Programming II

Assignment #3 – Exception Handling, Files & Serialization

Due Date: **Sunday 24th Mar. 2019 - at the start of the class – 8:30am**

Marks/Weight: **30 / 10%**

Purpose: The purpose of this Assignment is to:

- Practice the use of exception handling, files and serialization in C#

References: - Text book **"Visual C# 2017, Intro to Object Oriented Programming"**, Chapter 11 and 14
- Links available under **Week 5** and **Week 6** on eCentennial;

Instructions: Be sure to read the following general instructions carefully:
This assignment should be completed in groups of 3 students. Submit the project **through e-Centennial, Assessments / Assignment**. You must name your Visual Studio solution according to the following rule: ***GroupCode_COMP123_AssignmentNumber***
For Example: **sec006-6_COMP123_02**

Submit your assignment in a **zip file** that is named according to the following rule:
GroupCode_COMP123_AssignmentNumber.zip
Example: **sec006-6_COMP123_02.zip**

Apply the naming conventions for variables, methods, classes, and namespaces:
- *variable names, parameters and fields* – use camelCasing
- *classes, namespaces, methods, properties, enumerations* – use PascalCasing
- *constants:* SNAKE_UPPERCASE

Exercise:

This assignment attempts to model the social network twitter. It involves two main classes: **`Tweet`** and **`TweetManager`**. You will load a set of tweets from a local text file into an array. You will perform some simple queries on this array.

The **`Tweet`** and the **`TweetManager`** classes must be in separate files and must not be in the `Program.cs` file.

1 - The Tweet Class

The Tweet class consist of 9 members, 2 of them static and 7 non-static. You will implement this in Visual Studio according to the UML class diagrams below. A short description of the class members is given below:

Tweet **Class**
Fields
-$ currentId : int
Properties
+ Id : int + From : string + To : string + Message : string + Hashtag : string
Methods
+ Tweet(from : string, to : string, message: string hashtag: string) + ToString() : string +$ Parse(line : string) : Tweet

Field:

0.5 marks 1.1 **`currentId`** – this private field is a class variable, it represents the number to be used in setting the id of this tweet. Set it to 1 by default.

Properties:

All properties are self-explanatory.

0.5 marks 1.2 **`Id`** – this property is an int representing the id of this tweet. It is used to uniquely identify a tweet.

0.5 marks 1.3 **From** – this property is a string representing the originator of this tweet.

0.5 marks 1.4 **To** – this property is a string representing the intended recipient of this tweet.

0.5 marks 1.5 **Message** – this property is a string representing the actual message body of this tweet.

0.5 marks 1.6 **Hashtag** – this property is a string representing the hash tag of this tweet.

Methods:

3 marks 1.7 `public Tweet(string from, string to, string message, string hashtag)` – This public constructor takes four string parameters. This constructor does the following:

a. Assigns the arguments to the appropriate properties.
b. Sets the **Id** property using the class variable **currentId**.
c. After the **Id** property is set, the **currentId** is then incremented so that the next assignment will be unique. (see description of Id above)

2 marks 1.8 `public override string ToString()` – This method overrides the same method of the Object class. It does not take any parameter but return a string representation of itself. You decide on the format for the output. You should also consider outputting only part of the Message. Use the **Substring()** method of the string class to do this.

5 marks 1.9 `public static Tweet Parse(string line)` – This is a public class method that takes a string argument and returns a Tweet object. It is used to create a Tweet object when loading the tweets from a file. The argument represents a single line of input read from the file. This method does the following:

a. Uses the **Split()** method of the string to break up the input into four strings. The default delimiter for the **Split()** method is a space, however in this case the delimiter should be a TAB. To specify an argument, use the following code: `Split(new char[]{'\t'});`
b. Invokes the constructor with the four arguments;
c. Return the result of the above invocation as a new tweet;
d. In case there is an exception when parsing the line, for instance an index out of range exception resulted from a badly formatted line, return the tweet object with `"Invalid"` set to all properties. Use a `try catch` block to handle this by catching the specific exception (index out of range exception);

2 - The TweetManager Class

This class consists of 6 static members. You will also implement this in Visual Studio. A short description of the class members is given below:

TweetManager **Class**
Fields
-$ tweets : Tweet[] -$ fileName : string
Methods
$ «constructor» TweetManager() +$ ShowAll() : void +$ ShowAll(hashtag : string) : void +$ ConverToJson() : void

Fields:

0.5 marks 2.1 **tweets** – this private field is a class variable, it is an array with all the tweets in the system. It is initialized in the static constructor. It is populated in the static constructor.

0.5 marks 2.2 **fileName** – this private field is a class variable, it represents the name of the file that contains all the tweets. It is used in the static constructor to read in the tweets. You will have to set this to the name of file that has the information about the tweets.

Note: A file with tweets to test has been provided and it is called ***tweets.txt***.

Methods:

6 marks 2.3 **static TweetManager()** – This is the static constructor. It does not require any parameter. This constructor does the following:

A static constructor does not take any argument nor does it require any accessibility modifier

a. Initialize the **tweets** field to a new array of tweets
   Challenge: How are you going to figure out the size of the array?
b. Opens the file specified by the filename field for reading (***tweets.txt***)
c. Using a looping structure, it does the following:
   i. Reads one line from the file
   ii. Passes this line to the static **Parse()** method of the **Tweet** class to create a tweet object
   iii. The resulting object is added to the tweets array
   iv. This is repeated until the input from the file is empty (**null**).

2 marks 2.4 `public static void ShowAll()` – This is a public class method that does not take any argument and does not return a value. It displays all tweets.

2 marks 2.5 `public static void ShowAll(string hashtag)` – This is a public class method that takes a string argument that does not return a value. It displays all the tweets with hashtag matching the argument.

This is good example of method overloading, i.e. methods with the same name

4 marks 2.6 `public static void ConvertToJson()` – This is a public class method that serializes the array of Tweets and save them to a file called ***tweets.json***.

3 - Testing

0.5 marks 3.1 In the **Main()** method in the **Program** class, write the code to test **ShowAll()** method of the TweetManager class.

1 mark 3.2 Then ask the user to type a hashtag and print out to the console a list of tweets that matches the hashtag entered by the user by calling the overloaded **ShowAll(string hashtag)** method.

0.5 marks 3.3 Invoke the **ConvertToJson()** method to ensure it works.

3.4 Add a **Console.ReadKey()** to ensure we can see the output to the console

Aspects that will also be evaluated

- Proper identifier casing
- Correct implementation of classes (instance variable declarations, constructors, getter and setter methods)
- Declaring and creating objects, calling their methods, interacting with user, displaying results
- Comments, correct naming of variables, methods, classes
- Friendly input/output