**Computer Science Programming Courses**Ron Kessler’s Classes

**Programming in .Net: Overview of the .Net Programming Model
VB, C#, C++/CLI Classes**

This document outlines the steps necessary to create a finished .Net project. We need Visual Studio to convert our source code into a format that a CPU/Microcontroller can use and then do something useful/fun for us. Let’s take a closer look at how the Microsoft platform does this.

The VB, C#, or C++/CLI source code we write in Visual Studio in my classes is converted into MSIL (Microsoft Intermediate Language) by a compiler application when we build/run a project. A compiler is an application that is designed to convert source code into some other format. There are tons of these around and they are designed for a specific CPU and O/S. Microsoft created special compilers for each of the .Net languages so they can get our source code ready to execute on our computer.

The MSIL file that is created when we build our project is called the **assembly** and has an “.exe” file extension. Any file with this extension can be executed by double-clicking on it. This is the file that you ship to a customer or give to your friend to run on their computer. I will show you in class where this file is.

Figure : CLR Model from Vidya Agarwal (C-sharpcorner.com)

When your customer/friend runs this assembly file on a Windows computer that already has the .Net framework installed (any computer with XP SP-3 or newer), a program called the CLR takes over.

This Common Language Runtime (CLR) is a huge program that comes with Visual Studio. It manages our application by communicating with the O/S so the code we write will be executed correctly. It “links” the features contained on the hard drive inside the .Net Framework namespaces to create our finished product. It handles memory for us and tells the CPU which instructions to execute at just the right time. Thus, these types of programs are called “**Managed Code**” applications.

So when you hear the term “**Managed Code**”, I want you to understand it refers to VB.Net, C#.Net, and C++/CLI applications. In my C++ classes, we use C++/CLI projects. The CLI stands for **Common Language Infrastructure**. This terminology is used because you can also create “unmanaged” projects in C++. These projects create what is called “**Native Code**” without any MSIL becoming involved. Programmers use these types of programs when they need a high speed application that is designed to run on a specific CPU. You can see this in figure 1 above.

Finally, there is another program included inside the CLR called the “**Just-In-Time**” compiler (JIT). This program converts our MSIL into the **Machine Code** or Native code that the CPU on each machine needs in order to execute our application. They call this Just-In-Time for a specific reason.

Picture a form with two buttons on it. One is an “OK” button and the other is the “Quit” button. When someone clicks the OK button it triggers an event. The computer is notified that someone clicked the button and so it needs to see what action to take. That action is based on the code we wrote in that event handler. Now, the FIRST time the button is clicked, the MSIL code for that button is compiled into machine code. That code is then stored in RAM in a special location. Note, only the code for the OK button is converted. This allows the app to respond quickly. If the entire project had to be converted into machine code, it would be really slow.

When someone clicks the Quit button the FIRST time, the code for that click event is then converted to machine code and that is stored in RAM. This process continues whenever a control is used the first time. When someone clicks the button a second or third time, the machine code that has already been created and saved in memory is executed. The JIT only runs the FIRST time an event is triggered. This makes the program respond very quickly. When the program ends, this whole process starts over the next time it is run by the user.

Please remember that even though this sounds really complicated, all we are asking Visual Studio to do is convert our source code into a format that a CPU/Microcontroller can use to make something cool happen on our computer/phone/tablet/robot or whatever.