Assignment 1 Comp 1510

Section 1

Due Sunday, October 8th, 2017, 9:00 PM in the D2L dropbox.

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 should be solved using the material in chapters 1 and 2 of the text, unless material from chapter 3 is needed.

Project 1: Write an application that prompts for and reads a `double` value representing a monetary amount. Then determine the fewest number of each bill and coin needed to represent that amount, starting with the highest (assume that a ten-dollar bill is the maximum size needed, and that pennies exist). For example, if the value entered is 47.63 (forty-seven dollars and 63 cents), then the program should print the equivalent amount as:

```
4 ten dollar bills
1 five dollar bills
1 two dollar coin
2 quarters
1 dimes
0 nickles
3 pennies
```

Hint: if you are having problems with floating point errors giving you incorrect results, do the calculations in integer pennies.

Call the class `Change` and put it into package `q1`.

Project 2: Write an application that reads an integer number in from the user via the prompt. That integer value is a value in seconds. Convert the seconds into hours, minutes and seconds in the format `hh:mm:ss`. For example, 5322 seconds would be 1:28:42. Call the class `SecondsConvert` and put it into package `q2`.

Project 3: Write an application that reads two floating point numbers and prints their sum, difference, quotient, and product. Call this class `Arithmetic` and put it into the `q3` package. Be sure to try the cases where one or the other input (or both) is zero.

Project 4: Write an application that prompts for and reads an integer representing the length of a cube's side, then prints the cube's volume and surface area. Call the class `Cube` and put it into package `q4`.

Project 5: 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	'

Write an application that has 5 `final char` variables, called `c1, c2, c3, c4, c5` which contain characters, with values chosen from the range `A` to `I`, and encodes them all as one `int` value, prints the original 5 characters, followed by the encoded value. The program should then decode the value back into the 5 characters, and print the decoded characters. Call the class `Pack` and put it into package `q5`.

The output format should look as follows (for the input IFEBA):

```
Original: IFEBA
Encoded: 89579953
Decoded: IFEBA
```

The logic should proceed as follows:

1. Convert each character to its numeric value. This can be done by using the formula (valid for `'A'-'I'`) *value = character – 'A' + 1* (arithmetic can be done on Java Unicode characters)
2. Use the 5 values as "digits" in a base 56 number and calculate the resulting value (the "digit corresponding to c1 being the most significant "digit"). This can be done by alternating multiplication by the base and adding the next digit.
3. Decode the value by using remainder to extract the least significant digit and integer division to move all the remaining digits down one place.

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, each program will consist of a single class, with the name as given above.

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 templates) 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.

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.

Since a significant part of the assignment is correct style and program usage, a utility called checkstyle is part of the package. You should be sure to run checkstyle frequently and fix any problems as you are coding (after you complete each section of code). If you wait until you are done, you may be overwhelmed by the number of errors.

You should also make backup copies often, so if you lose your code, the previous backup is useful to you.

The zip file produced by the ant program are to be submitted in the D2L dropbox for the assignment. 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 (this shows possibilities, you will not need to be so elaborate):

```
/**
 * 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 in assignment 1:

```
@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: How to Write Doc Comments with Javadoc

As well, for coding style in Java you are required to conform to the checkstyle configuration included in the assignment template. Reasons are given in Sun Java Coding Conventions.

Quick checklist for your code conventions:

- Proper indenting (4 spaces for each indentation level - don't use tabs).
- 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 80 column rule <= 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 provide a readme.txt file in the `src` (source) directory. This file will be bundled up (in your zip file produced by running the ant script) with your included source code. The readme.txt file should contain the status of your assignment, for each problem whether it is complete or, if not, what errors it has. 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 for this.*
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 build by the ant script, which contains the source code files and the readme.txt file. Although Ant provides this service to you automatically, it is always a good idea to check and ensure that the zip file contains the correct files.
4) Upload your file into the D2L dropbox. This must be before the due date and time. The naming convention you must follow for naming your zip file is: lastname,firstname-assign-1.zip

   example: ferguson,arron-assign-1.zip

   If you do not use this exact format, your instructor will change it for you for a small fee: 5 marks out of your 50 marks for the assignment.

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.