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 Inheritance (Lab 3)
  - **Kage was out due to unforeseen circumstances.**

Inheritance

This lab is all about using inheritance through interfaces and parentage to simplify the code you have to write.
Think about:

- What you must have every time (things that change)
- What you want to avoid having every time (things that don’t change)
- Things that you already have elsewhere, and how that helps you where you are (Inheritance)
  - What use are your parents?
    - What use are their parents?
  - Am I doing anything myself that they have already done for me?

Exceptions

This lab is all about using exceptions to trigger behavior when something wrong (but expected) happens, and writing your own exception using Inheritance.

Think about:

- try{ /* code */ } catch(<Type of exception> <name>) { /* code */}
  - NOTE: for this lab, when it says to throw the exception, that does not mean to throw and then catch your own exception, just throw it.
- What is it that throws the exception?
- What do we want to do when the exception is thrown?
- Where all could this be useful?
- What is the difference between a checked and an unchecked exception?

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**.

```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?