Assignment 4 Comp 1510

Section 1

Due Wednesday, December 6th, 2017, 9:00 pm in the share-in

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 three programming projects, each is worth 10 marks. The projects can and should be solved using the material through chapter 10 of the text.

Project 1: (package q1) Design and implement an application, called `Statistics` (that's the name of the class), that computes and prints the mean and standard deviation of a list of integers x_1 through x_n. Assume there will be no more than 50 input values and that the values are valid integers. Read the values into an array from System.in with no prompt, terminating when there is no more data. Test using file redirection (or you can use the keyboard to enter data and the operating system-dependent end of file marker to stop entering input). For example, with data values of 4, 9, 11, 12, 17, 5, 8, 12, 14, the standard deviation will be 4.18. Compute both the mean and standard deviation as floating-point values, using the following formulae:

$$\text{mean} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\text{sd} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - mean)^2}{n-1}}$$

If you're still confused, a useful link may help:

http://www.gcseguide.co.uk/standard_deviation.htm

Project 2: (package q2) Design and implement an application that draws an equilateral triangle using a rubber banding technique. The triangle size and orientation are determined by a mouse drag. Use the original mouse press location as the center, and the dragged position of the mouse as one of the corners. Draw a filled circle at the center. Call the main program `DrawTriangle`.

Hint: JavaFX has a Point2D class that may be helpful.

Project 3: (package q3)**:** The MIX computer (Knuth, *The Art of Computer Programming, Volume 1*) has a 56 characters, with numeric values 0 .. 55 as in the following table:

Value	Char	Value	Char	Value	Char	Value	Char
0	space	14	M	28	Y	42	(
1	A	15	N	29	Z	43	)
2	B	16	O	30	0	44	+
3	C	17	P	31	1	45	-
4	D	18	Q	32	2	46	*
5	E	19	R	33	3	47	/
6	F	20	Σ	34	4	48	=
7	G	21	Π	35	5	49	$
8	H	22	S	36	6	50	<
9	I	23	T	37	7	51	>
10	Δ	24	U	38	8	52	@
11	J	25	V	39	9	53	;
12	K	26	W	40	.	54	:
13	L	27	X	41	,	55	'

Define a class, `MIXChar` to represent a single `MIXChar` character. The class should be designed with the following methods (minimum):

1. `static boolean isMIXChar(char c)` returns true if c corresponds to a MIXChar character, false otherwise
2. `static MIXChar toMIXChar(char c)` converts c to the corresponding MIXChar, with exception thrown if conversion not possible
3. `static char toChar(MIXChar x)` converts MIXChar character to corresponding Java char
4. `static String toString(MIXChar[])` returns String with characters corresponding to those of the input array

5. `static MIXChar[] toMIXChar(String s)` returns array of MIXChar characters corresponding to the chars in s. Throws exception if any if the string's characters do not correspond to MIXChar characters
6. `static long[] encode(MIXChar[] m)` returns an array of long's holding the values of the m packed 11 MIXChar characters per long.
7. `static MIXChar[] decode(long[] l)` returns an array of MIXChar characters corresponding to unpacking the input array, assuming 11 characters are packed per long, with the last long possibly having fewer than 11 characters
8. `int ordinal()` returns numerical value of this MIXChar
9. `String toString()` returns string containing this MIXChar as a Java char.
10. static long[] encode(MIXChar[]) converts the MIXChar array into a long array by packing 11 MIXChar characters in each long, with the last long element having perhaps fewer than 11 characters packed into it
11. `static MIXChar[] decode(long[] l)` returns a MIXChar array with the characters unpacked from the long array
12. `static void main(String[] args)` reads a line from the input, and prints an error message if any characters are invalid MIX characters (i.e. not in the above table). If all the characters are MIX characters, convert them to have the correct numerical values, then encode them into a long[] and print out the long[]. Then decode the long[], convert the MIXChar characters to a Java string and print out the resulting Java String.

Hint: The packing is essentially a base conversion calculation converting an 11 "digit" number to binary, where a digit is a value between 1 and 55. To fit 11 MIXChars into a long, you will need to use unsigned arithmetic, which can be handled with methods in the Long class. You may find helpful Long.toUnsignedString(), Long.remainderUnsigned(), Long.divideUnsigned() among other things. To do the packing, all you need is * and +.

Your main program should result in the same string after encoding following by decoding. A good test is an input string of 50 single quotes.

Eclipse tip: to show the Greek letters on the Eclipse console, go to the common tab of the run configuration for your program and select UTF-8 for the Encoding.

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 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.

For the first assignment, the problems are simple enough that you are only asked to submit the implementation code, which you must have thoroughly tested.

Assignments are to be submitted in *soft copy* (in D2L). See below for details on the physical requirements for your 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>
 * Some important points to consider are that drawing a figure that
 * covers a given rectangle will occupy one extra row of pixels on
 * the right and bottom edges compared to filling a figure that is
 * bounded by that same rectangle.
 * Also, drawing a horizontal line along the same y coordinate as
 * the baseline of a line of text will draw the line entirely below
 * the text except for any descenders.
 * Both of these properties are due to the pen hanging down and to
 * the right from the path that it traverses.
 * <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 including constructors.

Lastly:
Tutorial on javadoc: http://java.sun.com/j2se/javadoc/writingdoccomments/

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

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 variable, class and method
- comments go before the thing commented. .
- only one declaration per line.
- adhere to the 100 column rule <= 100 columns

Section 4

Assignment Grading

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

- up to **3 points** for commenting and following the style guide, and
- up to **5 points** for class correctness
- up to **2 points** for program correctness (basically the "main" method)

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 `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 for each of the questions (e.g., Q1 is 100% complete, Q2 is 95% complete and so on). 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 for this.*

2) Ensure that your code follows all rules of style in the style document, `http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html, with the modifications described above.` **If you do not understand any of the code conventions, please see your lab instructor well before the assignment is due**.

3) Ensure that your `javadoc` documentation follows the rules found in this document. For more help on `javadoc` documentation, go to: `http://java.sun.com/j2se/javadoc/writingdoccomments/` or ask your lab instructor.

4) All of your source code should be contained in a `src` directory. The source code files will be in a subdirectory of `src` corresponding to the package (i.e., src\q1, src\q2, src\q3). You must provide all of the required files, even if they are not complete.

5) **You will lose more marks if your assignment does not compile.** If you do not have time to complete one of the questions or a portion of the question, in the very least ensure that it compiles.

6) After the programs are working, run the ant script to create the zip file. You create the proper name for the zip file by editing the build.xml file to set your student name and version. We will mark the highest version.

7) Before uploading your zip file please ensure that all of the source files and readme are present. Then upload your file into D2L in your dropbox for assignment 4. 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.