Assignment #2

Due: 11:59 pm Friday November 2nd
Submit all source code as a .jar file in the D2L dropbox

In this assignment, we will work on setting up classes with good documentation – preconditions, postconditions and class invariants.

This is a group assignment – look in D2L to see which group you belong to.
Make sure you are contributing to the discussions in the group. There is a D2L discussion forum for your group where you can share ideas and store files.
Choose a primary role, and a secondary role if you wish, in your group but make sure there is at least one Coordinator, Recorder, and Checker.

Download the a2.jar from from D2L. Start up Eclipse and create a new Java project in Eclipse with an **`src`** folder. Right-click on this src folder and import the a2.jar.

Take a look at the file named **`A2.java`**. This program should compile and run. When you run this program, you will see a menu printed in the Console pane. Enter the number 1 and hit the Enter (Return on Macs) key. Enter the number 3 to display and note what happens. Then enter the number 2 and Enter and so on until you have 10 shapes.
Note what happens when you try to create an 11th shape. See if you can figure out where the problem is.

Add two new classes:

1. **`Circle`** class. This class should extend the **`Ellipse`** class.
   - Add a constructor with 3 parameters: x and y positions and a diameter
   - Do not add any data members – use the data members of Ellipse without modifying their visibility.
   - Add a query method named diameter that will return the diameter of the Circle object (without modifying the visibility of the data members of Ellipse).
2. **`Square`** class. This class should extend the **`Rectangle`** class
   - Add a constructor with 3 parameters: x and y positions and a size
   - Do not add any data members – use the data members of Rectangle without modifying their visibility.
   - Add a query method named size that will return the size of the Square object (without modifying the visibility of the data members of Rectangle).

Modify these classes:

1. **`ShapeUserInterface`** – this is the class that sets up the text-based user interface for this program
   - add static final int data members so that the user can choose to create new Circle or Square objects.
   - Modify the `displayMenu` method so that it prints choices for a new circle and a new square.
   - Modify the `executeChoice` method so that it calls the `createShape` method if the user chooses to create a new Circle or Square. Note that this method could be simpler.
   - Modify the `createShape` method so that it is now capable of creating Circles and Squares. Use a switch statement instead of all the if statements. You will have to modify what the last println in this method does so that the appropriate messages are printed for circles and squares.
   - Modify the `createShape` method so that it is now checks (see below) if the `shapes` object has room for more before it adds another shape to `shapes`. After this change, the program should not crash if the user tries to add more shapes than there is room for.

2. **`ShapeCollection`** – this class is used to store a collection of Shape objects.
   - Without modifying any of the existing code (you can add methods but you cannot make changes to visibility of data members or change existing methods) make it possible for the `ShapeUserInterface.createShape` method to check if there is empty space in the store array.

3. **`A2.java, Circle.java`**, **`Ellipse.java`**, **`Rectangle.java, Shape.java, ShapeCollection.java, ShapeUserInterface.java, Square.java`**:
   Add Javadoc comments to the top of each of these files. These comments should appear as shown in the Course Syllabus (also see below). These comments should have a short description of the class and should have your names, the roles you played in the assignment, and your emails.
4. **`A2.java, Circle.java`**, **`Ellipse.java`**, **`Rectangle.java, Shape.java, ShapeCollection.java, ShapeUserInterface.java, Square.java`**:
   Add Javadoc comments to every method in these classes.
   Add preconditions and postconditions for all methods that have an input parameter.
   Remember that preconditions and postconditions, when possible, should be valid Java boolean expressions.
5. **`Ellipse.java`**, **`Rectangle.java, Shape.java, ShapeCollection.java`**: Add class invariants for the data members for these classes.

Javadoc HTML files

To generate a complete set of HTML files from the Javadoc comments in your source code:

i. In the Package Explorer pane, locate the Project you were working on and click on the src folder inside the project.
ii. Click on Project in the main menu bar and choose "Generate Javadoc ..."
   This opens up the Javadoc Generation window.
   a) On the lab computers, if the box under the label "Javadoc command:" is blank or not correct, you will have to click on the Configure … button and locate the javadoc program. On Windows systems this will be in C:\Program Files\Java\jdk1.6.0nnn\bin where the nnn is the version number of the JDK on the system.
   b) In the pane titled "Select types for which Javadoc will be generated:" make sure that all the .java files in our project are chosen.
   c) Below this is the part labeled "Create Javadoc for members with visibility:"; choose Private
   d) Use the standard doclet and note the Destination – this should be somewhere inside your Eclipse workspace.
      Windows bug: On some Windows system, this procedure works properly the first time but the second time around, the spaces in the Destination are replaced with the character sequence "%20" and so you will have to edit this and make sure the documentation goes to the right spot.
   e) Click on Next and provide a title if you wish.
   f) Click on Next and in the pane titled "Extra javadoc options" add
      -tag require:a:"Preconditions:"
      -tag ensure:a:"Postconditions:"
   g) Click on Finish
iii. It may take some time for Javadoc to get started but eventually you should see some output from the Javadoc program in the Console pane. If you see errors saying that the @date tag is "unknown" and you want this tag to work, re-try this step.
   Once everything works, you should see a new folder in your Project titled "doc". Open this folder up and you should see a number of .html files.
iv. Right-click on the "index.html" file and select Open with → Web Browser. You should see all the classes in your program nicely documented as a series of web pages – with Precondition and Postcondition sections.

When exporting the project as a .jar file, be sure to select the doc folder and all its contents for export to your .jar file.

Your program should run without compiler or runtime errors – when fixing runtime errors fix the actual causes of errors; wrapping code with try-catch blocks with the aim of suppressing error messages is not a good fix.

Documentation:

Every .java file should have Javadoc comments at the top that look like:

```
/**
 * <p>Title: Example Class</p>
 * <p>Description: CS 235 Assignment #2</p>
 * @author Name1
 * @role   Primary Coordinator, secondary Recorder
 * @email  name1.here@my.uwrf.edu
 * @author Name2
 * @role   Primary Recorder
 * @email  name2.here@my.uwrf.edu
 * @author Name3
 * @role   Primary Checker, secondary Coordinator
 * @email  name3.here@my.uwrf.edu
 * @date   November 1st 2012
 */
```

Every method should have comments immediately above the method briefly describing how it works and what the preconditions and postconditions of the method are.

```
/**
 * <p>compute</p>
 * <p>Description: This method computes ...</p>
 * @param  int data – used to compute …
 * @require data >= 0
 * @ensure this.count() >= 0 && this.count() <= Integer.MAX_INT / 2
 *
 */
```

Your code must be well documented – add comments explaining (to a peer) how each method works. Submit (upload into the Dropbox for Assignment #2 in D2L) a complete package as a .jar file with documented source code files and Javadoc web-pages. **Late submissions are not accepted**.

References:

1. Information Hiding
2. Nino & Hosch, An Introduction to Programming and Object-Oriented Design using Java. This is the course textbook. Chapters 4 and 5 are especially important for this assignment.