CS4141 Introduction to Programming Assignment

Part 3 Specification

We want to develop a collection of methods that we can use to manage tweets. We're going to begin with five methods that we think will be particularly useful. Some of the methods are *function methods* (i.e. they return a value) and some are *procedure methods* (.i.e. they perform a particular task).

We are going to organise the method collection into a Java class called MyToolBox.java so that we can use the methods in any future programs that we write. Thus, we will have a single instance, or copy or version or implementation, of the code for each method, in a single location, and will be able to use it in any programming project we undertake. If we need to modify any of the methods, for example to improve it so that it is more efficient or to correct an error in the original implementation, we will only have to alter the code in the MyToolBox.java file and then every program that uses the methods will benefit from the improvements.

The following method headers and descriptions provide the details of each method and code extract examples of their use.

1. `public static String timeOfDay()`

   The method returns a String that contains the current time of day in the format HH:MM:SS, with leading zeros for any value less than 10. For example, the method might return any of the values "12:34:56" or "07:22:06" or "00:00:09" or "23:59:21". Note: You may find the String *format* operation helpful when coding this method.

   Here is a code extract that shows some examples of the method in use

   ```
   // initialise currentTime to the current time on the system clock
   String currentTime = MyToolBox.timeOfDay();

   // Display the time a Tweet was posted

   System.out.println("Posted at " + MyToolBox.timeOfDay());
   ```

2. `public static void tweetHeader(int width)`

   If the value of the `width` parameter is less than or equal to zero then the method outputs nothing.

If the value of the `width` parameter is less than 10 then the method should default to the minimum width of 10.

If the value of the `width` parameter is greater than 80 then the method should default to the maximum width of 80.

The method outputs a header showing the specified width as a pair of 'tens' and 'units' lines followed by a separator line of '=' characters.

The following examples show the method in use and the output that it produce.

```
MyToolBox.tweetHeader(43);
         1     2     3     4
1234567890123456789012345678901234567890123
===========================================
MyToolBox.tweetHeader(100); // uses maximum of 80
         1     2     3     4     5     6     7     8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
================================================================================
MyToolBox.tweetHeader(4); // uses minimum of 10
         1
1234567890
==========
MyToolBox.tweetHeader(-4); // no output, width is < 0
```

3. **`public static void displayTweet(String tweetText, int width)`**

The method displays a Tweet Header followed by the text contained in `tweetText` as a series of lines. Each line contains the number of characters specified by the *width* parameter except, possibly, the last line, which may have fewer characters.

If the length of the `tweetText` parameter is zero then the method outputs nothing.

If the value of the `width` parameter is less than 10 then the method should default to the minimum width of 10.

If the value of the `width` parameter is greater than 80 then the method should default to the maximum width of 80.

The following examples show the method in use and the output that should be produced.

```
MyToolBox.displayTweet("Edsger Dijkstra", 4) // uses minimum of 10
         1
1234567890
==========
Edsg
er D
ijks
tra
String aTweet = "Bono saves but not in Ireland";
MyToolBox.displayTweet(aTweet, 6); // uses minimum of 10
         1
1234567890
==========
Bono s
aves b
ut not
in Ire
land
String anotherPost = "";
MyToolBox.displayTweet(anotherPost, 25) // no output - empty string
String munsterRugby = "Munster 12 The All Blacks 0";
MyToolBox.displayTweet(munsterRugby, 1978); // uses maximum of 80
         1      2      3      4      5      6      7      8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
================================================================================
Munster 12 The All Blacks 0
String parnas = "We must not forget that the wheel is reinvented " +
                "so often because it is a very good idea; " +
                "I've learned to worry more about the soundness " +
```

```
                "of ideas that were invented only once.";
MyToolBox.analyse(parnas,35);
         1     2     3
12345678901234567890123456789012345
===================================
We must not forget that the wheel i
s reinvented so often because it is
 a very good idea; I've learned to
worry more about the soundness of i
deas that were invented only once.
```

4. `public static int countOf(String tweetText, char symbol)`

   If the `symbol` character is not a '#' or a '@' then the method returns a -1 result.

   If the `symbol` character is a '#' or a '@' the method returns a count of the number of times the `symbol` character appears in the text of the `tweetText` parameter. Remember, the count could be zero.

   Note, the `symbol` parameter is of type `char`.

   For example

```
String COPon = "#COP26 Unite Behind The Science @ThePlanet";
int hashCount = MyToolBox.countOf(COPon, '#'); // should return 1
int userCount = MyToolBox.countOf(COPon, '@'); // should return 1
int x = MyToolBox.countOf(COPon, 'U'); // should return -1 (symbol not # or @)
```

5. `public static void analyse(String tweetText, int width)`

   The method displays the `tweetText` using the specified width and shows the '#' and '@' symbol counts.

   The following examples show the method in use and the output produced.

```
String twiterati = "#COP26 Unite Behind The Science @ThePlanet";
MyToolBox.analyse(twiterati,11);
         1
12345678901
```

```
==========
#COP26 Unit
E Behind Th
E Science @
ThePlanet
01 Hashtag
01 Username
String profoundTweet = "#Complexity is not a goal. I don't want " +
                       "to be remembered as an engineer of " +
                       "#complex #systems. @DavidParnas" ;
MyToolBox.analyse(profoundTweet,25);
         1     2
12345678901234567890123 45
==========
#Complexity is not a goal
. I don't want to be reme
mbered as an engineer of
#complex #systems. @David
Parnas
03 Hashtags
01 Username
```

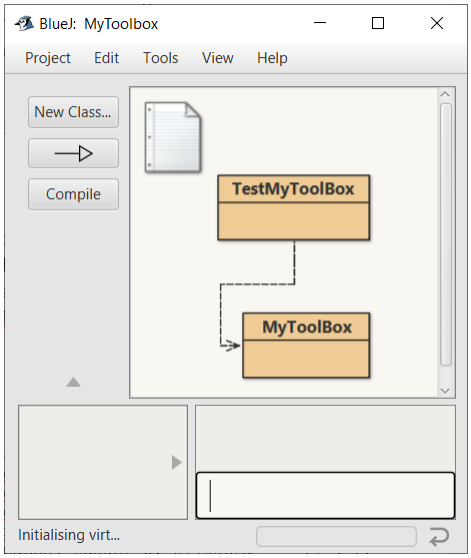
Testing Your Code

To test the methods you should create a "driver" program that uses the methods in the MyToolBox class. Please note, this is just an example.

```
public class TestMyToolBox {
        public static void main(String[] args) {
        // initialise currentTime to the current time on the system clock
         String currentTime = MyToolBox.timeOfDay();
        // Display the time a Tweet was posted
        System.out.println("Posted at " + MyToolBox.timeOfDay());
```

```
        int i = 0, randomWidth = 0;
        // Display 10 random width headers
        for( i= 0; i < 10; i++) {
        randomWidth = (int) (Math.random() * 101); // random width 0...100
         MyToolBox.tweetHeader(randomWidth);
        }
        MyToolBox.displayTweet("Edsger Dijkstra", 4)
        String aTweet = "Bono saves but not in Ireland";
        MyToolBox.displayTweet(aTweet, 6);
        String anotherPost = "";
        MyToolBox.displayTweet(anotherPost, 25)
        String munsterRugby = "Munster 12 The All Blacks 0";
        MyToolBox.displayTweet(munsterRugby, 1978);
        String parnas = "We must not forget that the wheel is reinvented " +
                    "so often because it is a very good idea; " +
                   "I've learned to worry more about the soundness " +
                         "of ideas that were invented only once.";
        MyToolBox.analyse(parnas,35);
        String COPon = "#COP26 Unite Behind The Science @ThePlanet";
        int hashCount = MyToolBox.countOf(COPon, '#'); // should return 1
        int userCount = MyToolBox.countOf(COPon, '@'); // should return 1
        int x = MyToolBox.countOf(COPon, 'U'); // should return -1 (symbol not # or @)
        }
}
```

You should create your "driver" program class and your ToolBox class in the same project. You can add as many classes to a project as you wish but for this assignment you will need only two. The driver program class should be called TestMyToolBox.java and the code for the five methods should be stored in a file called MyToolBox.java. The TestMyToolBox class "uses" the methods in the MyToolBox class. Here is how BlueJ shows the relationship between the code in the MyToolBox class and the driver program TestMyToolBox class.



Submission Requirements

Your solution to the assignment should be submitted on Sulis on or before 16h00 on Monday 22nd November 2021. We will accept submissions up to 23h55 Monday 22nd November 2021 but please note that Sulis identifies submissions that are after the deadline. For this part of the assignment there is no penalty for submitting on or before 23h55.

You should submit TWO files (1) MyToolBox.java and (2) TextMyToolBox.java. In each file you should include prominent comments that contain your ID number and your Name. Sulis will accept any filenames you use but it helps us if you adhere to these conventions.

This part of the assignment is worth 30% of the total of 100%.

Marking Scheme for CS4141/CS6371 Programming Assignment Part 3

Requirement	Marks
`timeOfDay` method	5
`tweetHeader` method	5
`displayTweet` method	5

`countOf` **method**	**5**
`analyse` **method**	**5**
Driver Program	**5**
Total	**30**