Frameworks

CS151

Chris Pollett

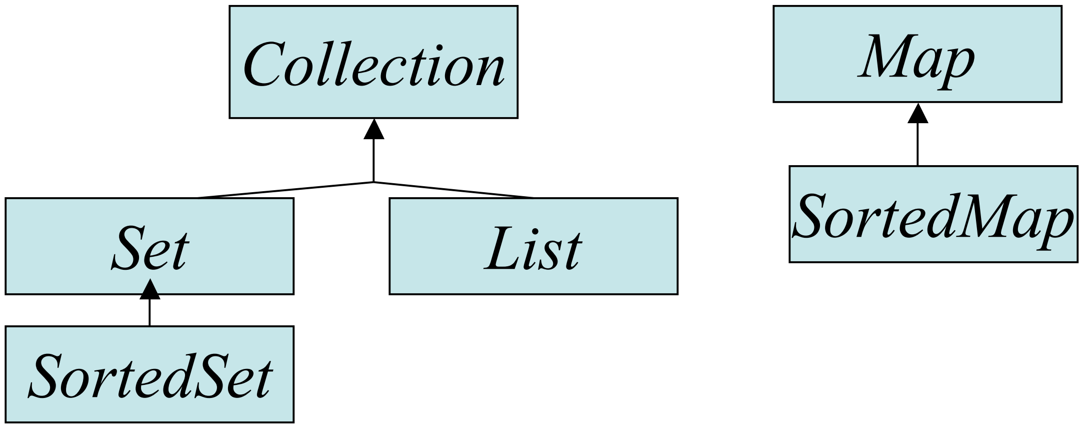
Oct. 26, 2005.

Outline

- Collections Framework
- GUI Frameworks - AWT and Swing

The Java Collection Framework

- Last day, we began discussing the Java collection Framework.
- We said what a collection was, and defined what bags, sets, lists, and maps were.
- Java defines several interfaces to support each of these:



Collection Framework Interfaces

- Collection
  - add(o), addAll(c), clear(), contains(o), containsAll(c), isEmpty(), iterator(), remove(o), removeAll(c), retainAll(c), size()
- Set
  - add(o), addAll(c) -contracts say now cannot have duplicates
- List
  - add(i, o), add(o) (add to end of list), addAll(c), andAll(i, c), get(i), indexOf(o), lastIndexOf(o), listIterator(), listIterator(i), remove(i), remove(o), set(i,o), subList(i,j)
- Map
  - clear(), containsKey(k), containsValue(v), entrySet(), get(k), isEmpty(), keySet(), put(k,v), putAll(m), remove(k), size(), values()

Concrete Collections

- HashSet implements Set -- uses a hash table to store objects
- LinkedHashSet implements Set -- uses a hash table and doubly linked list to store objects
- TreeSet implements SortedSet -- uses balanced binary tree to store objects
- ArrayList implements List -- uses resizeable arrays
- LinkedList implements List -- uses doubly linked lists
- Vector implements List -- uses resizeable arrays, is synchronized
- HashMap implements Map -- uses hash table
- IdentityHashMap implements Map -- uses identity not equality to do check in hash table
- LinkedHashMap implements Map -- uses hash table and doubly linked list
- TreeMap implements SortedMap -- uses balanced binary tree
- Hashtable implements Map legacy

Using Collections (JDK 1.4 and earlier)

- You might do code fragments like:

```
Set mySet = new HashSet();
String myString="hello";
Integer myInt = new Integer(5);
mySet.add(myString);
mySet.add(myInt);
System.out.println(""+mySet.size());
//etc. Note if used iterator would iterate over objects
```

- For a map you might do:

```
Map h = new HashMap();
h.put(new Integer(5), new String("Hello"));
String s = (String)h.get(new Integer(5)); /* notice the cast b/c return type of
    get is Object */
```

Generics

- As of JDK 1.5, Java support generics. We can now say things like a List of Strings

```
List<String> list = new List<String>();
list.add(“hello”);
String s = list.iterator().next(); //notice no cast
for(String str : list)
{
        System.out.println(str);
}
```

- JDK 1.5 also supports autoboxing unboxing of base types:

```
 TreeMap<Integer, String>  t =new TreeMap<Integer, String>();
 t.put(5,”hello”); //notice didn’t do new Integer(5)
 List<Integer> l = new LinkedList<Integer>();
 l.add(4);
 int i = l.get(0);
```

Iterators

- We already mentioned the Iterator pattern and the Java Iterator interface.
- Iterator has three methods, hasNext(), next(), remove()
- For lists this interface has a use sub-interface -- ListIterator.
- ListIterator supports add(o), hasNext(), hasPrevious(), next(), nextIndex(), previous(), previousIndex(), remove(), set(o).

GUI Frameworks - AWT and Swing

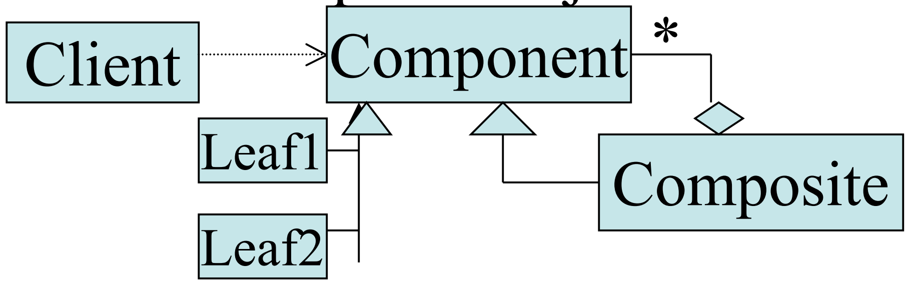
- Java’s GUI Framework consists of several categories of classes:
  - GUI Components: Components also known as widgets are the building blocks of the visual aspects of the GUI. Example components include: Button, Label, Checkbox, Scrollbar, Frame, and Dialog.
  - Layout managers: These define strategies for laying out GUI components in windows. For example, FlowLayout, BorderLayout, GridLayout
  - Events and event listeners: events represent user input or actions. Each aevent class represent a particular kind of input. For instance, KeyEvent for keyboard inputs, MouseEvent. For each event type, there is an associated Listener responsible for handling events of that type. For example, KeyListener, MouseListener.
  - Graphics and imaging classes: Color, Font, Graphics, Point, Rectangle, Dimension, Image, Icon, etc.

AWT and Swing

- The main GUI component in the AWT is Component.
- It has the following subclasses of primitive components: Button, Canvas, Checkbox, Choice, Label, List, Scrollbar, and TextComponent. TextComponent is further subclassed as TextArea and TextField.
- Component also have several subclasses of container components.
- In general, a container components is allowed to have 0 or more other components on it.
- Container is a subclass of Component to handle container components. It has two main subclasses Panel and Window. Applet is a subclass of Panel, Window is further subclassed as Frame and Dialog. Dialog has FileDialog as a subclass.
- Swing has a similar list of classes as above but each is preceded by the letter J. For example, JComponent, JButton,etc.
- Swing components are lightweight - that is they are drawn directly by Java and look the same across platforms. AWT components are heavyweight -- they are mapped to the native components of the particular OS. There is an overhead in doing this.

Composite Design Pattern

- Category: Structural Design Pattern
- Intent: Compose objects into tree structures to represent a part-whole hierarchy.
- Applicability: Use the Composite pattern
  - when you want to represent a par-whole hierarchy of objects
  - when you want clients to be able to ignore the difference between composite objects and individual objects.



Layout Managers

- Container has a LayoutManager on it which is responsible for where components on the container are drawn.
- LayoutManager has the following subclasses: BorderLayout, CardLayout, FlowLayout, GridLayout, LayoutManager2. This last is further subclassed as GridBadLayout.
- To set the layout of a container one does: setLayout(lm)
- To add a component to a container one uses add(comp) or add(comp, cst)