Learnings

Module – 1

1. Visual Studio IDE 2019 Overview

- Visual Studio is an Integrated Development Environment (IDE) developed by Microsoft.
- It provides support for 36 different programming languages. It is available for Windows as well as for macOS.

- **Different types of windows**
  - The IDE has two basic window types, *tool windows* and *document windows*. Tool windows include Solution Explorer**,** Server Explorer**,** Output Window**,** Error List, the designers, the debugger windows.
  - When you have to view or edit two locations at once in a document, you can split windows.

- **Solution and Projects**
  - A solution is simply a container Visual Studio uses to organize one or more related projects.
  - Create a solution
    I. Open Visual Studio.
    II. On the start window, select Create a new project**.**
    III. On the Create a new project page, enter blank solution into the search box, select the Blank Solution template, and then select Next.
    IV. Name the solution and then select Create.

- **Code editor features**
  - The Visual Studio editor provides many features that make it easier for you to write and manage your code and text.

- Editor features:
    I. Syntax Coloring
    II. Error and Warning Marks
    III. Brace Matching
    IV. Selecting Code and Text
    V. Change Tracking

- **Shortcuts**

Task	**Shortcut**
Maximize floating window	Double-click on title bar
Maximize/minimize windows	Win+Up arrow / Win+Down arrow
Redock floating window	Ctrl+double-click on title bar
Move/dock floating windows	Win+Left arrow / Win+Right arrow
Close active document	Ctrl+F4
Show open file list	Ctrl+Alt+Down arrow
Show all floating windows	Ctrl+Shift+M
Show jump list	Win+Alt+N
Start new instance	Win+Shift+N
Switch between windows	Win+N

2.Project Types

- **Windows App, Class Library**
  - A *class library* defines types and methods that are called by an application.
  - If the library targets .NET Standard 2.0, it can be called by any .NET implementation that supports .NET Standard 2.0.
  - We can use a class library in our app or project by passing reference into it and then in the file using it's namespace.
  - Windows App is an app that can be run only on windows platform and can be executed by the operating system on the system directly.
  - Windows app in installed in windows platform using windows operating system.
  - Its interface is shown in the form of windows form.

- **Web application**
  - A consistent web development environment that is also used for creating the other components of the application.
  - Availability, performance, and scalability of web processing.
  - Access to the integrated .NET security model

3) Create First C# Program "Hello World"

- **What is namespace?**

    - namespaces are used to logically arrange classes, structs, interfaces, enums and delegates.
    - The namespaces in C# can be nested.
    - he .NET framework already contains number of standard namespaces like System, System.Net, System.IO etc.

- **Declaring a Namespace**

The C# language provide a keyword namespace to create a user defined name space. The general form of declaring a namespace is as follows.

```
namespace <namespace_name>
{
// Classes or interface or struct etc.
}
```

- It is not possible to use any access specifiers like private, public etc with a namespace declaration. The namespaces in C# are implicitly have public access and this is not modifiable.
- The namespace elements can't be explicitly declared as private or protected. The namespace allows only public and internal elements as it members. The default is internal.

- **What is class?**
- A class is like a blueprint of a specific object. In the real world, every object has some color, shape, and functionalities.
- In C#, a class can be defined by using the class keyword.

```
Public class (class name)
{
// fields, property, methods etc…
}
```

- **Variable And Method Declaration**

1) Method:
   - A method is a group of statements that together perform a task. Every C# program has at least one class with a method named Main.
   - The Main method is the entry point for every C# application and it's called by the common language runtime when the program is started.

```
<Access Specifier> <Return Type> <Method Name>(Parameter List)
{
  //Method Body
}
```

Following are the various elements of a method −

- Access Specifier − This determines the visibility of a variable or a method from another class.
- Return type − A method may return a value. The return type is the data type of the value the method returns.
- Method name − Method name is a unique identifier and it is case sensitive.
- Parameter list − Parameters are optional, enclosed between parentheses, the parameters are used to pass and receive data from a method.
- Method body − This contains the set of instructions needed to complete the required activity.

2) Variable:

- C# variable contains a data value of the specific data type.
- variable Declaration in C# is as follows-
- [modifier] [dataType] [variableName] = variable;
- Variable names must be unique.
- Variable names can contain letters, digits, and the underscore _ only.
- Variable names must start with a letter.
- Variable names are case-sensitive, num and Num are considered different names.
- Variable names cannot contain reserved keywords. Must prefix @ before keyword if want reserve keywords as identifiers.

4) Understanding C# Program

- **Program flow**
  - C# program consists of the following parts −
    1. Namespace declaration
    2. A class
    3. Class methods
    4. Class attributes
    5. A Main method
    6. Statements and Expressions
    7. Comments
  - C# is case sensitive.
  - All statements and expression must end with a semicolon (;).
  - The program execution starts at the Main method.
  - Unlike Java, program file name could be different from the class name.

- **Understanding Syntax**

```
using System;

namespace HelloWorldApplication
{
    class HelloWorld
    {
        static void Main(string[] args)
        {

            Console.WriteLine("Hello World");
        }
    }
}
```

- he first line of the program using System; - the using keyword is used to include the System namespace in the program. A program generally has multiple using statements.
- The next line has the namespace declaration. A namespace is a collection of classes. The HelloWorldApplication namespace contains the class HelloWorld.
- The next line has a class declaration, the class HelloWorld contains the data and method definitions that your program uses. Classes generally contain multiple methods. Methods define the behavior of the class. However, the HelloWorld class has only one method Main.
- The next line defines the Main method, which is the entry point for all C# programs. The Main method states what the class does when executed.
- The Main method specifies its behavior with the statement Console.WriteLine("Hello World");
- WriteLine is a method of the Console class defined in the System namespace. This statement causes the message "Hello, World!" to be displayed on the screen.

5. Working with code files, projects & solutions

- **Understanding structure of solution**
  - Solution is collection of projects which are related each other and are containing information of dependencies between them.
  - A solution can contain multiple projects and a project can be part of multiple solutions.
  - A solution contains projects, projects contain source files and solution also contains solutionName.sln file.

- **Understanding structure of project**
  - Structure of Windows forms app o File name is same as app name and [appname].csproj file contains references of other projects and of packages used and also version of project etc.
  - It has got dependencies which has all the server side NuGet Packages and other third-party packages required in the project
  - It has a .cs file which contains code of windows form that we create.
  - It also contains Program.cs file which has Main method as required to run app as console application. The configuration for app is in CreateHostBuilder Method called in Main method.

- **Structure of Web app**
  - File name is same as app name and [appname].csproj file contains references of other projects and of packages used and also version of project etc.
  - It contains Connected Services folder which is used to connect project to services like Azure and is basically used during deployment.
  - It has got dependencies which has all the server side NuGet Packages and other third party packages required in the project
  - Then there is properties folder which contains launchSettings.json which contains debug settings
  - Then it contains Pages folder which contains some demo interface pages.
  - wwwroot folder contains static files like images, css, JavaScript, etc. These are the only files which are served over http request.
  - It also contains Program.cs file which has Main method as required to run app as console application.

- It also contains Startup.cs file which runs always first when the project is executed and it contains
- appsettings.json is an application configuration file and contains configurations like database settings, any global variables for whole application.

- **Structure of Class Library**
- File name is same as app name and [appname].csproj file contains references of other projects and of packages used and also version of project etc.
- It has got dependencies which has all the server side NuGet Packages and other third party code files required in the project
- It has a .cs file which contains code of a particular class that we define in it.

- **Structure of Web Api Project**
  - File name is same as app name and [appname].csproj file contains references of other projects and of packages used and also version of project etc.
  - It contains Connected Services folder which is used to connect project to services like Azure and is basically used during deployment.
  - It has got dependencies which has all the server side NuGet Packages and other third party packages required in the project
  - Then there is properties folder which contains launchSettings.json which contains debug settings
  - Then it contains Controller folder which contains controllers which has code of controlling or manipulating an api.
  - www root folder contains static files like images, css, JavaScript, etc. These are the only files which are served over http request.
  - It also contains Program.cs file which has Main method as required to run app as console application.
  - It also contains Startup.cs file which runs always first when the project is executed and it contains configurations of services used in project
  - appsettings.json is an application configuration file and contains configurations like database settings, any global variables for whole application.

- **Familiar with different type of file extensions**
  - .sln – It contains information of the projects contained in the solution as it is the solution file.
  - .csproj – It contains the references of other projects and of packages used and also version of project etc.
  - .cs – class file of C#.
  - .json – JavaScript Object Notation file which stores simple objects and data structures.

6) Understanding datatypes & variables with conversion

- C# is a strongly-typed language.
- It means we must declare the type of a variable that indicates the kind of values it is going to store, such as integer, float, decimal, text, etc.

- **Value Data type:**
- Assigns a value directly in both signed/unsigned form and the system allocates memory to store the value.
- They are derived from the class System. ValueType.
- The value types directly contain data which stores numbers, alphabets, and floating-point numbers etc.

1. Predefined Data Type:

   These are the already defined datatypes in C#

   Integer – int, Decimal – decimal, Float – float, Character – char, Double – double, etc.
2. User Defined data type:
   It is a Datatype which is defined and used by the users.
   E.g., Structure – struct, Enumerations – Enum

- **Reference Data type:**
  - The reference data types do not contain the actual data stored in a variable, but they contain a reference to the variables.
  - In other words, they refer to a memory location.
  - If the data in the memory location is changed by one of the variables, the other variable automatically reflects this change in value.

1. Predefined data type:
   such as Objects, String.
2. User Defined data type:
   such as Classes, Interface.

- **Pointer Data type:**
  The pointer is a variable that points to an address of a value

- **Datatype Conversion**
  - **Implicit conversions**: No special syntax is required because the conversion always succeeds and no data will be lost. Examples include conversions from smaller to larger integral types, and conversions from derived classes to base classes.
  - **Explicit conversions**: Explicit conversions require a cast expression. Casting is required when information might be lost in the conversion, or when the conversion might not succeed for other reasons. Typical examples include numeric conversion to a type that has less precision or a smaller range, and conversion of a base-class instance to a derived class.
  - **User-defined conversions**: User-defined conversions are performed by special methods that you can define to enable explicit and implicit conversions between custom types that do not have a base class–derived class relationship. For more information, see User-defined conversion operators.
- **Boxing And Unboxing**
  - Boxing: Boxing is the process of converting a value type to the object type or any interface type implemented by this value type. Boxing is implicit.

    Example:

    int number1=100;

    number2=number1

  - Unboxing: It is the process of converting a reference type to value type. Unboxing extract the value from the reference type and assign it to a value type.
  - Unboxing is explicit. It means we have to cast explicitly.

    Example:

    number1=100;

    int number2=(int)number1

7) Understanding Decision making & statements

- **if else, switch**
  - **If Statement**: The if statement contains a boolean condition followed by a single or multi-line code block to be executed. At runtime, if a boolean condition evaluates to true, then the code block will be executed, otherwise not.

```
if(condition)
{
        // code block to be executed when if condition evaluates to true
}
```

  - **else if Statement**: Multiple else if statements can be used after an if statement. It will only be executed when the if condition evaluates to false. So, either if or one of the else if statements can be executed, but not both.

```
if(condition1)
        {
        // code block to be executed when if condition1 evaluates to true
        }
else if(condition2)
         {
        // code block to be executed when
         // condition1 evaluates to false
        //condition2 evaluates to true
        }
```

  - **else Statement**: The else statement can come only after if or else if statement and can be used only once in the if-else statements.
  - The else statement cannot contain any condition and will be executed when all of the previous if and else if conditions evaluate to false.

```
if(condition)
{
        // code block to be executed when if condition evaluates to true
 }
Else
{
// code block will executed when id condition evaluates to false
}
```

- **switch Statement**:
  - The switch statement can be used instead of if else statement when you want to test a variable against multiple conditions.
  - The switch statement starts with the switch keyword that contains a match expression or a variable in the bracket switch (match expression).
  - The result of this match expression or a variable will be tested against conditions specified as cases, inside the curly braces { }. Each case includes one or more statements to be executed.
  - The case will be executed if a constant value and the value of a match expression/variable are equal.
  - The switch statement also contains a default label. The default label will be executed if no cases executed.
  - break keyword is used to exit the program control from a switch case.

Example:

```
switch(match expression/variable)
{
case constant-value:
statement(s) to be executed;
 break;
default:
 statement(s) to be executed;
 break;
}
```