PHASE-I

(Nadar Sanchana)

1) Visual Studio 2019 IDE Overview:

Different types of windows (solution exp.,properties etc.)

- Tool windows:

  It includes Toolbox, Solution Explorer, Properties, Help, and Server Explorer, Output Window**,** Error List**,** the designers, the debugger windows, and so on

- Solution Explorer:
  - It enables you to explore and manage your solutions and projects.
  - **Solution node** →manages your solution
  - **Project node** →manages your project
  - **Dependencies node** → manages your solution & project dependencies.
  - **Program node** → you can view, edit, and manage your program or application.

- Server Explorer:
  - Can be used to test connections and view SQL Server databases, any other databases that have an ADO.NET provider installed, and some Azure services.

- Output Window:
  - The Output window displays status messages for various features in the integrated development environment.
  - The following controls are shown in the toolbar of the Output window: Find message code,go to previous message, go to next message,clear all.

- Document window:
  - Document windows contain source code files, arbitrary text files, config files, and so on.

- Code Window:
  - We can use the Code window to write, display, and edit Visual Basic code. We can open as many Code windows as we have modules, so we can easily view the code in different forms or modules, and copy and paste between them.

- Dock Window:

- When you click and drag the title bar of a tool window,a guide diamond appears.
- During the drag operation, when the mouse cursor is over one of the arrows in the diamond, a shaded area will appear that shows you where the window will be docked if you release the mouse button now.

- Properties Window:
  - It can be found in View menu(F4 shortcut key)
  - Displays different types of editing fields. These edit fields include edit boxes, drop-down lists, and links to custom editor dialog boxes.
  - They are the settings of controls.
    It contains UIelement list such as
    - Object name
    - Categorized
    - Property Pages
    - Sort by Property Source
      Groups properties by source, such as inheritance, applied styles, and bindings. Only available when editing XAML files in the designer.
    - Events

Solution and Project:

Project:

- Project contains all files that are compiled into an executable, library, or website.
- Those files can include source code, icons, images, data files, and so on.
- Visual Studio uses MSBuild to build each project in a solution.
- The file extension reflects the type of project, for example, a C# project (.csproj), a Visual Basic project (.vbproj), or a database project (.dbproj).
- The project file is an XML document that contains all the information and instructions that MSBuild needs in order to build your project, including the content, platform requirements, versioning information, web server or database server settings, and the tasks to perform.
- Project files can be accessed from project node in solution explorer.

Solution:

- Visual Studio uses two file types to store settings for solutions:

1) .sln:
   - Organizes projects, project items, and solution items in the solution.
   - The .sln file contains text-based information the environment uses to find and load the name-value parameters for the persisted data and the project VSPackages it references.

2) .suo:
   - (Solution User Options) stores user-level settings and customization.
   - The solution user options file is used to store user preference settings, and is created automatically when Visual Studio saves a solution.
   - When the environment opens a .suo file, it enumerates all currently loaded VSPackages.

Code features:

- syntax highlighting
- bracket-matching
- auto-indentation
- box-selection
- snippets
- Intuitive keyboard shortcuts
- easy customization

- Visual Studio Code includes an interactive debugger, so you can step through source code, inspect variables, view call stacks, and execute commands in the console.
- Using Electron, VS Code combines web technologies such as JavaScript and Node.js with the speed and flexibility of native apps.

Shortcuts:

- Solution Explorer search → Ctrl+
- Search in Tools Options → Ctrl+E
- Open solution → Ctrl+Shift+O
- Add New Project → Ctrl+Shift+N
- Open Properties window → F4
- Build → Ctrl+Shift+B
- Run → F5
- Restart → Ctrl+Shift+F5

- Rename → F2
- Close All the document → Ctrl +W
- Navigate to the previously opened file → Ctrl+tab or Ctrl+Alt+ Down Arrow
- Search in files → Ctrl+Shift+F
- Replace in files → Ctrl+Shift+H
- Add New File → Ctrl+N
- Add New Item → Ctrl+Shift+A
- Add Existing Item → Alt+Shift+A
- Add class → Type cl + tab tab
- Add constructor → Type ctro + tab tab
- Comment → Ctrl+K+C
- Remove comment → Ctrl+K+U
- Cut / Delete line→ Cut: Ctrl +L Delete: Ctrl+Shift+L
- Got to definition → F12
- Go to implementation → Ctrl+F12

2) Project Types:

- Windows App:
  - A Windows forms application is one that runs on the desktop computer.
  - It will have collection of controls such as labels,textboxes, listboxes etc.
  - Windows Forms is a UI framework for building Windows desktop apps.

Create a WinForms app

1. Select create a new project.
2. In the Search for templates box, select winforms.
3. In the code language dropdown, choose C# or Visual Basic.
4. In the templates list, select Windows Forms App (.NET) and then click Next.
5. Set the Project name and click Create**.**
6. Add controls to the form such **as** labels,textboxes, listboxes from toolbox plane.
7. Now Select the button control on the form.In the Properties pane, click on the events icon to list the events of the button.

- Class Library:

  - Contains → program code, data & resources.
  - It is collection of prewritten classes or coded templates, any of which can be specified and used by a programmer when developing an application program.
  - We can also add class library by Right-click on the solution in Solution Explorer and select Add > New Project.
  - On the Add a new project page, enter library in the search box. Choose C# or Visual Basic from the Language list, and then choose All platforms from the Platform list. Choose the Class Library template, and then choose Next.
  - **A Dynamic Link library** (DLL) is a library that contains functions and codes that can be used by more than one program at a time.

- Web Application:

  - It performs task over the internet.
  - It is a computer program which uses browser and web technology.
  - In visual studio ASP.net is used to design web application.

Create a project:

1. In the start window, choose Create a new project.

2. After you apply the language, platform, and project type filters, choose the ASP.NET Core Web App template, and then choose Next.
3. Write the project name & additional information.
4. In the Solution Explorer, expand the folder, and then choose .cshtml file.
5. This file corresponds to a page that's named Home in the web app, which runs in a web browser.

---

3)Creating c# program:

➢ Namespace:

- Used to organize the classes.
- To access class use namespace.classname.
- For eg: System.Console.WriteLine, where System is namespace, Console is classname and WriteLine is the method.
- Used to resolve classname conflict.
- Doesn't correspond to file, directory or assembly name.
- **Namespace Alias Qualifier(::)** makes the use of alias name in place of longer namespace and it provides a way to avoid ambiguous definitions of the classes. It is always positioned between two identifiers. The qualifier looks like two colons(::) with an alias name and the class name. It can be global.

➢ Class:

- Consists of data and behaviour.
- user-defined blueprint.
- combines the fields and methods into a single unit.

class declarations can include these components, in order:

- **Modifiers:** A class can be public or internal etc. By default modifier of class is internal.
- **Keyword class:** A *class* keyword is used to declare the type class.
- **Class Identifier:** The variable of type class is provided. The identifier should begin with a initial letter which should be capitalized by convention.

- **Base class or Super class:** A base class is a class from which other classes are derived.

- **Interfaces:** A comma-separated list of interfaces implemented by the class, if any, preceded by the **: .**A class can implement more than one interface.

- **Body:** The class body is surrounded by { }

- Constructors in class are used for initializing new objects.
- Fields are variables→ state of class & its object.
- Methods→used to implement the behaviour of the class and its objects.

OBJECTS:

- Real time entity.

An object consists of:

- **State:** It is represented by attributes of an object.
- **Behaviour:** It is represented by methods of an object.
- **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

➢ Variables and method declaration

- Variables declared within a class are called **fields**.

Syntax
Datatype variable_name;

➢ Methods:

- A method is a code block that contains a series of statements.
- The Main method is the entry point for every C# application.

METHOD SIGNATURE:

- Declaration is done in: class, interface, or struct.
- Specified by access level such as :public, private, abstract or sealed.

METHOD ACCESS:

- Calling a method on an object is like accessing a field.

- A program causes the statements to be executed by calling the method and specifying any required method arguments.

METHOD CALLING:

- For **calling a method by reference**, ref keyword is used.
- Passing a value type parameter to a method by reference means passing a reference of the variable to the method.
- So the changes made to the parameter inside the called method will affect the original data stored in the argument variable.
- In **call by value**, the changes made to the parameter inside of the called method will not have an effect on the original value. The Variable value is directly stored in the memory.

RETURN VALUES:

- Methods can return a value to the caller. If the return type is not void, the method can return the value by using the return keyword.
- A statement with the return keyword followed by a value that matches the return type will return that value to the method caller.

ASYN METHODS:

- Asynchronous methods can be invoked.
- C# asynchronous method is a special method that executes asynchronously. C# provides **async** modifier to make a method asynchronous. It is used to perform asynchronous tasks.
- C# **await** expression is used to suspend the execution of a method.
- If a method which uses **async** modifier does not contain **await** expression, executes synchronously.

4) Understanding C# Program:

Consider the following example of C#:

```
Using System;

namespace demo1{

Class demo{

public static void Main()

{

Console.WriteLine("Hello World");

}

}
```

- Program flow & Understanding the syntax:

Document section
Using directive section
Interfaces Section
Classes section
Main method

- The **using** keyword is used to import System class files.
- Here System is the namespace.
- System namespace contains the classes that most applications use for interacting with the operating system.
- Once the required classes are imported then the program goes to namespace.
- The **namespace** consist of class. It is used to organize the classes.
- Then the **class** is defined. The class may consist of constructors, fields, destructors etc.
- The program then goes to the main() method and it is the first function that is called when program execution begins. It is declared as public to make it accessible from just about anywhere in the program.

- It is the entry point of a C# program.
- Every C# program will contain at least one class with at least one Main method.
- Then the code inside the function is executed.
- The **WriteLine** function displays text on the console.
  - Console is a class of the System namespace, which has a WriteLine() method that is used to print text.
  - All the C# components have their own access level that can be controlled by defining the scope of accessibility for the member objects inside the class by using access modifiers.
  - A data type tells the compiler what kind of value a variable will hold. C# has several built-in data types such as String, Integer, Float, Boolean, etc.

---

5) Working with code files, projects & solutions:

➢ Understanding structure of solution:

A solution is a structure for organizing projects in Visual Studio. The solution maintains the state information for projects in two files:

- .sln file →text-based, shared

- The environment calls ReadSolutionProps to read in the .sln file.
- When a user opens a solution, the environment cycles through the preSolution, Project, and postSolution information in the .sln file to load the solution, projects within the solution, and any persisted information attached to the solution.
- The .sln file contains text-based information the environment uses to find and load the name-value parameters for the persisted data and the project VSPackages it references.

- .suo file →binary, user-specific solution options

- The solution user options (.suo) file contains per-user solution options. This file should not be checked in to source code control.
- The solution user options file is used to store user preference settings, and is created automatically when Visual Studio saves a solution.
- We can save user information into streams with the name of the stream being the key that will be used to identify the information in the .suo file.
- When the environment opens a .suo file, it enumerates all currently loaded VSPackages.

➢ Understanding structure of project:

- **Program.cs** is the entry Point for application. Like any application the execution of Application starts from public static void Main(string[] args){} This Program. cs creates the application Host. It configures the settings file like appsettings.
- All projects have a list of **references**, which is shown in the Solution Explorer directly beneath the project nodeEach item in this list represents a reference to some external component that your project uses.
- With .NET projects, it causes Visual Studio .NET to tell the compiler that your code uses the component, enabling you to use the types it contains—if you don't have the appropriate references in your project, you will get compiler errors complaining that a type or namespace cannot be found.
- While adding a project reference automatically adds a **dependency**, you can also manage dependencies directly.
- Dependencies are solution-scoped properties that affect the build order of the projects in your solution.
- If Project A depends on Project B, VS.NET will always make sure Project B has been built before building Project A.

- Startup.cs file is a replacement of Global.asax file in ASP.NET Web Form application.
- Startup.cs file is entry point, and it will be called after Program.cs file is executed at application level. It handles the request pipeline. Startup class triggers the second the application launches.

➢ Different types of file extension:

- SOURCE TYPE FILES:
  - .asmx → Deployment file.
  - .asp → Active Server Page file.
  - .cpp, .c → Main source code files for your application.
  - .disco → The dynamic discovery document file. Handles XML Web service discovery.
  - .h → header file

- RESOURCE TYPE FILES:
  - .cur → Cursor bitmap graphic file.
  - .ico → Icon bitmap graphic file.
  - .rc, .rc2 → Resource script files to generate resources.
  - .txt → A text file, usually the "readme" file.

- PROJECT FILES:
  - .atp → Application template project file.
  - .dbp → Database project file.
  - .exe, .dll → Executable or dynamic-link library files.
  - .vap → A Visual Studio Analyzer project file.
  - .vbp, .vip, .vbproj → The Visual Basic project file.
  - .vmx → The macro project file.
  - .vup → The utility project file.

- SOLUTION FILES:
  - .sln → solution file
  - .suo → solution options file

---

6)Understanding datatypes & variables with conversion:

➢ Base Datatypes:

The datatypes in C#, are categorized into the following types −

1. Value types
2. Reference types
3. Pointer types

- **Value type** variables can be assigned a value directly. They are derived from the class System.ValueType.
- Some examples are int, char, and float, double, long, sbyte, short, ulong etc.
- The **reference types** do not contain the actual data stored in a variable, but they contain a reference to the variables.
- Example of built-in reference types are: object, dynamic, and string.
- The **Object Type** is the ultimate base class for all data types in CTS.
- Any type of value can be stored in the **dynamic data type** variable.
- The **String Type** allows you to assign any string values to a variable.
- **Pointer type** variables store the memory address of another type.

➢ Datatype Conversion:

Two types of conversion: Implicit conversion
Explicit conversion

IMPLICIT CONVERSION:

- It is done by the compiler.

- It is done when there is no loss of information.
- It there is no possibility of throwing exception during the conversion.
- For example: converting int to float will not lose any data and no exception will be thrown.
- But while converting float to int, we lose the fractional part and also possibility of overflow exception.
- Therefore in this case explicit conversion is required.

EXPLICIT CONVERSION:

- It is done by the user.
- Convert class can be used for explicit conversion.
- For example : Convert.ToInt32() can be used to convert string value to integer.
- Int.parse() or float.parse() is used for explicit conversion.

➢ Boxing/Unboxing:

- The process of Converting a **Value Type (char, int etc.) to a Reference Type(object)**is called **Boxing**.
- Boxing is implicit conversion process in which object type is used.
- The Value type is always stored in Stack. The Referenced Type is stored in Heap.
- **Example :**
  int num = 2; // 2 will assigned to num
  Object Obj = num; // Boxing
- First declare a value type variable (num), which is integer type and assigned it with value 2. Now create a references object type (obj) and applied Explicit operation which results in num value type to be copied and stored in object reference type obj

- The process of converting **reference type into the value type** is known as **Unboxing**.
- It is explicit conversion process.
- **Example :**
  int num = 2; // value type is int and assigned value 2
  Object Obj = num; // Boxing
  int i = (int)Obj; // Unboxing

- Declaration a value type variable (num), which is integer type and assigned with integer value 2. Now, create a reference object type (obj).The explicit operation for boxing create an value type integer i and

applied casting method. Then the referenced type residing on Heap is copy to stack.

7)Understanding Decision making & statements:

➢ If/else:

- 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
  }else
  {
  //code
  }

➢ else/if :

- Multiple else if statements can be used after an if statement.
- It will only be executed when the if condition evaluates to false.
- Syntax:

```
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 if(condition3)
{
  /* code block to be executed when
     condition1 evaluates to flase
      condition2 evaluates to false
      condition3 evaluates to true */
}
```

➢ Nested if / else:

- In this there are multiple if/else statement in nested form.
- It makes the code more readable and complicated programs can be done easily.
- Syntax:

```
  if(condition1)
{
  if(condition2)
   {
      /*code block to be executed when
     condition1 and condition2 evaluates to true */
 }
   else if(condition3)
   {
      if(condition4)
      {
         /*code block to be executed when
only condition1, condition3, and condition4 evaluates to true */
      }
      else if(condition5)
      {
         /*code block to be executed when
            only condition1, condition3, and condition5 evaluates to true */
      }
      else
      {
         /*code block to be executed when
            condition1, and condition3 evaluates to true
            condition4 and condition5 evaluates to false  */
      } } }
```

➢ **Switch case:**

- The switch statement can be used instead of if else statement when we want to test a variable against three or more conditions.
- The switch statement starts with the switch keyword
- It contains a match expression: switch(match expression)
- A case must be specified with the unique constant value and ends with the colon :
- The default label will be executed if no cases executed.
- Syntax:

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

default:
statement(s) to be executed;
break;
}
```