


Pemrograman Berbasis Objek

Collections

Politeknik Elektronika Negeri Surabaya




Pemrograman Berbasis Objek

Collections Framework

- Dikenalkan pada Java 2 SDK.
- Collection sudah ada sejak JDK 1.0
 - Hashtable
 - Vector

Politeknik Elektronika Negeri Surabaya




Pemrograman Berbasis Objek

Collections

- Collection adalah suatu obyek yang bisa digunakan untuk menyimpan sekumpulan obyek
- Obyek yang ada dalam Collection disebut elemen
- Collection menyimpan elemen yang bertipe Object, sehingga berbagai tipe obyek bisa disimpan dalam Collection

Politeknik Elektronika Negeri Surabaya

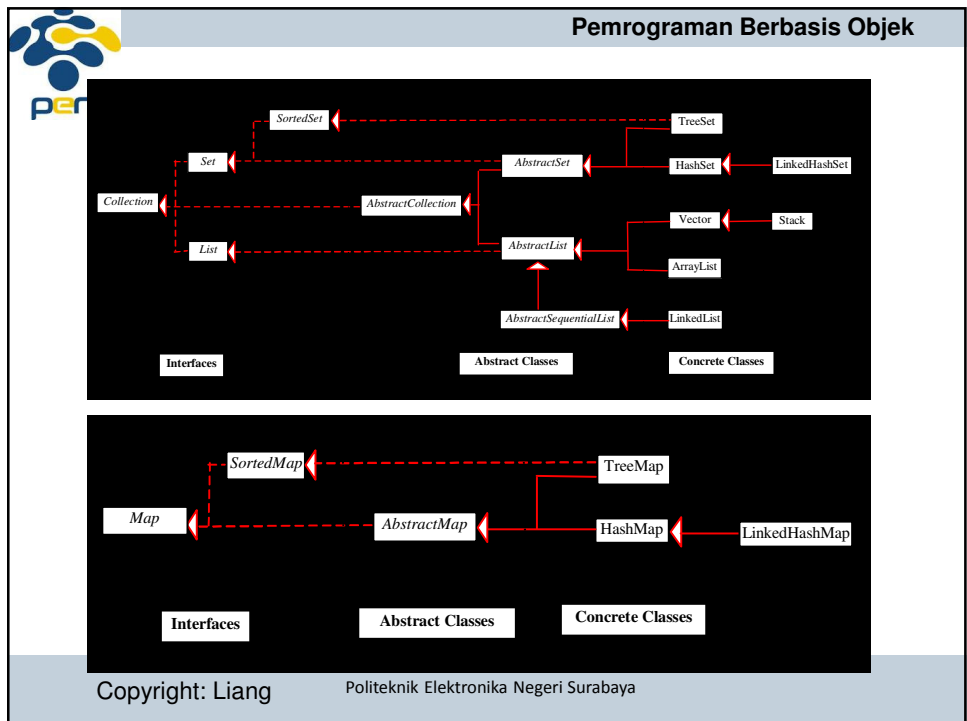
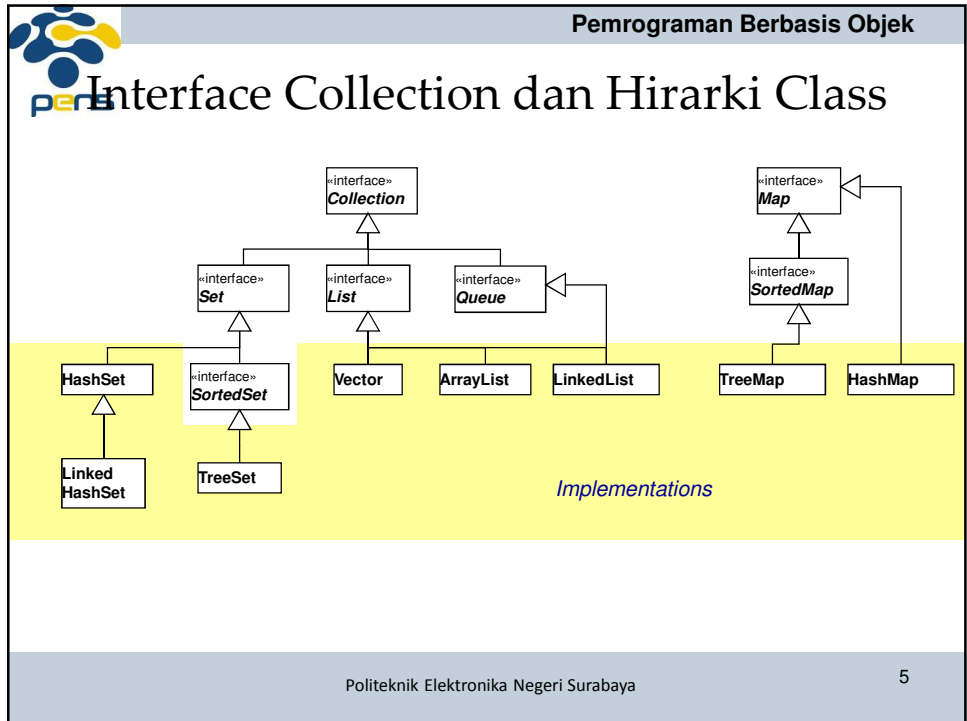


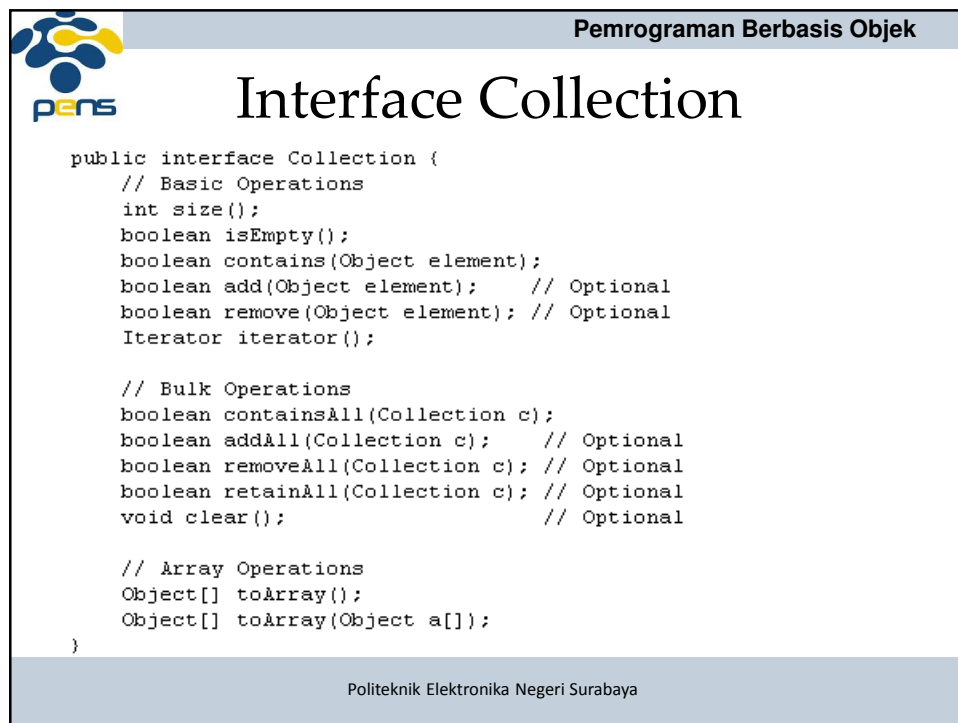
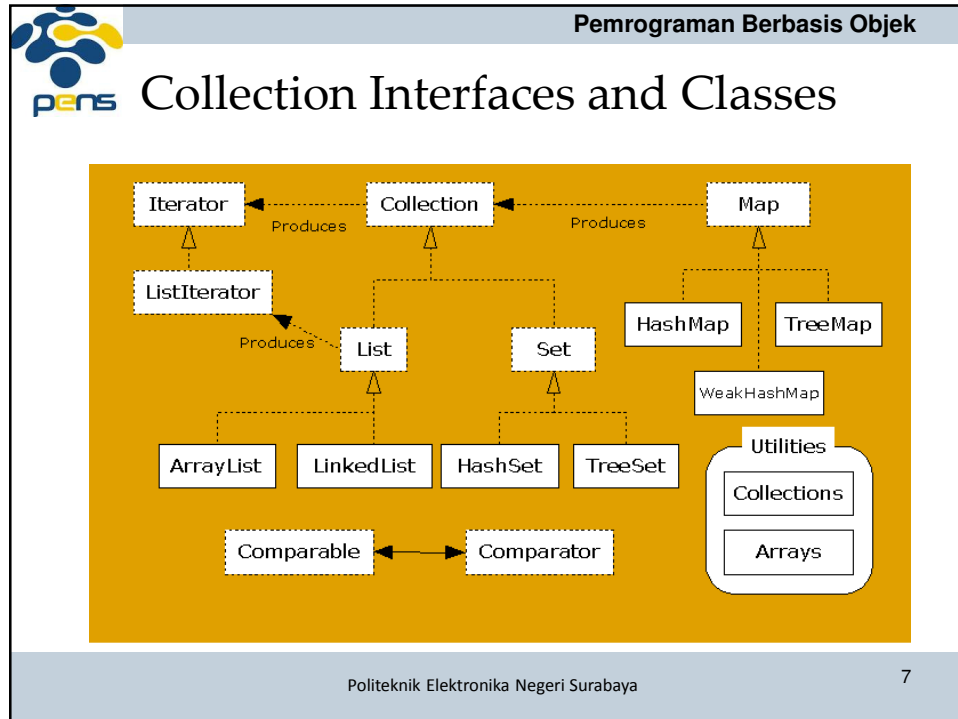
Pemrograman Berbasis Objek

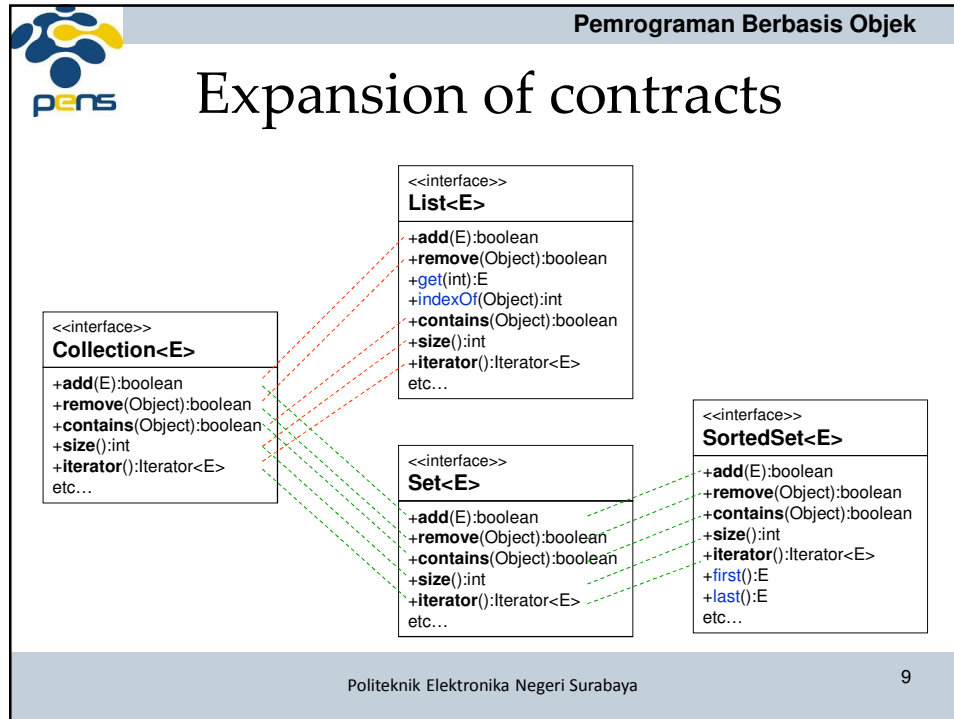
The Java Collections API

- Java Collections API terdiri dari interface:
 - **Collection** : sekumpulan obyek yang tidak mempunyai posisi yang tetap (no particular order) dan menerima duplikat.
 - **List** : sekumpulan obyek yang berdasarkan urutan (ordered) dan menerima duplikat.
 - **Set** : sekumpulan obyek yang tidak berdasarkan urutan (unordered) dan menolak duplikat.
 - **Map** : mendukung pencarian berdasarkan key, key ini harus unik. Has no particular order.

Politeknik Elektronika Negeri Surabaya








Pemrograman Berbasis Objek

Set : HashSet

- Elemen pada Set selalu unik
- Set menolak duplikat
- Elemen yang tersimpan tidak berdasarkan urutan(unorder) dan tidak di sorting (unsorted)
- Berhubungan dengan definisi matematika mengenai himpunan.
- Interface Set merupakan subinterface dari interface Collection
- Class yang **mengimplementasikan** interface Set adalah HashSet

Politeknik Elektronika Negeri Surabaya


Pemrograman Berbasis Objek

```


public interface Set {
    // 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[]);
}

```

Politeknik Elektronika Negeri Surabaya


Pemrograman Berbasis Objek

Set: HashSet

```


import java.util.*;

public class SetExample {
    public static void main(String[] args) {
        Set set = new HashSet();
        set.add("one");
        set.add("second");
        set.add("3rd");
        set.add(new Integer(4));
        set.add(new Float(5.0F));
        set.add("second"); // duplicate, not added
        set.add(new Integer(4)); // duplicate, not added
        System.out.println(set);
    }
}

```

Hasil: `[one, second, 5.0, 3rd, 4]`

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek

Set: HashSet

```
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);
    }
}

% java FindDups i came i saw i left

Duplicate detected: i
Duplicate detected: i
4 distinct words detected: [came, left, saw, i]
```

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek

Set: HashSet

```
import java.util.*;

public class FindDups2 {
    public static void main(String args[]) {
        Set uniques = new HashSet();
        Set dups = new HashSet();

        for (int i=0; i<args.length; i++)
            if (!uniques.add(args[i]))
                dups.add(args[i]);

        uniques.removeAll(dups); // Destructive set-difference

        System.out.println("Unique words: " + uniques);
        System.out.println("Duplicate words: " + dups);
    }
}

% java FindDups2 i came i saw i left

Unique words: [came, left, saw]
Duplicate words: [i]
```

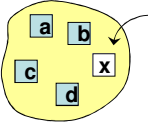
Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek

Set <E>

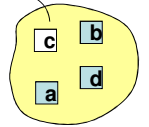
- Mathematical Set abstraction – contains **no duplicate** elements
 - i.e. no two elements e_1 and e_2 such that $e_1.equals(e_2)$

add(x)
→ true



add(b)
→ false

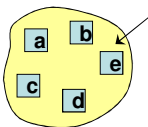
remove(c)
→ true



remove(x)
→ false

```
<<interface>>
Set<E>
+add(E):boolean
+remove(Object):boolean
+contains(Object):boolean
+isEmpty():boolean
+size():int
+iterator():Iterator<E>
etc...
```

contains(e)
→ true



contains(x)
→ false

isEmpty()
→ false

size()
→ 5

```
<<interface>>
SortedSet<E>
+first():E
+last():E
etc...
```

Politeknik Elektronika Negeri Surabaya

15


Pemrograman Berbasis Objek

**Operasi Besar (Bulk operations)
pada Set: HashSet**

Merupakan operasi pada Himpunan

- $s_1.containsAll(s_2)$
mengembalikan nilai true jika s_2 adalah subset s_1 . (Set s_2 adalah subset s_1 apabila s_1 berisi semua anggota s_2)
- $s_1 = s_1 \cup s_2$
 $s_1.addAll(s_2)$
hasil dari s_1 adalah gabungan (union) dari s_1 dan s_2
- $s_1 = s_1 \cap s_2$
 $s_1.retainAll(s_2)$
hasil dari s_1 adalah irisan(intersection) dari s_1 dan s_2 .
- $s_1 = s_1 - s_2$
 $s_1.removeAll(s_2)$
hasil dari s_1 adalah selisih dari s_1 dengan s_2
Selisih ($s_1 - s_2$) adalah set yang berisi semua elemen yang ada pada s_1 tetapi tidak ada pada s_2 .

Politeknik Elektronika Negeri Surabaya


Pemrograman Berbasis Objek

```

import java.util.*;

public class Sets{
    public static void main(String args[]){
        Set s1 = new HashSet();
        Set s2 = new HashSet();

        for(int i=0;i<5;i++){
            s1.add(new Integer(i));

        for (int i=3; i<7; i++)
            s2.add(new Integer(i));
        System.out.println("s1 = " + s1);
        System.out.println("s2 = " + s2);

        System.out.println(s1.containsAll(s2));


        s1.retainAll(s2);
        System.out.println(s1);

        s1.removeAll(s2);
        System.out.println(s1);
    }
}

```

C:\j2sdk1.4.1_01\bin\j
s1 = [2, 4, 1, 3, 0]
s2 = [4, 6, 3, 5]
false
[4, 3]
[]
Finished executing


Politeknik Elektronika Negeri Surabaya


Pemrograman Berbasis Objek

SortedSet:TreeSet

- Aturan sama dengan interface Set → menolak duplikat.
- Ingat → SortedSet adalah subinterface Set.
- Beda: elemen tersimpan dalam urutan ascending → sorted.
- Contoh SortedSet: TreeSet.

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek

SortedSet: TreeSet


```

import java.util.*;
class SortedSetTest{
    public static void main(String [] arg){
        SortedSet set = new TreeSet();
        set.add("Chess");
        set.add("Whist");
        set.add("Checkers");
        set.add("BlackJack");
        set.add("Chess");
        System.out.println(set);
    }
}

```

Output: [BlackJack, Checkers, Chess, Whist]

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek

Class HashSet and TreeSet

- **HashSet and TreeSet mengimplementasikan interface Set.**

HashSet


- Diimplementasikan menggunakan hash table
- Element tidak terurut
- **Method add, remove, and contains, kompleksitas waktu $O(c)$.**

TreeSet

- Diimplementasikan menggunakan struktur pohon.
- Dijamin elemen dalam keadaan terurut.
- **Method add, remove, and contains kompleksitas waktu logarithmic $O(\log(n))$,
dimana n adalah jumlah elemen pada set.**

Politeknik Elektronika Negeri Surabaya

20




Pemrograman Berbasis Objek

List

- Elemen tersimpan berdasarkan urutan masukan (ordered).
- Menerima duplikat.
- Contoh List:
 - LinkedList : elemen dalam LinkedList masuk dari awal dan dihapus dari akhir.
 - Vector : a growable array of object.
 - ArrayList: mirip vector, bersifat unsynchronized (jika multiple threads mengakses object ArrayList, object ini harus synchronized secara eksternal)

Politeknik Elektronika Negeri Surabaya




Pemrograman Berbasis Objek

List

- Pengembangan dari Interface Collection.
 - Pengaksesan elemen melalui indeks seperti array
add (int, Object), get(int), remove(int), set(int, Object)
 - Pencarian element
indexOf(Object), lastIndexOf(Object)
 - Menggunakan Iterator tertentu disebut **ListIterator**
 - Dapat mengambil subList
subList(int fromIndex, int toIndex)

Politeknik Elektronika Negeri Surabaya

22




Pemrograman Berbasis Objek

List

- **add(Object)** : menambahkan elemen diakhir list
- **remove(Object)** : menghapus di awal list
- **list1.equals(list2)** : dikatakan sama dengan memperhatikan urutan elemen

Politeknik Elektronika Negeri Surabaya

23



Pemrograman Berbasis Objek

List

```

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); // Optional


    // Search
    int indexOf(Object o);
    int lastIndexOf(Object o);

    // Iteration
    ListIterator listIterator();
    ListIterator listIterator(int index);

    // Range-view
    List subList(int from, int to);
}

```

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek

Class ArrayList dan LinkedList

- Class **ArrayList** dan **LinkedList** mengimplementasikan interface **List**.
- **ArrayList** adalah sebuah implementasi array dimana elemen-elemennya dapat diakses secara langsung menggunakan **get and set methods**.
Biasanya ArrayList untuk urutan sederhana (simple sequence).
- **LinkedList** menggunakan **double linked list**
- Memberikan performance yang lebih baik untuk **method add dan remove** dibandingkan **ArrayList**.
- Memberikan performance yang lebih jelek untuk method **get and set** dibandingkan **ArrayList**.

Politeknik Elektronika Negeri Surabaya 25



Pemrograman Berbasis Objek

List : ArrayList


```

1  import java.util.*
2
3  public class ListExample {
4      public static void main(String[] args) {
5          List list = new ArrayList();
6          list.add("one");
7          list.add("second");
8          list.add("3rd");
9          list.add(new Integer(4));
10         list.add(new Float(5.0F));
11         list.add("second"); // duplicate, is added
12         list.add(new Integer(4)); // duplicate, is added
13         System.out.println(list);
14     }
15 }

```

[one, second, 3rd, 4, 5.0, second, 4]

Politeknik Elektronika Negeri Surabaya


Pemrograman Berbasis Objek

List: Vector


```

import java.util.*;
class VectorTest{
    public static void main(String [] arg){
        Vector v = new Vector();
        v.add("Zak");
        v.add("Gordon");
        v.add(0, "Duke");
        v.add("Lara");
        v.add("Zak");
        System.out.println(v);
        String name = (String) v.get(2);
        System.out.println(name);
    }
}

```

Output: [Duke, Zak, Gordon, Lara, Zak]
Gordon

Politeknik Elektronika Negeri Surabaya


Pemrograman Berbasis Objek

```

import java.util.*;
public class TestVector{
    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);
    }
    public static void shuffle(List list, Random rnd) {
        for (int i=list.size(); i>1; i--)
            swap(list, i-1, rnd.nextInt(i));
    }
    public static void main(String args[]){
        Vector v = new Vector();

        for(int i=0;i<10;i++)
            v.add(new Integer(i));
        System.out.println(v);
        v.setElementAt("Andi",1);
        System.out.println(v);
        v.set(5,"Rita");
        System.out.println(v);

        swap(v,2,5) ;
        System.out.println(v);

        shuffle(v,new Random());
        System.out.println(v);
    }
}


```

```

C:\j2sdk1.4.1_01\bin\java.exe -class
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, Andi, 2, 3, 4, 5, 6, 7, 8, 9]
[0, Andi, 2, 3, 4, Rita, 6, 7, 8, 9]
[0, Andi, Rita, 3, 4, 2, 6, 7, 8, 9]
[3, 7, 8, 9, Andi, 0, 2, 6, Rita, 4]
Finished executing

```

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek

List

- Collection menyediakan method untuk merandom isi dari Collection

```
import java.util.*;

public class Shuffle {
    public static void main(String args[]) {
        List l = new ArrayList();
        for (int i=0; i<args.length; i++)
            l.add(args[i]);
        Collections.shuffle(l, new Random());
        System.out.println(l);
    }
}
```

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek

List

- Output program sama dengan sebelumnya tapi lebih singkat

```
import java.util.*;

public class Shuffle {
    public static void main(String args[]) {
        List l = Arrays.asList(args);
        Collections.shuffle(l);
        System.out.println(l);
    }
}
```

Politeknik Elektronika Negeri Surabaya


Pemrograman Berbasis Objek

List

```

import java.util.*;


public class TestList{
    public static void main(String args[]){
        List list = new ArrayList();
        list.add("Anis");
        list.add("Budi");
        list.add("Candra");
        list.add("Dewi");
        int i = list.indexOf("Candra");
        System.out.println(i);
    }
}

```

Output

2

Politeknik Elektronika Negeri Surabaya


Pemrograman Berbasis Objek

```

import java.util.*;

public class TestList{

    public static void replace(List l, Object val, Object newVal) {
        for (ListIterator i = l.listIterator(); i.hasNext(); ) {
            if (val==null ? i.next()==null : val.equals(i.next())) {
                i.set(newVal);
            }
        }
    }

    public static void replace(List l, Object val, List newVals) {
        for (ListIterator i = l.listIterator(); i.hasNext(); ) {
            if (val==null ? i.next()==null : val.equals(i.next())) {
                i.remove();
                for (Iterator j = newVals.iterator(); j.hasNext(); )
                    i.add(j.next());
            }
        }
    }

    public static void main(String args[]){
        List list = new ArrayList();
        List list2 = new ArrayList();

        list.add("Anis");
        list.add("Budi");
        list.add("Candra");
        list.add("Dewi");
        System.out.println(list);
        replace(list,"Anis","Rika");
        System.out.println(list);

        list2.add("Intan");
        list2.add("Imam");
        replace(list,"Budi", list2);
        System.out.println(list);
    }
}

```

```

C:\j2sdk1.4.1_01\bin\java.exe -cl:
[Anis, Budi, Candra, Dewi]
[Rika, Budi, Candra, Dewi]
[Rika, Intan, Imam, Candra, Dewi]
Finished executing

```



Pemrograman Berbasis Objek

List

- Tambahkan program sebelumnya

```

System.out.println(list.subList(0,5).indexOf("Dewi"));
list.add(2,"Dewi");
System.out.println(list);
System.out.println(list.subList(2,list.size()).indexOf("Dewi"));
System.out.println(list.subList(2,list.size()).lastIndexOf("Dewi"));


```

```

4
[Rika, Intan, Dewi, Imam, Candra, Dewi]
0
3
Finished executing

```

Politeknik Elektronika Negeri Surabaya


Pemrograman Berbasis Objek

```

public static List dealHand(List deck, int n) {
    int deckSize = deck.size();
    List handView = deck.subList(deckSize-n, deckSize);
    List hand = new ArrayList(handView);
    handView.clear();
    return hand;
}

class Deal {
    public static void main(String args[]) {
        int numHands = Integer.parseInt(args[0]);
        int cardsPerHand = Integer.parseInt(args[1]);


        // Make a normal 52-card deck
        String[] suit = new String[] {"spades", "hearts",
                                     "diamonds", "clubs"};
        String[] rank = new String[]
            {"ace", "2", "3", "4", "5", "6", "7", "8",
             "9", "10", "jack", "queen", "king"};
        List deck = new ArrayList();
        for (int i=0; i<suit.length; i++)
            for (int j=0; j<rank.length; j++)
                deck.add(rank[j] + " of " + suit[i]);

        Collections.shuffle(deck);

        for (int i=0; i<numHands; i++)
            System.out.println(dealHand(deck, cardsPerHand));
    }
}

```

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek

Output


```

% java Deal 4 5

[8 of hearts, jack of spades, 3 of spades, 4 of spades, king of diamonds]
[4 of diamonds, ace of clubs, 6 of clubs, jack of hearts, queen of hearts]
[7 of spades, 5 of spades, 2 of diamonds, queen of diamonds, 9 of clubs]
[8 of spades, 6 of diamonds, ace of spades, 3 of hearts, ace of hearts]

```

Politeknik Elektronika Negeri Surabaya




Pemrograman Berbasis Objek

List

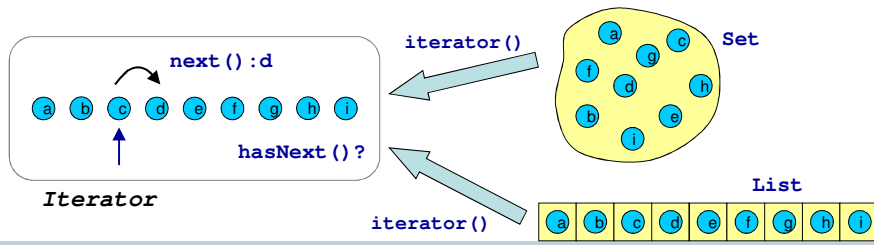
- Sebagian besar algoritma(method) pada class Collections diaplikasikan ke List. Sehingga dengan algoritma ini memudahkan untuk memanipulasi data pada List.
- sort(List)
mengurutkan List dengan algoritma merge sort
- shuffle(List)
Permutasi secara random pada List
- reverse(List)
membalik urutan elemen pada List
- fill(List, Object)
Mengganti setiap elemen pada List dengan value yang ditentukan
- copy(List dest, List src)
Mengkopikan source List ke destination List.
- binarySearch(List, Object)
Mencari sebuah element pada List dengan algoritma binary Search

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek


 `java.util.Iterator<E>`

- Think about typical usage scenarios for Collections
 - **Retrieve** the list of all patients
 - **Search** for the lowest priced item
- More often than not you would have to traverse every element in the collection – be it a List, Set, or your own datastructure
- Iterators provide a generic way to traverse through a collection regardless of its implementation



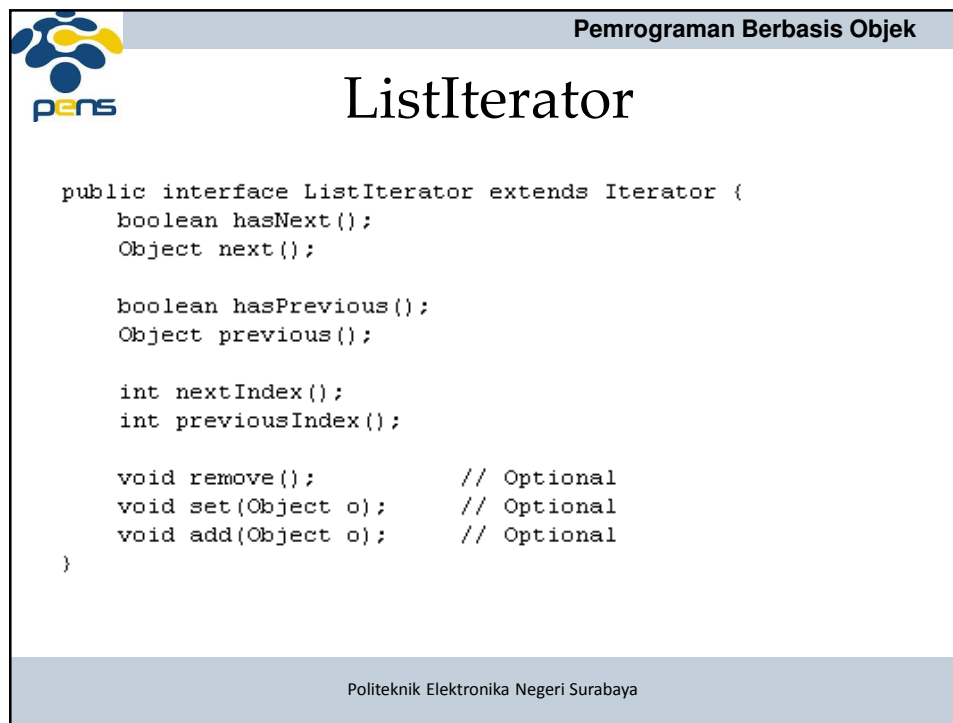
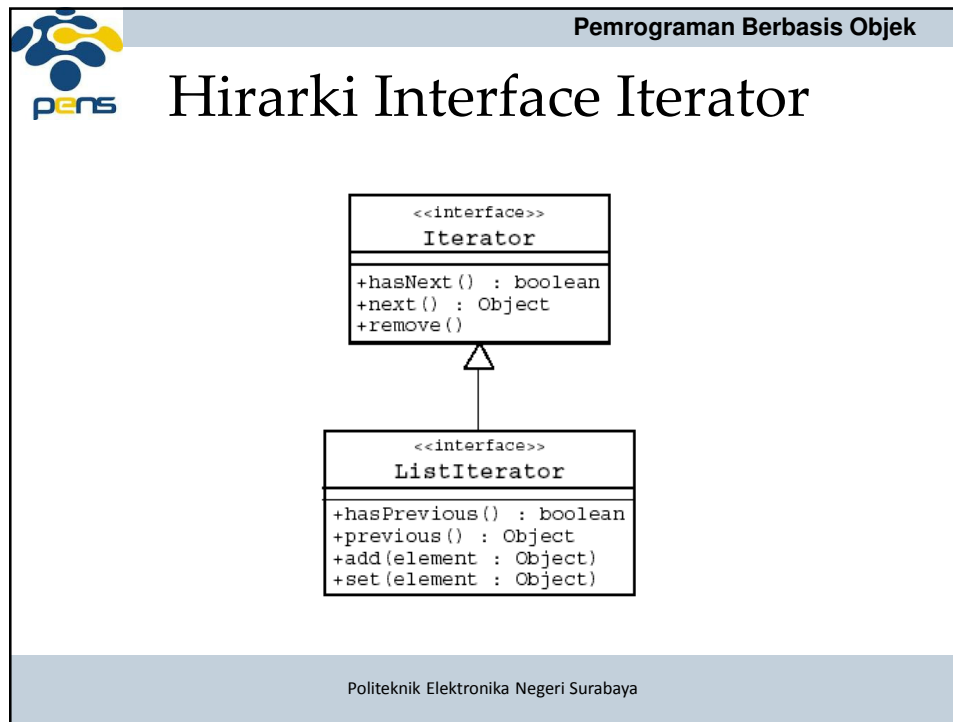
Politeknik Elektronika Negeri Surabaya 37


Pemrograman Berbasis Objek

 **ListIterator**

- **ListIterator** adalah subinterface dari **Iterator**.
- Dengan menggunakan ListIterator pada List, maka elemen dapat diambil secara backward.
- Gunakan method **next** atau **previous** sebagai navigasi.

Politeknik Elektronika Negeri Surabaya






Pemrograman Berbasis Objek

Map

- Menyimpan elemen dengan key unik.
- Satu key untuk satu elemen.
- Key disimpan dalam bentuk object.
- Map tidak bisa menyimpan duplicate key.
- Map bisa menyimpan duplicate element.
- Has no particular order.
- Contoh:
 - Hashtable
 - HashMap
 - not synchronized for threads
 - permits null values to be stored

Politeknik Elektronika Negeri Surabaya




Pemrograman Berbasis Objek

Map

- Map dapat dipandang sebagai Collection dengan 3 cara:
 - keySet
menghasilkan Set key yang ada pada Map
 - Values
Collection values yang ada pada Map. Collection disini bukan Set karena multiple key dapat mempunyai nilai yang sama.
 - entrySet
Hasil disimpan pada Set, pasangan key-value yang ada pada Map

Politeknik Elektronika Negeri Surabaya




Pemrograman Berbasis Objek

Map

- **Map** adalah object yang memetakan key dengan value. Disebut juga dengan *associative array* atau *dictionary*.
- Method untuk menambah dan menghapus
 - **put(Object key, Object value)**
 - **remove (Object key)**
- Method untuk mengambil object.
 - **get (Object key)**
- Method untuk mengambil key, value dan pasangan (key, value)
 - **keySet() // returns a Set**
 - **values() // returns a Collection,**
 - **entrySet() // returns a set**

Politeknik Elektronika Negeri Surabaya

43



Pemrograman Berbasis Objek

Map

```

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();

    // 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);
    }
}

```

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek

```

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}

```

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek

Map: Hashtable


```

class CollectionTest{
    public static void main(String [] arg){
        Hashtable ht = new Hashtable();
        ht.put("key1", new Integer(12));
    }
}

```

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



Map: HashMap

```


import java.util.*;
class HashMapTest{
    public static void main(String [] arg){
        HashMap hm = new HashMap();
        hm.put ("Game1", "Hearts");
        hm.put (null, "Chess");
        hm.put ("game3", "Checkers");
        hm.put ("game3", "Whist");
        hm.put ("game4", "Chess");
        System.out.println(hm);
    }
}

```

Output: {Game4=Chess, Game3=Whist, Game1=Hearts, null=Chess}

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



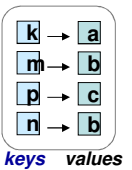
Map<K, V>

- Stores mappings from (unique) keys (type **K**) to values (type **V**)
 - See, you can have more than one type parameters!
- Think of them as “arrays” but with objects (keys) as indexes
 - Or as “directories”: e.g. "Bob" → 021999887

get(k)
→ a

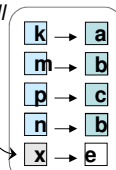
get(x)
→ null

size()
→ 4

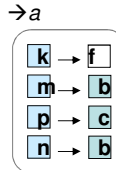


keys values

put(x,e)
→ null

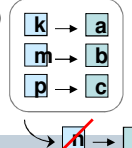


put(k,f)
→ a

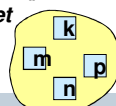


remove(n)
→ b

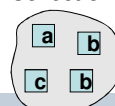
remove(x)
→ null



keySet()
→ Set



values()
→ Collection



<<interface>>
Map<K,V>

+put(K,V):V
+get(Object):V
+remove(Object):V
+size():int
+keySet():Set<K>
+values():Collection<V>
etc...


<<interface>>
SortedMap<K,V>

+firstKey():K
+lastKey():K
etc...

Politeknik Elektronika Negeri Surabaya

48

Pemrograman Berbasis Objek



```

import java.util.*;

public class TestMap {
    public static void main(String args[] ) {
        Map m1 = new HashMap();
        m1.put(new Integer(1),"abc");
        m1.put(new Integer(2),"abc");
        m1.put(new Integer(3),"def");

        Map m2 = new HashMap();
        m2.put(new Integer(1),"klm");
        m2.put(new Integer(2),"abc");
        m2.put(new Integer(3),"def");
        m2.put(new Integer(4),"Klm");
        m2.put(new Integer(5),"abc");

        m1.putAll(m2);
        System.out.println(m1);

        Set set = m1.keySet();
        System.out.println(set);

        System.out.println(m1.values());

        System.out.println(m1.entrySet());

        for (Iterator i=m1.entrySet().iterator(); i.hasNext(); ) {
            Map.Entry e = (Map.Entry) i.next();
            System.out.println(e.getKey() + " : " + e.getValue());
        }
    }
}

```


```

C:\j2sdk1.4.1_01\bin\java.exe -clas
{2=abc, 4=klm, 1=klm, 3=def, 5=abc}
[2, 4, 1, 3, 5]
[abc, klm, klm, def, abc]
{2=abc, 4=klm, 1=klm, 3=def, 5=abc}
2: abc
4: klm
1: klm
3: def
5: abc
Finished executing

```

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

Pemrograman Berbasis Objek



MultiMap

```

import java.util.*;

public class MultiMap {
    public static void main(String[] args) {
        Map m = new HashMap() ;

        String str[] ={"Andi","Ani","Anisa"};
        List l = Arrays.asList(str);

        m.put(new Integer(1),l);

        String str2[]= {"Budi","Badu","Bina"} ;
        l = Arrays.asList(str2);

        m.put(new Integer(2),l);

        System.out.println(m);
    }
}


```

```

C:\j2sdk1.4.1_01\bin\java.exe -classpath "D:
{2=[Budi, Badu, Bina], 1=[Andi, Ani, Anisa]}
Finished executing

```

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA




Pemrograman Berbasis Objek

SortedMap: TreeMap

- Aturan mirip Map
- Beda: obyek tersimpan secara sorted berdasarkan key.
- No duplicate key.
- Elements may be duplicate.
- Key tidak boleh null value.

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek


SortedMap: TreeMap

```
import java.util.*;
class TreeMapTest{
    public static void main(String [] args){
        SortedMap title = new TreeMap();
        title.put(new Integer(3), "Checkers");
        title.put(new Integer(1), "Euchre");
        title.put(new Integer(4), "Chess");
        title.put(new Integer(2), "Tic Tac Toe");
        System.out.println(title);
    }
}
```

Output: {1=Euchre, 2=Tic Tac Toe, 3=Checkers, 4=Chess}

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek

 `java.util.Collections`

- Offers many very useful utilities and algorithms for manipulating and creating collections
 - **Sorting** lists
 - Index searching
 - Finding min/max
 - Reversing elements of a list
 - Swapping elements of a list
 - Replacing elements in a list
 - Other nifty tricks
- Saves you having to implement them yourself
→ **reuse**


```
List<Movie> movies = new ArrayList<Movie>();
Collections.
```

- max(Collection<? extends T> coll) T - Collections
- max(Collection<? extends T> coll, Comparator<? super T> c) T - Collections
- min(Collection<? extends T> coll) T - Collections
- min(Collection<? extends T> coll, Comparator<? super T> c) T - Collections
- nCopies(int n, T o) List<T> - Collections
- replaceAll(List<T> list, T oldVal, T newVal) boolean - Collections
- reverse(List<T> list) void - Collections
- reverseOrder() Comparator<T> - Collections
- reverseOrder(Comparator<T> cmp) Comparator<T> - Collections
- rotate(List<T> list, int distance) void - Collections
- shuffle(List<T> list) void - Collections
- shuffle(List<T> list, Random rnd) void - Collections
- singleton(T o) Set<T> - Collections
- singletonList(T o) List<T> - Collections
- singletonMap(K key, V value) Map<K, V> - Collections
- sort(List<T> list) void - Collections**
- sort(List<T> list, Comparator<? super T> c) void - Collections
- swap(List<T> list, int i, int j) void - Collections
- synchronizedCollection(Collection<T> c) Collection<T> - Collections

Press 'Ctrl+Space' to show Template Proposals

Politeknik Elektronika Negeri Surabaya 53

Pemrograman Berbasis Objek

 `Collections.sort()`

- Java's implementation of merge sort – ascending order

e	b	c	d	b	f	a	⇒	a	b	b	c	d	e	f
0	1	2	3	4	5	6		0	1	2	3	4	5	6

- What types of objects can you sort? Anything that has an **ordering**
- Two sort() methods: sort a given List according to either 1) *natural ordering* of elements or an 2) externally defined ordering.

- `public static <T extends Comparable<? super T>> void sort(List<T> list)`
- `public static <T> void sort(List<T> list, Comparator<? super T> c)`

- Translation:
 1. Only accepts a List parameterised with type implementing **Comparable**
 2. Accepts a List parameterised with any type as long as you also give it a **Comparator** implementation that defines the ordering for that type

Politeknik Elektronika Negeri Surabaya 54