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 8. PREDAVANJA (23.11.2006)

Študijsko leto 2006/07

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

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

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)

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

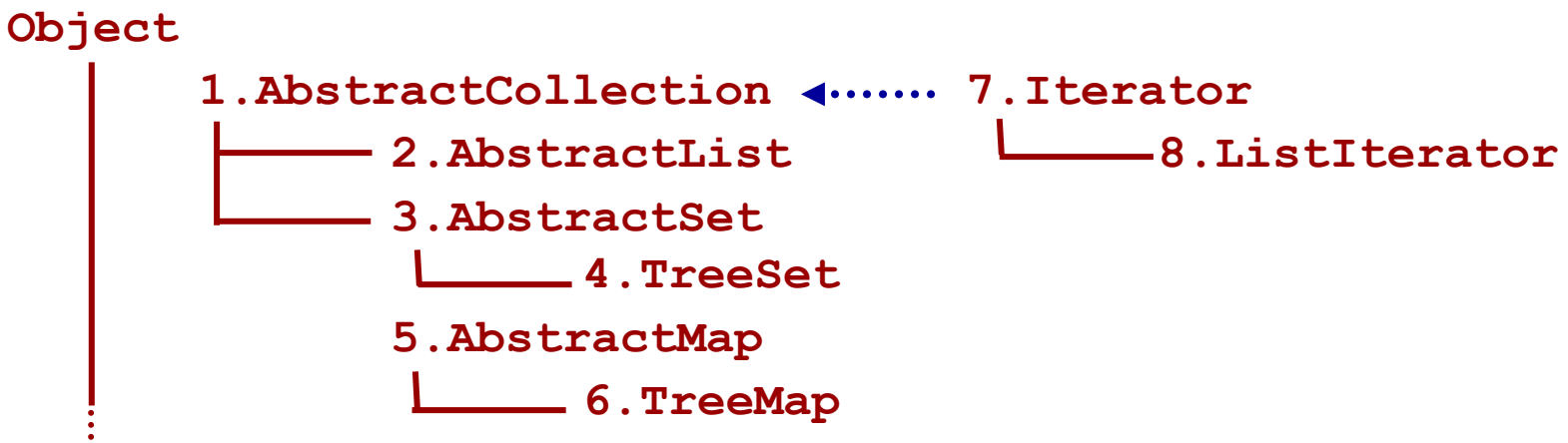
Primer: sklad (LIFO), seznam, vrsta (FIFO), množica, ...

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 zbirk

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)

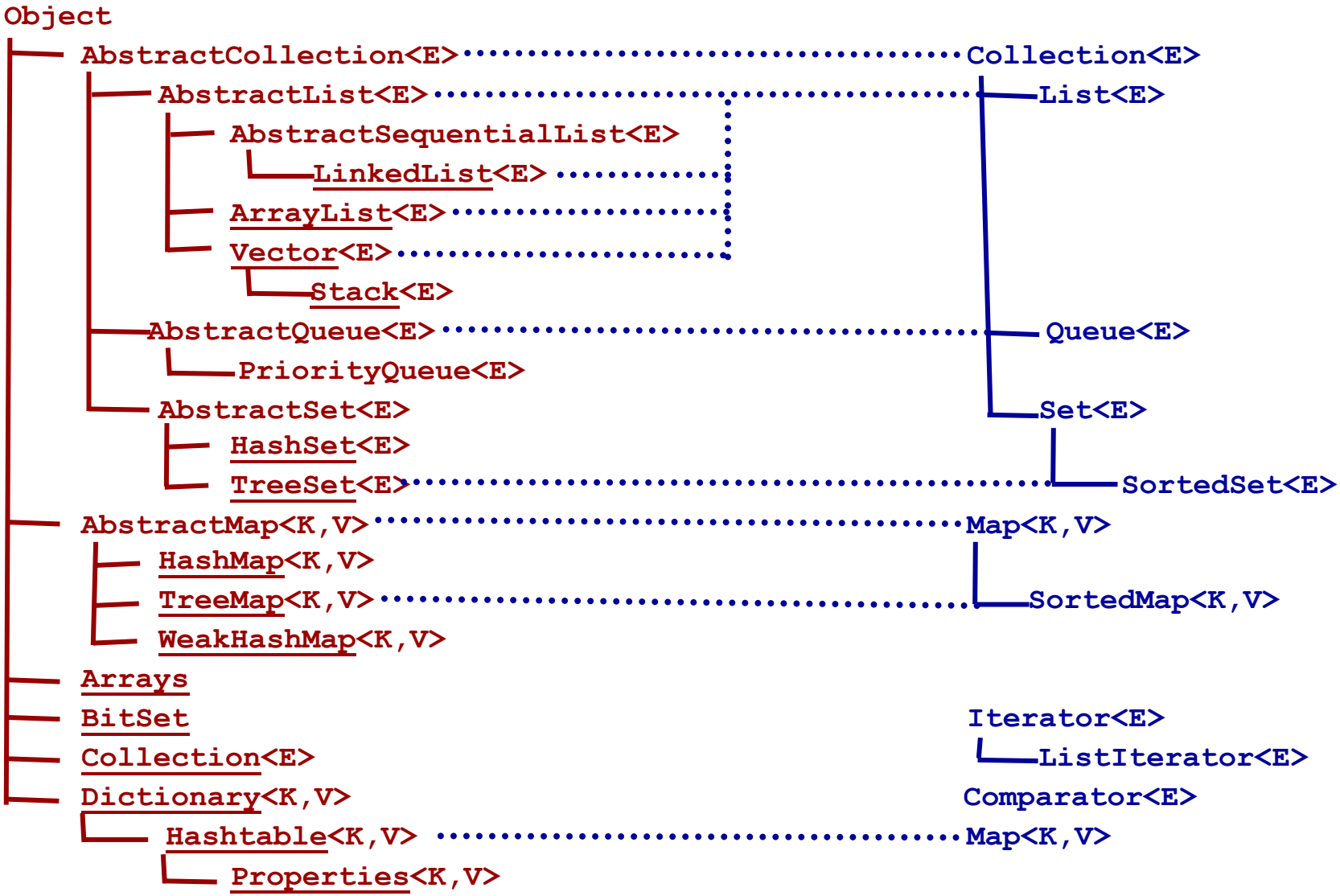


Tabela (razred `Array`)

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`)

- 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`)

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 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!!!`

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

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();
    }
    ...
}
```

Vektor (razred `Vector<E>`)

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)`

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();`