Masyvai. Objektų talpyklos

(Arrays, collections)

Masyvai Javoje

- Dinamiškai sukuriami java objektai iš anksto apibrėžtam komponenčių skaičiui saugoti.
- Komponentės g.b. primityvaus tipo arba nuorodos tipo (atskiru atveju - interfeisai arba abstrakčiosios klasės).

Masyvo tipas

- Jeigu T - tipas, T[] nuorodos į masyvą tipas.
- Jeigu a - tipo T[] nuoroda, tai T[i] - i-ji komponentė (indeksuojama nuo 0).
- Masyvo ilgis nurodomas sukūrimo metu: new T[n] ir pasiekiamas kaip laukas final public length, n ≥ 0;
- *ArrayIndexOutOfBoundsException*

Deklaravimo pavyzdžiai

```
int[] ai;          // array of int
short[][] as;      // array of array of short

Object[]  ao,  // array of Object
          otherAo;  // array of Object

short  s,    // scalar short
       aas[][];  // array of array of short (C)
```


Deklaracija ir sukūrimas

```
Exception ae[] = new Exception[3]; // null values
Object aao[][] = new Exception[2][3];
int[] factorial = { 1, 1, 2, 6, 24, 120, 720, 5040 };
char ac[] = { 'n', 'o', 't', ' ', 'a', ' ',
              'S', 't', 'r', 'i', 'n', 'g' };
String[] aas = { "array", "of", "String" };
```

Pastaba: objektai – masyvo elementai automatiškai nesukuriami, bet gali būti pateikti inicializaciniu sąrašu

Masyvo superklasė - *Object*

- Masyvai yra klonuojami (negiliai), serializuojami.
- Tipams: T := TT galioja T[] := TT[]. Tačiau priskyrimo metu galimas *ArrayStoreException*:

```
Object x[] = new String[3]; // OK
 x[0] = new Integer(0); //---> Exception
```

Utilitos darbui su masyvais

- **System.arraycopy(...)** – efektyvus kopijavimas
- Klasė **java.util.Arrays** apibrėžia stat. manipuliavimo masyvais metodus, pvz., paieškai, rūšiavimui.

```
import java.util.Arrays;

public class ArraySort{
    public static void main(String[] args) {
      Arrays.sort(args);
      for (int i = 0; i < args.length; i++)
        System.out.println(args[i]);
    } }
```

7

Collection – talpykla -

- Objektas – kolekcija / aibė / sąrašas..., užtikrinantis objektų saugojimą, paiešką,...
- Prisiminkime duomenų struktūras ...
- Java *Collection* klasių šeima pateikia standartinių duom. struktūrų bei algoritmų realizacijas
- Operuojama objektais, bet ne primityviaisiais tipais

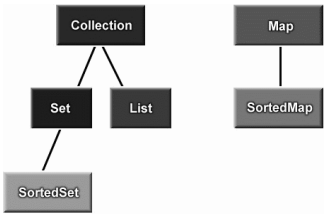
8

Vystymasis

- Javoje 1.x ribotos priemonės – *Vector, Hashtable, Enumeration*
- *1.2* – *Collection* klasės, vieningas požiūris
- 1.5 – Parametrizacija, papildomos klasės (*Queue*), primit. tipų – Apvalkalų automatinė konversija (*Integer* <-> *int*)

9

Interfeisai



– Apibrėžti **java.util** pakete

– Yra realizuojančiosios klasės

10

Ypatybės

- ***Collection*** – objektų elementų rinkinys
- ***Set*** – rinkinys be pasikartojančių elementų
- ***List*** – elementų seka
- ***Map*** – porų <raktas, reikšmė> aibė
- ***SortedSet***
- ***SortedMap***
- Pastaba: Objektai rikiuojami realizuojant *Comparable, Comparator* interfeisus.

11

Collection interfeisas

- Nusako bendriausias operacijas
- Peržiūra iteratoriumi

12

```
public interface Collection {
    // ------ Basic Operations
    int size();
    boolean isEmpty();
    boolean contains(Object element);
    boolean add(Object element);    // Optional
    boolean remove(Object element); // Optional
    Iterator iterator();
    // ------- Bulk Operations
    boolean containsAll(Collection c);
    boolean addAll(Collection c);    // Optional
    boolean removeAll(Collection c); // Optional
    boolean retainAll(Collection c); // Optional
    void clear();                    // Optional
    // ------- Array Operations
    Object[] toArray();
    Object[] toArray(Object a[]);
}
```

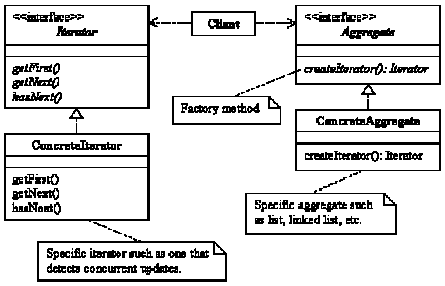


```
public interface Iterator {
    boolean hasNext();
    Object next();
    void remove();    // Optional
}

//================================//

static void filter(Collection c) {
    for (Iterator i = c.iterator(); i.hasNext(); )
        if (!cond(i.next()))
            i.remove();
}
```


Iterator'iaus šablonas



Šaltinis: http://www.bit.umkc.edu/burris/pl/design-patterns/

Set interfeisas

- Neturi naujų metodų.
- Nauja semantika: pasikartojantys elementai draudžiami.


```
import java.util.*;
public class FindDups {
    public static void main(String args[]) {
        Set s = new HashSet();
        for (int i=0; i<args.length; i++)
            if (!s.add(args[i]))
                System.out.println("Duplicate detected: "+args[i]);
        System.out.println(s.size()+" distinct words detected: "+s)
;
    }
}
// Now let's run the program:
// % java FindDups i came i saw i left
//
// Duplicate detected: i
// Duplicate detected: i
// 4 distinct words detected: [came, left, saw, i]
```


List - sąrašas

- Nuosekli elementų seka
- Kreiptis pagal indeksą
- Paieška gražina elemento poziciją
- Išplėstas iteratorius
- Posarašiai (*Range-view*)
- Realizuoja *ArrayList, LinkedList*
- *Collections* klasė įgalina pritaikyti įvairius algoritmus *List* egzemplioriams

```
public interface List extends Collection {
   //----- Positional Access
   Object get(int index);
   Object set(int index, Object element);      // Optional
   void add(int index, Object element);        // Optional
   Object remove(int index);                   // Optional
   abstract boolean addAll(int index, Collection c); // Opt
   //----- Search
   int indexOf(Object o);
   int lastIndexOf(Object o);
   //----- Iteration
   ListIterator listIterator();
   ListIterator listIterator(int index);
   // Range-view
   List subList(int from, int to);
}
```

19

List'o panaudojimas

```
private static void swap(List a, int i, int j) {
   Object tmp = a.get(i);
   a.set(i, a.get(j));
   a.set(j, tmp);
}
```

20

```
public interface ListIterator extends Iterator {
   boolean hasNext();
   Object next();

   boolean hasPrevious();
   Object previous();

   int nextIndex();
   int previousIndex();

   void remove();          // Optional
   void set(Object o);     // Optional
   void add(Object o);     // Optional
}
```

21

Map

- Susieja raktą su reikšme
- Įgyvendina: *HashMap, TreeMap*

22

```
public interface Map {

   // Basic Operations
   Object put(Object key, Object value);
   Object get(Object key);
   Object remove(Object key);
   boolean containsKey(Object key);
   boolean containsValue(Object value);
   int size();
   boolean isEmpty();

   // Bulk Operations
   void putAll(Map t);
   void clear();
```

23

```
   // Collection Views
      public Set keySet();
      public Collection values();
      public Set entrySet();

      // Interface for entrySet elements
      public interface Entry {
         Object getKey();
         Object getValue();
         Object setValue(Object value);
      }
   } // End Map
```

24

```
import java.util.*;
public class Freq {
   private static final Integer ONE = new Integer(1);
   public static void main(String args[]) {
      Map m = new HashMap();
      // Initialize frequency table from command line
      for (int i=0; i<args.length; i++) {
         Integer freq = (Integer) m.get(args[i]);
         m.put(args[i], (freq==null ? ONE :
                  new Integer(freq.intValue() + 1)));
      }
      System.out.println(m.size()+" distinct words detected:");
      System.out.println(m);
   }}
```

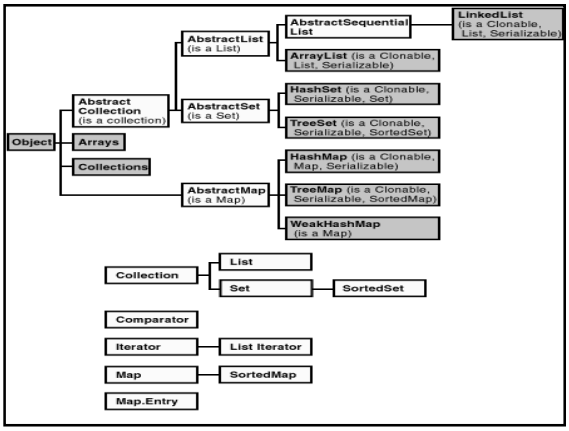
```
// % java Freq if it is to be it is up to me to delegate
// 8 distinct words detected:
// {to=3, me=1, delegate=1, it=2, is=2, if=1, be=1, up=1}
```

25

Realizacijos

		Implementations			
		Hash Table	Resizable Array	Balanced Tree	Linked List
Interfaces	Set	HashSet		TreeSet	
	List		ArrayList		LinkedList
	Map	HashMap		TreeMap	

26



Pabaiga

28