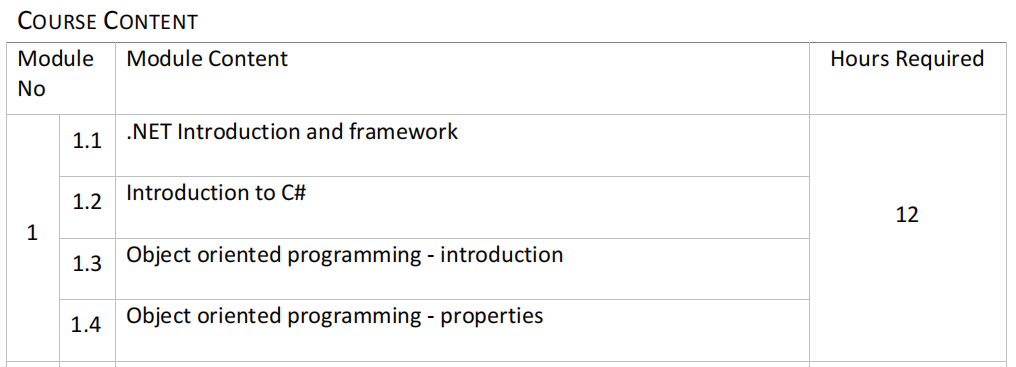
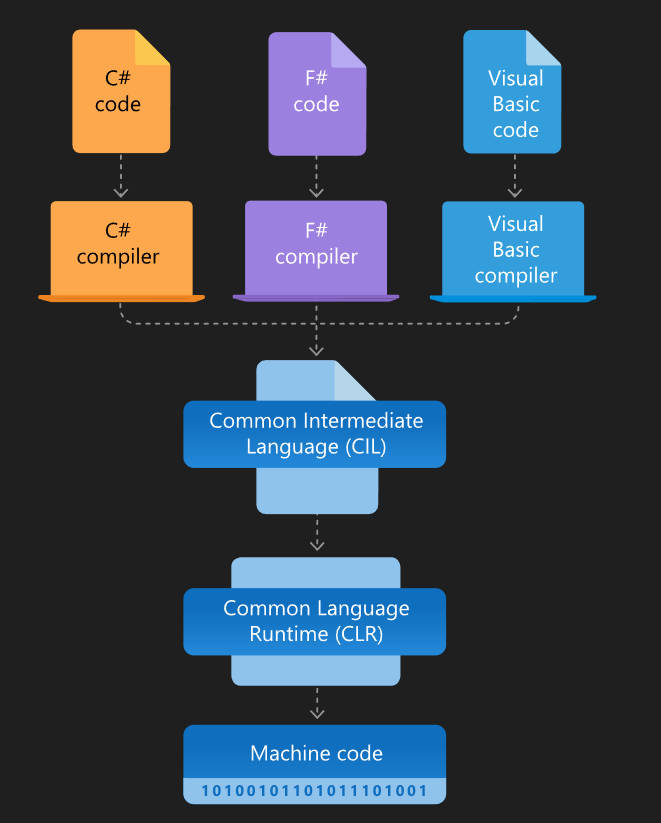
**Microsoft .NET Framework using C#**



1. ***.NET Introduction and framework*** :

* It is a comprehensive software development platform **developed by Microsoft**.
* .NET is a **cross-platform** implementation for running websites, services, and console apps on Windows, Linux, and macOS.
* It is a **virtual machine** that provide a common platform to run an application that was built using the different language such as C#, VB.NET, Visual Basic, etc.
* .NET framework is used for developing and creating applications such as:
* Console applications
* Web applications
* Windows forms applications
* Web services
* Event-driven applications.
* The **main objective** of this framework is to develop an application that can run on the windows platform.

### KEY Components of .NET Framework:



1. **Common Language Runtime (CLR):**

Manages the execution of .NET programs, providing services such as memory management, security, and exception handling.

1. **.NET Framework Class Library (FCL):**

A vast collection of reusable classes, interfaces, and value types. Includes APIs for reading and writing files, connecting to databases, drawing, and more.

**KEY Features**:

1. **Multiple Language Support.**
2. **Cross-Platform Development.**
3. **Versatile Application Development: (**A vast collection of reusable classes, interfaces, and value types.)

**Characteristics**



**KEY Advantages** :

1. **Security :**

Features such as **code access security (CAS)** and **role-based security**.

1. **Performance :**

**JIT compilation** and **optimization techniques** enhance the speed and efficiency of applications.

1. **Integration :**

Smooth integration with other Microsoft products like SQL Server, SharePoint, and Office.

1. **Introduction to C#** :

* C# is an **object-oriented programming language** created by **Microsoft** that runs on the .NET Framework.
* FOUNDER : “****Anders Hejlsberg”.****
* It is used to develop **web apps,** **desktop apps**, **mobile apps**, **games** and much more.

**KEY Concepts:**

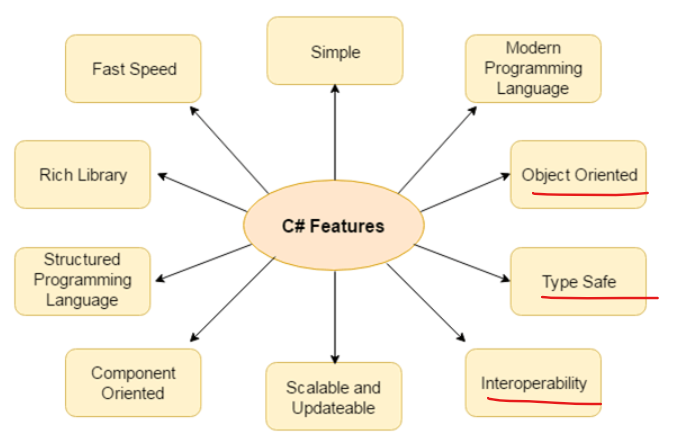
· **Syntax and Structure :** Similar to other C-based languages.

· **Type Safety :** C# enforces strict type checking, reducing bugs and enhancing reliability.

· **Memory Management : Automatic garbage collection** .

· **Language Interoperability:** C# can interoperate with other languages on the .NET platform, thanks to the CLR.

**KEY Features:**



1. **Object Oriented.**
2. **Type Safe:**

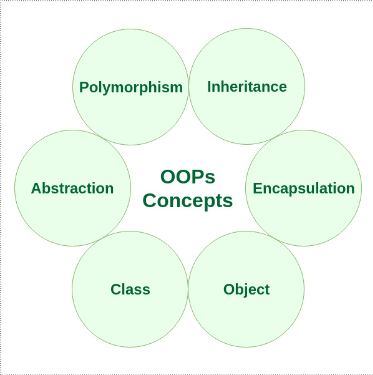
can only access the memory location that it has permission to execute. Therefore it improves a security of the program.

1. **Interoperability.**
2. **Scalable and Updatable.**
3. **Object Oriented Programming - Introduction** :

* Object-Oriented Programming (OOP) is a **programming paradigm based on the concept of "objects**," which can contain data and code to manipulate that data. C# is an object-oriented language.

1. **Object Oriented Programming - Properties** :

**Characteristics:**



* **Class : [ is a user-defined data type that has **data members** and **member functions**.]**
* **Object : [Instance of a Class]**
* **Abstraction : [Displaying only essential information and hiding the details.]**
* **Inheritance : [Acquire the properties of parent class into a child class]**
* **Polymorphism : [Having many forms.An operation may exhibit different behaviors in different instances.]**
* **Encapsulation : [binding together the data and the functions]**

**Benefits :**

* **Modularity :**

Code is organized into discrete objects, making it easier to manage and understand.

* **Reusability .**
* **Scalability.**

**LAB PROGRAMs :**

**DAY 1:BASICS**

|  |
| --- |
| namespace sampleproject2  {      internal class Program      {          static void Main(string[] args)          {              Console.WriteLine("Hello, World!");              long myNum = 15000000;              Console.WriteLine(myNum);              int myInt = 9;              double myDouble = myInt;              Console.WriteLine(myDouble);              Console.WriteLine(myInt);              //Automatic casting:Int to Double              myInt = (int)myDouble;              Console.WriteLine(myDouble);              //Program :To display Day              Console.WriteLine("\nEnter a num b/w 1 to 7 to find the day:" );              int input = int.Parse(Console.ReadLine());                switch (input)              {                  case 1:                      Console.WriteLine("Monday");                      break;                  case 2:                      Console.WriteLine("Tuesday");                      break;                  case 3:                      Console.WriteLine("Wednesday");                      break;                  case 4:                      Console.WriteLine("Thursday");                      break;                  case 5:                      Console.WriteLine("Friday");                      break;                  case 6:                      Console.WriteLine("Saturday");                      break;                  case 7:                      Console.WriteLine("Sunday");                      break;                  default:                      Console.WriteLine("Invalid Input");                      break;              }              //PROGRAM : Odd or Even              Console.WriteLine("Enter the num to find Odd or Even : ");              int num = int.Parse(Console.ReadLine());                if(num%2 == 0)              {                  Console.WriteLine("Even");              }              else if(num == 0)              {                  Console.WriteLine("Number is Neither Odd Nor Even");              }              else              {                  Console.WriteLine("ODD");              }              //Program : Factorial              int i, fact = 1, num2;              Console.WriteLine("\nEnter the number to find factorial");              num2 = int.Parse(Console.ReadLine());              for(i = 1; i <= num2; i++)              {                  fact \*= i;              }              Console.WriteLine("\nFactorial = ", fact);            }      }  } |

**DAY 2: Exercise 1**

|  |
| --- |
| Extend the Mobile class and create an Android class that implements multiple interfaces. Create two interfaces:   * **ICamera**: With a method **TakePhoto()**. * **IGPS**: With a method **GetLocation()**.   The Android class should implement these interfaces in addition to inheriting from the Mobile class.  After creating the Android class, write a program to:   1. Create an instance of the Android class. 2. Set the attributes brand, model, and osVersion. 3. Call the **ChargeBattery** method to set the battery level to 75%. 4. Call the**MakeCall** method to simulate making a call. 5. Call the **TakePhoto** method to simulate taking a photo. 6. Call the **GetLocation** method to simulate getting the current location. 7. Print the details of the Android device using a method. |

**File : “Mobile.cs”**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ProjectMobile  {      public class Mobile      {          //Attributes :          public string brand;          public string model;          public int battery\_level;          //Constructor          public Mobile(string brand, string model)          {              this.brand = brand;              this.model = model;              this.battery\_level = 0; //Setting intial battery\_level a 0%            }          //Method to simulate making a call :          public void makeCall(string phoneNumber)          {              Console.WriteLine(phoneNumber);              Console.WriteLine("\nMaking a Call.....");              useBattery(10); //10% Battery is consumed when making a call          }          public void chargeBattery(int amount) {              battery\_level += amount;              if (battery\_level > 100) {                  battery\_level = 100;              }              Console.WriteLine($"Battery charged to {battery\_level}%");          }          public void useBattery(int amount)          {              battery\_level -= amount;              if (battery\_level < 0) {                  battery\_level = 0;              }              Console.WriteLine($"Battery level is now {battery\_level}%");          }          //method to print mobile details          public void printDetails()          {              Console.WriteLine($"Brand : {brand}");              Console.WriteLine($"Model: {model}");              Console.WriteLine($"Battery Level : {battery\_level}%");          }      }    } |

**File : “Android.cs”**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ProjectMobile  {      interface ICamera      {          void TakePhoto();      }      interface IGps      {          void GetLocation();      }      public class Android : Mobile, ICamera, IGps      {          public void TakePhoto()          {              Console.WriteLine("Pic ");          }          public void GetLocation()          {              Console.WriteLine("Get loc");          }          public Android(string brand,string model) : base(brand, model)          {          }          public void installApp(string appName)          {              if (battery\_level > 20) {                  Console.WriteLine($"Installing {appName} app....");                  useBattery(5);              }              else              {                  Console.WriteLine("Battery is not suffiecient to Install an App.Please charge your phone ");              }          }      }  } |

**File : “Program.cs”**

|  |
| --- |
| namespace ProjectMobile  {      internal class Program      {          static void Main(string[] args)          {                Mobile myPhone = new Mobile("Apple", "Iphone 15");              myPhone.chargeBattery(50);              myPhone.makeCall("8157847663");              myPhone.useBattery(20);              myPhone.printDetails();              Android android1 = new Android("Samsung", "S21 FE");              android1.chargeBattery(50);              android1.makeCall("6238000260");              android1.useBattery(10);              android1.printDetails();              android1.installApp("Valorant");              android1.GetLocation();              android1.TakePhoto();            }      }  } |

**DAY 3: Exercise 2(Abstraction)**

|  |
| --- |
| IMG_256 |

**File : “Mobile.cs”**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace secondab  {     public abstract class Mobile      {          protected string brand;          protected string model;          protected int battery\_level;          protected Mobile(string brand,string model,int battery\_level) {              this.brand = brand;              this.model = model;              this.battery\_level = battery\_level;          }          public abstract void StartDevice();          public abstract void   UseDevice();          public void ShowDetails()          {              Console.WriteLine($"Brand : {brand}");              Console.WriteLine($"Model: {model}");              Console.WriteLine($"Battery Level : {battery\_level}%");          }      }  } |

**File : “Featurephone.cs”**

|  |
| --- |
| using Microsoft.VisualBasic;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace secondab  {      public class Featurephone : Mobile      {          int buttonCount;          public Featurephone(string brand, string model, int battery\_level, int buttonCount) : base(brand, model, battery\_level)          {              this.buttonCount = buttonCount;          }          public override void StartDevice()          {              Console.WriteLine("featurephone device start");          }          public override void UseDevice()          {              Console.WriteLine(" featurephone use device");          }          public void ShowDetails()          {              base.ShowDetails();              Console.WriteLine($"Button count:{buttonCount}");          }      }  } |

**File : “Program.cs”**

|  |
| --- |
| namespace secondab  {      internal class Program      {          static void Main(string[] args)          {              Smartphone s1 = new Smartphone("Samsung", "S21FE", 35, "v4");              s1.StartDevice();              s1.UseDevice();              s1.ShowDetails();              Featurephone f1 = new Featurephone("Samsung", "Note 8", 20, 10);              f1.StartDevice();              f1.UseDevice();              f1.ShowDetails();          }      }  } |

**DAY 3: Property(using get and set) and Enum**

[Note: I have included both Property and Enum’s code together in one program.]

* **private** variables can only be accessed within the same class.
* However, sometimes we need to access them - and it can be done with **properties**.
* A property is like a **combination of a variable and a method**, and it has two methods: a **get** and a **set** method:

**File : “Person.cs”**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace property\_get\_set  {      public class Person      {          private string name; //field          public string Name          { //Property              get { return name; }              set { name = value; }          }      }  } |

**File : “Program.cs”**

|  |
| --- |
| using System.Security.Cryptography.X509Certificates;  namespace property\_get\_set  {      public class Program      {          //ENUM:enum is a special "class" that represents a group of constants          private enum Gender //enum          {              Male,              Female,              Others          }          static void Main(string[] args)          {              //GET SET property Part :              Person pobj1 = new Person();              pobj1.Name = "Anshad";              Console.WriteLine($"Name : {pobj1.Name}");              //ENUM part              Gender p1 = Gender.Male;              Console.WriteLine($"Gender of p1 is {p1}");              Gender p2 = Gender.Female;              Console.WriteLine($"Gender of p2 is {p2}");            }      }  } |

**DAY 4: Delegates**

* Delegate is a **reference to the method**.
* It works like function pointer in C and C++.

**File : “Program.cs”**

|  |
| --- |
| namespace DelegateExample  {      //Delegate Declaration:      delegate int ArithOp(int x, int y);      delegate void MDelegate();      public class Program      {          static void Main(string[] args)          {              //Delegate instances:              ArithOp operation1 = new ArithOp(MathOperation.Add);              ArithOp operation2 = new ArithOp(MathOperation.Sub);              //Invoking delegates:              int result1 = operation1(200, 100);              int result2 = operation2(200, 100);              Console.WriteLine("Result 1 = " + result1);              Console.WriteLine("Result 2 = " + result2);              //Multicast Delegate :-              MDelegate m1 = new  MDelegate(DM.Display);              MDelegate m2 = new MDelegate(DM.Print);              MDelegate m3 = m1 + m2;              MDelegate m4 = m2 + m1;              MDelegate m5 = m3 - m2;              m3();              m4();              m5();          }      }  } |

**File : “MathOperation.cs”**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace DelegateExample  {      public class MathOperation      {          //Delegate Method definition:          public static int Add(int a, int b)          {              return (a + b);          }          public static int Sub(int a, int b)          {              return (a - b);          }      }  } |

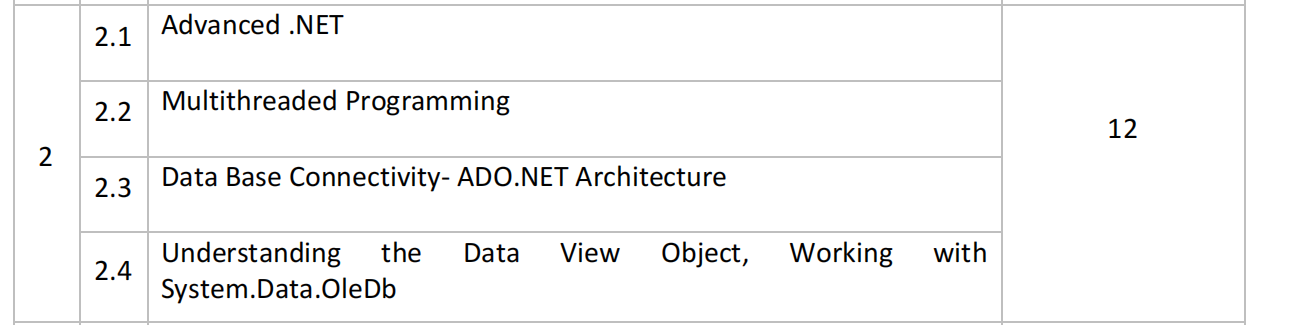
**File : “DM.cs”**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace DelegateExample  {      public class DM      {          public static void Display()          {              Console.WriteLine("NEW DELHI");          }          public static void Print()          {              Console.WriteLine("NEW YORK");          }      }  } |

**Partial Class**

* It allows us to write partial ****class, interface, struct and method**** in two or more separate source files. **All parts are combined** when the application is compiled.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Runtime.Intrinsics.X86;  using System.Text;  using System.Threading.Tasks;  namespace PartialClass  {      public partial class Books      {          public string Author\_name;          public string bookno;          public void printDetails()          {              Console.WriteLine($"Author Name : {Author\_name} \n Book number : {bookno}");          }      }      public partial class Books      {          public string Bookname;          public string publishername;          public void PrintPublisherDetails(string bn,string pn)          {              this.Bookname = bn;              this.publishername = pn;              Console.WriteLine($"Book name : {Bookname} \n Publisher Name : {publishername}");          }      }      public partial class Books      {          public static void Main(string[] args)          {              Books b1 = new Books();              Console.WriteLine("Enter the Author name");              b1.Author\_name = Console.ReadLine();              Console.WriteLine("Enter the Book no");              b1.bookno = Console.ReadLine();              b1.printDetails();              Books b2 = new Books();              b2.PrintPublisherDetails("Dilsha", "a002");              //Console.WriteLine($"{Author\_name} \n {bookno}");          }      }  } |



1. **Advanced .NET** :

**String Handling**

* String is an object of “System.string”.

**Methods**

* **Clone() :**
* **Compare(String,String)**

**Exception Handling(**try,catch,finally,throw**)**

**Exception Classess:**

**System.IO.IOEXCEPTION**

**System.IndexOutOfRangeException**

**System.ArrayTypeMismatchException**

**System.DivideByZeroException**