CS 251 - Lab 004

TA: Kage Weiss
Office Hours: R 2-2:50 FEC 2000, or by appointment.
Contact: mmweiss@unm.edu
Website: **http://cs.unm.edu/~kageweiss/TA/cs251.html -- SLIDES POSTED**

- Sign in sheet located on desk by TA
- Today we are working on finishing up GUI Practice (Lab 8), and starting the final project Bubble Shooter (Lab 9)

How I Grade Your Code:

- Chenoweth provides the rubric and occasionally tester code.
- I write tester code that thoroughly tests your code.
- If your code runs as expected with my tester, I'm happy.
- I then review your code (.java) to make sure it follows CS251 Code Standards.
  - That means variable and method names, comments, privacy, **NO TABS**, etc.
- I **can** tell how much work you put into your code.
- I can tell who didn't cite StackOverflow.
- If you obviously put work in, I'm happy to give points and comments!
- If you obviously didn't put work in, I have to really look at your code and end up finding (and counting off for) minor infractions.
- Every point I take off is listed in the comments for the grade. **Read them**.

GUIs

- Frames
- Panels
  - Layouts and placing
- Buttons, clicks
  - Events
  - Action Listeners
- Labels
- Images/Drawing
- repaint()

Don’t forget:

- **frame.pack();**
- **frame.setDefaultCloseOperation(JFrame.EXIT…);**
- **frame.setLocationRelativeTo(null);**
- **frame.setVisible(true);**

Making a Game

You just wrote a simple GUI, so you know how complex they are, any guesses as to how complex it would be to try and make your GUI run a game? 251 me would never recommend that mess.

That's why we break down the problem into pieces, we'll let the GUI do GUI things and only GUI things later on when we write it, and we'll write the game as a simple program by itself: Bubble Manager

- Bubble Manager is the core of your game, and by reading the pdf (and maybe the linked article) you will be able to write the game so that you can use a tester to "play" it.
- This allows you to keep game issues and GUI issues separate.

Making a Game

Organization

- The GUI comes after your base program is running smoothly
  - A GUI is graphical, so maybe draw your plan for what you want it to look like before you build it.
  - The only methods your GUI should be calling are those your BubbleTester called.
  - Fancy graphics aren’t worth any points if your game doesn’t run properly.
  - Remember to work in your instance, if you can’t run two completely separate windows of the game, you have too many static variables and methods.

Making a Game

Organization

- A main class merges your manager and GUI instances
  - Though possible to do in your GUI, a main class allows for separating out the startup from the internal game.
  - It also allows an easy *new game* solution by simply replacing the old manager in your current GUI with a new Manager OR calling a newGame() method on your Manager.
- You'll learn a lot more about this in 351!

```java
/**
 * @version date ( in_CS_XXX_00X format : YYYY - MM - DD )
 * @author FirstName LastName
 **/

/** My class ClassName does ... */
public class ClassName {

    private memVarType memberVariable;

    /**
     * Getter for specified member variable.
     * @return this.memberVariable The memberVariable of this instance
     */
    public memVarType methodName() {
        return this.memberVariable;
    }
    /**
     * What does this method do?
     * @param param1 What is this parameter?
     * @return What are we returning?
     * @throws ExceptionName Why are we throwing this/what triggers this?
     */
    public varType methodName2 (memVarType param1) throws ExceptionName {
        if(param1 == this.memberVariable) {
            for(int i = 0; i <= param1; i++) {
                System.out,println(i + ". Our var = " + this.memberVariable)
            }
            throw new ExceptionName(ExceptionParameters);
        }
        return (this.memberVariable + param1);
    }
}
```

Info Block, tells who/when/what

Method is public, so it gets a JavaDoc comment

Method is public, so it gets a JavaDoc comment, this one's a bit longer because it has three fields

Indentations must be spaces, NOT Tabs

```java
import java.all.my.imports.*; // What imports from java do I need?
import cs251.interface; //(this isn't the import, but you need one)
/**
 * @version date ( in_CS_XXX_00X format : YYYY - MM - DD )
 * @author FirstName LastName
 **/
public class Demo {
    private NestedClass variable;
    private OtherCollection otherVariable;
    /** What my constructor does to prepare a new instance */
    public Demo() {
        this.variable = new NestedClass();
        this.otherVariable = prepareNewCollection();
    }
    /** This is that preparation method so we contain everything.
     * @return OtherCollection our own custom collection */
    private OtherCollection prepareNewCollection() {
        OtherCollection output = new OtherCollection();
        output.
        return output;
    }
    /* This is a nested class, it's private so no javadoc comment needed,
     * but we can say what it does here to make life easier */
    class NestedClass extends Collection implements ThatInterfaceWeAreUsing {
        @Override
        public int ThatOneMethodFromInterface() {
            return 9001;
        }
        @Override
        public double OtherMethodFromCollection(Collection args) {
            double[] math = /* oh hey we did something with args here*/;
            return math;
        }
```

Info Block, tells who/when/what
No room on the slide, but you need a class comment too.

Constructors you write need comments too

Method is public, so it gets a JavaDoc comment.
It takes no arguments, but what does it do?

Nested class is private, so no need for JavaDoc, but comments are still a part of documenting your code, what is this class?

WHO WOULD WIN?

a computer program with millions of lines of code

one C U R L Y B O Y with no friend

{

Though it may not seem like much, every character is important to the composition of a work of code.