**Session 11: Class Objects and Types of Class.**

**Class: -** user defined blue print or prototype or Template. Its datatype. It represent entity or object. Define state and behaviour of an object. Using class we can create object. Class can be provide derived and child class. Class keyword is used to declare and define class body.its **refrence type.** Default access modifier is internal. Default access modifier for method and variable is private. It contain memebrs of element called class member.

**Nested Class:-** Class within Class.

**Object:** represents the real life entities. Each non static method is having object. New keyword is used to create instance or Object. When program runs object has been created. It’s a tangible entity. It contain identity, state, and behaviour.

* **Types of Class:** - **Abstract Class: --** not associated with any specific instace. it provide common definition for all.Declared using abstract class. Cannot create object of abstract class. If u want to use it must be inherited in subclass. It contain abstract and non-abstract method. Abstract method doesn’t contain body. For this method body will assign trough child class. Method can be have implementation or non-implementation. It contain only one subclass. Inside abstract class method cannot be private. It has modifiers for method and properties. It have constants and filed.
* **Partial Class:-** it allows dividing properties method and event into multiple source file at compile time these files are combined. Partial keyword is used. All parts have same accessibility (access modifier). Its in same assembly and nampesapce. Different part have different base types so final class will inherit all the base types. Class contain so many lines of code. If one part is inheriting all class can be inherited internally.

* **Sealed Class:-** it restrict the user to inherit the class. Sealed keyword is used. No class can be derived from sealed class. Method can be sealed but cannot be **override.** Class is specialised. Used for security purpose. Prevent the customisation library. Access modifiers can be applied. Object of the class can be created for using sealed class. Third party cannot be inherit the class.
* **Static Class:-** cannot be instantiated (Cannot create object).cannot inherit. Static keyword is used. Static class can contain only static member. Can be accessed by ClassName. When we want data should be constant.

S11\_\_ClassAndTypes.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace OOPS\_\_AllSession

{

class S11\_\_ClassAndTypes

{

//\*\*\*\*\*\*\*\*\*\* Nested Class \*\*\*\*\*\*\*\*\*\*\*

public abstract class BasicDetail // Base Class

{

public string firstName;

public int age { get; set; }

public char gender;

public long phoneNumber;

public abstract void AbstractClassAndDisplayDetail(); // Abstract Method

}

public class StudentDetail : BasicDetail //Hirarchical Inheritace. more than one child class.

{

public int rollNo;

public int fess;

public override void AbstractClassAndDisplayDetail()

{

Console.Write("Enter Student Name: ");

this.firstName = Console.ReadLine();

Console.Write("Enter Student Age: ");

this.age = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Student Gender: ");

this.gender = Convert.ToChar(Console.ReadLine());

Console.Write("Enter Student PhoneNumber: ");

this.phoneNumber = Convert.ToInt64(Console.ReadLine());

Console.Write("Enter Student RollNo: ");

this.rollNo = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Student Fess: ");

this.fess = Convert.ToInt32(Console.ReadLine());

Console.WriteLine($"\n\*\*\*\*\*\*\*\*Student Detail Are:\*\*\*\*\*\*\*\*\*\*\n Student Name : {this.firstName}\n Student Age: {this.age}\n Student Gender: {this.gender}\n Student Phone: {this.phoneNumber}\n Student RollNo: {this.rollNo}\n Student Fees: {this.fess}");

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\* Partial Class \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public partial class PartialDetail

{

public string firstName { get; set; }

public int age { get; set; }

public char gender { get; set; }

public long phoneNumber { get; set; }

}

public partial class PartialDetail

{

public void PartialClass()

{

Console.Write("Enter Student Name: ");

this.firstName = Console.ReadLine();

Console.Write("Enter Student Age: ");

this.age = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Student Gender: ");

this.gender = Convert.ToChar(Console.ReadLine());

Console.Write("Enter Student PhoneNumber: ");

this.phoneNumber = Convert.ToInt64(Console.ReadLine());

Console.WriteLine($"\n\*\*\*\*\*\*\*\*Student Detail Are:\*\*\*\*\*\*\*\*\*\*\n Student Name : {this.firstName}\n Student Age: {this.age}\n Student Gender: {this.gender}\n Student Phone: {this.phoneNumber}");

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\* Sealed Class \*\*\*\*\*\*\*\*\*\*\*\*\*

sealed class BaseClass //Parent Class

{

static int num1, num2;

public static void Sealed\_Data()

{

Console.Write($"Addition Is: {num1 + num2}");

}

}

class DerrivedClass //: BaseClass(Inheritance is not allowed in sealed class) //Derived ==> Child Class

{

public void Sealed\_DerrivedData()

{

BaseClass.Sealed\_Data();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Static Class \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

static class StaticClass

{

//int number = 20; //Error

//public int numb = 40; // Error

//internal int num = 60; //Error

//static const int nums= 60; //Error

static int numbers = 30;

//public void Instance\_\_NonStatic\_Method()

//{

// this.number = 40;

//}

public static void Static\_Method()

{

int numss = StaticClass.numbers;

Console.WriteLine("Value is :" + numss);

}

}

}

OopsSessions.cs

using OOPS\_\_AllSession;

using System;

using static OOPS\_\_AllSession.S11\_\_ClassAndTypes;

namespace Oops\_\_AllSession

{

class OopsSessions

{

static void Main(string[] args)

{

Console.WriteLine("\*\*\*\*\*\*\*\*Welcome To Main Method\*\*\*\*\*\*\*\*\*\*\*");

//S11\_\_ClassAndTypes classType = new S11\_\_ClassAndTypes();

StudentDetail abstractClass = new StudentDetail();

//abstractClass.AbstractClassAndDisplayDetail();

PartialDetail partialClass = new PartialDetail();

//partialClass.PartialClass();

StaticClass.Static\_Method();

//SealedClass sealedClass = new SealedClass();

}

}

}