Assignment 1 Comp 1510

Section 1

Due Sunday, September 29, 2019, 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 absolutely required.

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 toonies
0 loonies
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 (do

not worry about leading zeroes in the numbers). 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 encodes a 5 MIX-character string into a single int variable and decode an int value back into MIX-characters. Read the string from the user and print the encoded number and decoded string. The MIX-characters are restricted to the range from A to I (inclusive)

You may want to use 5 `char` variables, called `c1, c2, c3, c4, c5` to hold the five characters.

Call the class `Pack` and put it into package `q5`.

The output format should look as follows (given Input: `IFEBA)`:

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

The logic should include the following:

1. Prompt for and read the input string into a variable, `input`.
2. Use the string method `input.charAt(n)` to extract the n^{th} character from the string, where n is 0, 1, 2, 3, 4.
3. Convert each character to its numeric value. This can be done by using the formula (valid for `'A'-'I'`) *value = character – 'A' + 1* (note arithmetic can be done on Java Unicode characters)
4. 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.
5. Print the encoded integer.
6. Decode the value by using remainder to extract the least significant digit and integer division to move all the remaining digits down one place.
7. To get each digit, `d`, back to a character, use the formula `(char)(d - 1 + 'A')`.
8. Print the five decoded characters.

For all projects, each program is required to conform to the style guidelines in **Section 3**. In particular, each class, the main method and variables/constants declared outside the main method must have Javadoc comments, as in **Section 3**.

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 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 install checkstyle in your IDE and fix any problems as you are coding. 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 is 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

Item: All 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 want to use lists or tables in the description, you should use HTML tags.

What Javadoc tags to use for classes in assignment 1:
`@author`
`@version`

Item: Class level variable

Details:
The only variables that you will not comment using Javadoc comments are variables in the main method (local variables). Local variables *can* be commented by using the slash-slash style of commenting, but are not required to be commented if the variable name is clear enough.

Comment Item: Method
What Javadoc tags to use:
`@param when you have parameters passed to a method (see lecture examples).`

Details:
The only method in assignment 1 is the main method.

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.

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*** by ant. If it contains errors and you do not fix them, you will lose marks for style (i.e., 0-2 out of 10 for each project). **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: bloggs,fred-assign-1.zip

   If you do not use this exact format, the marker *may* 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, if 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 own 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.