Programming II

Assignment 2 – Implement a Tweet Class

Once you are done with your program, upload it to eCentennial under Assessments / Assignment / Assignment 2

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

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

The Tweet Class

The Tweet class consist of nine members that include two static ones (the members decorated with the **$** symbol). You will implement this and all of the classes in Visual Studio. A short description of the class members is given below:

Tweet Class
Fields
-$ CURRENT_ID : int
Properties
+ «property setter absent» From : string + «property setter absent» To : string + «property setter absent» Body : string + «property setter absent» Tag : string + «property setter absent» Id : string
Methods
+ «constructor»Tweet(from : string, to : string, body : string tag : string) + «constructor»Tweet(from : string, to : string, body : string tag : string, id : string) + ToString() : string **+$** Parse(line : string) : Tweet

The **$** symbol is used to denote that this member belongs to the type rather than a specific object and you must use the type to access that member.

Fields:

1. **CURRENT_ID** – this private field is a class variable, it represents the number to be used in setting the id of this tweet.

Properties:

All of the properties are readonly and are self-explanatory.

1. **From** – this property is a string representing the originator of this tweet. The getter is public and the setter is absent.
2. **To** – this property is a string representing the intended recipient of this tweet. The getter is public and the setter is absent.
3. **Body** – this property is a string representing the actual message body of this tweet. The getter is public and the setter is absent.
4. **Tag** – this property is a string representing the hash tag of this tweet. The getter is public and the setter is absent.
5. **Id** – this property is a string representing the id of this tweet. The getter is public and the setter is absent. This is used to uniquely identify a tweet.

Methods:

1. `public Tweet(string from, string to, string body, string tag)` – 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 **CURRENT_ID**.
   c. After the **Id** property is set, the **CURRENT_ID** is then incremented so that the next assignment will be unique. (see description of Id above)

   This is an example of constructor overloading
2. `public Tweet(string from, string to, string body, string tag, string id)` – This public constructor takes five string parameters. This is called by the `static Tweet Parse(string)` method. This constructor does the following:
   a. Assigns the arguments to the appropriate properties.

Methods:

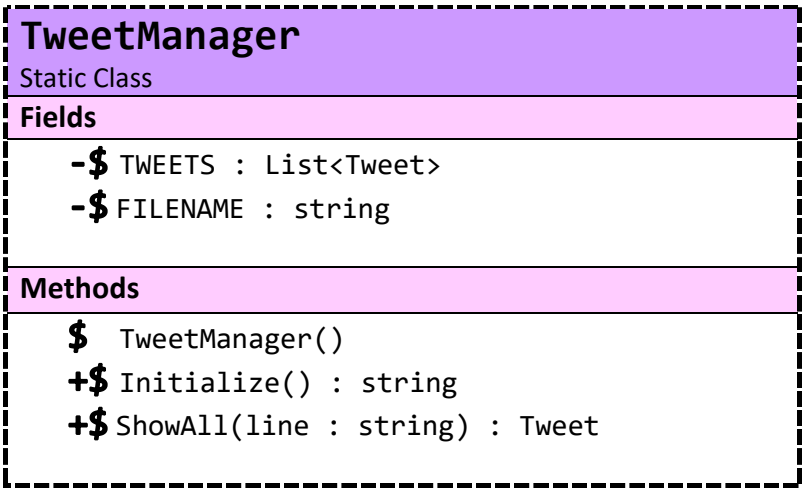
1. `public override string ToString()` – This method overrides the same method of the Object class. It does not take any parameter but return a string representation of itself. You decide on the format for the output.

Because the body of a tweet can be as long as 280 letters, you should limit the size of your output to about 40 letters long.
Check if the length of the body exceeds 40 letter before using the **`Substring()`** method of the string class.

2. **`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 method of the string class is to chunk 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 five arguments. Because all the arguments are string, it is easy to inter-change the order. You need to examine the text file to make sure that you are sending the arguments to the constructor in the required order.
    c. Return the result of the above invocation

The TweetManager Class

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



ALL MEMBERS ARE STATIC!

Fields:

1. **`TWEETS`** – this private field is a class variable; it is a collection of all the tweets in the system. It is initialized and populated in the static constructor.
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 ***Assignment_02_TweetFile.txt***.

> A static constructor does not take any arguments, nor does it require any accessibility modifier.
>
> It is called before any member is accessed

Methods:

1. **`static TweetManager()`** – This is the static constructor. It does not require any parameter. This constructor does the following:
   a. Initialize the **`tweets`** field to a new list of tweets
   b. Opens the file specified by the filename field for reading (***Assignment_02_TweetFile.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 tweet collection
      iv. This is repeated until the input from the file is empty (**`null`**).
2. **`public static void Initialize()`** – This class method it used to facilitate the development of this project. It will not be used in the production code, just while developing. This method does the following:
   a. Assigns the TWEETS field
   b. Creates about 5 tweets objects and add them to the tweet collection.

> This will be used to test your code in the event you cannot figure out the file reading part

3. **`public static void ShowAll()`** – This is a public class method that does not take any argument that does not return a value. It displays all the tweets in the collection.
4. **`public static void ShowAll(string tag)`** – This is a public class method that takes a string argument that does not return a value. It displays all the tweets with tag matching the argument. This comparison must be case in-sensitive.

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

Testing

Write a unit test for each method of the `Tweet` class.