Assignment 2 Comp 1510

Section 1

Due Sunday, November 5, 2016, 8:59 pm in D2L (be careful about daylight savings time). If you complete projects 1-4 before the midterm it will help you learn the material. You are not required to work on the assignment during midterm week.

Read the entire assignment carefully, especially the sections about assignment preparation and Plagiarism / Collaboration.

This is an individual assignment, no partners allowed and no sharing of any code whatsoever.

There are five programming projects, each is worth 10 marks. The projects can and should be solved using the material in chapters 3 and 4 of the text (no loops).

Project 1: (package q1) Write an application, called PhoneNumbers, that creates and prints a random phone number of the form XXX-XXX-XXXX. Include the dashes in the output. The phone number has some constraints. Do not let the first three digits contain an 8 or 9 (but do not be more restrictive than that), and ensure that the second set of three digit is not greater than 635. Note that any of the digits can be zero.

Project 2: (package q2) Write an application, called CylinderStats, that reads the radius and height of a cylinder and prints its surface area and volume. Use the following formulas. Print the output to four decimal places. In the formulas, *r* represents the radius and *h* the height.

Surface area: $2\pi r(r + h)$

Volume: $\pi r^2 h$

Project 3: (package q3) Design and implement a class called Cylinder that contains instance data that represents the cylinder's radius and height. Define a Cylinder constructor to accept and initialize the radius and height, and include getter and setter methods for both instance variables. Include methods that calculate and return the volume and surface area of the cylinder (see preceding problem for formulas). Include a toString method that returns a one-line description of the cylinder. Create a driver class, called MultiCylinder, that instantiates and updates two Cylinder objects, printing their radius and height before and after modification, and prints the final volume and surface area of each cylinder.

Project 4: (package q4) Design and implement a class called Box that contains instance data that represents the height, width, and depth of the box. Also include a boolean variable called `full` as instance data that represents whether the box is full or not. Define a Box constructor to accept and initialize the height, width, and depth of the box. Each newly created box is empty (the constructor should initialize `full` to false). Include getter and setter methods for all instance data. Include a toString method that returns a one-line description of the box. Create a driver, called BoxTest, whose main method instantiates and updates two Box objects. BoxTest should call each Box method at least once and print enough information to indicate that the methods worked correctly.

Project 5: (package q5) Develop a JavaFX GUI application called Email that implements a prototype user interface for composing an email message. The application should have labeled text fields for the To, CC, and Bcc address lists, a subject line, and one for the message body (optionally use a TextArea for the message body – see the JavaFX API at http://docs.oracle.com/javafx/2/api/index.html). Include a button labeled Send. When the Send button is clicked, the program should print the contents of all fields to standard output using println() statements.

Each program is required to conform to the style guidelines in **Section 3**. Each class must have Javadoc comments, as in **Section 3**. For this assignment, the Javadoc comments are required for classes, methods and variables but see **Section 3** for details.

For this assignment, programs may consist of multiple classes, with the names as given above. The classes for project 1 should be in package q1, project 2 in package q2, etc. We will provide an ant script to compile, run, and package your code for submission.

Section 2

Assignment Preparation

In the corresponding assignment template zip file, there is a starting project layout for building and packaging your assignment. You will need to fill in your name in the build.xml file. You then need to fill in the code for each of the classes (we give you a template for the main class, you will need to write all of the others) and run the ant script to compile / package your assignment.

In developing each programming project, you will follow the program development steps as discussed in the text book. Specifically, when the problem requires it, you need to perform all of the following steps:

1. establishing the requirements,
2. creating a design,
3. implementing the code, and
4. testing the implementation.

Assignments are to be submitted in *soft copy* (in D2L). See below for details on the physical requirements for you submission.

Section 3

Comments and documentation

Comment Item: All Classes (including inner classes)
Details:
Do not create comments for the sake of creating comments. Focus is on quality not on quantity. Comments should be succinct and to the point. If you can be brief, then do so. Please use English that is grammatically correct.

Each class must have a Javadoc header block comment immediately before the class statement. This must have the following format:

```
/**
 * Introductory summary sentence describing the class.
 * More complete description of everything the class is supposed
 * to do(may be several lines long.
 *
 * @author name of author of the code and set
 * @version version number, such as 1.0
*/
```

Each paragraph should be separated from the next by a <p> tag. If you need to use lists or tables in the description, you should use HTML tags.

Example header block:

```
/**
```

```
 * Graphics is the abstract base class for all graphics contexts
 * which allow an application to draw onto components realized on
 * various devices or onto off-screen images.
 * A Graphics object encapsulates the state information needed
 * for the various rendering operations that Java supports.  This
 * state information includes:
 * <ul>
 * <li>The Component to draw on
 * <li>A translation origin for rendering and clipping coordinates
 * <li>The current clip
 * <li>The current color
 * <li>The current font
 * <li>The current logical pixel operation function (XOR or Paint)
 * <li>The current XOR alternation color
 *     (see <a href="#setXORMode">setXORMode</a>)
 * </ul>
 * <p>
 * Coordinates are infinitely thin and lie between the pixels of the
 * output device.
 * Operations which draw the outline of a figure operate by traversing
 * along the infinitely thin path with a pixel-sized pen that hangs
 * down and to the right of the anchor point on the path.
 * Operations which fill a figure operate by filling the interior
 * of the infinitely thin path.
 * Operations which render horizontal text render the ascending
 * portion of the characters entirely above the baseline coordinate.
 * <p>
 * All coordinates which appear as arguments to the methods of this
 * Graphics object are considered relative to the translation origin
 * of this Graphics object prior to the invocation of the method.
 * All rendering operations modify only pixels which lie within the
 * area bounded by both the current clip of the graphics context
 * and the extents of the Component used to create the Graphics object.
 *
 * @author      Sami Shaio, Set K
 * @author      Arthur van Hoff, Set J
 * @version     %I%, %G%
*/
```

What Javadoc tags to use:

```
@author
@link (where applicable) - for example when you want to link to superclasses
@literal (where applicable)
@version
<b> ... </b>
<code> ... </code>
<i> ... </i>
<p> ... </p>
<ul> <li> ... </li> ... </ul>
```

Comment Item: Class Variable, Instance Variable (including private, protected, public & default)

What Javadoc tags to use:

```
@literal (where applicable)
<b> ... </b>
<code> ... </code>
<i> ... </i>
<p> ... </p>
<ul> <li> ... </li> ... </ul>
```

Details:
The only variables that you will not comment using Javadoc comments are local variables. Local variables can be commented by using the slash-slash style of commenting.

Comment Item: Method (including private, protected, public & default)
What Javadoc tags to use:

```
@literal (where applicable)
@param (where applicable) - when you have parameters passed to a method.
@return (where applicable) - when you have a return value.
@see (where applicable)
@throws (where applicable) - when you have a method that throws an exception -
Any exception!
<b> ... </b>
<code> ... </code>
<i> ... </i>
<p> ... </p>
<ul> <li> ... </li> ... </ul>
```

Details:
Don't forget to comment all methods and constructors.

Lastly:
More information on javadoc:
http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

As well, for coding style in Java you are required to conform to the document: http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html. Note that some of the style items in this document refer to things not yet covered. You can safely ignore these until they become applicable after we have covered the corresponding constructs (loops, for example). We are making the following modifications and additions for assignment 2:

1. Indentation tab spacing is 4 spaces
2. * is not allowed in import statements, they must refer to a specific class.
3. Trailing comments (those after source code on a line) should not be used.

Quick checklist for your code conventions:

- proper indenting (4 spaces for each indentation level)
- proper Javadoc documentation (as per above)
- proper variable naming conventions
- one Javadoc comment per instance variable, class and method
- comments go before the thing commented.
- only one declaration per line.
- adhere to the 80 column rule: line length must be <= 80 columns

Section 4

Assignment Grading

The grade for this assignment will be assigned on the basis of 10 points for each part.

- up to **2 points** for commenting and following the style guide, and
- up to **8 points** for correctness.

Section 5

Schedule

Your project is due on the assigned date at the assigned time. Late assignments will count as zero – *absolutely no exceptions*. It will be your responsibility to ensure that you've submitted the appropriate files and that you've done so on time. Do not wait until an hour before the assignment is due – you never know if you'll have network trouble.

This policy will be waived only for documented medical situations (not including cold, flu, or simply not feeling well) or other extraordinary circumstances (e.g. war, natural disaster). "The computer was down" is not an unusual circumstance; our response will always be "the computer often goes down; you should have allowed yourself more time."

How to hand in COMP 1510 Assignments

1) You must ensure to fill out the readme.txt file found in the assignment template `src` (source) directory. This file will be bundled up (in your zip file) with your included source code. The readme.txt file should contain the status of your assignment. Please fill it out accurately. Note: *You must ensure that your submitted readme.txt file accurately reflects what you've submitted in your assignment. If the assignment is not complete but your readme.txt file states that it is, you've misrepresented yourself and therefore you will lose marks.*

2) Ensure that you've checked over the report generated by ***Checkstyle***. If it contains errors and you do not fix them, you will lose marks for style (i.e., 0/2). **If you do not understand any of the code conventions, please see your lab instructor well before the assignment is due**.

3) Remember, you are only submitting the zip file which contains the source code files and the readme.txt file. You should run the *ant* command to produce this zip file for submission.

4) *Upload your zip file into your **D2L dropbox** for the assignment. This must be before the due date and time.*

Section 6

Plagiarism and Collaboration on Programming Projects

The assignment you turn in *must* represent your own work and not the work of anyone else. On the other hand, it is unreasonable to expect that you will work in a complete vacuum, without ever speaking to a classmate. The purpose of this section is to give you guidance about the areas in which it is appropriate to discuss assignment topics with your classmates. Violating these guidelines may result in a charge of academic dishonesty.

Plagiarism

The term plagiarism describes an attempt to claim work as your own, which you have copied from another person, whether that other person knows about it or not. In a class like this, plagiarism includes copying program code, data, documentation, etc. Plagiarism is simply not allowed. If you submit another student's work as your own, you will be charged with a violation of the BCIT Academic Integrity Code.

Collaboration

Collaboration is defined as two or more students working together on a phase of a project. Working together does not mean that one student does the work and the other student just copies it! Collaboration is allowed under certain conditions, as long as you are honest about it.

You are taking this class to learn important fundamental things about computing, and we must give you a grade that fairly represents what we think you've learned. Therefore, we need to know that your work is your work, so we need to limit the collaboration somewhat. For purposes of projects in this class, here are some guidelines as to which phases of a project are appropriate for collaboration, and which are inappropriate. This may change from assignment to assignment.

OK	Preliminary analysis of problem
OK	Developing an algorithm
OK	Developing a plan for testing
NO	Coding
NO	Proof-reading the program before compiling
OK	Interpreting errors at compilation time
OK	Interpreting errors at execution time
NO	Testing

Working in pairs

If the assignment explicitly allows or requires people to work in pairs, then both names must appear on the one assignment, which is handed in for the pair. In this case, the rules on collaboration apply to the students in that pair collaborating with anyone else.

Save Your Projects!

You are required to save a copy of all your projects until the end of the semester, after grades have been reported. Be prepared to re-submit these to the instructor if he or she asks you to do so.

Protect Yourself

If you suspect that another student is misusing your work (for example, one of your printouts disappeared), report this immediately to the instructor, to protect yourself against a charge of plagiarism if another student copies your work. If there is ever any confusion as to who copied from who, it is institute policy to charge both with plagiarism and punish both (or all) parties.

Read the BCIT Policy on Conduct carefully.