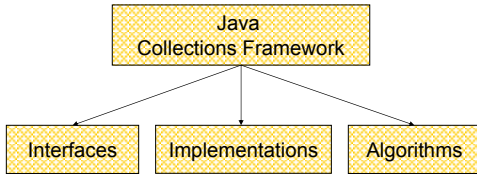


Java Collections Framework

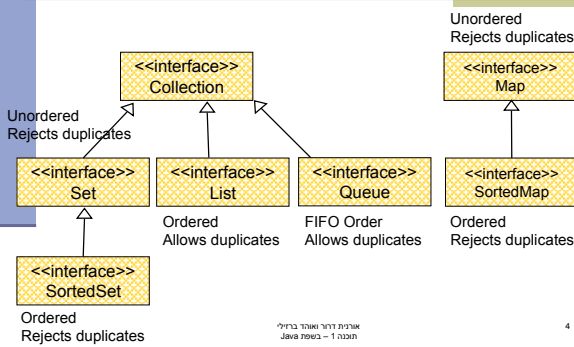
- **Collection:** a group of elements
- **Interface Based Design:**



Software 1 with Java

Recitation No. 8 (Collections)

Collection Interfaces



Online Resources

- **Java 5 API Specification:**
<http://java.sun.com/j2se/1.5.0/docs/api/index.html>
- **Sun Tutorial:**
<http://java.sun.com/docs/books/tutorial/collections/>

Collection Implementations

- **Class Name Convention:** <Data structure> <Interface>

General Purpose Implementations		Data Structures			
		Hash Table	Resizable Array	Balanced Tree	Linked List
Interfaces	Set	HashSet		TreeSet (SortedSet)	
	Queue				LinkedList
	List		ArrayList		LinkedList
	Map	HashMap		TreeMap (SortedMap)	

The Collection Interface

- Doesn't hold primitives (use wrapper classes)
- Since Java5 collections are type-safe:

```

Collection<String> collectionOfStrings = new LinkedList<String>();
Collection<Integer> collectionOfIntegers = new LinkedList<Integer>();

collectionOfStrings.add("Hello");           ✓
collectionOfIntegers.add(5);                ✓
collectionOfIntegers.add(new Integer(6));   ✓

collectionOfStrings.add(7);                 ✗
collectionOfIntegers.add("world");          ✗
    
```

Best Practice

- Specify an implementation only when a collection is constructed:

```

Set s = new HashSet();
    Interface      Implementation

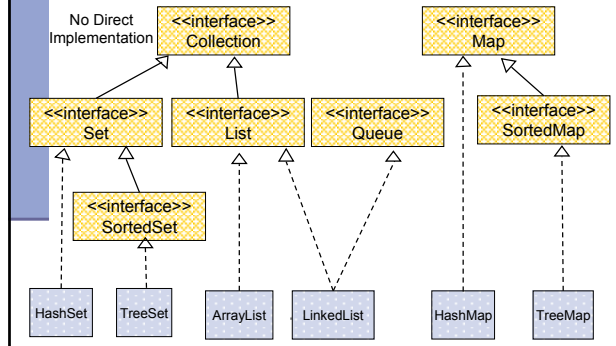
public void foo(HashSet s) {...} Works, but...
public void foo(Set s) {...} Better!

s.add() invokes HashSet.add()
    
```



8

General Purpose Implementations



Set Example

```

Set<Integer> set = new HashSet<Integer>();
set.add(3);
set.add(1);
set.add(new Integer(1));
set.add(new Integer(6));
set.remove(6);
System.out.println(set);
    
```

remove() can get only reference as argument

Output: [1, 3]

Insertion order is not guaranteed

A set does not allow duplicates. it may *not* contain:

- two references to the same object
- two references to null
- references to two objects a and b such that a.equals(b)

List Example

```

List<Integer> list = new ArrayList<Integer>();
list.add(3);
list.add(1);
list.add(new Integer(1));
list.add(new Integer(6));
list.remove(list.size()-1);
System.out.println(list);
    
```

Implementation

List holds Object references (auto-boxing)

List allows duplicates

Invokes List.toString()

Output: [3, 1, 1]

Insertion order is kept

remove() can get index or reference as argument

Map Example

```

Map<String,String> map = new HashMap<String,String>();
map.put("Dan", "03-9516743");
map.put("Rita", "09-5076452");
map.put("Leo", "08-5530098");
map.put("Rita", "06-8201124");
System.out.println(map);
    
```

No duplicates

Unordered

Output:

{Leo=08-5530098, Dan=03-9516743, Rita=06-8201124}

Keys (names)	Values (phone numbers)
Dan	03-9516743
Rita	06-8201124
Leo	08-5530098

12

Queue Example

```

Queue<Integer> queue = new LinkedList<Integer>();
queue.add(3);
queue.add(1);
queue.add(new Integer(1));
queue.add(new Integer(6));
queue.remove();
System.out.println(queue);
    
```

Elements are added to the tail of the queue

remove() may have no argument - head is removed

Output: [1, 1, 6]

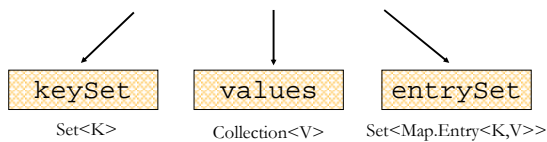
FIFO order

מאת: ד"ר. דניאל גולדברג
דוא"ר: daniel@cs.huji.ac.il

11

Map Collection Views

Three views of a `Map<K, V>` as a collection



SortedMap Example

```
SortedMap<String,String> map = new TreeMap<String,String>();
map.put("Dan", "03-9516743");
map.put("Rita", "09-5076452");
map.put("Leo", "08-5530098");
map.put("Rita", "06-8201124");
System.out.println(map);
```

Output:

```
{Dan=03-9516743, Leo=08-5530098, Rita=06-8201124}
```

lexicographic order

Keys (names)	Values (phone numbers)
Dan	03-9516743
Rita	06-8201124
Leo	08-5530098

Iterating Over the Keys of a Map

```
Map<String,String> map = new HashMap<String,String>();
map.put("Dan", "03-9516743");
map.put("Rita", "09-5076452");
map.put("Leo", "08-5530098");
map.put("Rita", "06-8201124");
```

```
for (String key : map.keySet()) {
    System.out.println(key);
}
```

Output: Leo
Dan
Rita

Iterating Over the Keys of a Map

```
Map<String,String> map = new HashMap<String,String>();
map.put("Dan", "03-9516743");
map.put("Rita", "09-5076452");
map.put("Leo", "08-5530098");
map.put("Rita", "06-8201124");
```

```
for (Iterator<String> iter= map.keySet().iterator(); iter.hasNext(); ) {
    System.out.println(iter.next());
}
```

Output: Leo
Dan
Rita

Iterating Over the Key-Value Pairs of a Map

```
Map<String,String> map = new HashMap<String,String>();
map.put("Dan", "03-9516743");
map.put("Rita", "09-5076452");
map.put("Leo", "08-5530098");
map.put("Rita", "06-8201124");
```

```
for (Map.Entry entry : map.entrySet()) {
    System.out.println(entry.getKey() + " : " + entry.getValue());
}
```

Output: Leo: 08-5530098
Dan: 03-9516743
Rita: 06-8201124

Iterating Over the Key-Value Pairs of a Map

```
Map<String,String> map = new HashMap<String,String>();
map.put("Dan", "03-9516743");
map.put("Rita", "09-5076452");
map.put("Leo", "08-5530098");
map.put("Rita", "06-8201124");
```

```
for (Iterator<Map.Entry<String,String>> iter= map.entrySet().iterator(); iter.hasNext(); ) {
    Map.Entry<String,String> entry = iter.next();
    System.out.println(entry.getKey() + " : " + entry.getValue());
}
```

Output: Leo: 08-5530098
Dan: 03-9516743
Rita: 06-8201124

Sorting

```
import java.util.*;

public class Sort {
    public static void main(String args[]) {
        List<String> list = Arrays.asList(args);
        Collections.sort(list);
        System.out.println(list);
    }
}
```

import the package of List, Collections and Arrays

returns a List-view of its array argument.

Arguments: A C D B

Output: [A, B, C, D]

lexicographic order

20

Collection Algorithms

- Defined in the [Collections](#) class
- Main algorithms:
 - sort
 - binarySearch
 - reverse
 - Shuffle
 - Min
 - max

אוניברסיטת תל אביב
הפקה 1 - ביטוח עסק

19

Sorting (cont.)

- Sort a List `l` by `Collections.sort(l)`;
- If the list consists of `String` objects it will be sorted in lexicographic order. Why?
- `String` implements `Comparable<String>`:

```
public interface Comparable<T> {
    public int compareTo(T o);
}
```
- Exception when sorting a list whose elements
 - do not implement `Comparable` or
 - are not *mutually comparable*.

אוניברסיטת תל אביב
הפקה 1 - ביטוח עסק

21