Department of Computer Science

08120 Programming 2
Week 28 Practical 2013/2014
Crazy Timers

You are working for a company called "Crazy Timers". They make the countdown timers which you see for things like product launches and the like. They count down the seconds remaining before a particular event. You have been employed to work on the part of the product which asks the user for time information.

Countdown calculator

You have been employed to create a program which will work out the number of seconds there are in a time expressed in hours, minutes and seconds. The user enters the number of hours, minute and seconds and the program then displays the number of seconds equivalent to that time. The program should be used in the following way:

```
Countdown Timer Calculator by Rob Miles
Version 1.0
Enter the number of hours: 3
Enter the number of minutes: 20
Enter the number of seconds: 50
The total number of seconds is: 12050
```

The program has worked out that a time of 3 hours, 20 minutes and 50 seconds is equivalent to 1,2050 seconds. The program uses the facts that there are 60 seconds in a minute, and 3600 seconds in an hour.

Before you write the program you must consider the following issues:

- How you will represent data in the program (the type of the data and the names of the items used to hold the values)
- What the program will do with the data that you give it
- How you will test the program to prove that it works

Now you can create the program itself:

1. Create a program which meets the given specification.

Show a demonstrator the program working and get it signed off on the attendance sheet.

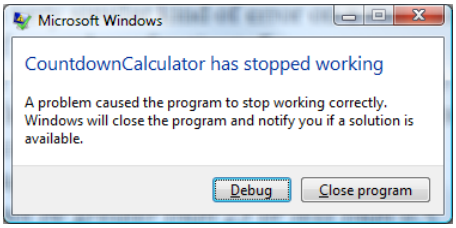
Timer Trouble

Recently you developed a program for "Crazy Timers". The managing director of the firm is looking for you and she is not a happy bunny. If a user enters silly numbers the program does not notice this:

```
Enter the number of hours: 10000
Enter the number of minutes: -99
Enter the number of seconds: 360
```

This would produce a mathematically correct time, but the users don't like the fact that they can mis-type a value and get a wrong answer. They are also upset that the program can be crashed very easily:

```
Enter the number of hours: Har Har
```



The managing director would like to sue you but you offer to change the program for free. You also remind her that, since she did not set any limits on the numbers when you agreed the specification, she is as guilty as you are…You agree the following limits:

The hours cannot be greater than 23 or less than 0.
The minutes cannot be greater than 59 or less than 0.
The seconds cannot be greater than 59 or less than 0

To solve the problem you decide to create a method that will read numbers and validate them.

getValue Method

The getValue method will reject text and only allow values which are between the minimum and the maximum supplied when it is called. The method will also be given a string which will be used as a prompt for the value. It would be used as follows to read in the number of hours:

```
hours = getValue("Enter the number of hours: ", 23, 0);
```

You can use the C# **`if`** construction to test the value of a particular variable:

```
if ( ( intputValue > max ) || (intputValue < min ) )
{
    Console.WriteLine ( "Invalid value" );
}
else
{
    // value is OK
}
```

The code above uses the *logical OR* (which is the two vertical bars **`||`**) to test for the input value being either too big or too small. You will have to use this code inside a loop which will repeatedly read values until a value one is supplied. Your method will also have to catch any exceptions which are thrown if the call of the Parse fails when trying to convert the string entered by the user.

Before you write the program you must consider the following issues:

- The values that you will use to test your program, and the results that they should produce. You need to create 5 individual test sets, each of which will detect a particular behaviour of your program.

Before you go any further; perform the following:

1. Modify your Countdown Calculator to use a method as shown above. Make the program as reliable as you possibly can and ensure that it passes all your tests.
2. Use appropriate **`if`** constructions, exception catching and looping to ensure that your program will work.

3. Show a demonstrator the program working and get it checked off on the attendance sheet. They will want to see your tests and may have some tests of their own. If they can crash your program you have to give them a Mars Bar. Or not.

From now on when you design a program you need to consider the valid values that your variables may have. The program design forms are going to be amended to reflect this change. Deciding on valid ranges like this should make your test methods easier to design.

Reverse Countdown

Your customer has asked you to make a "reverse countdown" calculator which can be used to convert seconds back into hours, minutes and seconds:

```
Reverse Countdown Timer Calculator by Rob Miles
Version 1.0
Enter the number of seconds: 12050
This is equal to 3 hours, 20 minutes and 50 seconds
```

The maximum number of seconds that are to be converted in this way is 86399. To do this you will have to reverse the calculation which you performed above. Here is an English description of one way to do it:

1. Start with the number of seconds which the user has given you.
2. Divide the number of seconds by 3,600. Call this the number of hours.
3. Multiply the number of hours by 3,600 and subtract the result from the initial number of seconds (this gives you the number of seconds left) .
4. Divide the number of seconds left by 60. Call this the number of minutes.
5. Multiply the number of minutes left by 60, and subtract this from the number of seconds left, this gives you the remaining number of seconds.

Remember that the C# operator that performs division between integers will always "round down".

You can base this program on your original one. Devise some test data and work out the results you should see before running the program. Your new program should use the `getValue` method to read in the number

Now you can create the program itself:

1. Create a program which meets the given specification. Again you should be able to base the program on an existing one (use the copy command to make a new source file).

Show a demonstrator the program working and get it signed off on the attendance sheet.

Seconds to Event

The final version takes a particular target date and works out how many seconds until that day starts.

```
Seconds to Calculator by Rob Miles
Version 1.0
Enter the hour: 12
Enter the day: 1
Enter the month: 9
Enter the year: 2013
The total number of seconds is: 18571139
```

This version of the program works out the number of seconds between the time now and a particular hour in the future.

Devise some test data and work out the results you should see before running the program. Make sure your tests include months and day combinations that are not valid, for example the 30th of February. Can you see any problems creating test data?

Take a look at the `DateTime` and `TimeSpan` classes provided as part of the System libraries. You should find that by using those you can make the program very short indeed. Pay special attention to `DateTime.Now` and `DateTime.Subtract`

Now you can create the program itself:

1. Create a program which meets the given specification.

Show a demonstrator the program working. Do the multiple choice test on eBridge.

Rob Miles
Feb 2014