CS1073 Assignment #7 - Fall 2020

Submission Deadline: Friday, October 30th before 12:00 NOON (Atlantic Daylight Time Zone) in the Assignment 7 dropbox in Desire2Learn. (Read the submission instructions at the end of this document carefully).

The purpose of this assignment is to review/gain practice with:

- decision statements,
- the `while` loop,
- the `Math` class,
- and arithmetic in Java.

This assignment is to be done individually. If you have questions, direct them to a tutor/assistant during a help session in the "Faculty of Computer Science Student Success Centre" team or to your course instructor.

1. Fencing App:

Suppose you are writing software for a fencing company that designs fenced corrals in two different configurations: rectangles, and regular polygons. A regular polygon has all sides of equal length. They're interested in knowing how much each configuration costs and the area of land it encloses. Their fences are all built to a standard height, so they know the unit cost is $9.50/m of fencing.

a) Begin by writing two classes: RectangularCorral (to represent a rectangular fenced corral), and PolygonalCorral (to represent a fenced corral in the shape of a regular polygon).

Continued on next page…

For each RectanglularCorral, we must record its length and width. (Note: The length and width will both be positive numbers. You may assume that valid data will be provided when the object is constructed; you do not need to insert checks here to validate the data.) We also know that the unit cost of the fencing is $9.50/m; this is a fixed value (it will never change). Include 2 instance variables, a constant, and an appropriate constructor method. You should also have an accessor method to retrieve the length, another to retrieve the width, and one to retrieve the unit cost. In addition, include 2 more methods: one to calculate and return the cost of the fencing, and another to calculate and return the area of land the corral encloses.

For each PolygonalCorral, we must record the length of the side and the number of sides the enclosure will have. (Note: Every PolygonalCorral will have a minimum of 3 sides, and the length of each side will be a positive number. You may assume that valid data will be provided when the object is constructed; you do not need to insert checks here to validate the data.) We also know that the unit cost of fencing is $9.50/m; this is a fixed value (it will never change). Include 2 instance variables, a constant, and an appropriate constructor method. You should also have an accessor method to retrieve the side length, another to retrieve the number of sides, and one to retrieve the unit cost. In addition, include 2 more methods: one to calculate and return the cost of the fencing, and another to calculate and return the area of land the corral encloses.

Note: To calculate the area of a regular polygon requires the use of trigonometric functions. Use radians in your calculations and the value of π that is defined in the Math class.

Write Javadoc comments for your RectanglularCorral and PolygonalCorral classes. Include a comment for each class, for each instance variable, each constant and each method. Use @author, @param & @return tags where appropriate. Run the Javadoc utility on your files and view the resulting RectanglularCorral.html and PolygonalCorral.html files in a browser to make sure that your Javadoc comments were inserted/formatted correctly. (Be sure to include author and private information when generating the documentation with Javadoc.)

Question 1 part b) begins on the next page…

b) Next, write a driver program that reads in the dimensions in metres, and/or number of sides for different fence configurations and displays the cost and area of each. After reading in the information for all the corrals, the program should indicate whether the corral with the largest area is a rectangle or a regular polygon, followed by its area. NOTE: each area should be displayed with exactly 3 digits to the right of the decimal, and each cost should be displayed as a currency.

Begin by presenting the user with a list of options, and keep looping until the user indicates that they would like to quit. You may assume that the user will provide valid input; you do not need to provide checks to validate the input. Below & continued on the next page is sample output to show you how your program should work (note: user input is shown here in green and italics):

```
What would you like to do?
1 - Get info for rectangular enclosure
2 - Get info for polygon enclosure
3 - Quit
Enter your choice: 2
Length (in m): 5.7
Number of sides: 6
The area is: 84.411 square metres.
The cost is: $324.90

What would you like to do?
1 - Get info for rectangular enclosure
2 - Get info for polygon enclosure
3 - Quit
Enter your choice: 1
Width (in m): 8.3
Length (in m): 10.0
The area is: 83.000 square metres.
The cost is: $347.70

What would you like to do?
1 - Get info for rectangular enclosure
2 - Get info for polygon enclosure
3 - Quit
Enter your choice: 2
Length (in m): 6.75
Number of sides: 8
The area is: 219.995 square metres.
The cost is: $513.00
What would you like to do?
```

```
1 - Get info for rectangular enclosure
2 - Get info for polygon enclosure
3 - Quit
Enter your choice: 1
Width (in m): 4.6
Length (in m): 3.25
The area is: 14.950 square metres.
The cost is: $149.15

What would you like to do?
1 - Get info for rectangular enclosure
2 - Get info for polygon enclosure
3 - Quit
Enter your choice: 1
Width (in m): 12.0
Length (in m): 0.75
The area is: 9.000 square metres.
The cost is: $242.25

What would you like to do?
1 - Get info for rectangular enclosure
2 - Get info for polygon enclosure
3 - Quit
Enter your choice: 3
The corral with the largest area is a polygon.
Its area is: 219.995 square metres.
```

After you have tested your application (with several different input values) and you're sure that it works properly, save sample output for submission. Include at least 3 corrals and at least 1 of each of the different configurations (rectangular and regular polygon) in the sample output you submit.

For this question, take a snapshot of the terminal window to include in your .zip file. That way, the marker will be able to see the input values as well. Since the output may be rather long, you may need multiple screen captures to get it all. For your assignment report, you may copy and paste the output from the terminal window into your report (again, this will ensure that the input values are also included.)

Remember that you should include a Javadoc comment at the top of your driver class for part 1. This comment should include a one-line description of the class and @author information.

2. Arabic to Ancient Egyptian Number System

Ancient Egyptian civilizations used hieroglyphs to represent their number system. The following table shows the Arabic equivalent to the Egyptian hieroglyph:

Symbol Description	Keyboard Symbol	Hieroglyph	Value
Egyptian god	W		1000000
Frog	&		100000
Bent Finger	)		10000
Lotus Flower	*		1000
Coil of Rope	@		100
Heel Bone	n		10
Staff	|		1

Source: http://mathstat.slu.edu/escher/index.php/History_and_Numbers

When given a hieroglyph, the value of each symbol is added up to indicate the Arabic value.
When there are multiples of the same hieroglyph, then they are positioned according to the following pattern:

```
|    | |    | | |    | | | |    | | |    | | |    | | | |    | | | |    | | |
                               | |      | | |    | | |      | | | |    | | |
                                                                       | | |
```

The hieroglyphs are drawn starting with the largest place value, and each smaller place value written below. Your program should print the pattern of the hieroglyphs using loops and/or conditional statements – do not "hard-code" the print out for each pattern 1-9. (Hint: use the pattern that numbers 1,2,3,5,6,9 allow up to 3 symbols on each line, and 4,7,8 allow up to 4 symbols on each line.)

Write a program that prompts the user for a number between 1 and 9999999 (inclusive) and then prints out the number in Egyptian hieroglyphs.

Continued on next page…

Sample Output (note: user input is shown in green and italics):

```
Please enter a number between 1 and 9999999:  45206

45206 in Egyptian hieroglyphs is:
) ) ) )
* * *
* *
@ @
| | |
| | |
```

If the user enters a number outside of the acceptable range (i.e. less than 1 or greater than 9999999), your program should not try to convert it to hieroglyphs. Instead, it should print out a message and prompt the user for a new input value. For example:

```
Invalid input.  You must enter a number between 1 and 9999999.

Please enter another number now:
```

Do this repeatedly until a valid input value is provided.

After you have tested your application (with several different in put values) and you're sure that it works properly, save sample output from running the program 3 times (3 different test cases). Include both valid input and invalid input. For this question, take a snapshot of the terminal window to include in your .zip file. That way, the marker will be able to see the input values as well. For your assignment report, you may copy and paste the output from the terminal window into your report (again, this will ensure that the input values are also included.)

Submission instructions are on the next page…

Your electronic assignment submission (submitted via Desire2Learn) will consist of two files:

i. a written report. This should begin with a title page; just as we described in Assignment #1, your title page should include: the course (CS 1073), your section (FR01A, FR02A, FR03A or FR04A), the assignment number, your full name, and your UNB student number. That should be followed by four sections, with each part clearly identified with a section heading. Include:

   a. the source code for Part 1,
   b. the sample output for Part 1,
   c. the source code for Part 2, and
   d. the sample output for Part 2.

   (Aside: Your source code should contain all of the Javadoc comments mentioned above. However, you do not need to include the .html files.)

   This written report should be prepared using a word processor; we recommend using Microsoft Word (i.e. create a .docx file for your report). Copy & paste your java source code & required output into the report document. Add appropriate headings for each part. Fix up the formatting where necessary, adjusting line breaks & page breaks to ensure that your document is easy to read. Use a monospaced font for your code to maintain proper indentation.) Once the report is complete and you've checked it all over, save the .docx file for your own records, and then **save a second copy in pdf format for submission**. (Note: Be sure to open that file in a pdf viewer to verify that the pdf was generated correctly.) The **SINGLE pdf file** containing your report will be submitted to the appropriate assignment drop box on Desire2Learn. (It is important that you submit a pdf file and NOT the original Word document. This pdf will allow the marker to write comments directly on your work to give you better feedback.)

   Note: Please name this report as follows: **YourName_As7_Report.pdf**

ii. an archive file (**.zip**) that contains your Java source code and output for this assignment. Make sure that your archive includes all .java files (in case the marker wishes to compile & run your code to test it). You should not include the report document or the .class files in your archive. This archive should be submitted as a **single file** to the appropriate drop box on Desire2Learn.

   Note: Please name this archive file as follows:
   YourName_As7_Archive.zip