Java - Collections, Maps and Iterators

CS 4354
Summer II 2015

Jill Seaman

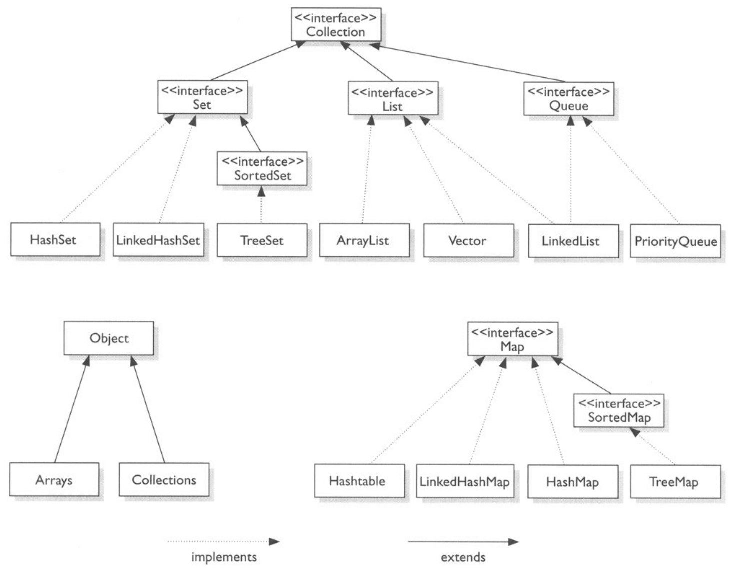
Collections in Java

- A collection is a data structure for holding elements
- java.util.Collection<T> is an interface implemented by many classes in Java. It has 3 extended interfaces:
  - ✦List<T> implemented by ArrayList<T> and LinkedList<T>, etc.
  - ✦Set<T> implemented by HashSet<T> and others
  - ✦Queue<T> implemented by PriorityQueue<T> and others
- Some methods in the Collection interface:
  - ✦isEmpty(), contains(e), add(e), remove(e), iterator()

Maps in Java

- A map is an object that associates keys with values.
- A map cannot contain duplicate keys; each key can map to at most one value.
- java.util.Map<K,V> is an interface implemented by many classes in Java
  - ✦HashMap<K,V>, Hashtable<K,V>
  - ✦TreeMap<K,V>
- Some methods in the Map interface:
  - ✦isEmpty, containsKey(e), put(k,v), get(k), remove(k)
  - ✦values(): Collection<V>, keySet(): Set<K>

Diagram of Collections and Maps in Java



Iterators in Java

- An iterator is an object that cycles through all the elements in a collection.
- java.util.Iterator<T> is an interface with the following methods:
  - ✦public T **next**()   returns the next element in the collection (and advances)
  - ✦public boolean **hasNext**() returns true if next() is not done.
  - ✦public void **remove**() (Optional) removes the last element returned by next.
- You can get Iterators from Collections (and Maps):
  - ✦ArrayList<Double> x = new ArrayList<Double>;
    Iterator<Double> it = x.iterator();
  - ✦HashMap<String,Double> hm = new HashMap<String,Double>;
    Iterator<Double> it = hm.values().iterator();

Collections and Iterators: example

```
import java.util.*;
public class SimpleIteration {
  public static void main(String[] args) {
    List<Pet> pets = Pets.arrayList(12);
    Iterator<Pet> it = pets.iterator();
    while(it.hasNext()) {
      Pet p = it.next();
      System.out.print(p.id() + ":" + p + " ");
    }
    System.out.println();
    // A simpler approach (because List implements Iterable)
    for(Pet p : pets)
      System.out.print(p.id() + ":" + p + " ");
    System.out.println();
    // An Iterator can also remove elements:
    it = pets.iterator();
    for(int i = 0; i < 6; i++) {
        it.next();
        it.remove();
    }
    System.out.println(pets);
  }
}
```