

**IT 230 Module 1 Activity 3 Coding Submission**

**Name:** Nathan Hallam

**Date:** January 9th, 2024

**Class:** IT 230

**Module:**

| **1.** | Insert a copy of your of the ZIP file of all of your Visual Studio project files here so that it can be loaded and run in another Visual Studio:  ***See Attached*** |
| --- | --- |
| class Program  {  static void Main()  {  Console.WriteLine  (  "Hallam" +  "\nHello World!" +  "\nPress any key to continue..."  );  Console.ReadKey();  }  } |
| **2.** |  |
| **3.** | Explain the design of your program, the steps you took to complete it, and how you coded it:  Printing to the console is simple in Visual C# as there exists a class called “Console” containing many methods that can be used to determine the various states of a console application. In this program two console methods are used - the first being Console.WriteLine(). WriteLine, as its name suggests, writes a string to be displayed on the console. In this case 3 lines are written with the second and third lines starting with a newline character ‘\n’ so that each phrase is on its own line instead of using three separate WriteLines, this helps to consolidate everything. In hindsight I could have used Console.Write to get the same output but I am accustomed to using WriteLine so it is generally my goto when outputting to the console. The formatting of the string argument in the WriteLine is only for readability in the IDE and has no bearing on its execution in the program.  The second Console method used is Console.ReadKey() to pause the process of the program and await a keyboard input from the user before shutting down the program. This is only needed when creating a build of the program as the IDE automatically waits for an input from the user once the program finishes executing. |
| **4.** | Reflect on this experience and the lessons you learned from it:  While I did not learn anything new in regards to creating the console application I dd find out that you do not need to encapsulate the code into a class and method, meaning that the code could have been what is shown below instead of what was given back in #1. I only found this out when creating the project when there was some starting example code given alongside a link to the MSDN.  I could have stuck with this but it made me feel icky as I have never made a program without a class before in C# so I went with what I knew over the simpler method. This also included making a functional build for the program instead of simply using the debugger. This is why the Console.ReadKey() is present as the console would simply close without it.  Console.WriteLine  (  "Hallam" +  "\nHello World!" +  "\nPress any key to continue..."  );  Console.ReadKey(); |