CSCI 136
Data Structures & Advanced Programming

Bill Jannen

Lecture 6

Feb 15, 2017

Announcements

- How was Lab 1?
- Lab 2 is a little tricky (but fun)
  - Bring your design docs to Lab!
  - Useful references in the book: read the handout carefully

Last Time

- Learned about assertions and pre/post conditions

```
        assert <condition> : <error message>;
```

  - Compile code normally, but run with:

```
        $ java –enableassertions <program>
```

- Discussed Associations
  - Key-value pairs
  - General-purpose class: keys and values are Objects

Review: Association Class

```
import structure5.*;
class Association {
   protected Object key;
   protected Object value;

   //pre: key != null
   public Association (Object key, Object value) {
        assert (key!=null) : “Null key”;
        this.key = key;
        this.value = value;
   }

   public Object getKey() {return this.key;}
   public Object getValue() {return this.value;}

   public Object setValue(Object value) {
        Object old = this.value;
        this.value = value;
        return old;
   }
}
```

Person.java (once More)

Shaquille O'Neal: 7' 1"
(aka The Big Shamrock, Shaq Fu, …)

Simone Biles: 4' 9"

Review: Association Class

```
import structure5.*;
class Association <K, V> {
   protected K key;
   protected V value;

   //pre: key != null
   public Association (K key, V value) {
        assert (key!=null) : “Null key”;
        this.key = key;
        this.value = value;
   }

   public K getKey() {return this.key;}
   public V getValue() {return this.value;}

   public V setValue(V value) {
        V old = this.value;
        this.value = value;
        return old;
   }
}
```

Generics

- Casting is dangerous
    - …but sometimes unavoidable
- Generics let us catch type errors at *compile* time
- We can't construct generic arrays
    - Vector.java shows how to handle this

Today's Outline

- Learn about Vectors
    - Dynamically resizable array
    - Easier to use (in most cases) than arrays
- How are Vectors implemented?

Searching Vectors

- If we were implementing Vector.contains(myObject), what would we do?
    - Loop through elements and return true if one element equals myObject
- What does this require?
    - Properly defined equals() method in myObject class!
    - (== checks if two objects are the same object, not if they are logically equivalent)

Notes About Vectors

- Primitive Types and Vectors

```
Vector v = new Vector();
v.add(5);
```

  - This (technically) shouldn't work! Can't use primitive data types with vectors…they aren't Objects!
  - (But Java is now smart about some data types, and converts them automatically for us -- called autoboxing)

- We used to have to "box" and "unbox" primitive data types:

```
Vector<Integer> v = new Vector<Integer>();
Integer num = new Integer(5);
v.add(num);
…
Integer result = v.get(0);
int res = result.intValue();
```

- Similar wrapper classes (Double, Boolean, etc) exist for all primitives

Vector Summary So Far

- Vectors: “extensible arrays” that automatically manage adding elements, removing elements, etc.
  1. Use generics to specify type when creating a new Vector<E>
  2. Use wrapper classes (with capital letters) for primitive data types (use “Integers” not “ints”)
  3. Define equals() method for Objects being stored if contains(), indexOf(), etc. is needed

Implementing Vectors

- Vectors are really just arrays of Objects
- Key difference is that the number of elements can grow and shrink dynamically
- How are they implemented in Java?
  - What instance variables do we need?
  - What methods? (start simple)
- Constructor(s): `Vector()`, `Vector(size)`, `get(index)`, `set(index, Obj)`, `add(Obj)`, `add(index, Obj)`, `remove(index)`, `isEmpty()`, `size()` (we'll finish some of these next time!)

Vector.java

Lab 2

- Three classes:
  - Table.java: Vector< Association< String, FrequencyList> >
  - FrequencyList.java: Vector< Association<String, Integer> >
  - WordGen.java: main method
- Two Vectors of Associations
- Implement toString() in Table and FrequencyList for debugging!
- What are the key stages of execution?
  - Test code thoroughly before moving on to next stage