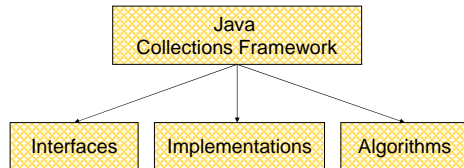


Java Collections Framework

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

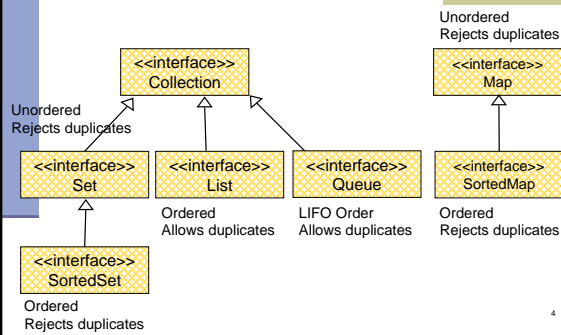


2

Software 1 with Java

Recitation No. 6
(Collections)

Collection Interfaces



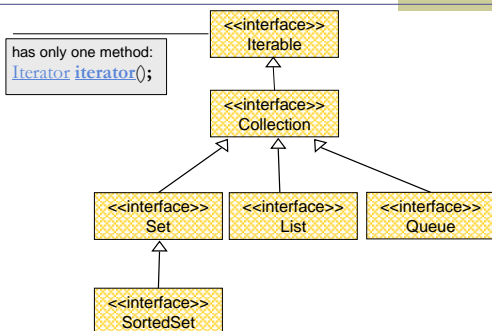
4

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/>

3

Collection extends Iterable



6

The Collection Interface

- Holds any Object references
 - Not type safe
 - Use casting
- Doesn't hold primitives
 - Use wrapper classes
- Since Java5 collections are type-safe
 - Will be discussed later in the course

5

Iterating over a Collection

```
for (Iterator iter = collection.iterator() ;
     iter.hasNext();) {
    System.out.println(iter.next());
}
```

8

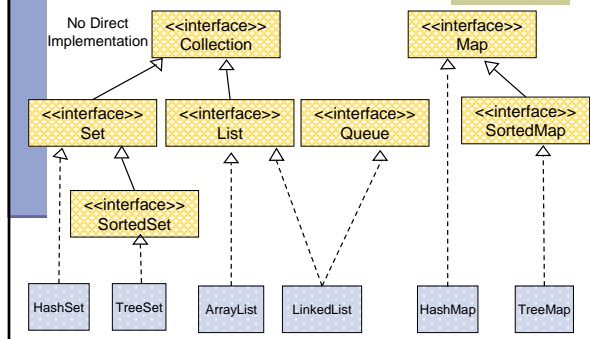
The Iterator Interface

- Provide a way to access the elements of a collection sequentially without exposing its underlying representation
- Methods:
 - hasNext() - Returns true if there are more elements
 - next() - Returns the next element
 - remove() - Removes the last element returned by the iterator (optional operation)

Command and Query

7

General Purpose Implementations



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)	

9

List Example

```
List list = new ArrayList();
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);
```

Invokes
list.toString()

Output:
[3, 1, 1]

List allows
duplicates

remove() can get
index or reference
as argument

Insertion
order is kept

Implementation

List holds
Object
references
(auto-boxing)

12

Best Practice

- Specify an implementation only when a collection is constructed:

```
Set s = new HashSet();
```

Interface Implementation

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

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

polymorphism

11

Queue Example

```
Queue queue = new LinkedList();
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

14

Set Example

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

A set does not allow duplicates. it does not contain two references to the same object two references to null references to two objects a and b such that a.equals(b)

remove() can get only reference as argument

Output: [1, 3]

Insertion order is not guaranteed

13

SortedMap Example

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

lexicographic order

Output:

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

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

16

Map Example

```
Map map = new HashMap();
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

15

Iterating Over the Keys of a Map

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

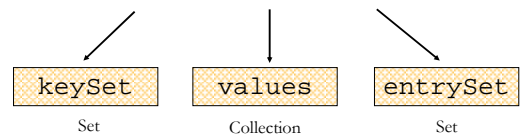
```
for (Iterator iter= map.keySet().iterator(); iter.hasNext(); ) {
    System.out.println(iter.next());
}
```

Output: Leo
Dan
Rita

18

Map Collection Views

Three views of a Map as a collection



The Set of key-value pairs (implement Map.Entry)

17

Iterating Over the Key-Value Pairs of a Map

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

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

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

casting

20

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

Output:
Leo
Dan
Rita

19

Sorting

```
import java.util.*;
```

import the package of List, Collections and Arrays

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

returns a List-view of its array argument.

Arguments: A C D B
Output: [A, B, C, D]

lexicographic order

22

Collection Algorithms

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

21

The System.out.printf command

- Useful for exercise 6
- A method of the java.io.PrintStream class
- Format:**
 - fixed text + format specifiers
 - printf([String](#) format, [Object](#)... args)
 - format specifier:**
%[argument_index\$][[flags][width][.precision]conversion
- A simple example:
System.out.printf("hello %s %d!!!\n", "world", 999);
Output: hello world 999!!!

24

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*.

23

The System.out.printf command

`%[argument_index$][flags][width][.precision]conversion`

conversion: s=string (any object), f=float (double, float)

d=decimal x= hexadecimal (int, byte,short, long)

Example:

```
System.out.printf(
    "d=%1$3d,s=%1$-3s,x=%1$x,f=%2$7.3f,%%", 10,
    12.2);
```

Output:

```
d= 10,s=10 ,x=a,f=12.200 ,%
```