University of Central Lancashire
UCLan

CO1404: Introduction to Programming

Lab 7 : Methods and Coursework Preparation

Lab Summary

The lab worksheets are designed to help you write your first programs. Students who have experience programming should find it straightforward at first. However, there will be advanced questions later on. There's a lot of reading in this worksheet as we introduce the tools required. Later worksheets will tend to be shorter, with more work for you to do!

Remember:

- If you have any questions or require assistance, please get your tutors attention. We are all here to help!
- Refer to previous weeks materials for guidance.

Objectives

This Lab aims to achieve the following objective(s) below:

- **PO.1 :** Methods
- **PO.2 :** Prepare for Coursework!

Lab Activity

This lab will explore how we create and interact with arrays. It consists again of just a single stage. As usual this is required to be completed by the end of this week. As always refer to the 4 **L**s: Lectures, Lecture Notes, Lab sheets and Lab Projects.

Questions

1. **What is the value of this expression:** `4 + 12 / 4`
    - (a) 7
    - (b) 4

- (c) 8
- (d) This is an invalid expression.

2. **What is the key difference between a while loop and a do-while loop?**
   - (a) A do-while loop executes in reverse.
   - (b) A while loop has one condition, but a do-while loop may have several.
   - (c) A do-while loop is guaranteed to execute at least once.
   - (d) A do-while loop calls a method to do it's task.

3. **Which condition tests if `lotteryBall` is in the range 1 to 49 inclusive**?
   - (a) `lotteryBall <= 1 && lotteryBall >= 49`
   - (b) `lotteryBall >= 1 && lotteryBall <= 49`
   - (c) `lotteryBall > 1 || lotteryBall < 49`
   - (d) `lotteryBall >= 1 || lotteryBall <= 49`

4. **What will be displayed on the output from the following code:**

```
int numbers[] = new int[6] {3, 7, 11, 2, 4, 17};
int result = numbers[2] + numbers[4];
Console.Write(result);
```

   - (a) 9
   - (b) 15
   - (c) 19
   - (d) 23

5. **Write a for loop that displays the odd numbers from 79 down to 3 in exactly this format (*3 marks for this question)***:

```
79, 77, 75, ... 7, 5, 3,
```

Methods

1. Create a new Visual Studio C# Console project called **Methods**. All the first few exercises should be written in the same project.

2. Add a method called `WaitForEnter` as shown in the lecture notes.

- Make sure you add the method in the correct location. You should not put the method inside Main, it should go inside the block for class Program

3. Call the method *two times* in the Main method to check that your method works and to show how using methods saves repeated code.

4. In the same project add a method `DrawHorizontalLine`, as shown in the presentation . Call this method in-between the two calls to `WaitForEnter`.

A Method Returning a Value

1. Still in the same project add a method `ReadNumberFromUser`, as illustrated in the lecture. >

> However, your version of this method should allow the user to enter an `int` not a `double` and the method should `return` an `int`. Be careful to make all the appropriate changes for this.

2. Just before your call to `DrawHorizontalLine`, display the text "How wide do you want the line?", then call your new `ReadNumberFromUser` method. Put the result of the method into an integer called `lineWidth`. Refer to the example on in the lecture and avoid making the mistake shown too.

> We will use this integer `lineWidth` in the next exercise...

A Method with a Parameter

1. Now update your `DrawHorizontalLine` method to accept a single `int` parameter called `width`. Refer to lecture slides for examples of this. Now change the horizontal line code so the line is not 80 wide, but uses the parameter width instead.

2. Finally change the *call* to `DrawHorizontalLine`. You now need to pass an integer to this method to say how wide you want the line - pass the variable `lineWidth` that you input from the user.

3. If you have done everything correctly to here you should end up with output as shown below. The length of the line should match the user input:

```
Press Enter to continue...
How wide do you want the line? **5**
\-----
Press Enter to continue...
```

Your main code should be simple and easy to read, and you have a collection of useful, reusable methods. The advantage of methods should be clear.

STAGE 1 - Network toolbox

1. Create a new Visual Studio C# Console project called **NetworkToolbox**. Delete all the shell code so you have an empty file.

2. Download the **NetworkToolbox** code from the module webpage, week 7. Copy the code from that file into your empty project file.

3. Run the program and you should see three network tests (if you don't make sure you have set up the project correctly with your tutor):
    - Finding the IP addresses associated with a hostname
    - Finding the hostname associated with an IP address (it's Google's address)
    - *Pinging* a hostname (Google again): A network message is sent to www.google.com, and sent back again, to make sure we can reach that site. The total trip time for the message is shown.

    > **Note:**
    >
    > - The university firewall may prevent ping from reaching its destination. If that happens, change the hostname to 127.0.0.1 to get a response - this is pinging your own machine!
    > - You may get a warning from your firewall when running these tests, but they are perfectly harmless - your firewall is just alerting you to the fact that this program accesses the network.
    > - The detail of the code to do these tests is not important right now, the next exercise is about converting this program to use methods.

Converting the Program to use Methods

1. First notice that the user is asked to press Enter three times in the program. Put this code into a `WaitForEnter` method in the same way as you did in the first exercise above. The Main code should call this method three times.

2. Next create three methods, `DisplayIPs`, `DisplayHostName` and `PingHost`. Each method should contain the appropriate block of code currently in the `Main` method. At first the methods should not have parameters and not return values. Replace the blocks of code in the Main method with calls to your new methods. When you are complete, your Main method should be very simple, like this:

```
DisplayIPs();
WaitForEnter();
DisplayHostname();
```

3. Notice that all three methods begin by initialising a string. These methods would be much more flexible if we passed the string as a parameter. So update each of the methods so the host or IP used is passed as a parameter. Then update your main code to pass suitable strings. You Main code will now look like this:

```
DisplayIPs("www.google.com");
WaitForEnter();
DisplayHostname("173.194.65.105");
...
```

4. The `PingHost` method will fail if the destination is unreachable. The method displays the success or failure as a string using `pingreply.Status.ToString()`. It would be useful if the method returned this string to caller, that way the main program can decide what to do if the ping fails. This is an example of making a method more useful, even though we don't currently need that feature.

5. Update the `PingHost` method to `return` the ping reply status as a `string`. In the `main` method put this return value into a `string` variable called `pingReply`.

6. After the `PingHost` call display the text "Ping failed" if pingReply is not equal to "Success". Test your code by pinging "10.255.255.255" instead of "www.google.com".

   You have now created a useful toolbox of methods, and your main code should look simple and informative. This is the style you should aim for in all your programs. Do remember to add comments to your methods though!

Stage 2 - Advanced Challenges

Instead of running just three fixed tests, it would be more useful to provide these network tools through a menu.

- Display a menu of three options, "1. Display IPs", "2. Display Host Name" and "3. Ping Host". Allow the user to type an option from 1 to 3 and call the appropriate method based on their input. Validate their input.

- For each option ask the user to type in the host name / IP address that will be used for the operation.

- Clear the screen and loop round to the menu again after each operation.

- **!MORE ADVANCED:** The Ping method will 'crash' (throw an exception) if given a host name that it can't resolve. Research how to catch and recover from an exception in C#. You could also take this opportunity to improve the Ping method, e.g. by attempting to ping multiple times on failure, or accept a parameter to specify a number of attempts and average out the round trip time across all the attempts, etc.

Time for Review and Coursework Preparation

Use the remainder of this lab session to go back and review previous labs that you were unsure about or didn't complete. Try the advanced sections that you didn't do first time round, or attempt to reach Stage 2 to push your skill further. Also use this time to ask questions of your tutor on topics you feel weak or need extra help on. Alternatively, work on your Coursework!