CS193J: Programming in Java

January 22, 1999

Section #2

Outline

- Small complete Die program
- Modular development and testing
- Using Java API Specifications
- Built-ins Math, Vector, Hashtable
- Please ask questions!
  - Office hours in 193C from 2:30 to 4

Packages

- Partitions the name space for classes
- Large collection of built-in packages
  - java.util.* (Vector, Stack, Hashtable,…), ...
- java.lang implicitly available
- Importing a package makes it visible
  - can use methods without package prefix
    Date.getTime() instead of
    java.util.Date.getTime()

The Object Class

- Everything in Java is an (O)object.
- Root of the class hierarchy
  - all Java classes are subclasses of this
  - public class Object
- Provides a set of methods common to every Java class
  - clone(), equals(Object)…
  - public String toString()

Java API Specification

- http://java.sun.com/docs
- Link from class pages (Other Materials)
- Indexed at three levels
  - Packages java.lang, java.util, …
  - Classes java.lang.Math, java.util.Vector, …
  - Variables,Constructors, Methods, Exceptions, Errors
- Don’t know where to look?
  - Look in the Class Hierarchy!

The Die Class (Summary)

- Packages and visibility
- Method specifications from (online) docs
- Java Class hierarchy
- toString() and System.out.println()
- Separate testing

The Vector Class

- Variable length arrays of objects
- Docs: Where is the Vector Class?
  - java.util
  - “find” (Alt-F on some browsers) in the hierarchy
- Constructing a vector of a specified capacity
  - Vector(int)

Some Vector Methods

- *addElement(Object)*
  - Adds the specified component to the end of this vector, increasing its size by one.
- *insertElementAt(Object, int)*
  - Inserts the specified object as a component in this vector at the specified index.
- *isEmpty()*
  - Tests if this vector has no components.
- *elementAt(int)*
  - Returns the component at the specified index.
- *size()*
  - Returns the number of elements in the vector.

The Dice Class (Summary)

- Creates a collection of dice
- Interacts with the Die class
- Uses the built-in Vector class
- Separate testing module

The Hashtable class

- Create hashtables, what else?!
  - map keys to values
  - any non-null object can be used as a key
- java.util.Hashtable
  - Browse online docs for specifications
- Constructing a hashtable
  - Hashtable()

Some Hashtable methods

- *isEmpty()*
  Tests if this hashtable maps no keys to values.
- *size()*
  Returns the number of keys in this hashtable.
- *containsKey(Object)*
  Tests if this hashtable contains a specific key.
- *put(Object, Object)*
  Maps the specified key to the specified value in this hashtable.
- *get(Object)*
  Returns the value to which the specified key is mapped in this hashtable.
- *remove(Object)*
  Removes the key (and its corresponding value) from this hashtable.

Using a Hashtable?

- Insertion:
  - You provide a key which is to be associated with a object and a value which is the object you want to store.
- Retrieval:
  - To retrieve, you only need to provide the key again and out pops the right object.

Hashtable Objects

- Must implement hashCode()
- Must have a consistent equals() method
- The top level Object class implements hashCode() so every object has this default hashCode() function. The default function simply uses the object address.
- You can either use that or override the hashCode() method to write your own.
  - If you override hashCode(), you have to make sure that equals() is still consistent with it. Namely, if two different objects compare equal, they should have the same hash value.

Wrapper Classes

- Wrap basic types into classes
  - int, float, boolean, double, char
  - provided by classes Number, Boolean, Character,… in java.lang
- Can use wrapped basic types in collection classes
  - Hashtable, Vector, Stack,…
- Integer class
  - Integer(int) // Wrap an integer
  - int intValue() // Unwrap it

Summary

- Java API specifications
- Organization of classes into packages
- Testing classes independently
- Using built-ins
  - Vector, Hashtable, Math,...
  - docs as an efficient tool
- Wrapper classes
- Good luck on Homework One !!