Assignment - 1 Basics of C#

Q.1. Write a program to create a class department which has department information and also create an employee class which inherits the department and displays all the information.

Aim:To create a class department and display its information.

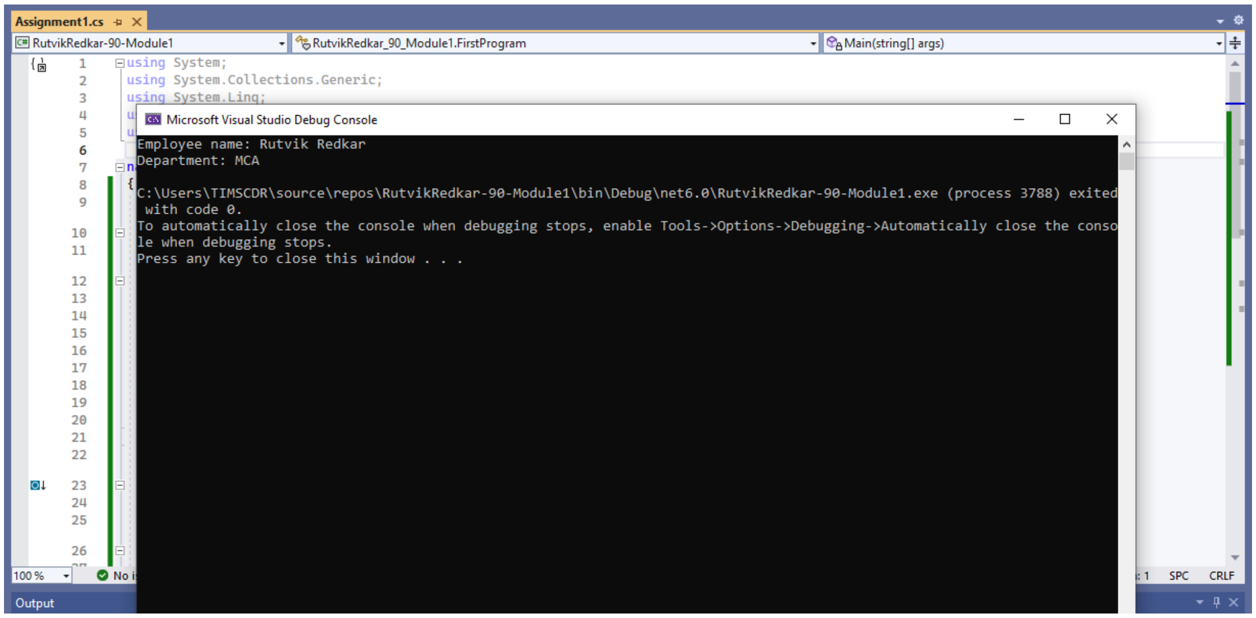
Objective: To understand the functionality of class and inheritance

Theory : In C# inheritance allows us to create a new class from an existing class. It is a key feature of Object-Oriented Programming (OOP). The class from which a new class is created is known as the base class (parent or superclass). And, the new class is called derived class (child or subclass)The derived class inherits the fields and methods of the base class. This helps with the code reusability in C#.

Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace RutvikRedkar_90_Module1
{

    class FirstProgram
    {
        static void Main(string[] args)
        {

            Emp emp = new Emp();
            emp.getEmp("Rutvik Redkar");
            emp.printDept();
            Console.ReadKey();

        }
    }

    public class Dept
    {
        public string dept = "MCA";
        public void printDept()
        {
            Console.WriteLine("Department: " + dept);
        }
    }

    public class Emp : Dept
    {
        public void getEmp(string name)
        {
            Console.WriteLine("Employee name: " + name);
        }
        public void getDept()
        {
            string dpname = dept;
            Console.WriteLine("Deptartment name: " + dpname);
        }
    }

}
```

Output :


Assignment1.cs
RutvikRedkar-90-Module1
RutvikRedkar_90_Module1.FirstProgram
Main(string[] args)
Microsoft Visual Studio Debug Console
100 %
Output

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
Employee name: Rutvik Redkar
Department: MCA

C:\Users\TIMSCDR\source\repos\RutvikRedkar-90-Module1\bin\Debug\net6.0\RutvikRedkar-90-Module1.exe (process 3788) exited
 with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Conclusion:
Hence we learnt how to create and implement inheritance

Q.2. Write a program to create an abstract class named plan which has an abstract method that sets rates of plan and also a method which calculates total bill by taking a number of seats for different plans. Create two different planes domestic and international which have different rates.

Aim: To write a program to create and implement abstract class

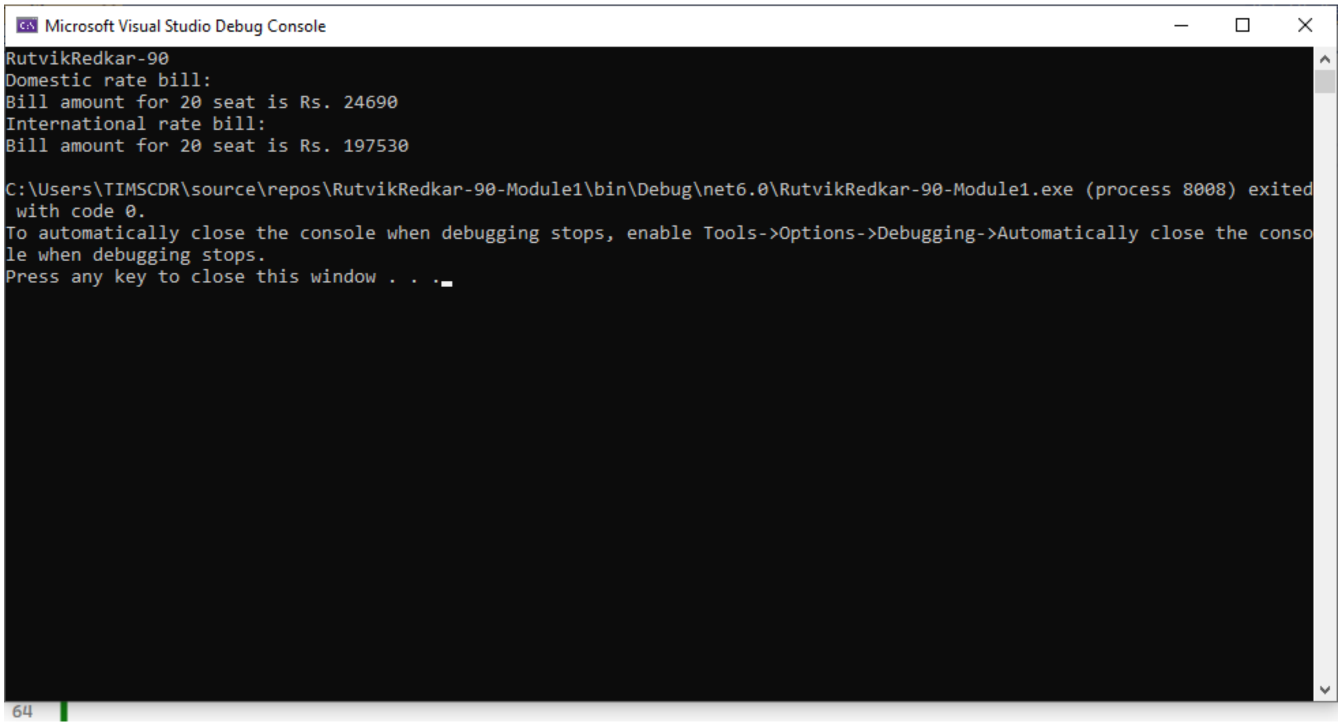
Objective : create an abstract class for hiding some data

Theory : Abstraction in C# is the process to hide the internal details and show only the functionality. The abstract modifier indicates the incomplete implementation. The keyword abstract is used before the class or method to declare the class or method as abstract. Also, the abstract modifier can be used with indexers, events, and properties.
A method that is declared abstract, has no "body" and is declared inside the abstract class only. An abstract method must be implemented in all non-abstract classes using the override keyword. After overriding, the abstract method is in the non-Abstract class. We can derive this class in another class, and again we can override the same abstract method with it

Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace RutvikRedkar_90_Module1
{
    public abstract class plan
    {
        protected double rate;
        public abstract void setRate();
        public void calcBill(int seats)
        {
            Console.WriteLine("Bill amount for " + seats + " seat is Rs. " + rate * seats);
        }

    }

    class domestic : plan
    {
        public override void setRate()
        {
            rate = 1234.5;
        }
    }

    class international : plan
    {
        public override void setRate()
        {
            rate = 9876.5;
        }
    }

    internal class Assignemnet2
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("RutvikRedkar-90");
            domestic d = new domestic();
            d.setRate();
            international i = new international();
            i.setRate();

            Console.WriteLine("Domestic rate bill: ");

            d.calcBill(20);
            Console.WriteLine("International rate bill: ");
            i.calcBill(20);

            Console.ReadKey();
        }

    }
}
```

Output :


Microsoft Visual Studio Debug Console

```
RutvikRedkar-90
Domestic rate bill:
Bill amount for 20 seat is Rs. 24690
International rate bill:
Bill amount for 20 seat is Rs. 197530

C:\Users\TIMSCDR\source\repos\RutvikRedkar-90-Module1\bin\Debug\net6.0\RutvikRedkar-90-Module1.exe (process 8008) exited
 with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Conclusion:

Hence we learnt how to create and implement abstract classes in C#

Q.3..Write a program to create an interface iControl which has two properties Height and Width. Create a class ListBox and CheckBox to implement the interface

Aim: To write a program to create interface and then implement it

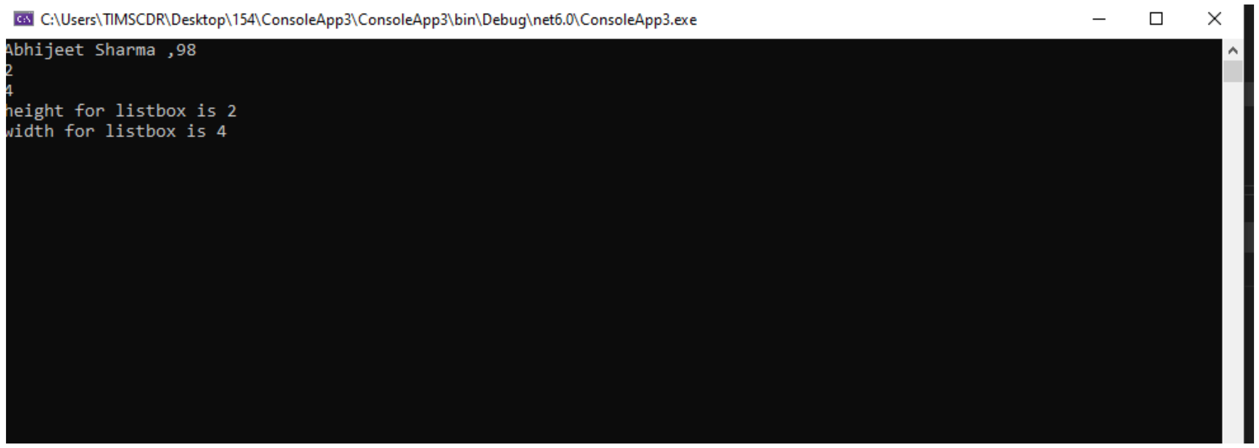
Objective : Create an interface iControl which has two properties and implement the interface

Theory : Another way to achieve abstraction in C#, is with interfaces.An interface is a completely "abstract class", which can only contain abstract methods and properties (with empty bodies):

Code :

```
using System;


namespace ConsoleApp3
{
   internal class Class1
   {
      static void Main(string[]args)
      {
         Console.WriteLine("Abhijeet Sharma ,98 ");
         ListBox l = new ListBox();
         CheckBox c = new CheckBox();
         int lh, lw, ch, cw;
         lh = Convert.ToInt16(Console.ReadLine());
         lw=Convert.ToInt16(Console.ReadLine());
         l.height = lh;
         l.width=lw;
         Console.WriteLine("height for listbox is " + l.height);
         Console.WriteLine("width for listbox is " + l.width);


         ch = Convert.ToInt16(Console.ReadLine());
         cw= Convert.ToInt16(Console.ReadLine());
         c.height = ch;
         c.width = cw;
         Console.WriteLine("height for checkbox is " + c.height);
         Console.WriteLine("width for checkbox is " + c.width);
      }
   }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp3
{
    internal interface Interface1
    {
        int height
        {
            get;
            set;
        }
        int width
        {
            get;
            set;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp3
{
    internal class ListBox : Interface1
    {
        int h, w;


        public int height
        {
            get { return h; }
            set { h = value; }
        }
        public int width
        {
            get { return w; }
            set { w = value; }
        }


    }
}

using System;


namespace ConsoleApp3
{
    internal class CheckBox:Interface1
    {
        int h, w;
         public int height
        {
            get { return h; }
            set { h = value; }
        }
          public int width
        {
            get { return w; }
            set { w = value; }
        }

    }
}
```

Output :


C:\Users\TIMSCDR\Desktop\154\ConsoleApp3\ConsoleApp3\bin\Debug\net6.0\ConsoleApp3.exe

```
Abhijeet Sharma ,98
2
4
height for listbox is 2
width for listbox is 4
```

Conclusion:
Hence we learnt how to create and implement interfaces