CPSC1012 Advanced Portfolio

Weight: 25% of your final mark

Introduction

In the advanced portfolio, you will complete **5 of the 6** programming questions. Each programming question is marked out of 5 marks. The marking rubric for each question is as follows:

Mark	Description
5	Excellent – program passes all test cases and coding follows best practices and class standards
4	Very Good – program passes all test cases, but coding does not follow best practices and class standards
3	Acceptable – coded all the requirements and program produce the expected results for some of the test cases
2	Needs Work – coded all the requirements but program fails to produce expected results
1	Unsatisfactory – coded less than 50% of the requirements
0	Not done.

Requirements

- Create a separate Console application for each question.
- The name of your each Console application must be in the format **AP-Question#-*YourName***., (eg: AP-Question1-CodeKing)
- Each question must be demonstrated to your instructor before you submit a compressed (zip) copy of the Console application Visual Studio solution folder to Moodle.

Programming Questions

1. **Acetaminophen class**

   Design a class named **Acetaminophen** that contains:

   - A `int` property named **`Age`** in years for the children (default **2**).
   - A `double` property name **`Weight`** in pounds for the children (default **24**).
   - A no-argument constructor that creates a default children Acetaminophen.
   - A constructor that creates a children Acetaminophen with the specified age and weight.
   - The mutator for **`Age`** will check if the new value is between 2 and 11 before using the new value otherwise it will throw an exception.
   - The mutators for **`Weight`** will check if the new value is between 24 and 95 before using the new value otherwise it will throw an exception.
   - A method named **`LiquidDosageByWeight()`** that returns as a `double` the dosage of Acetaminophen in ml.
   - A method named **`LiquidDosageByAge()`** that returns as a `double` the dosage of Acetaminophen in ml.
   - The single oral dose is:

Weight (lbs)	Age (years)	Liquid Dosage (ml)
24 – 35	2 – 3	5
36 – 47	4 – 5	7.5
48 – 59	6 – 8	10
60 – 71	9 – 10	12.5
72 – 95	11	15

Figure 1 – Single Oral Dose

Write a program to test your **Acetaminophen** class as shown in the sample run:

```
****************
* Fever Control *
****************
Enter the children's weight in pounds: 20
Invalid weight for children acetaminophen. Weight must be between 24 and 95 lbs.
Enter the children's weight in pounds: 100
Invalid weight for children acetaminophen. Weight must be between 24 and 95 lbs.
Enter the children's weight in pounds: 35
Enter the children's age in years: 1
Invalid age for children acetaminophen. Age must be between 2 and 11.
Enter the children's age in years: 12
Invalid age for children acetaminophen. Age must be between 2 and 11.
Enter the children's age in years: 4
Weight: 35 lbs, Age: 4 years old
Is the information above about the children correct (Y/N): Y
Dosage by weight (35lbs) is: 5 ml
Dosage by age (4 years) is: 7.5
```

2. **Carpet Calculator.**

Write an application that calculates the price of carpeting for rectangular rooms. To calculate the price, you multiply the area of the floor (width times length) by the price per square foot of carpet. For example, the area of floor that is 12 feet long and 10 feet width is 120 square feet. To cover the floor with carpet that costs $8 per square foot would cost $960. (12 x 10 x 8 = 960.)

First, you should create a class named **`RoomDimension`** that has two fields: one for the length of the room and one for the width. The **`RoomDimension`** class should have a method that returns the area of the room. (The area of the room is the room's length multiplied by the room's width.)

Next you should create a **`RoomCarpet`** class that has a **`RoomDimension`** object as a field. It should also have a field for the cost of the carpet per square foot. The **`RoomCarpet`** class should have a method that returns the total cost of the carpet.

Figure 1 is a UML class diagram that shows possible class designs and the relationships among the classes. Once you have written these classes, use them in an application that ask the user to enter the dimensions of a room and the price per square foot of the desired carpeting. The application should display the total cost of the carpet.

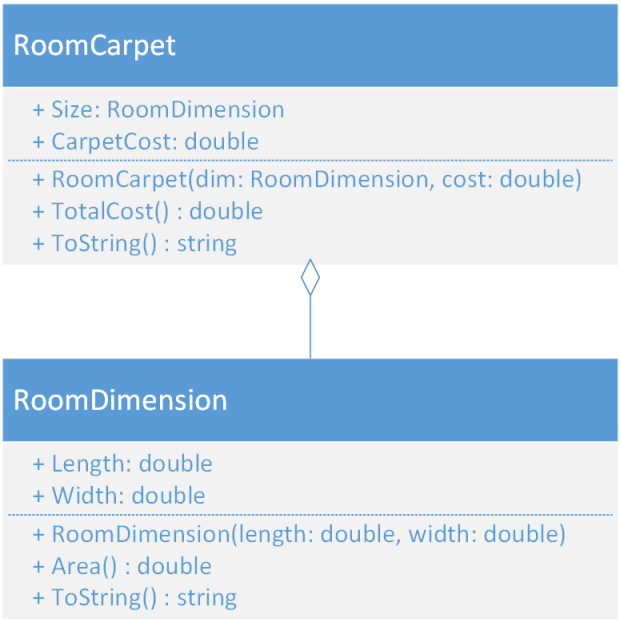

Figure 1

Here is a sample run:

```
**********************
* Carpet Calculator *
**********************
Enter the room length in feet: 12
Enter the room width in feet: 10
Enter the carpet cost per square feet: 3.99
Length = 12, Width = 10, Area = 120
The total cost of the carpet is $478.80
```

3. **Trivia Game Administration**

Write a program that allows an administrator to manage the questions for a trivia game. When the program is run, it should check to see if a data file exists. If the data file exists, then the trivia questions should be loaded from the data file into an array/list in memory. If the data file does not exist, start the program with no trivia questions in memory. The program should then present a menu that allows the administrator to list all trivia items (question, answer, and value) in the database, add a new trivia item, or delete an existing trivia item. Upon exiting the program, the trivia data in memory should be stored in a text file.

You are to design a **`TriviaQuestion`** class to hold the data for a trivia question. The **`TriviaQuestion`** class should have `string` fields for the following data:

- A trivia question
- A list of four possible answers
- The number of the correct answer (1, 2, 3, or 4)

The **`TriviaQuestion`** class should have appropriate constructor(s) and fully implemented properties.

```
*****************************
* Trivia Game Administration *
*****************************
1.  List Trivia Items
2.  Add Trivia Item
3.  Delete Trivia Item
4.  Quit
Your choice: 1
There are currently no items in the database.

*****************************
* Trivia Game Administration *
*****************************
1.  List Trivia Items
2.  Add Trivia Item
3.  Delete Trivia Item
4.  Quit
Your choice: 2
Enter the trivia item question:
Who is the current premier of Alberta?
Enter answer #1:
Rachel Homan
Enter answer #2:
Rachel McAdams
Enter answer #3:
Rachel Notley
Enter answer #4:
Rachel Ray
Enter the correct answer choice (1-4):
3

*****************************
* Trivia Game Administration *
*****************************
1.  List Trivia Items
2.  Add Trivia Item
3.  Delete Trivia Item
```

```
4.  Quit
Your choice: 3
Item#       Question
-----       --------------------------------------------------------
1           Who is the current premier of Alberta?
2           Who is the current mayor of Edmonton?
3           Who is the current prime minister of Canada?
Enter the item number to delete:
2
Item #2 has been deleted from the database.

*****************************
* Trivia Game Administration *
*****************************
1.  List Trivia Items
2.  Add Trivia Item
3.  Delete Trivia Item
4.  Quit
Your choice: 1
1.  Who is current premier of Alberta?
    1.  Rachel Homan
    2.  Rachel McAdams
    3.  Rachel Notley
    4.  Rachel Ray
Current answer: 3

2.  Who is current prime minister of Canada?
    1.  Jean Chretien
    2.  Justin Trudeau
    3.  Paul Martin
    4.  Stephen Harper
Current answer: 2

*****************************
* Trivia Game Administration *
*****************************
1.  List Trivia Items
2.  Add Trivia Item
3.  Delete Trivia Item
4.  Quit
Your choice: 4
The trivia items has been saved to the file “C:\Intel\TriviaQuestons.csv”.
Good-bye.
```

4. **Trivia Game**

Write a program that plays a simple trivia game for two players. The game will read the trivia items (question, answer, and value) from the text file that was created using the Trivia Game Administrator program. The program will work like this:

- When the program is run, it should check to see if a data file exists. If the data file exists, then the trivia questions should be loaded from the data file into an array/list in memory. If the data file does not exists, the program will exit.
- The game will evenly split the questions between the two players.
- Starting with player 1, each player gets a turn at answering the trivia questions. When a question is displayed, four possible answers are displayed. Only one of the answers is correct, and if the player selects the correct answer, he or she earns a point.
- After answers have been selected for all the questions, the program displays the number of points earned by each player and declares the player with the highest number of points the winner.

You are to design a **`TriviaQuestion`** class to hold the data for a trivia question. The **`TriviaQuestion`** class should have `string` fields for the following data:

- A trivia question
- A list of four possible answers
- The number of the correct answer (1, 2, 3, or 4)

The **`TriviaQuestion`** class should have appropriate constructor(s) and fully implemented properties.

```
***************
* Trivia Game *
***************
There are 11 questions in the question bank. Each player will answer 5 questions.

Player 1 Question
------------------
Which Canadian province currently has the highest minimum wage?
a.  Alberta
b.  British Columbia
c.  Ontario
d.  Quebec
Your answer: B

Player 1 Question
-----------------
Who is the current premier of Alberta?
a.  Rachel Homan
b.  Rachel McAdams
c.  Rachel Notley
d.  Rachel Ray
Your answer: B

...
Player 2 Question
-----------------
Who is the current mayor of Edmonton?
a.  Bill Smith
```

```
b.  Don Iveson
c.  Jan Reimer
d.  Stephen Mandel
Your answer: B

Player 2 Question
-----------------
Who is the current prime minister of Canada?
a.  Jean Chretien
b.  Justin Trudeau
c.  Paul Martin
d.  Stephen Harper
Your answer: B

Player 1 has 3 points. Player 2 has 5 points. Player 2 is the winner.
```

5. **Username and Password Verifier**

You are the system administrator for the company responsible for creating user accounts for employees. The company requires that account username and password meet the following criteria:

- The username is **case insensitive**, between 3 to 15 characters, and contain only alphanumeric characters (letters A-Z, numbers 0-9) and the underscore character (_).
- The password is **case sensitive**, it **must not contain your username**, and contain the following:
  - Between 8 to 80 characters long (**no spaces**)
  - At least one upper case character (**`A-Z`**)
  - At least one lower case character (**`a-z`**)
  - At least one number (**`0-9`**)
  - At least one special character (**`!@#$%^*_=+.?-`**)

Design a class named **UserAccount** that contains:

- A `string` property named **`Username`** of the account (default user1).
- A `string` property name **`Password`** of the account (default P@ssword1).
- A no-argument constructor that creates a default UserAccount.
- A constructor that creates a UserAccount with the specified username and password.
- The mutator for **`Username`** will check if the new value matches the username criteria before using the new value.
- The mutator for **`Password`** will check if the new value matches the password criteria before using the new value.

Demonstrate the class in a program that allows the user to enter a username and a password and then displays a message indicating whether it is valid or not. Here is a sample run:

```
**********************
* New Account Details *
**********************
Enter the username: j
Invalid username format! The username must between 3 to 15 characters.
Enter the username: j@nait.ca
Invalid username format! The username may use only the characters: Aa-Zz, 0-9, -_
Enter the username: jw1
Enter the password: password
Invalid password format! The password must contain at least one lowercase letter, one uppercase
letter, one digit, and one special character (! @ # $ % ^ * _ = + . ? -).
Enter the password: passwordJW1123
Invalid password format! The password cannot contain your username
Enter the password: passwordJW 123
Invalid password format! The password cannot contain spaces
Enter the password passwordJW!23
The following account information has been successfully created.
Username: jw1, Password: passwordJW!23
```

6. **Tic-Tac-Toe Game**

In a game of tic-tac-toe, two players take turns marking an available cell in a 3 x 3 grid with their respective tokens (either X or O). When one player has placed three tokens in a horizontal, vertical, or diagonal row on the grid, the game is over and that player has won. A draw (no winner) occurs when all the cells on the grid have been filled with tokens and neither player has achieved a win. Create a program for playing a tic-tac-toe game.

The program prompts the two players to alternately enter an X token and O token. Whenever a token is entered, the program redisplays the board on the console and determines the status of the game (win, draw, or continue). Here is a sample run:

```
********************
* Tie-Tac-Toe Game *
********************
The cell numbers for the game is shown below.
-------------
| 7 | 8 | 9 |
-------------
| 4 | 5 | 6 |
-------------
| 1 | 2 | 3 |
-------------
Enter cell number (1-9) for player X: 5
-------------
|   |   |   |
-------------
|   | X |   |
-------------
|   |   |   |
-------------
Enter cell number (1-9) for player O: 7
-------------
| O |   |   |
-------------
|   | X |   |
-------------
|   |   |   |
-------------
Enter cell number (1-9) for player X:
...
-------------
| O |   | X |
-------------
| O | X |   |
-------------
| X |   |   |
-------------
Player X wins!
Would you like to play again (y/n)? n
Good-bye and thanks for playing.
```