1.Write a c# program using constructors, this keyword, static keyword and access modifiers

using System;

namespace Studentspace

{

public class Student {

static int id = 215; //declaring a static variable

public string name;

public int marks;

protected int stud\_hltktno; // declaring a protected variable

private int rollno; // declaring a private variable

public int RollNo { // accessing private variable

get

{

return rollno;

}

set {

rollno = value;

}

}

public Student(string \_Name,int \_Marks)

{

this.name = \_Name;

this.marks = \_Marks;

}

public void Display\_details() {

Console.WriteLine(" Student Id is: {0}" ,Student.id); //accessing static variable

Console.WriteLine(" Student Name is: {0}" ,this.name);

Console.WriteLine(" Student Marks are:{0}" ,this.marks);

}

}

/\* public class Student2 : Student {

public void printHallticket()

{

Student2 s2 = new Student2();

s2.stud\_hltktno = 1223415; //accessing protected variable in derived class

Console.WriteLine("Student Hall Ticket No{0}:",s2.stud\_hltktno);

}

} \*/

public class Sample

{

static void Main() {

Console.WriteLine("======= Student Details =======");

Student s1 = new Student("Rajesh",500); //creating object and invoking constructor

s1.Display\_details(); // accessing student class method

//s2.printHallticket(); // we can't access this method because Sample class not derived from Student class

}

}

}

**Output:**

Text

Description automatically generated

**Internal Access Modifier Program:**

using System;

namespace AssemblyOne

{

public class Assemblyclassone

{

internal int Id = 101; // declaring an internal variable

}

public class Assemblytwoclass

{

public void display()

{

Assemblyclassone a1 = new Assemblyclassone();

Console.WriteLine(a1.Id); // 2nd class can also access ID Variable

}

}

}

**Protected Internal Modifier Program:**

using System;

namespace AssemblyOne

{

public class Assemblyclassone

{

protected internal int Id = 101; // declaring an protected internal variable

}

public class Assemblytwoclass

{

public void display()

{

Assemblyclassone a1 = new Assemblyclassone();

Console.WriteLine(a1.Id); // 2nd class can also access ID Variable

}

}

}

**2nd Program:**

using System;

using AssemblyOne; // using Assemblyone reference in Assemblytwo

namespace Assemblytwo

{

public class Assemblyclassthree : Assemblytwoclass // deriving class from base class of another Assembly

{

public void display1()

{

Assemblyclassthree a2 = new Assemblyclassthree();

Console.WriteLine(a2.Id); // accessing ID variable of

}

}

}

2. Write a c# program using encapsulation using properties (backing field) and auto-implemented properties

Ex1: Encapsulation by using set() and get() Methods

using System;

public class Encaps

{

private int \_id;

public void setID(int Id)

{

if (Id < 0)

{

Console.WriteLine(" Student Id can't be Negative");

}

this.\_id = Id;

}

public int getID()

{

return this.\_id;

}

public class Sample

{

public static void Main()

{

Encaps e1 = new Encaps();

e1.setID(215); //setting a value for Id

Console.WriteLine(" Student ID is: {0}", e1.getID()); //printing Id value by using get Method

}

}

}

**Output:**

Graphical user interface, text

Description automatically generated

Ex2 : Encapsulation by using set and get accessors (Properties)