CHAPTER 1: THE WAY OF THE PROGRAM

Fall 2019 – CSC 180 – Introduction to Programming

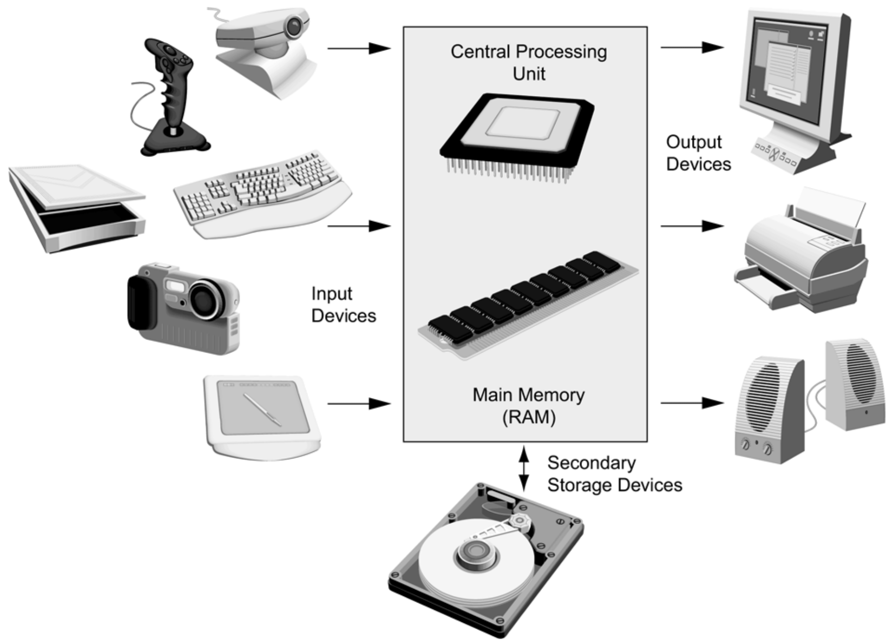
TAKE OWNERSHIP OF YOUR OWN LEARNING

- PLEASE READ BOTH THE SLIDES AND THE BOOK!!!
  - some of my slides may contain information not found in the book but essential for your learning
  - the book may contain information not presented in here due to time constraints

HARDWARE VS SOFTWARE

- Computer systems consist of
  - hardware devices: the physical pieces of the computer
    - The central processing unit (CPU), or the processor
    - Main memory, or RAM
    - Secondary storage devices
    - Input and Output devices (I/O devices)
      - Can you give examples?

  - software components: the programs that run on a computer.
    - Operating Systems – can you give examples?
    - Application Software – can you give examples?



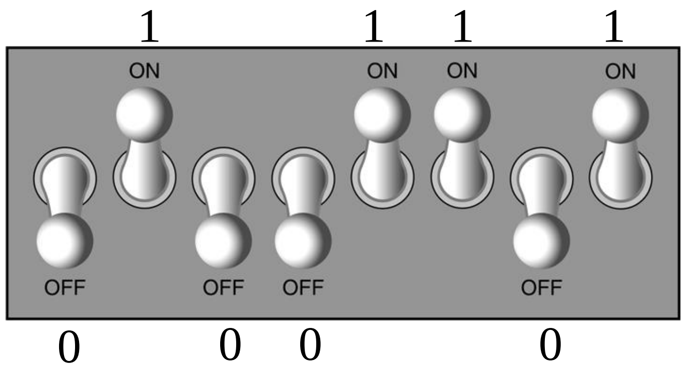
Images source: google images

BITS VS BYTES

- RAM is divided into units called ***bytes***.

- A byte consists of eight ***bits*** that may be either on or off.
  - A bit [BInary digiT] is either on or off:
    - 1 = on
    - 0 = off

  - How many values can you represent using :
    - 1 bit
    - 2 bits
    - 3 bits
    - 8 bits
    - 16 bits
    - 32 bits
    - 64 bits



Images source: google images

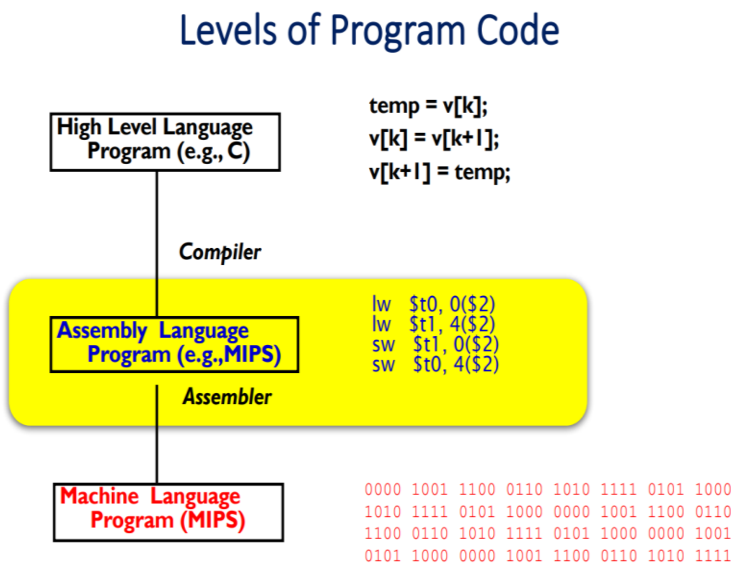
PROGRAMMING LANGUAGES

- A (**computer**) **program** is a set of instructions a computer follows in order to perform a task.
  - For example, generate a random number and display it to the user
  - Or, a program that converts a value from one base (say base-2) into another base (say base-10).
- A **programming language** is a special language used to write **computer programs**.
  - In our course we will use C# (C# is pronounced C sharp).
  - A programming language has a specific set of rules that must be followed when writing programs in that language. These rules form what is called the syntax of that programming language.
- An **algorithm** is a set of well defined steps to completing a task (this is a simplified definition).
  - This is the logic of a program

PROGRAMMING LANGUAGES (2)

- A **CPU** (processor) can only execute machine-code
  - Machine language is written using ***binary numbers***.
  - Each CPU has its own machine language.

- To make things easier for the developer new languages were created:
  - low-level languages (assembly)
    - processor dependent
  - high-level languages (C#, Java, Python, C, C++)
    - processor independent


Levels of Program Code
High Level Language Program (e.g., C)
Compiler
Assembly Language Program (e.g.,MIPS)
Assembler
Machine Language Program (MIPS)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

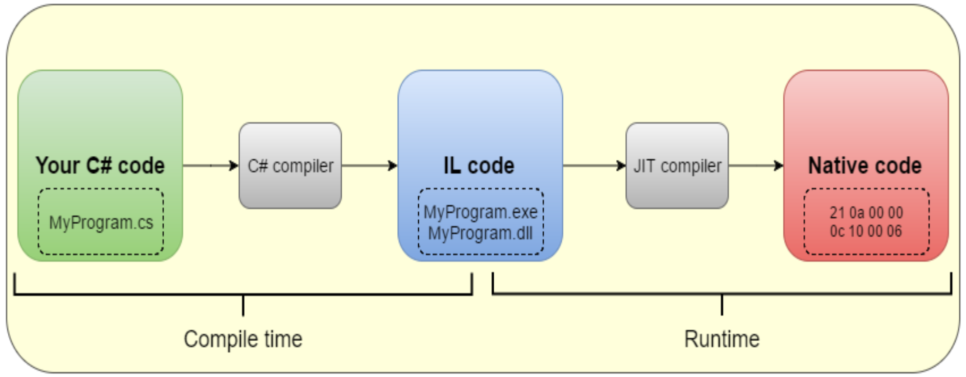
```
lw  $t0, 0($2)
lw  $t1, 4($2)
sw  $t1, 0($2)
sw  $t0, 4($2)
```

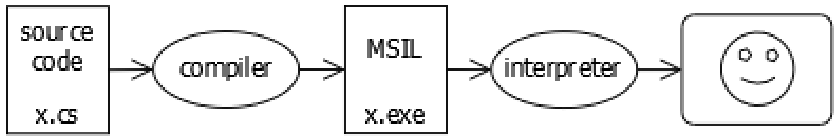
```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Image source: google images

PROGRAMMING LANGUAGES (3)

- Two ways to translate the program into machine code:
  - Compile (this is how C, C++ work)
    - translates it all at once, before running any of the commands.
      - "A compiled language is one where the program, once compiled, is expressed in the instructions of the target machine. For example, an addition "+" operation in your source code could be translated directly to the "ADD" instruction in machine code." (source)
  - Interpret (this is how Python works)
    - it translates the program line-by-line
      - "An interpreted language is one where the instructions are not directly executed by the target machine, but instead read and executed by some other program (which normally is written in the language of the native machine). For example, the same "+" operation would be recognised by the interpreter at run time, which would then call its own "add(a,b)" function with the appropriate arguments, which would then execute the machine code "ADD" instruction." (source)
  - Something in between (this is how Java and C# work) – see this source
    - For C#: the compiler read the source code and generates Microsoft Intermediate Language (MSIL) [or IL for short]
    - Common Language Runtime (CLR) is a program running on your computer that manages the execution of IL code. It uses a just-in-time (JIT), compiler to translate the IL code into machine code





WHAT IS A PROGRAM?

- A program is a sequence of instructions that specifies how to perform a computation

- The instructions, which we will call statements, look different in different programming languages, but there are a few basic operations most languages perform:
  - **input**: Get data from the keyboard, or a file, or some other device.
  - **output**: Display data on the screen or send data to a file or other device.
  - **math**: Perform basic mathematical operations like addition and multiplication.
  - **testing**: Check for certain conditions and run the appropriate sequence of statements.
  - **repetition**: Perform some action repeatedly, usually with some variation.

WHAT IS DEBUGGING?

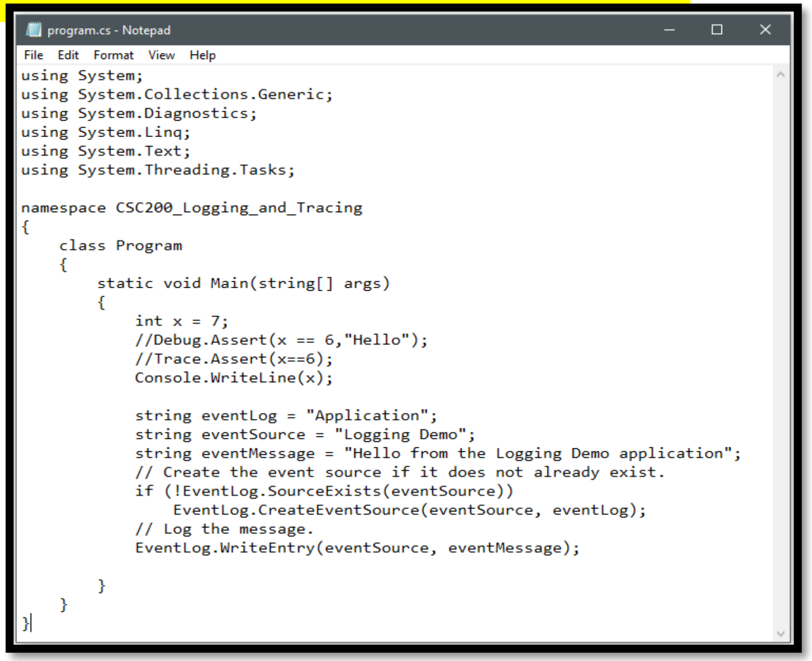
- programming errors are called bugs and the process of tracking them down and correcting them is called debugging.

- there are three types of errors:
  - Syntax errors ← easiest to find – the compiler won't accept your code until you fix these errors
    - The compiler can only translate a program if the program is syntactically correct; otherwise, the compilation fails and you will not be able to run your program.
    - These are the easiest to fix once you have sufficient experience. So please have patience the first few weeks until you learn how to deal with them.
  - Runtime errors
    - these errors don't appear until you run the program.
    - C# is a type-safe language, which means that the compiler catches a lot of errors. So run-time errors are rare, especially for simple programs.
  - Logical errors
    - the program you wrote is not the program you wanted to write.

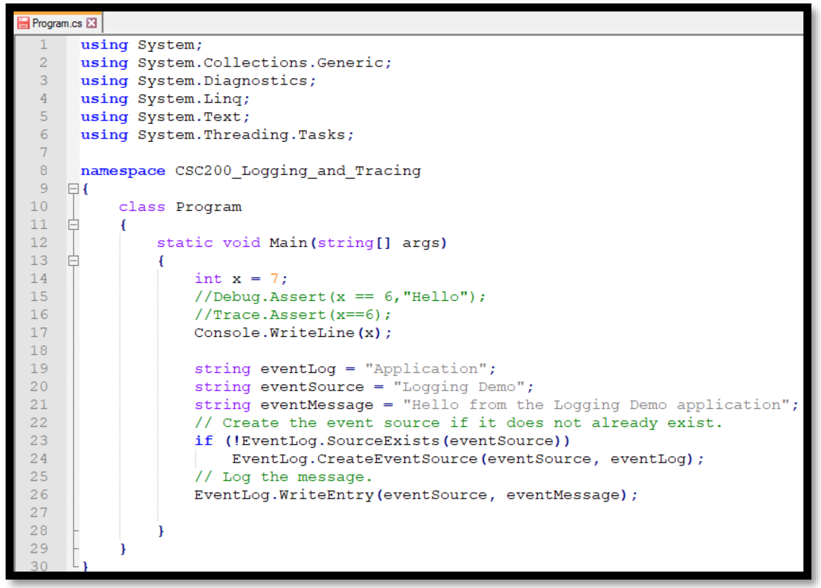
INTEGRATED DEVELOPMENT ENVIRONMENT

- To write and edit source code any text editor can be used.
  - For example, one could use notepad or **notepad++**

- Many programmers prefer however to use an **integrated development environment (IDE)** instead of simple text editors.
  - Visual Studio is an IDE from Microsoft and we'll make use of it in here. It includes (see source):
    - An advanced code editor supporting
      - IntelliSense (the code completion component) as well as code refactoring.
      - color coding
      - syntax highlighting
      - bookmarks
      - code snippets
      - background compilation - as you write code, Visual Studio compiles it in the background in order to provide feedback about syntax and compilation errors
    - An integrated debugger works both as a source-level debugger and a machine-level debugger.
    - Visual Studio includes a host of visual designers to aid in the development of applications
      - Windows Forms Designer, WPF Designer, Web designer/development, Data designer, Mapping designer, etc.
    - A compiler.
    - Many other …



```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CSC200_Logging_and_Tracing
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 7;
            //Debug.Assert(x == 6,"Hello");
            //Trace.Assert(x==6);
            Console.WriteLine(x);

            string eventLog = "Application";
            string eventSource = "Logging Demo";
            string eventMessage = "Hello from the Logging Demo application";
            // Create the event source if it does not already exist.
            if (!EventLog.SourceExists(eventSource))
                EventLog.CreateEventSource(eventSource, eventLog);
            // Log the message.
            EventLog.WriteEntry(eventSource, eventMessage);

        }
    }
}
```



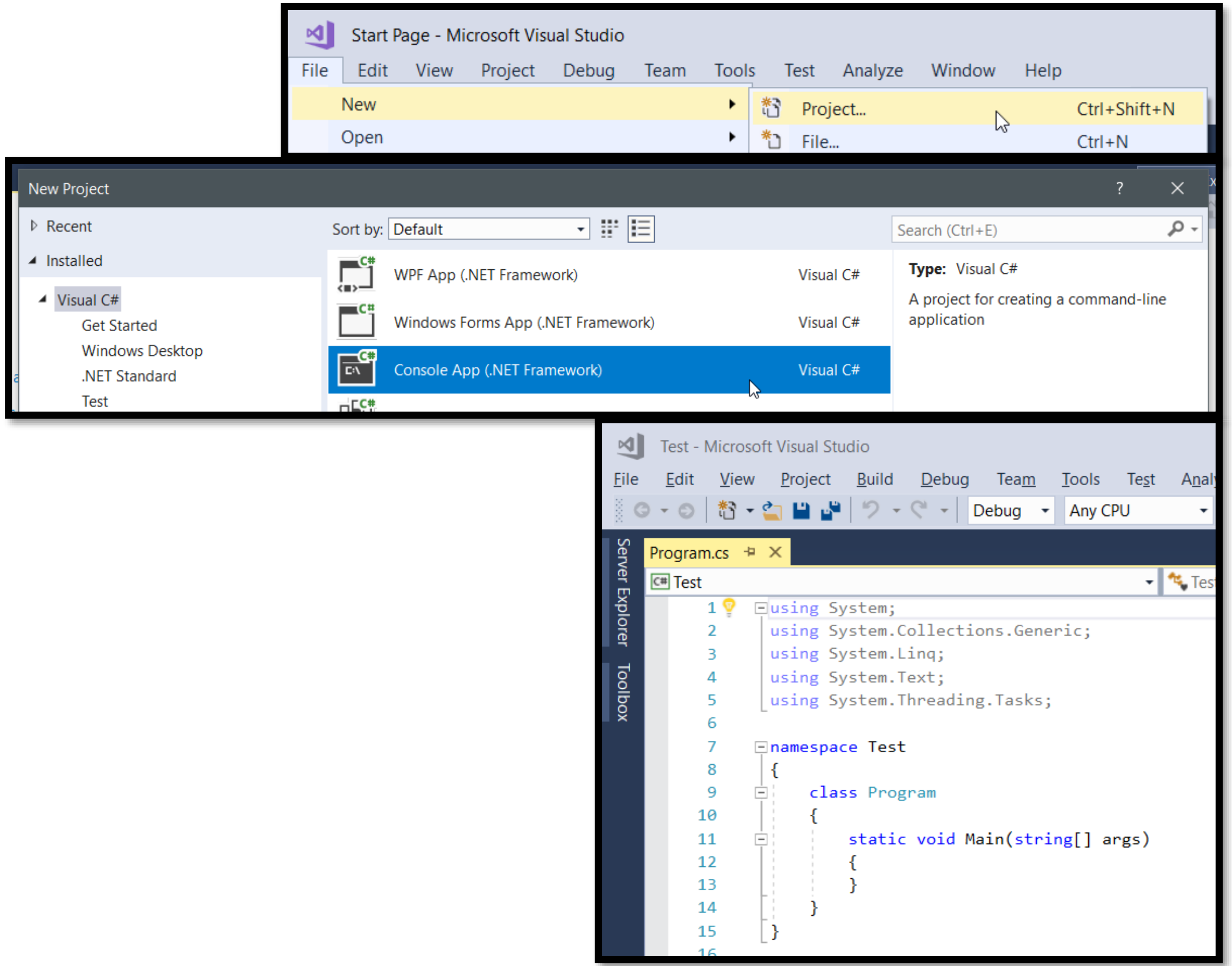
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CSC200_Logging_and_Tracing
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 7;
            //Debug.Assert(x == 6,"Hello");
            //Trace.Assert(x==6);
            Console.WriteLine(x);

            string eventLog = "Application";
            string eventSource = "Logging Demo";
            string eventMessage = "Hello from the Logging Demo application";
            // Create the event source if it does not already exist.
            if (!EventLog.SourceExists(eventSource))
                EventLog.CreateEventSource(eventSource, eventLog);
            // Log the message.
            EventLog.WriteEntry(eventSource, eventMessage);

        }
    }
}
```

WHAT IS THE .NET FRAMEWORK?

- By using Visual Studio you can use the .NET Framework to create a wide range of solutions that operate across a broad range of computing devices. .NET Framework includes:
  - **The Common Language Runtime**
    - manages the execution of code and provides a robust and highly secure execution environment that includes:
      - Memory management.
      - Transactions.
      - Multithreading.
  - **The .NET Framework Class Library**
    - a library of reusable classes that you can use to build applications.
    - these classes provide a foundation of common functionality that help to simplify application development by, in part, eliminating the need to constantly reinvent logic.
  - **Development Frameworks**
    - use to build common application types, including:
      - Desktop client applications, by using Windows Presentation Foundation (WPF).
      - Windows 8 desktop applications, by using XAML.
      - Server-side web applications, by using ASP.NET Web Forms or ASP.NET MVC.
      - Service-oriented web applications, by using Windows Communication Foundation (WCF).
      - Long-running applications, by using Windows services.
    - Each framework provides the necessary components and infrastructure to get you started.
    - We will mostly use Console Applications in this course

CREATING A C# PROJECT

- Open Visual Studio
- File > New > Project …
- Visual C# > Console App



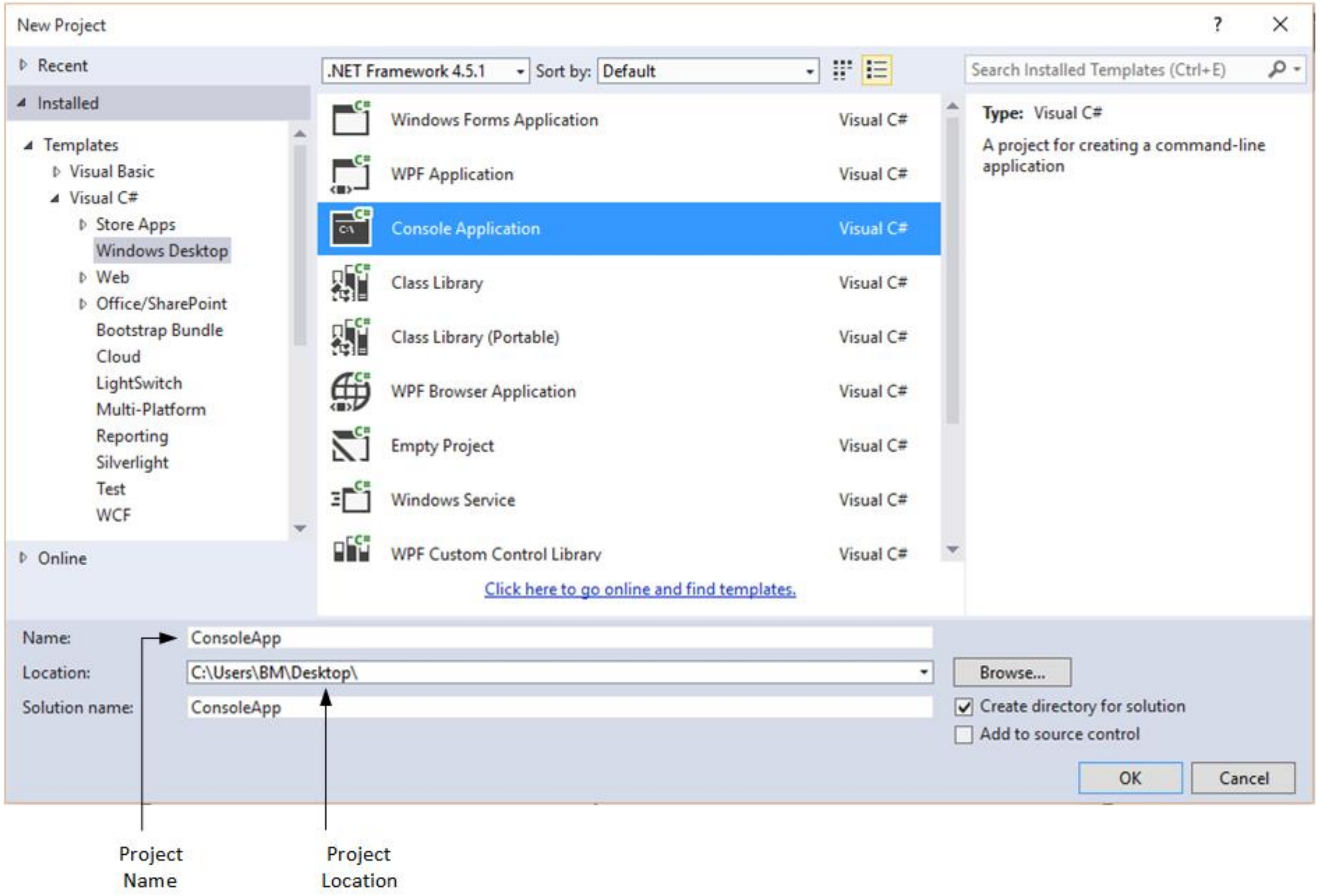
Type: Visual C#

A project for creating a command-line application

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Test
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

VISUAL STUDIO – HELLO WORLD!

- Here is another interesting online C# tutorial: http://www.infocodify.com/csharp/overview
  - Creating the Console App see this link


New Project

Recent
Installed
Templates
Visual Basic
Visual C#
Store Apps
Windows Desktop
Web
Office/SharePoint
Bootstrap Bundle
Cloud
LightSwitch
Multi-Platform
Reporting
Silverlight
Test
WCF
Online

.NET Framework 4.5.1 | Sort by: Default | Search Installed Templates (Ctrl+E)

Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
Class Library	Visual C#
Class Library (Portable)	Visual C#
WPF Browser Application	Visual C#
Empty Project	Visual C#
Windows Service	Visual C#
WPF Custom Control Library	Visual C#

Click here to go online and find templates.

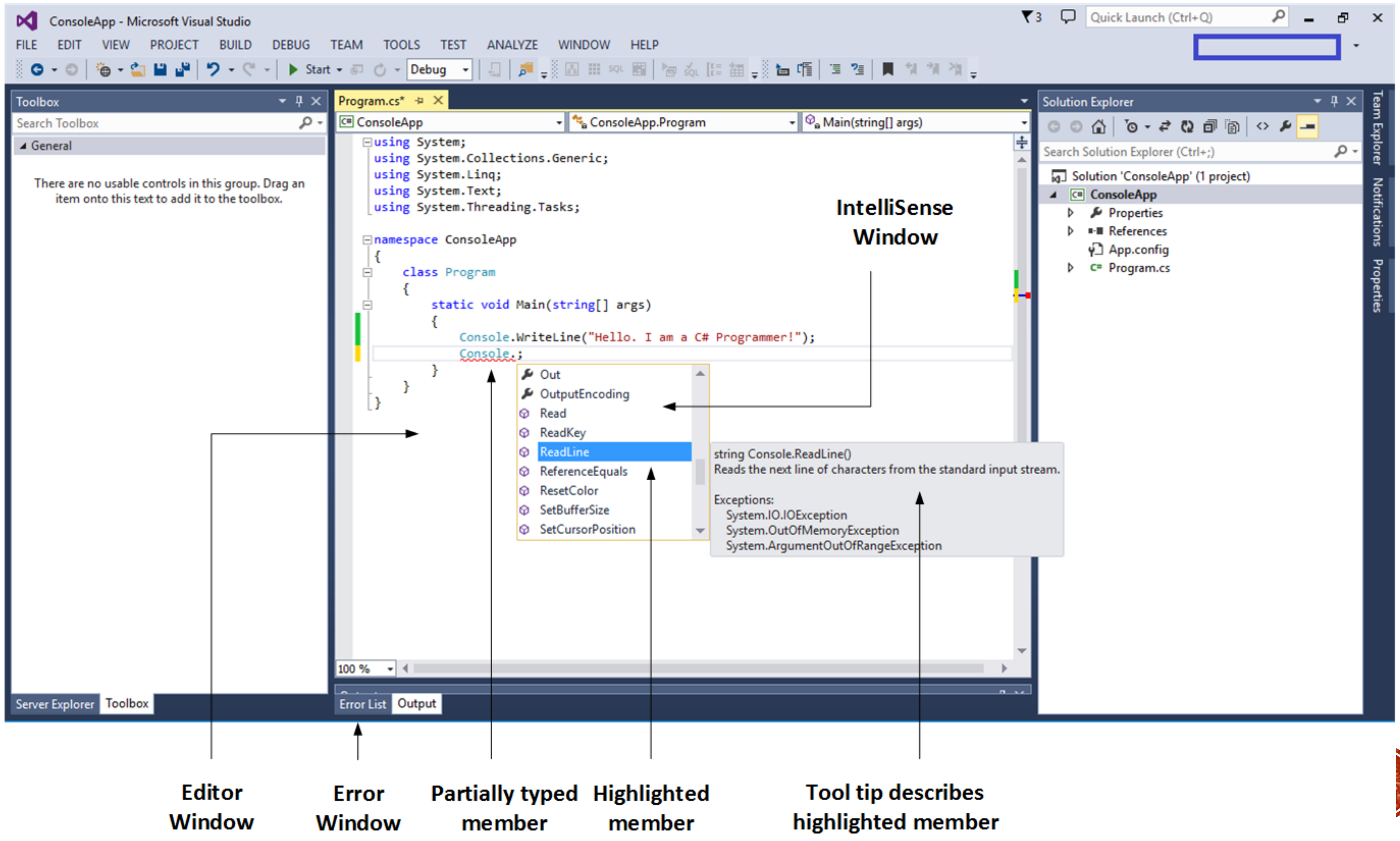
Type: Visual C#
A project for creating a command-line application

Name: ConsoleApp
Location: C:\Users\BM\Desktop\
Solution name: ConsoleApp

Browse...
[✓] Create directory for solution
[ ] Add to source control
OK Cancel

VISUAL STUDIO – HELLO WORLD!

- Here is another interesting online C# tutorial: http://www.infocodify.com/csharp/overview
  - see this [link]()



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello. I am a C# Programmer!");
            Console.;
        }
    }
}
```

```
Out
OutputEncoding
Read
ReadKey
ReadLine
ReferenceEquals
ResetColor
SetBufferSize
SetCursorPosition
```

```
string Console.ReadLine()
Reads the next line of characters from the standard input stream.

Exceptions:
  System.IO.IOException
  System.OutOfMemoryException
  System.ArgumentOutOfRangeException
```

"HELLO WORLD!"

- Traditionally the first program people write in a new language is "hello world"
- Use Ctrl+F5 to keep the console window open at the end of the program

```
using System;

namespace HelloWorldApplication {
    class HelloWorld {
        static void Main(string[] args) {
            /* my first program in C# */
            Console.WriteLine("Hello World!");
        }
    }
}
```

LET'S START … READ THE BOOK!!!

- This just a preview. **NOTE: C# is Case Sensitive!!!**
  - By the end of this semester you should understand every piece of this code
  - Note that the line numbers (on the left) are given so we can refer to a specific line, they are not part of the program.

- C# programs are made up of **namespaces** and **classes**.
  - A namespace is a way of separating names so that different programs can use the same name without getting confused.
    - Your program should all ways be included in a **namespace**.
    - A namespace definition has the form:

```
namespace ⟨NameSpace⟩ {
    . . .
}
```

  - A **namespace** is made up of class definitions
    - A class definition has the form:

```
class ⟨ClassName⟩ {
    . . .
}
```

  - A **class** is made up of a number of **method definitions** (and other members).
    - **Main method** is a special method …it marks the place the program starts
    - A method is a named collection of statements. It can have any number of statements.

  - A statement is an instruction to perform one particular operation, for example add two numbers together and store the result.
    - A **program** is just a sequence of one or more statements.
    - Statements typically end with semi-colon (;).
      - Example of a statement: Console.WriteLine("Hello, world.");

- **Console.WriteLine** is a method provided by one of C#'s **libraries**.
  - A library is a collection of class and method definitions.
    - In this case the **WriteLine** method comes from the **Console** class. The **Console** class is defined in the **System** namespace, which is why we had to tell the system we are using the System namespace

```
using System;
```

- C# uses squiggly-braces ({ and }) to group things together.

- Comments are fragments of text ignored by the compiler They can start with // or be enclosed in /* and */

```
1  using System;
2
3  namespace ThinkSharp {
4     public class Hello {
5        // main: generate some simple output
6        public static void Main(string[] args) {
7           Console.WriteLine("Hello, world.");
8        }
9     }
10 }
```

HOMEWORK FOR CHAPTER 1

- Requirements: see moodle for details
- Deadline: see moodle