Lecture 2: Overview & Java

CS 62
Spring 2012
Kim Bruce & Kevin Coogan

TAs: Ben Fish, Sam Konowich, Eli Omernick, Matthew Stock-Matthews, and Will Tachau
Course web page: http://www.cs.pomona.edu/classes/cs062

Homework

- Solutions to odd problems are in back of text.
- Ask questions at the beginning of class.

Card Deck Examples

- CardInterface -- interface
- AbsCard
  - abstract class, implements CardInterface
- Card extends AbsCard
- OtherCard extends AbsCard

} *alternate implementations*

- Deck
  - Class using cards

Java Keywords

- Abstract class -- can't be instantiated
  - usually some methods missing
- Information hiding qualifiers:
  - public
  - private
  - protected
- Static -- copy associated with class, not objects
- Final -- only assigned to once
  - in its declaration or constructor

Interfaces &Inheritance

- Class implements interface if supports all methods defined in interface
  - Try to use interfaces as types for flexibility
- Interface can extend another by adding methods
  - If A extends B and x has type A, then also has type B
- One class can extend another
  - inherits fields and methods
  - can override existing methods, add new ones
- instanceof & casts

Generics

- Can write classes parameterized by types
- See Association class
- Can only instantiate type parameters by interfaces or classes, not primitive types
- "Wrapper" versions of primitive types can be used instead of primitive types:
  - int -> Integer, double -> Double, boolean -> Boolean

JavaDoc

- Stylized form of comments, w/tools to extract

```
/**
 * comments here
 */
```

- Common tags:
  - @author *author name*
  - @version *date*
  - @param *param name and description*
  - @return *value returned, if any*
  - @throws *description of any exceptions thrown*

Comments

- Class header needs @author, @version
- Method header should include
  - Description of what (not how) it does
  - @param line for each parameter
  - @return if method returns a value
  - pre and post conditions as necessary
    - If no @return, then must have post
    - If checkable the add assert (see later) for postconditions

Pre and Post-conditions

- Pre-condition: Specification of what must be true for method to work properly
- Post-condition: Specification of what must be true at end of method if precondition held before execution.
- See Ratio class example

Assertions in Java

- Won't use Assert class from Bailey.
- Command to check assertions in standard Java
  - Two forms
    - assert boolExp
    - assert boolExp: message
- Article on when to use assert:
  - http://download.oracle.com/javase/7/docs/technotes/guides/language/assert.html
  - Short summary -- never use for preconditions of public methods -- make explicit checks
  - Use for postconditions & class invariants

Turning on assert

- Turn on assertions when run program, by adding "-ea" (without quotes) as virtual machine argument in arguments tab in Eclipse when set up runtime configuration.
- If leave it off, then ignores assert statements.
- If on and assertion is false, then will raise an AssertionError exception and will print associated message

Array and ArrayList

Arrays

- Containers that hold objects
  - Different syntax from objects
  - Public instance variable "length"
- Because of limitations of Java virtual machine, cannot create array of type variable:
  - E.g., new T[5] illegal if T is type variable
  - new C[5] is legal if C is primitive, class, or interface name.

ArrayList

- What happens if need more space in array than originally allocated?
- ArrayList is class that dynamically expands as needed.
- Part of java.util package
- To get access write import java.util.ArrayList or import java.util.*

ArrayList Specification

- Class ArrayList<E>
- Important methods:
  - add, get, set, indexOf, isEmpty, remove, size, contains, clear
  - size, isEmpty, get, set take constant time
  - add is "amortized constant" time
- See javadoc at
  - http://download.oracle.com/javase/6/docs/api/

Java Graphics

For details, see document on course web page in documentation and handouts section.

Graphics context

- All drawing is done in "paint" method of component
- public void paint(Graphics g)
  - g is a Graphics context
  - "pen" that does the drawing
  - Programmer calls repaint(), not paint!!
- Need to import classes from java.awt.*, java.awt.geom.*, & javax.swing.*
- See MyGraphicsDemo

General graphic applications

- Create an extension of component (either JPanel, JFrame, or JApplet) and implement paint method in the subclass.
  - See main method of demo to get window to show
  - Start paint method by casting g to Graphics2D to get access to new methods
- Call repaint() on component every time make a change.
  - Causes OS to schedule call of paint in event queue
  - Called automatically if window obscured and revealed

Geometric Objects

- Objects from classes Rectangle2D.Double, Line2D.Double, etc. from java.awt.geom
  - There are also float versions
  - Common superclass is Rectangular
  - Constructors take params x,y,width,height,
    - but don't draw object
  - myObj.setFrame(x,y,width,height) can move object
  - g2.draw(myObj) -- gives outline
  - g2.fill(myObj) -- gives filled version
  - g2.drawString("a string",x,y) draws string

MyGraphicsDemo

- Class extends JFrame, which creates window.
  - Constructor calls super with title of window.
- Main method creates object, sets size, visibility, and enables go-away box in upper left
- Paint method creates and draws objects.