CSE 142

Unordered Collections

Introduction

- Quick Review:
  - Ordered vs. Unordered collections
  - an Ordered Collection: ArrayList
- Today:
  - an Unordered Collection: HashSet
  - an Unordered Collection: HashMap
- Readings
  - Dugan notes: end of ch. 14; middle of ch. 17

Ordered vs. Unordered Collections

- Some collections have a natural order of their elements:
  - the steps in a recipe
  - the list of daily weather observations
  - the list of shapes to be drawn (with later shapes layered over earlier shapes)
- ArrayLists are good for these collections.
- Some collections don't have any obvious natural order:
  - the ingredients in a recipe
  - the stars in the sky
  - the CDs at Tower Records
- ArrayLists are *not* ideal for these collections.

HashSet

- A HashSet can store a collection of elements without any order.
- A partial interface:

```
public class HashSet {
    // Add the argument object to the set, if it wasn't there already
    public boolean add(Object obj);
    // Return whether the argument object is an element of the set
    public boolean contains(Object obj);
    // Remove the argument object from the set, if it was present
    public boolean remove(Object obj);
    // Return the number of elements in the set
    public int size( );
    // Return an iterator that will go through all the set's elements, in some order
    public Iterator iterator( );
    …
}
```

Using HashSets

```
HashSet set = new HashSet( );
set.add("Parsley");    set.add("Sage");    set.add("Oregano");
set.add("Rosemary");  set.add("Thyme");              // draw the picture!
int count = set.size( );                              // what is count?
set.remove("Oregano");                                // what is the picture now?
if (set.contains("Arsenic")) {
    System.out.println("Beware!");
}
Iterator iter = set.iterator( );
while (iter.hasNext( )) {
    String ingredient = (String) iter.next( );
    System.out.println(ingredient);
}                                                      // what is printed?
```

Keyed Collections

- Some collections have a way to look up each element, using the element's ***key***.
- For example:
  - Each CD in a music collection could be looked up by title.
  - Each student in the class could be looked up by name, or by student ID.
  - Each entry in the dictionary can be looked up by the word the entry defines.
- A collection that links keys to data is called a ***map***, or sometimes a ***table*** or a ***dictionary***.
  - Each key must be unique! Cannot have two different entries have the same key.
  - Not always true in real life, so we often have to invent unique keys for things. (Can you think of any examples?)

HashMap

- **A HashMap can store a *keyed collection* of *values*.**

```
public class HashMap {
    // Make key map to value in the map
    // (either by adding a new mapping or by changing what key maps to)
    public Object put(Object key, Object value);
    // Return the value that key maps to, or null if it isn't in the map
    public Object get(Object key);
    // Return whether the argument object is a key of the map
    public boolean containsKey(Object key);
    // Return whether the argument object is a value in the map
    public boolean containsValue(Object value);
    // Remove key and the value it maps to from the map, if it was present
    public boolean remove(Object key);
    …
}
```

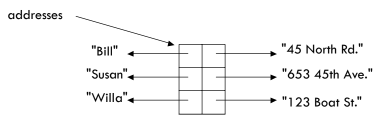
3/12/2002 (c) 2001-2, University of Washington U-7

Building a HashMap

- **Adding mappings:**

```
HashMap addresses = new HashMap( );
addresses.put( "Willa",    "123 Boat St."  );
addresses.put( "Bill",       "45 North Rd."  );
addresses.put( "Susan",  "653 45th Ave." );
```

- **The picture:**



3/12/2002 (c) 2001-2, University of Washington U-8

Examining a HashMap

```
HashMap addresses = …;
…
String addr1 = (String) addresses.get("Bill");            // what is addr1?
String addr2 = (String) addresses.get("Bobbie");        // what is addr2?

if (addresses.containsKey("Susan")) {
    System.out.println((String) addresses.get("Susan"));
}

addresses.remove("Willa");                      // what does the picture look like now?

// Bill moves in with Susan:
addresses.put("Bill", addresses.get("Susan"));          // what is the picture now?
```

3/12/2002 (c) 2001-2, University of Washington U-9

Null

- **In Java, there is a special value, null, which is used to represent "nothing" or "undefined."**
  - **Instance variables are initialized by default to null.**
- **Many collection methods return null to mean that no such object exists.**

```
HashMap notesToMyself = new HashMap ( );
…
String task = (String) notesToMyself.get("Most Important To-Do Item");
if (task == null) {
    System.out.println("Nothing to do; go play!");
} else {
    System.out.println("Get busy on " + task);
}
```

3/12/2002 (c) 2001-2, University of Washington U-10

More HashMap Methods

```
public class HashMap {
    …
    // Return the number of key/value pairs in the map
    public int size( );

    // Return a Set (the interface of HashSet) of the keys of the map
    public Set keySet( );

    // Return a Collection (the interface of all collections) of the values of the map
    public Collection values( );
    …
}
```

3/12/2002 (c) 2001-2, University of Washington U-11

Iterating through a HashMap

- **To iterate through a map, get either the keys or the values, and then iterate through them.**

```
HashMap musicCollection = …;
…
Set titles = musicCollection.keySet( );                  // get the set of keys
Iterator iter = titles.iterator( );                       // get an iterator on the keys
while (iter.hasNext( )) {
    String title = (String) iter.next( );                // get the next key
    CD disk = (CD) musicCollection.get(title);         // lookup the key
    System.out.println("Now playing " + title);
    disk.play( );
}
```

3/12/2002 (c) 2001-2, University of Washington U-12