Module 01 Lab 01 Worksheet

Your First C# Application

# Overview

This lab is designed to introduce you to developing C# applications with Microsoft Visual Studio Code (otherwise known as VS Code) and .NET Core.

|  |
| --- |
| **NOTE**: You should have a C# language reference handy. |

Before beginning, make sure you have followed the instructions in the VS Code Introduction and Setup <link to file> and have set up your lab software. You should also know how to create a simple console application project and open it in VS Code.

|  |  |
| --- | --- |
| * Using the aspnet generator, create a console application called App01. * Open it in VS Code * When prompted by VS Code, restore unresolved dependencies and add required assets to your project.   Open the Integrated Terminal and enter the command  dotnet run  to build and run your app.  Copy and paste the program output here. (5 pts.)  The code:  using System;  namespace App1  {  class Program  {  static void Main(string[] args)  {  Console.WriteLine("Hello World!");  }  }  }  The output was Hello World! | 5 |
| In the Explorer view, click on the file Program.cs to open it in an editor window.  At the top of the file, you will see at least one statement that begin with the using keyword. This is similar to import in Java or #include in C/C++, which bring in external code for your program to use. In C#, these are known as *namespaces*. You can think of a namespace as a file folder where the external code can be found.  (Your application folder is it's own namespace, labelled by the application name.)  What is the first namespace listed in Program.cs? (5 pts.)  Using System;  Namespace App1 | 5 |
| The C# extension for VS Code includes *Intellisense*, which constantly monitors your code to advise you on things that could or should or must be fixed. This also lets you easily get more information on items in your code  Hover your mouse over Main and you should get a pop-up with a message.  What is the message? (5 pts.)  void Program.Main(string[] args) | 5 |
| You can also get a readout on your project state with your status bar. Look at the lower left of the status bar. You should see numbers with associated icons.  Hover your mouse over the status icons.  What does the tooltip say? (5 pts.)  7 errors | 5 |
| Click on the project status icons. The Problems window will appear. This will show you the Intellisense messages organized by the files in which they appear.  What is listed in the Problems window? (5 pts.)  App1.AssemblyInfo.cs App1\objDebug\netcoreapp2.2 | 5 |
| Another feature of Intellisense is context-sensitive help. If it has suggestions for how to fix an problem, you will see a little light bulb icon near the message.  Click on Console and you should see a light bulb appear to the left. Click on the bulb.  What suggestion(s) is/are listed? (5 pts.)  Use expression body for methods | 5 |
| Go ahead and select the suggestion you listed in the previous question.  What happens? (5 pts.)  Changes from:  static void Main(string[] args)  {  Console.WriteLine("Hello World!");  }  To:  static void Main(string[] args) => Console.WriteLine("Hello World!");  }  Click on Undo to revert to the previous version of your code. | 5 |
| Take a moment to save the changes you made.  Let's take a moment to get our program to do something. Modify your program to:   * Prompt the user for their name * Store the input string into a variable named str01. * Output a greeting to the console using the string stored in str01. (ex. "Hello, Ethel!") * Keep prompting the user to enter a name. * If they enter an empty string (a simple carriage return), then drop out of the loop with an appropriate message (ex. "See you next time!").   Copy and paste your code here. (20 pts.)  using System;  namespace App1 {  class Program {  static void Main(string[] args) {  char token = 't';  string str1 = "name";  while(token != 'q'){  Console.Write("Enter name: ");  str1 = Console.ReadLine();  if (String.IsNullOrEmpty(str1)){  token = 'q';  Console.WriteLine("See you next time!");  } else {  Console.WriteLine("Hello, " + str1 + "!");  }  }  }  }  } | 20 |
| Modify your application to put the code (greeting, loop, if statement, etc.) into a single function. Then call that function from Main() while keeping the same output format.  Copy and paste your code here. (20 pts.)  using System;  namespace App1 {  class Program {  static void Main(string[] args) {  nameFunction();  }  static void nameFunction() {  char token = 't';  string str1 = "name";  while(token != 'q'){  Console.Write("Enter name: ");  str1 = Console.ReadLine();  if (String.IsNullOrEmpty(str1)){  token = 'q';  Console.WriteLine("See you next time!");  } else {  Console.WriteLine("Hello, " + str1 + "!");  }  }  }  }  } | 25 |
| Close your project folder and create a new console project called App02.  Modify App02 to declare an array of type string containing the days of the week.  Use a for loop to output the days to the console.  Copy and paste your code here. (20 pts.)  using System;  namespace App2 {  class Program {  static void Main(string[] args) {  string[] strArray = new string [] {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};  for (int i = 0; i < strArray.Length; i++){  Console.WriteLine(strArray[i]);  }  }  }  } | 20 |
| Modify App02 to use an enumeration (enum) instead of an array.  Copy and paste your code here. (20 pts.)  using System;  namespace App3 {  class Program {  static void Main(string[] args) {  foreach (string str in Enum.GetNames(typeof(WeekDays)))  Console.WriteLine(str);  }  enum WeekDays{  Monday,  Tuesday,  Wednesday,  Thursday,  Friday,  Saturday,  Sunday  }  }  } | 20 |
| **Total** |  |

Complete this worksheet and submit it to your instructor.