI1 ...

Problem velikosti sklada

Ralizacija metode v Javi: metoda QuicksortI2 ...

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

Leksikografsko sortiranje 31

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 32

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)
- dejansko le redko boljše od quicksorta, ker ne upoštevamo dodatnega dela

Vmesnik (ang. interface) 33

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) 34

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) 35

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) 36

Namen: zbirka objektov tipa `Object` ali drugega iz `Object` izpeljanega tipa.

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

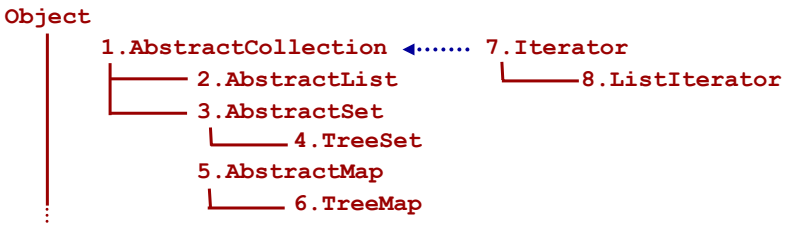
Realizacija v javi: Collections Framework – zbirka vmesnikov in razredov

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

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

Zbirka (Collection) 37

Obstajajo razredi, ki implementirajo te vmesnike:



Celoten diagram razmerij med vmesniki in razredi ...

V nadaljevanju:

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

Zbirka (Collection) 38

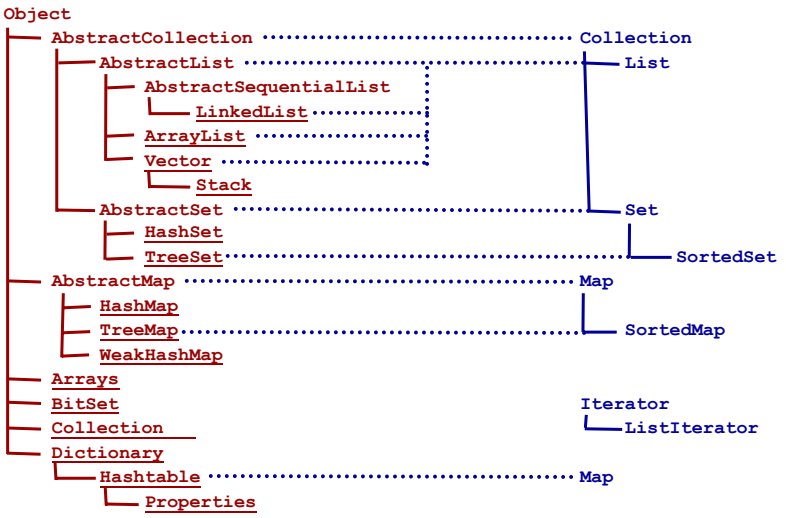


Tabela (razred `Array`) 39

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 `Array`) 40

- 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 `Array`) 41

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

Vektor (razred `Vector`) 42

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

Razlike v primerjavi s tabelo:

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

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

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

Kreiranje vektorjev - 4 konstruktorji:

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

Vektor (razred `Vector`) 43

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`) 44

Shranjevanje objektov v vektor:

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

Branje podatkov iz vektorja:

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

Vektor (razred `Vector`) 45

Potrebna je konverzija tipa

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`) 46

Prepis elementov iz vektorja v tabelo

- uporaba vektorja zahteva dodatno režijo, zato včasih smiselno

Metodi:

- `Object[] toArray();`
- `Object[] toArray(Object[]);` // pozor – eksplicitna pretvorba tipa

Odstranjevanje elementov iz vektorja:

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

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

Vektor (razred `Vector`) 47

Iskanje elementov v vektorju:

- `int indexOf(Object);`
- `int indexOf(Object, int);`
- `int lastIndexOf(Object);`
- `int lastIndexOf(Object, 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;
   }
}
```

Iterator

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

Razred **`Vector`** implementira vmesnik **`Collection`**.

`Iterator` ima konstruktor in 3 metode:

- `Iterator();`
- `Object next();`
- `boolean hasNext();`
- `remove();`

Primer: Obhod s pomočjo iteratorja:

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

Primer: Vektor.java

Sklad (razred `Stack`) 50

Podatkovna struktura, ki deluje po principu LIFO

Primer: zlaganje kovancev ...

Razred **`Stack`** (razširitev razreda **`Vector`**)

Operacije za delo s skladom:

- **`empty()`**
- **`peek()`**
- **`pop()`**
- **`push (Object)`**
- **`search (Object)`**

Deklaracija razreda `Stack` ...

Sklad (razred `Stack`) 51

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**

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

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`) 54

- **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:

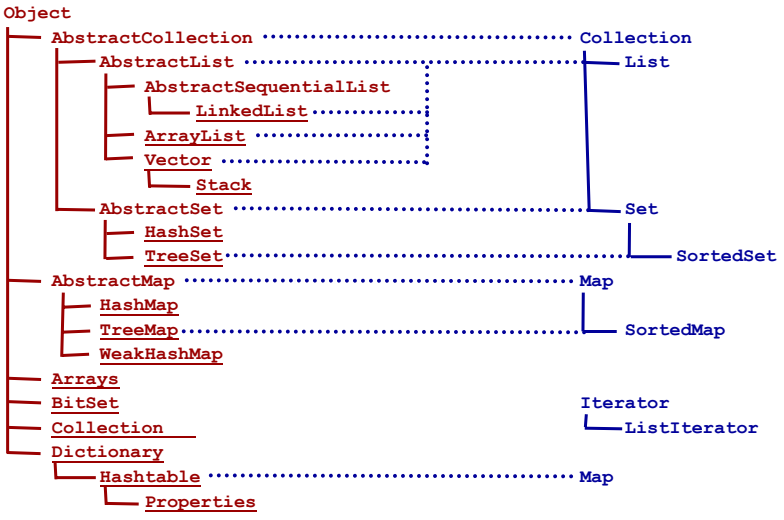
  n – število obročev, ki jih je treba prestaviti

  a – izvor

  b – pomožna palica

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

Collections Framework 55



Zbirka (vmesnik `Collection`) 56

Definiran v paketu `java.util`

Zahteva implementacijo naslednjih metod:

- `public boolean add(Object);`
- `public boolean addAll(Collection);`
- `public void clean();`
- `public boolean contains(Object);`
- `public boolean containsAll(Collection);`
- `public boolean equals(Object);`
- `public int hashCode();`
- `public boolean isEmpty();`

Zbirka (vmesnik `Collection`) 57

- `public Iterator iterator();`
- `public boolean remove(Object);`
- `public boolean removeAll(Collection);`
- `public boolean retainAll(Collection);`
- `public int size();`
- `public Object[] toArray();`
- `public Object[] toArray(Object[]);`

Zbirka (razred `AbstractCollection`) 58

Delna implementacija vmesnika `Collection`

- 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`) 59

Poljubna zbirka objektov, ki lahko vsebuje duplikate

bag = "torba" = multiset = množica z duplikati

Prikaz razširitve razreda **`AbstractCollection`**

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`) 60

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`) 61

- `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) 62

```
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) 63

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`

Seznam (`List`) 64

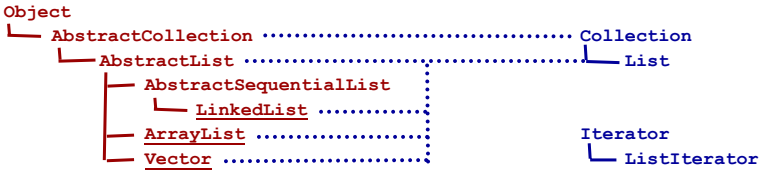
Podatkovna struktura predstavlja splošno zaporedje objektov:

- pojem pozicije
- dodajanje in izločanje objektov v predvidenem času

Primer: kompozicija vagonov ...

`Collection Framework` vsebuje:

- vmesnik **`List`**
- dva abstraktna razreda: **`AbstractList`** in **`AbstractSequentialList`**
- več realiziranih razredov: **`LinkedList`**, **`ArrayList`**, **`Vector`**
- iterator **`ListIterator`**



Seznam (vmesnik `List`) 65

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

Seznam (vmesnik `List`) 66

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


Seznam (abstraktni razred `AbstractList`) 67

Dve realizaciji:

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

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

Seznam (abstraktni razred `AbstractList`) 68

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


Seznam (abs.razred `AbstractSequentialList`) 69

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

Seznam (razred `ListIterator`) 70

Dvosmerni iterator

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


Seznam (razred `ArrayList`) 71

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

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

Seznam (razred `LinkedList`) 72

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

Primer: Seznam.java

Vrsta (`Queue`) 73

Podatkovna struktura, ki deluje po principu FIFO

Primer: vrsta pred blagajno v kinu ...

`Collection Framework` ne vsebuje posebnega vmesnika ali razreda za vrsto

Zastavimo svojo hierarhijo:

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

Vrsta (vmesnik `Queue` in razred `AbstractQueue`) 74

- 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`

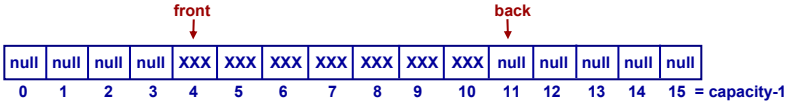
Deklaracija razreda `AbstractCollection`

Vrsta (razreda `ArrayQueue`) 75

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`

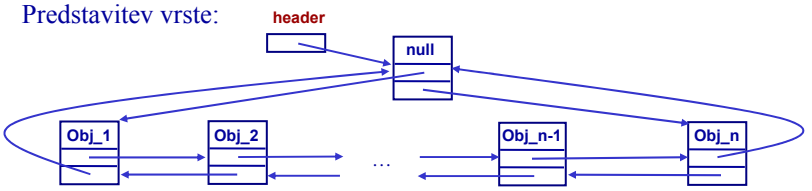
Vrsta (razred `LinkedQueue`) 76

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`

Generični razredi in vmesniki (Generics) 77

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: List<String> s = new LinkedList<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 **`<T>`**
- 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

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

Generični razredi in vmesniki (Generics) 78

Uporaba generičnega razreda brez določanja tipa:

```
List seznam = new ArrayList();

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

Izogibanje opozorilom:

```
List<?> seznam = new ArrayList<?>();
```

Uporaba pri argumentih metod:

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

Uporaba pri iteratorju:

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


Generični razredi in vmesniki (Generics) 79

Pozor pri avtomatski pretvorbi tipov:

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

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

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

Pisanje generičnih razredov:

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

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

Uporaba pri dedovanju:

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

Primer: QueueG.java

AbstractQueueG.java

ArrayQueueG.java

TestArrayQueueG.java

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 82

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 83

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));
```

`Queue`, `PriorityQueue` in `Comparator`:

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

```
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;
        }
    } );
for (int i=0; i<10; i++) {
    pv1.offer(10-i);
    pv2.offer(10-i);
}
```


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

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:

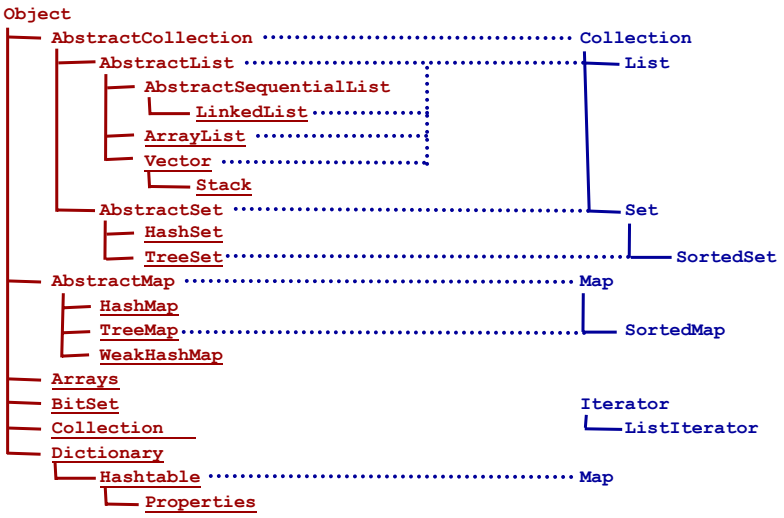
- primer: neizpolnjeno dvojiško drevo
- vrednosti v vozliščih so znaki
- preslikovalna funkcija: levi sin i = 2*i+1, desni sin i = 2*i+2

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`



Iskalna tabela (Map) 93

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) 94

```
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();
```


Iskalna tabela (Map) 95

```
   ...
   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();
}
```

Iskalna tabela (Map) 96

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`

Iskalna tabela (Map) 97

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 ...