Programming II

# Assignment 2 – Implement a Tweet Class (Groups of two students)

### Due: Demonstration of your code due at the beginning of the last class in Week 4

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 Class

The Tweet class consist of nine members that include two static ones. 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**  + From {get; set;} : string  + To {get; set;}: string  + Body {get; set;}: string  + Tag {get; set;}: string  + Id {get; set;}: string |
| **Methods**  + Tweet(from : string, to : string, body : string tag : 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.

### 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 have public getters and private setters 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 private.
2. **To** – this property is a string representing the intended recipient of this tweet. The getter is public and the setter is private.
3. **Body** – this property is a string representing the actual message body of this tweet. The getter is public and the setter is private.
4. **Tag** – this property is a string representing the hash tag of this tweet. The getter is public and the setter is private.
5. **Id** – this property is a string representing the id of this tweet. The getter is public and the setter is private.

### Methods:

1. **Tweet(string from, string to, string body, string tag)** – This public constructor takes four string parameters. This constructor does the following:
   1. Assigns the arguments to the appropriate properties.
   2. Sets the **Id** property using the class variable **CURRENT\_ID**.
   3. 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)
2. **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 Body. Use the **Substring()** method of the string class to do this.
3. **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:
   1. Uses the **Split()** method of the string class is to chunk the input into four strings. The delimiter is this case is a tab.
   2. Invokes the constructor with the four arguments
   3. 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:

|  |
| --- |
| **TweetManager**  Static Class |
| **Fields**  -$ tweets : List<Tweet>  -$ filename : string |
| **Methods**  $ TweetManager()  +$ Initialize() : string  +$ ShowAll(line : string) : Tweet |

### Fields:

1. **tweets** – this private field is a class variable, it is a collection of all the tweets in the system. It is initialized in the static constructor. It is populated in the
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

### Methods:

1. **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

* 1. Initialize the **tweets** field to a new list of tweets
  2. Opens the file specified by the filename field for reading
  3. Using a looping structure it does the following:
     1. Reads one line from the file
     2. Passes this line to the static **Parse()** method of the **Tweet** class to create a tweet object
     3. The resulting object is added to the tweet collection
     4. This is repeated until the input from the file is empty (**null**).

1. **Initialize()** – This class method it used to facilitate the development of this project. It will not be used in the production code. This method does the following:
   1. Creates about 5 tweets objects and add them to the tweet collection.
2. **ShowAll()** – This is a public class method that does not take any argument that does not return a value. It display all the tweets matching this tag.
3. **ShowAll(string tag)** – This is a public class method that takes a string argument that does not return a value. It display all the tweets matching this tag.

This is good example of method overloading

## Testing

In your test harness (the Main() method in the Program Class), write the code to test all the methods of the TweetManager class except the **Initialize()** method