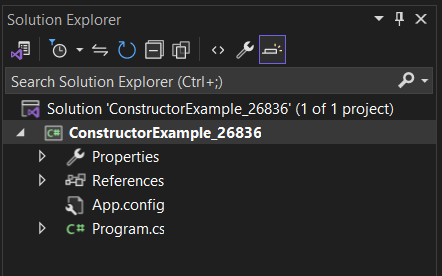
|  |  |
| --- | --- |
| **Lab No.** | **Lab Titles** |
| 01 | Write a C# program to perform addition using constructor. |
| 02 | Write a C# program to initialize and display jagged array elements with sum of each row. |
| 03 | Write a C# program to initialize and display 2D array elements with sum of each row. |
| 04 | Write a C# program to calculate area of rectangle using single interface. |
| 05 | Write a C# program to calculate area and paint cost of rectangle using multiple inheritance. |
| 06 | Write a C# program to call base class constructor using “base” keyword. |
| 07 | Write a C# program using hierarchical inheritance using virtual method. |
| 08 | Write a C# program which takes the length and breadth of 2 rectangle as input and store using an array object. Also print area of each rectangle. |
| 09 | Write a C# program to find the position of a specified word in a given string. |
| 10 | Write a C# program to count the total number of words and characters in a string. |
| 11 | Write a C# program to count the number of alphabets, digits and special characters in string. |
| 12 | Write a C# program to count the number of vowels or consonants in a string. |
| 13 | Write a C# program to calculate sum and difference of two digit using multicast delegates. |
| 14 | Write a C# program to achieve polymorphism using delegates. |
| 15 | Create student class with properties for id, name, gender, and address. It then creates a list<student> to store instances of this class the program adds 10 student, prints and searches for Student by their address using FindStudentByAddesss function. It should print the result of the search. |
| 16 | Write a C# program to raise an ApplicationException using Custom MyException class. |
| 17 | Write a C# program to show insert and select student record with given table (tblStudent) with fields (int id, nvarchar(50) name, nvarchar(50) address, nvarchar(50) gender). Also display total no of student from table. |
| 18 | Write a C# program to show insert and fetch student record by Gender from given table (tblStudent) with fields (int id, nvarchar(50) name, nvarchar(50) address, nvarchar(50) gender). |
| 19 | Write a C# program to perform (CRUD) operation from given table (tblStudent) with fields (int id, nvarchar(50) name, nvarchar(50) address, nvarchar(50) gender). |
| 20 | Write a simple program to create generic class with generic constructor, generic member variable, generic property and generic method. |
| 21 | Write an ASP.NET CORE program to explain working of memory caching for state management. |
| 22 | Create a new project and add a controller with endpoints to read and write values into cookies. |
| 23 | Write a program to validate form using a jQuery. |
| 24 | Create a controller with endpoints to set and read a value from the session. |
| 25 | Write a program in C# to implement a generic list data structure. |
| 26 | Write a generic method called ‘Swap’ that takes two parameters of the same type and swaps their values. |
| 27 | Write an ASP.NET Core program to demonstrate use of hidden fields. |
| 28 | Write a C# Program to insert and display records in table name tblEmployee (id int, name nvarchar(50), address nvarchar(50), salary decimal(18,0), gender nvarchar50) using EntityFramework. |
| 29 | Write a C# program to compute aggregate salary of 5 employee and then display employee record in descending order with respect to employee salary using EntityFramework. |
| 30 | Write a program to select employees whose salary is greater than 20000 and whose address is Kathmandu using EntityFramework. |

**Lab 1: Write a C# program of addition using constructor**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConstructorExample\_26836

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter first: ");

int a = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter second: ");

int b = Convert.ToInt32(Console.ReadLine());

AddTwoDigit obj = new AddTwoDigit(a,b);

Console.WriteLine(obj.Add());

Console.ReadLine();

}

}

}

public class AddTwoDigit

{

int first = 0;

int second = 0;

public AddTwoDigit(int x, int y)

{

first = x;

second = y;

}

public int Add()

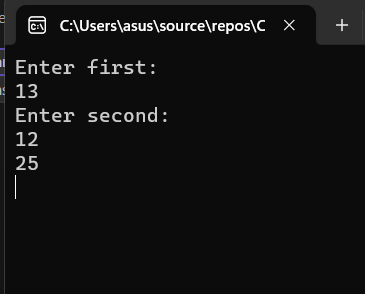
{

return first+second;

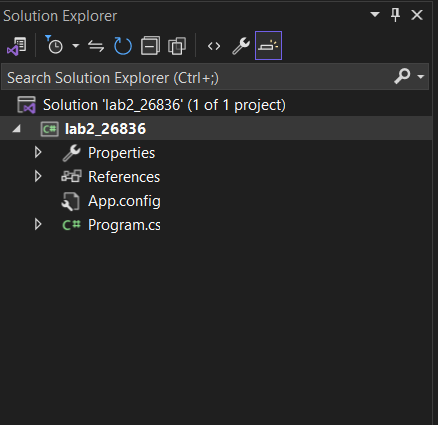
}

}

**Output:**



**Lab2: Write a C# program to find sum of the jugged array.**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab2\_26836

{

internal class Program

{

static void Main(string[] args)

{

//Write C# program to initialize and print jagged array

int[][] arr = new int[2][] { new int[] { 92, 93, 94 }, new int[] { 85, 66, 87, 88 } };

int sum = 0;

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < arr[i].Length; j++)

{

Console.WriteLine(arr[i][j] + " ");

sum += Convert.ToInt32(arr[i][j]);

}

Console.Write("=" + sum);

sum = 0;

Console.WriteLine();

}

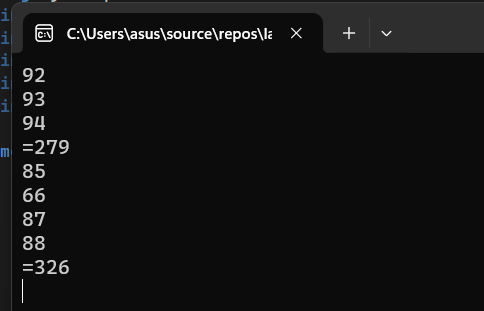
Console.ReadLine();

}

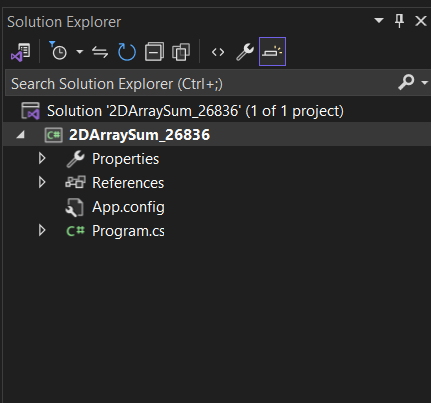
}

}

**Output:**



**Lab 3: Write a program to initialize and display 2D array and sum of each row.**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab4

{

internal class Program

{

static void Main(string[] args)

{

Rectangle rect = new Rectangle();

Console.WriteLine(rect.Add(2, 3));

Console.WriteLine(rect.Sub(4, 2));

Console.ReadLine();

}

}

public interface IA

{

int Add(int x, int y);

}

public interface IB

{

int Sub(int x, int y);

}

public class Rectangle : IA, IB

{

public int Add(int x, int y)

{

return x + y;

}

public int Sub(int x, int y)

{

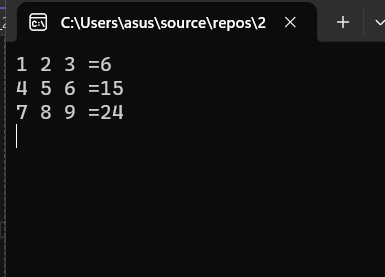
return (x - y);

}

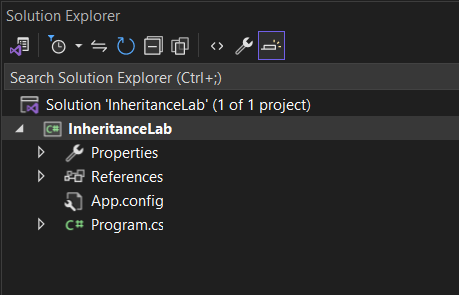
}

}

**Output:**



**Lab 4: Write a C# to calculate area of rectangle using single interface.**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace InheritanceLab

{

internal class Program

{

static void Main(string[] args)

{

Rectangle rect = new Rectangle();

rect.setHeight(5);

rect.setWidth(5);

Console.WriteLine("Area of rectangle : {0}", rect.CalculateArea());

Console.ReadLine();

}

}

}

class Shape

{

protected int width;

protected int height;

public void setWidth(int w)

{

width = w;

}

public void setHeight(int h)

{

height = h;

}

}

class Rectangle: Shape

{

public int CalculateArea()

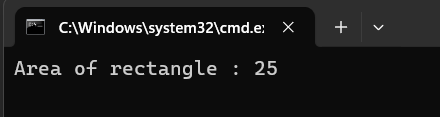
{

return (width \* height);

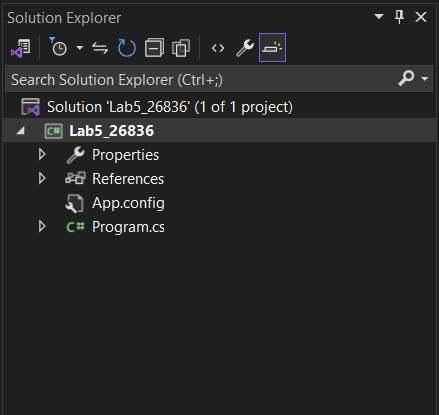
}

}

**Output:**

****

**Lab 5: Write a C# program to calculate area and paint cost of rectangle using multiple inheritance.**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_26836

{

class Program

{

static void Main(string[] args)

{

Rectangle rect = new Rectangle();

int area;

rect.setHeight(7);

rect.setWidth(23);

area = rect.getArea();

Console.WriteLine("Total Area:"+ rect.getArea());

Console.WriteLine("Total Paint Cost:"+ rect.getCost(area));

Console.ReadLine();

}

}

}

class Shape

{

protected int width;

protected int height;

public void setWidth(int w)

{

width = w;

}

public void setHeight(int h)

{

height = h;

}

}

//Base Class

public interface PaintCost

{

int getCost(int area);

}

//Derived Class

class Rectangle : Shape, PaintCost

{

public int getArea()

{

return (width \* height);

}

public int getCost(int area)

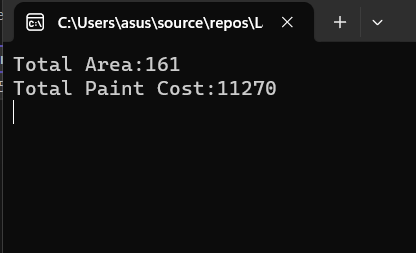
{

return area\*70;

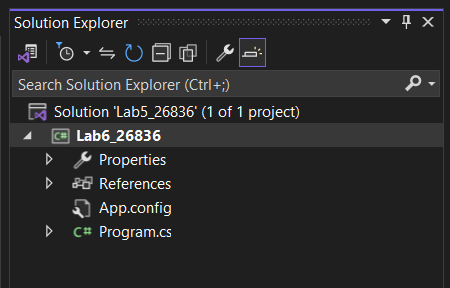
}

}

**Output:**



**Lab 6: Write a C# program to display car model and speed using base keyword**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

namespace paintcost\_26808

{

class Program

{

static void Main(string[] args)

{

Car car = new Car();

Console.WriteLine("Car Model: {0}, Speed {1}", car.model, car.speed);

Console.WriteLine();

Console.ReadLine();

}

}

}

class Vehicle {

public int speed;

public Vehicle() {

this.speed = 5;

}

}

class Car: Vehicle

{

public string model;

public Car(): base()

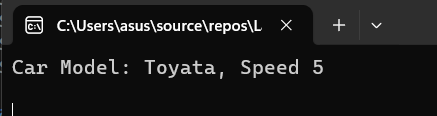
{

this.model = "Toyata";

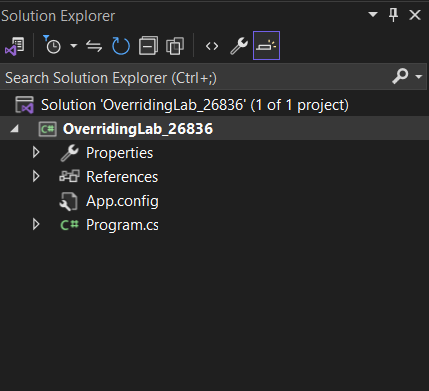
}

}

**Output:**



**Lab 7: Write a C# program using hierarchical inheritance using virtual method.**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OverridingLab\_26836

{

internal class Program

{

static void Main(string[] args)

{

God g = new Ram();

God g1 = new Shiva();

Console.WriteLine(g.GodName());

Console.WriteLine(g1.GodName());

Console.ReadLine();

}

}

}

public class God

{

public virtual string GodName() {

return "God";

}

}

public class Ram : God

{

public override string GodName()

{

return "Ram";

}

}

public class Shiva : God

{

public override string GodName()

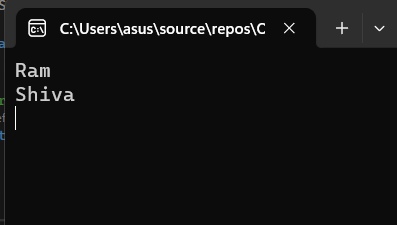
{

return "Shiva";

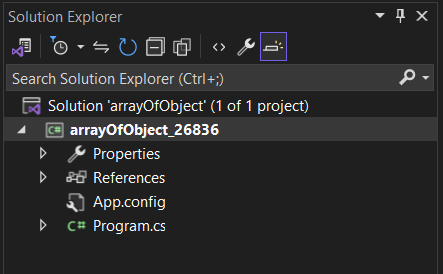
}

}

**Output:**



**Lab 8: Write a C# program to illustrate hierarchical inheritance using virtual method.**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace arrayOfObject

{

internal class Program

{

static void Main(string[] args)

{

Rectangle[] rect = new Rectangle[2];

for(int i = 0; i < 2; i++) {

Console.WriteLine("Rectangle :" + (i + 1));

Console.WriteLine("Enter Length");

int l = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter breadth");

int b = Convert.ToInt32(Console.ReadLine());

rect[i] = new Rectangle(l, b);

}

for(int i = 0;i<2;i++)

{

Console.WriteLine("Area of Rectangle : {0}= {1}", (i+1), rect[i].getArea());

}

Console.ReadLine();

}

}

public class Rectangle

{

private int length = 0;

private int breadth = 0;

public Rectangle(int l, int b) {

this.length = l;

this.breadth = b;

}

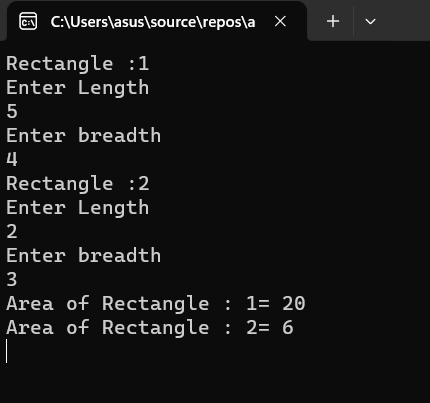
public int getArea()

{

return length\*breadth;

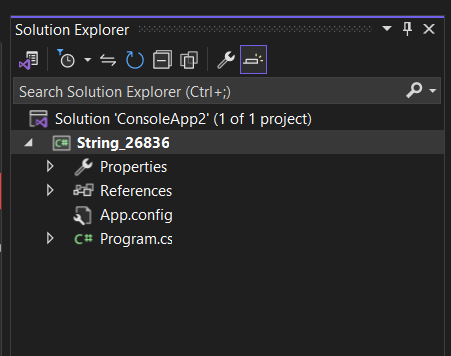
}

}

}

**Output:**

**Lab 9: Write a C# program to find the position of a specified word in a given string.**

****

**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace String\_26836

{

class Program

{

static void Main(string[] args)

{

Finder f = new Finder();

string str1 = "The cat sat on the mat";

Console.WriteLine("Original String:" + str1);

Console.Write("Position of the word 'sat' is:" + f.GetResult(str1, "sat"));

Console.Write("\nPosition of the word 'the' is:" + f.GetResult(str1, "the"));

Console.Write("\nPosition of the word 'cat' is:" + f.GetResult(str1, "cat"));

Console.ReadLine();

}

}

}

public class Finder

{

public int GetResult(string text, string word)

{

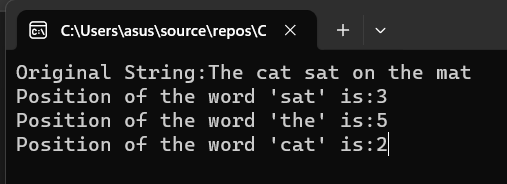
string[] str = text.Split(' ');

int a = Array.IndexOf(str, word) + 1;

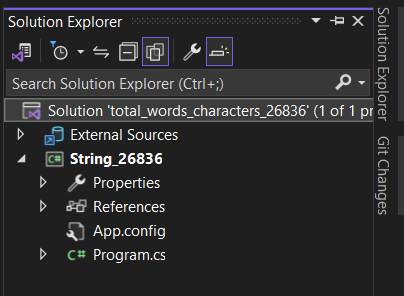
return a;

}

}

****

**Lab 10: Write a C# program to count the total number of words and characters in a string.**

****

**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace total\_words\_characters\_26836

{

class Program

{

static void Main(string[] args)

{

string inputString = "The quick brown fox jumps over the lazy dog.";

int wordCount = CountWords(inputString);

int charCount = inputString.Length;

Console.WriteLine("String: " + inputString);

Console.WriteLine("Total number of words: " + wordCount);

Console.WriteLine("Total number of characters: " + charCount);

Console.ReadLine();

}

static int CountWords(string input)

{

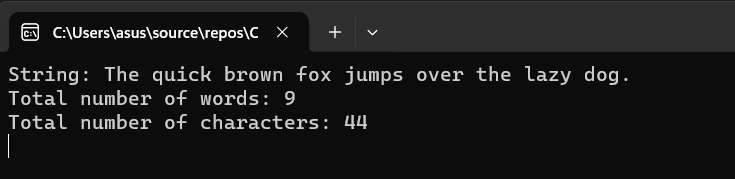
string[] words = input.Split(new char[] { ' ', '\n', '\t', '\r' }, StringSplitOptions.RemoveEmptyEntries);

return words.Length;

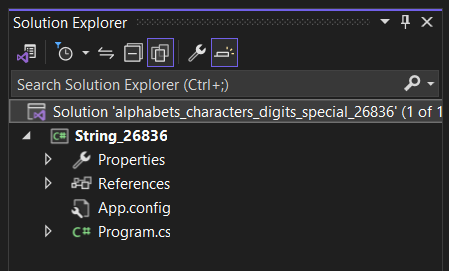
}

}

}

****

**Lab 11: Write a C# program to count the number of alphabets, digits and special characters in string.**

****

**Source Code:**

using System;

namespace alphabets\_characters\_digits\_special\_26808

{

class Program

{

static void Main(string[] args)

{

string inputString = "Hey!Would you like to have 1 cup of tea??";

int alphabetCount = 0;

int digitCount = 0;

int specialCharCount = 0;

foreach (char character in inputString)

{

if (Char.IsLetter(character))

{

alphabetCount++;

}

else if (Char.IsDigit(character))

{

digitCount++;

}

else

{

specialCharCount++;

}

}

Console.WriteLine("Number of alphabets:" + alphabetCount);

Console.WriteLine("Number of digits:" + digitCount);

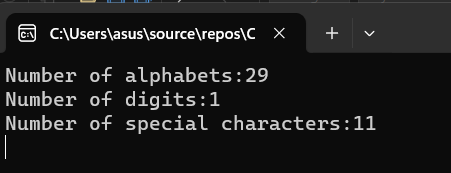
Console.WriteLine("Number of special characters:" + specialCharCount);

Console.ReadLine();

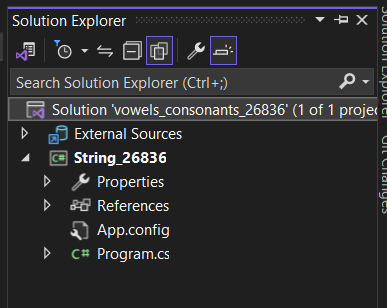
}

}

}

****

**Lab 12: Write a C# program to count the number of vowels or consonants in a string.**

****

**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace vowels\_consonants\_26808

{

class Program

{

static void Main(string[] args)

{

string inputString = "Checking of vowel and consonants";

inputString = inputString.ToLower();

int vowelCount = 0;

int consonantCount = 0;

foreach (char character in inputString)

{

if (Char.IsLetter(character))

{

if ("aeiou".Contains(character))

{

vowelCount++;

}

else

{

consonantCount++;

}

}

}

Console.WriteLine("Input String:" + inputString);

Console.WriteLine("Number of vowels:" + vowelCount);

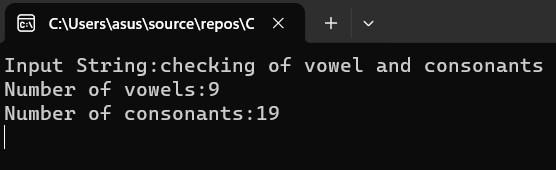
Console.WriteLine("Number of consonants:" + consonantCount);

Console.ReadLine();

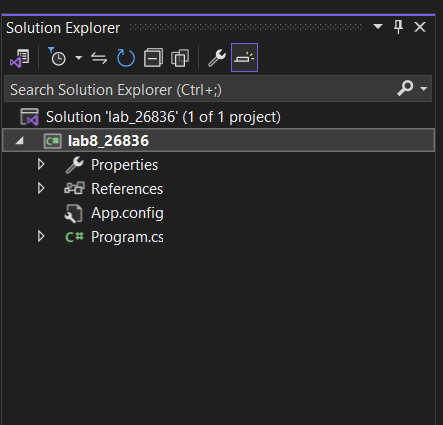
}

}

}

****

**Lab 13: Write a C# program to find sum and difference of two digit using multicast delegate.**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace @delegate

{ // c# program to find sum and difference of two digit using multicast delegate

public delegate void delmethod(int x, int y);

public class TestMultipleDelegate

{

public void plus\_Method1(int x, int y)

{

Console.WriteLine("This is plus mode");

Console.WriteLine(x + y);

}

public void sub\_Method2(int x, int y)

{

Console.WriteLine("This is subtract mode");

Console.WriteLine(x - y);

}

}

internal class Program

{

static void Main(string[] args)

{

TestMultipleDelegate obj = new TestMultipleDelegate();

delmethod del = new delmethod(obj.plus\_Method1);

//Multicast

del += new delmethod(obj.sub\_Method2);

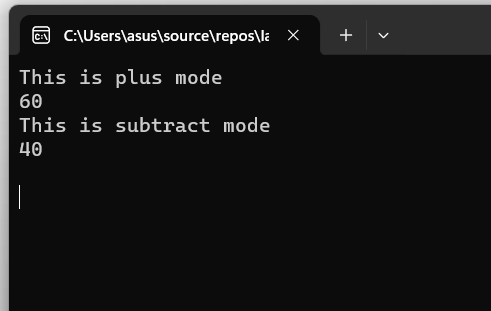
del(50, 10);

Console.WriteLine();

Console.ReadLine();

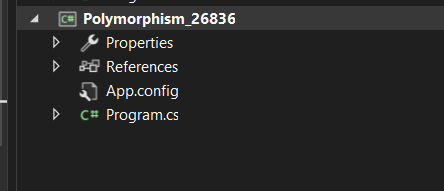
}

}

}

**Output:**

**Lab 14: Write a C# program to show polymorphism using delegates.**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Polymorphism\_delegates\_26836

{

class Program

{

static void Main(string[] args)

{

GodDelegate godMethod;

godMethod = new Ram().GodName;

godMethod();

godMethod = new Sita().GodName;

godMethod();

Console.ReadLine();

}

}

delegate void GodDelegate();

class Ram

{

public void GodName()

{

Console.WriteLine("God Ram");

}

}

class Sita

{

public void GodName()

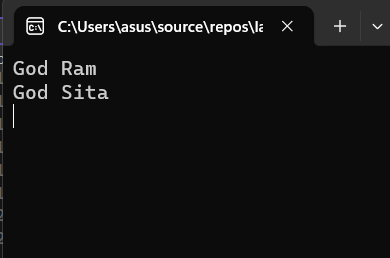
{

Console.WriteLine("God Sita");

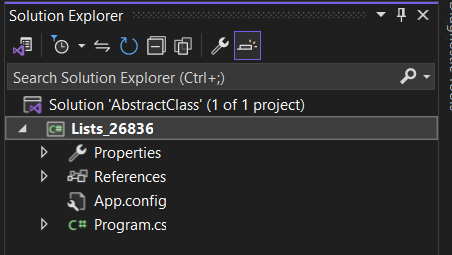
}

}

}



**Lab 15: Create student class with properties for id, name, gender, and address. It then creates a list<student> to store instances of this class the program adds 10 student, prints and searches for Student by their address using FindStudentByAddesss function. It should print the result of the search.**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AbstractClass

{

internal class Program

{

static void Main(string[] args)

{ //create student class with properties for id, name, gender, and address.

//it then creates a list<student> to store instances of this class

// the program adds 10 student, prints and searches for Student by their address using FindStudentByAddesss function

//finally it prints the result of the search.

List<Student> st = new List<Student>();

st.Add(new Student() { Id = 1, Name = "Sunil Chaudary", Address = "Kathmandu", Gender = "Male" });

st.Add(new Student() { Id = 2, Name = "Sunil jha", Address = "Kathmandu", Gender = "Male" });

st.Add(new Student() { Id = 3, Name = "Sunil shrextha", Address = "Bhaktapur", Gender = "Male" });

st.Add(new Student() { Id = 4, Name = "anil Chaudary", Address = "Lalitpur", Gender = "Male" });

st.Add(new Student() { Id = 5, Name = "Sanil Chaudary", Address = "Dang", Gender = "Male" });

st.Add(new Student() { Id = 6, Name = "Manil Chaudary", Address = "Kathmandu", Gender = "Male" });

st.Add(new Student() { Id = 7, Name = "Kunal Chaudary", Address = "Kathmandu", Gender = "Male" });

st.Add(new Student() { Id = 8, Name = "Rannil Chaudary", Address = "Kathmandu", Gender = "Male" });

st.Add(new Student() { Id = 9, Name = "Punil Chaudary", Address = "Bhaktapur", Gender = "Male" });

st.Add(new Student() { Id = 10, Name = "Cunil Chaudary", Address = "Kathmandu", Gender = "Male" });

List<Student> filterStudent = new List<Student>();

foreach (Student item in st)

{

if(item.Address == "Kathmandu")

{

filterStudent.Add(item);

}

}

foreach (var item in filterStudent)

{

Console.WriteLine("Name: {0} Address: {1} Gender: {2}", item.Name, item.Address, item.Gender);

}

Console.ReadLine();

}

public static List<Student> FindStudentByAddress(List<Student> students, string searchAddress)

{

List<Student> filterStudent = new List<Student>();

foreach (Student item in students)

{

if (item.Address == searchAddress)

{

filterStudent.Add(item);

}

}

return filterStudent;

}

}

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

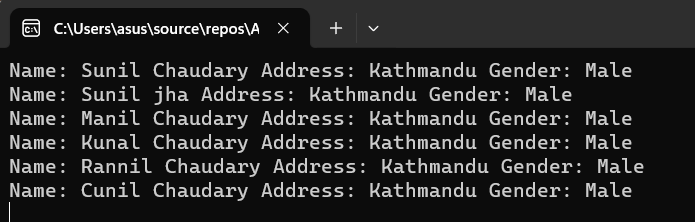
public string Address { get; set; }

public string Gender { get; set; }

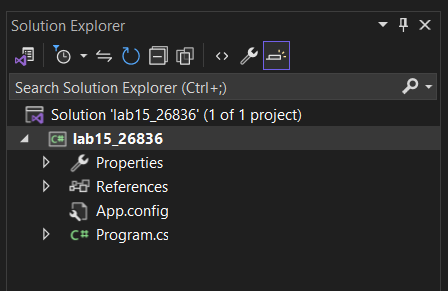
}

}

**Output:**



**Lab 16: Write a C# program to implement custom exception.**



**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab15\_26836

{

class MyException : ApplicationException

{

public void MyCustomException() {

Console.WriteLine("Exception occured, divisor should not be 0");

}

}

internal class Program

{

static void Main(string[] args)

{

//Custom exception lab

int d, div, res;

div = Int32.Parse(Console.ReadLine());

d = Int32.Parse(Console.ReadLine());

try

{

if(div == 0)

{

throw new MyException();

}

res = d / div;

Console.WriteLine("Result: {0}", res);

Console.ReadLine();

}

catch (MyException e)

{

e.MyCustomException();

}

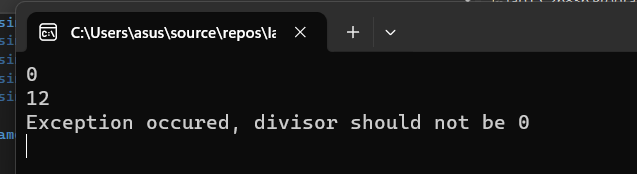
Console.ReadLine() ;

}

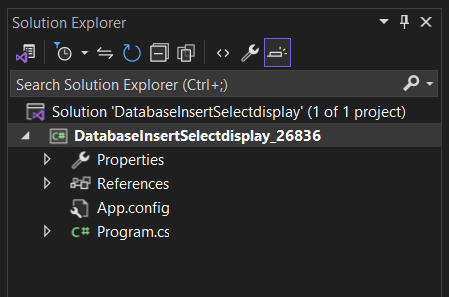
}

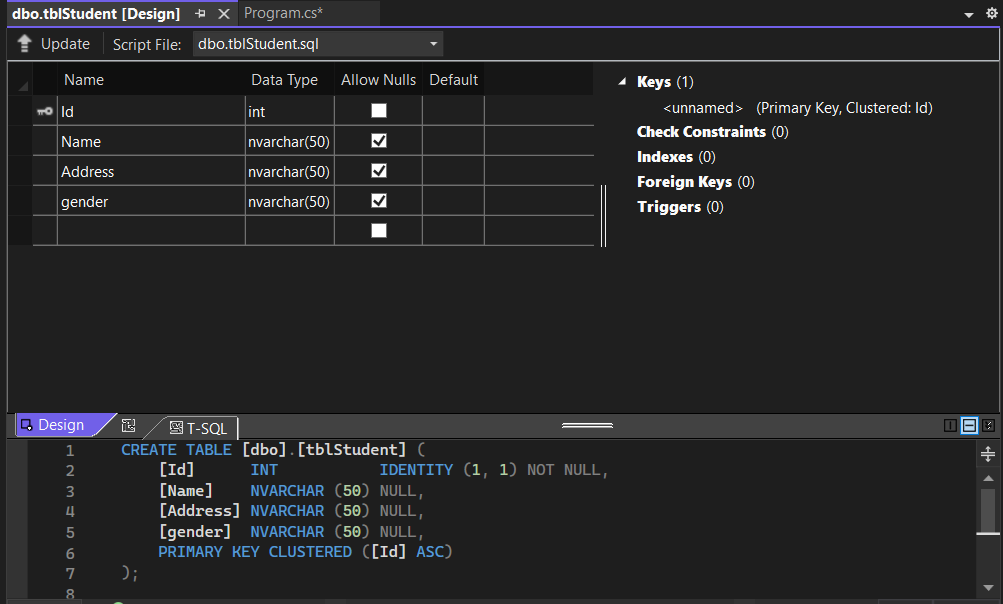
}

**Output:**



**Lab 17:** **Write a C# program to show insert and select student record with given table (tblStudent) with fields (int id, nvarchar(50) name, nvarchar(50) address, nvarchar(50) gender). Also display total no of student from table.**

****

****

**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient; //MS SQL server

using System.Data;

namespace DatabaseInsertSelectdisplay

{

//Write C# program to show inset and select student record with given table(tblStudent) with fields(int id, nvarchar(50)gender.

//Also display total no of students from table

internal class Program

{

static void Main(string[] args)

{

Student st = new Student();

//st.InsertStudent();

st.DisplayStudentData();

Console.ReadLine();

}

}

public class Student

{

public void InsertStudent()

{

string conStr = @"Data Source = (LocalDB)\MSSQLLocalDB; Database=SamriddhiDB; Integrated Security=true";

SqlConnection con = new SqlConnection(conStr);

SqlCommand cmd = new SqlCommand("insert into tblStudent values(@a,@b,@c)", con);

cmd.Parameters.AddWithValue("@a", "Suni Chaudhary");

cmd.Parameters.AddWithValue("@b", "Kathmandu");

cmd.Parameters.AddWithValue("@c", "Female");

con.Open();

cmd.ExecuteNonQuery();

con.Close();

Console.WriteLine("Record Inserted");

}

public void DisplayStudentData()

{

string conStr = @"Data Source = (LocalDB)\MSSQLLocalDB; Database=SamriddhiDB; Integrated Security=true";

SqlConnection con = new SqlConnection(conStr);

SqlCommand cmd = new SqlCommand("select \* from tblStudent", con);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

for(int i = 0; i < dt.Rows.Count; i++)

{

Console.WriteLine("Name: {0}, Address: {1}, Gender: {2}",

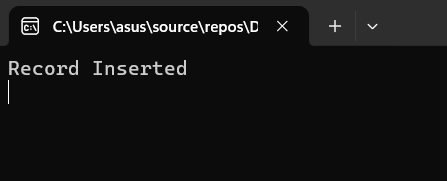
dt.Rows[i]["Name"], dt.Rows[i]["Address"], dt.Rows[i]["Gender"]);

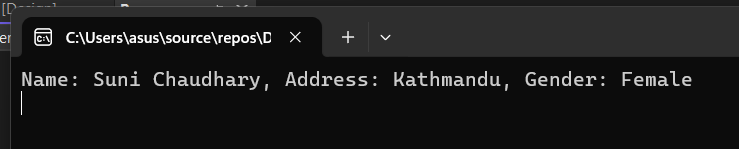
}

}

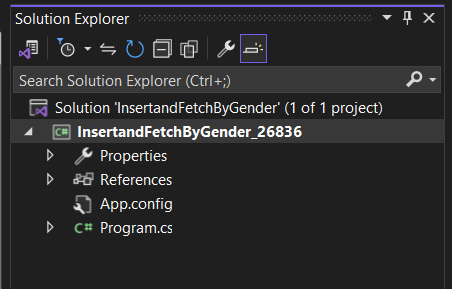
}

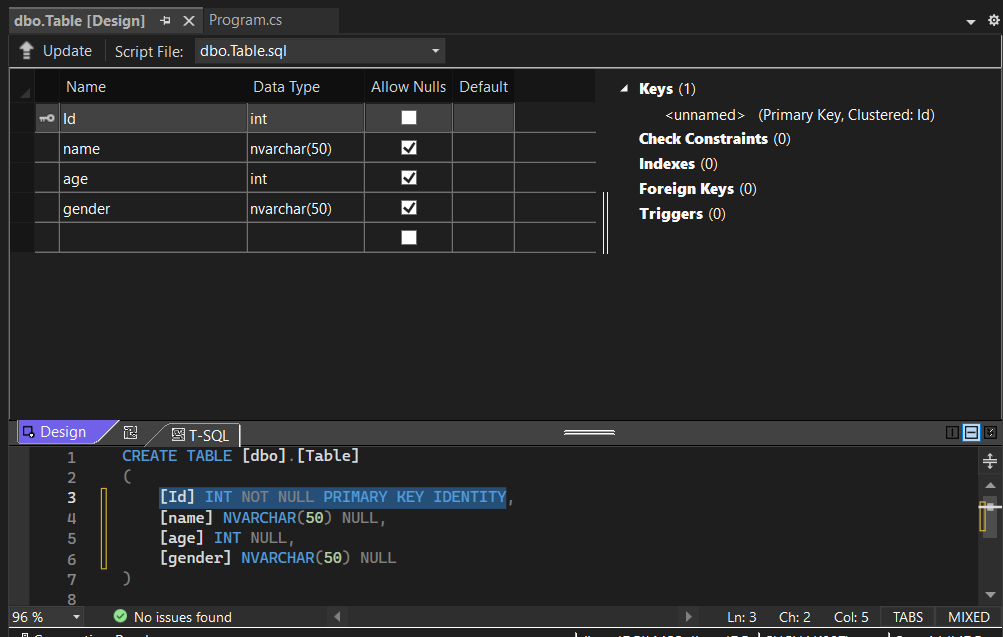
}





**LAB 18: Write a C# program to show insert and fetch student record by Gender from given table (tblStudent) with fields (int id, nvarchar(50) name, int age, nvarchar(50) gender).**

****

****

**Source Code:**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

internal class Program

{

static void Main(string[] args)

{

StudentInsert\_SelectRecord obj = new StudentInsert\_SelectRecord();

Console.Write("Enter Name:");

string name = Console.ReadLine();

Console.Write("Enter Address:");

string address = Console.ReadLine();

Console.Write("Enter Gender:");

string gender = Console.ReadLine();

obj.Insert(name, address, gender);

Console.WriteLine("Record Inserted");

Console.WriteLine();

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("All Student By Gender");

DataTable dt = obj.GetAllStudentByGender("Male");

for (int i = 0; i < dt.Rows.Count; i++)

{

string n = dt.Rows[i]["Name"].ToString();

string a = dt.Rows[i]["Address"].ToString();

string g = dt.Rows[i]["Gender"].ToString();

Console.WriteLine("Name:{0} Address:{1} Gender:{2}", name, address, gender);

}

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("Total No of Student: " + dt.Rows.Count);

Console.ReadLine();

}

}

public class StudentInsert\_SelectRecord

{

public void Insert(string name, string address, string gender)

{

string connStr = @"Data Source=(localdb)\MSSqlLocalDB; Database=SamriddhiDB; Integrated Security=true";

SqlConnection con = new SqlConnection(connStr);

string sql = "insert into tblStudent values(@name,@address,@gender)";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@name", name);

cmd.Parameters.AddWithValue("@address", address);

cmd.Parameters.AddWithValue("@gender", gender);

con.Open();

cmd.ExecuteReader();

con.Close();

}

public DataTable GetAllStudentByGender(string gender)

{

string connStr = @"Data Source=(localdb)\MSSqlLocalDB; Database=SamriddhiDB; Integrated Security=true";

SqlConnection con = new SqlConnection(connStr);

string sql = "select \*from tblStudent where Gender=@gender";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@gender", gender);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();//can hold data in tabular format

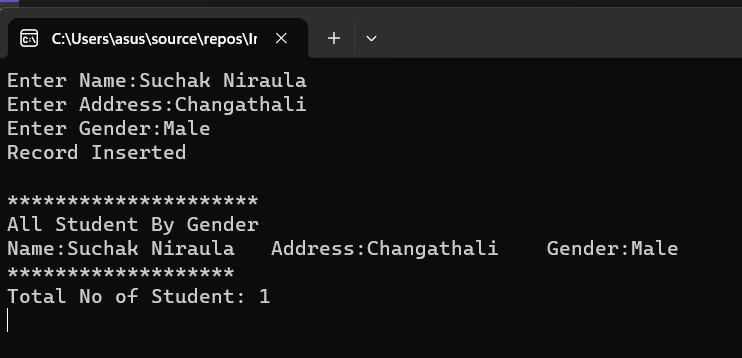
da.Fill(dt);

return dt;

}

}

}



**LAB 19: Write a C# program to perform (CRUD) Operation from given table (tblStudent) with fields (int id, nvarchar(50) name, int age, nvarchar(50) gender).**

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

Student st = new Student();

Console.WriteLine("Enter Option");

Console.WriteLine("1 For Insert Student");

Console.WriteLine("2 For Update Student");

Console.WriteLine("3 For Delete Student");

Console.WriteLine("4 For Fetch All Student");

string option = Console.ReadLine();

switch (option)

{

case "1":

Console.Write("Enter Name: ");

string name = Console.ReadLine();

Console.Write("Enter Address: ");

string address = Console.ReadLine();

Console.Write("Enter Gender: ");

string gender = Console.ReadLine();

st.InsertStudent(name, address, gender);

Console.WriteLine("Record Inserted");

break;

case "2":

Console.Write("Enter Id To Update: ");

int id = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Name: ");

string uname = Console.ReadLine();

Console.Write("Enter Address: ");

string uaddress = Console.ReadLine();

Console.Write("Enter Gender: ");

string ugender = Console.ReadLine();

st.UpdateStudent(uname, uaddress, ugender, id);

Console.WriteLine("Record Updated");

break;

case "3":

Console.Write("Enter Id To Update: ");

int did = Convert.ToInt32(Console.ReadLine());

st.DeleteStudent(did);

Console.WriteLine("Record Deleted");

break;

case "4":

DataTable dt = st.DisplayStudentData();

for (int i = 0; i < dt.Rows.Count; i++)

{

Console.WriteLine("Name:{0} Address:{1} Gender:{2}",

dt.Rows[i]["Name"], dt.Rows[i]["Address"], dt.Rows[i]["Gender"]);

}

break;

default:

break;

}

Console.ReadLine();

}

}

public class Student

{

public void InsertStudent(string name, string address, string gender)

{

string connStr = @"Data Source=(localdb)\MSSqlLocalDB; Database=SamriddhiDB; Integrated Security=true";

SqlConnection con = new SqlConnection(connStr);

string sql = "insert into tblStudent values(@name,@address,@gender)";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@name", name);

cmd.Parameters.AddWithValue("@address", address);

cmd.Parameters.AddWithValue("@gender", gender);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

public void UpdateStudent(string name, string address, string gender, int id)

{

string connStr = @"Data Source=(localdb)\MSSqlLocalDB; Database=SamriddhiDB; Integrated Security=true";

SqlConnection con = new SqlConnection(connStr);

string sql = "update tblStudent set Name=@name, Address=@address, Gender=@gender whwere Id=@id";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@name", name);

cmd.Parameters.AddWithValue("@address", address);

cmd.Parameters.AddWithValue("@gender", gender);

cmd.Parameters.AddWithValue("@id", id);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

public void DeleteStudent(int id)

{

string connStr = @"Data Source=(localdb)\MSSqlLocalDB; Database=SamriddhiDB; Integrated Security=true";

SqlConnection con = new SqlConnection(connStr);

string sql = "delete from tblStudent whwere Id=@id";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@id", id);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

public DataTable DisplayStudentData()

{

string conStr = "Data Source=(LocalDB)\\MSSQLLocalDB; Database=SamriddhiDB; Integrated Security=true";

SqlConnection con = new SqlConnection(conStr);//

SqlCommand cmd = new SqlCommand("select \*from tblStudent", con);

SqlDataAdapter da = new SqlDataAdapter(cmd);//works as mediator between datasource=datatable

DataTable dt = new DataTable();//row column

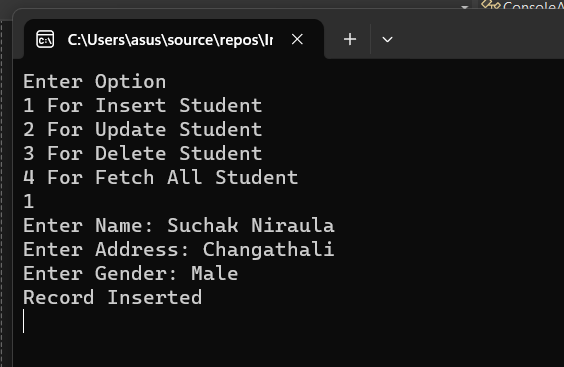
da.Fill(dt);

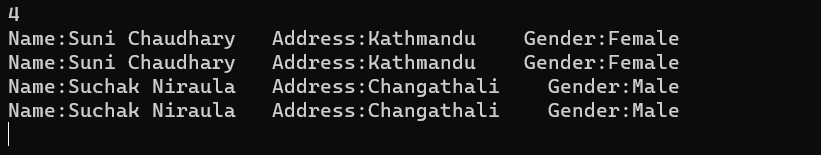
return dt;

}

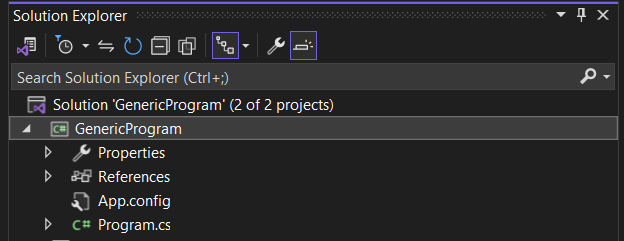
}

}

****

****

**LAB 20: Write a C# program to create generic class with generic constructor, generic member variable, generic property and generic method.**

****

using System;

using System.Collections.Generic;

using System.Linq;

using System.Linq.Expressions;

using System.Text;

using System.Threading.Tasks;

namespace GenericProgram

{

internal class Program

{

static void Main(string[] args)

{

MyGenericClass<int> intGenericClass = new MyGenericClass<int>(10);

intGenericClass.DisplayGenericValue();

intGenericClass.GenericProperty = 20;

intGenericClass.DisplayGenericValue();

MyGenericClass<string> stringGenericClass = new MyGenericClass<string>("Hello World!");

stringGenericClass.DisplayGenericValue();

stringGenericClass.GenericProperty = "Goodbye, Samriddhi College!";

stringGenericClass.DisplayGenericValue();

Console.ReadLine();

}

}

public class MyGenericClass<T>

{

private T variable;

public MyGenericClass(T value)

{

this.variable = value;

}

public T GenericProperty

{

get { return variable; }

set { variable = value; }

}

public void DisplayGenericValue()

{

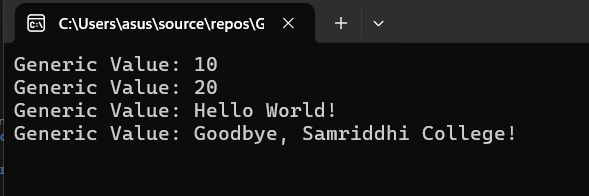
Console.WriteLine($"Generic Value: {variable}");

}

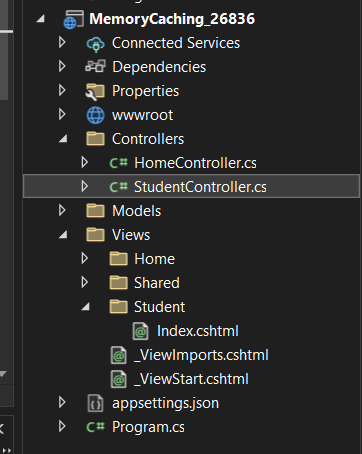
}

}

**Output:**

****

**LAB 21: Write an ASP.NET CORE program to explain working of memory caching for state management.**

****

**Program.cs**

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Student}/{action=Index}/{id?}");

app.Run();

**HomeController.cs**

using MemoryCachingProgram.Models;

using Microsoft.AspNetCore.Mvc;

using System.Diagnostics;

namespace MemoryCachingProgram.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

**StudentController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Caching.Memory;

namespace MemoryCachingProgram.Controllers

{

public class StudentController : Controller

{

private readonly IMemoryCache memoryCache;

public StudentController(IMemoryCache memoryCache)

{

this.memoryCache = memoryCache;

}

public IActionResult Index()

{

DateTime currentTime;

bool isExist = memoryCache.TryGetValue("CacheTime", out currentTime);

if (!isExist)

{

currentTime = DateTime.Now;

memoryCache.Set("CacheTime", currentTime, TimeSpan.FromSeconds(30));

}

return View("Index", currentTime.ToLongTimeString());

}

}

}

**Index.cshtml**

@\*

For more information on enabling MVC for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

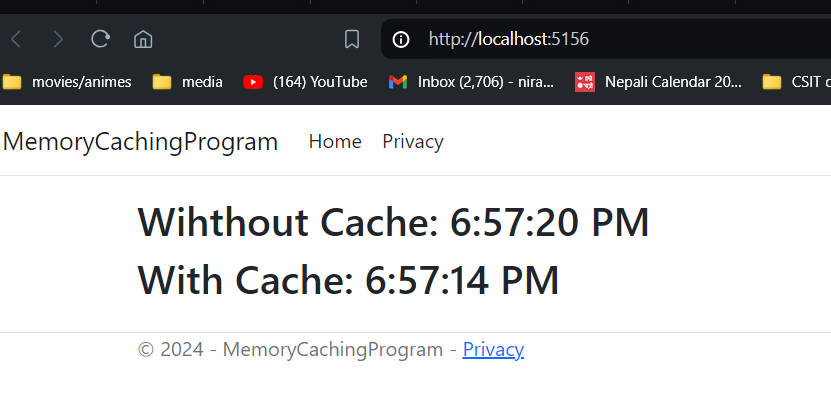
\*@

@model string

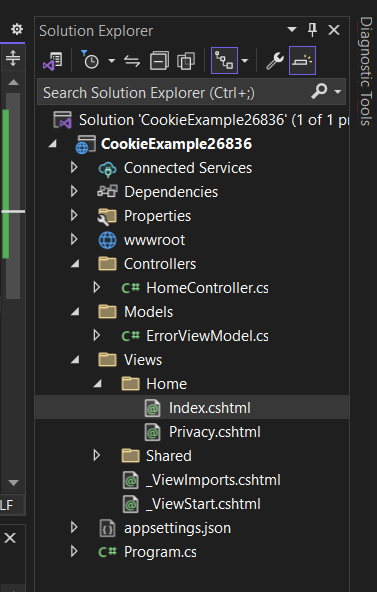
<h2>Wihthout Cache: @DateTime.Now.ToLongTimeString()</h2>

<h2>With Cache: @Model</h2>

**Output:**

****

**Lab 22: Create a new MVC project in ASP.NET Core and add controller with endpoints to read and write values into cookies.**

****

**HomeController.cs**

public IActionResult Index()

{

string userName = Request.Cookies["UserName"];

return View("Index", userName);

}

[HttpPost]

public IActionResult Index(IFormCollection form)

{

string userName = form["userName"].ToString();

//set the key value in Cookie

CookieOptions option = new CookieOptions();

option.Expires = DateTime.Now.AddMinutes(10);

Response.Cookies.Append("UserName", userName, option);

return RedirectToAction(nameof(Index));

}

public IActionResult RemoveCookie()

{

//Delete the cookie

Response.Cookies.Delete("UserName");

return View("Index");

}

**Index.cshtml**

@{

ViewData["Title"] = "Home Page";

}

@if (!string.IsNullOrWhiteSpace(Model))

{

<div>Welcome back, @Model</div>

@Html.ActionLink("Forget Me", "RemoveCookie")

}

else

{

<**form** **asp-action**="Index">

<span>Hey, seems like it's your first time here!</span><br />

<label>Please provide your name:</label>

@Html.TextBox("userName")

<div class="form-group">

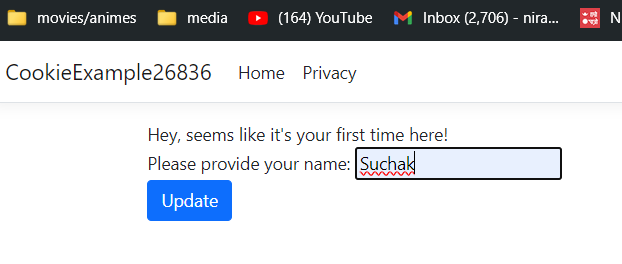
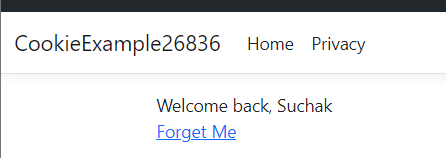
<input type="submit" value="Update" class="btn btn-primary" />

</div>

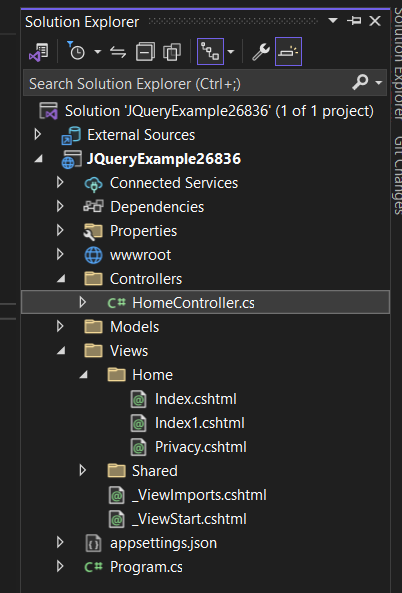
</**form**>

}

**Output:**

****

**Lab 23: Write a program to validate form using a jQuery .**



**Source Code:**

HomeController.cs

public IActionResult Index()

{

return View();

}

Index.cshtml

@{

ViewData["Title"] = "Home Page";

}

<div class="text-center">

<h1 class="display-4">Welcome</h1>

<p>Learn about <a href="https://learn.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>

</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>

<**form** id="first\_form" method="post" action="">

<div>

<label for="first\_name">First Name:</label>

<input type="text" id="first\_name" name="first\_name" />

</div>

<div>

<label for="last\_name">Last Name:</label>

<input type="text" id="last\_name" name="last\_name" />

</div>

<div>

<label for="email">Email:</label>

<input type="email" id="email" name="email" />

</div>

<div>

<label for="password">Password:</label>

<input type="password" id="password" name="password" />

</div>

<div>

<input type="submit" value="Submit" />

</div>

</**form**>

<script>

$(document).ready(function () {

$('#first\_form').submit(function (e) {

e.preventDefault();

var first\_name = $('#first\_name').val();

var last\_name = $('#last\_name').val();

var email = $('#email').val();

var password = $('#password').val();

if (first\_name.length < 1) {

alert("First Name Required")

}

if (last\_name.length < 1) {

alert("Last Name Required")

}

if (email.length < 1) {

alert("Email Required")

}

if (password.length < 8) {

alert("Password Required")

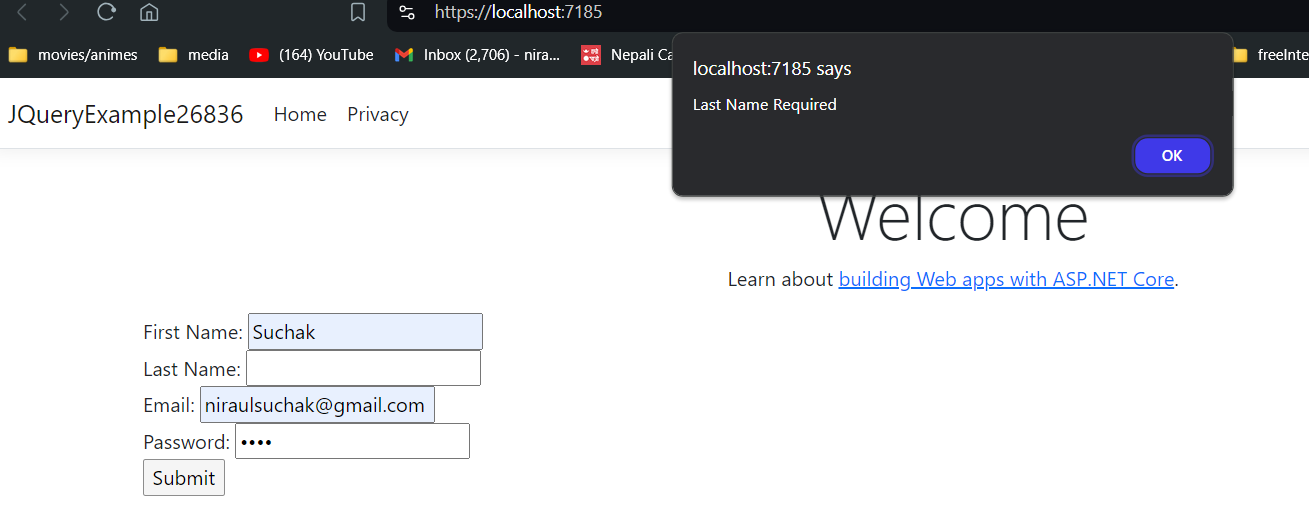
}

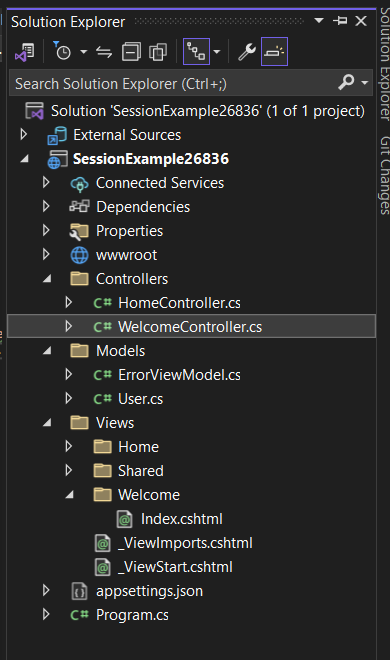
});

});

</script>

**Output:**



**LAB 24: Create a controller with endpoints to set and read a value from the session.**

**WelcomeController.cs**

public class WelcomeController : Controller

{

public IActionResult Index()

{

HttpContext.Session.SetString("Name", "Suchak Niraula");

HttpContext.Session.SetInt32("Age", 22);

User newUser = new User()

{

Name = HttpContext.Session.GetString("Name"),

Age = HttpContext.Session.GetInt32("Age").Value

};

return View(newUser);

}

}

**User.cs(Models)**

namespace SessionExample26836.Models

{

public class User

{

public string Name { get; set; }

public int Age { get; set; }

}

}

**Index.cshtml(Views -> Welcome)**

@model SessionExample26836.Models.User

@{

ViewData["Title"] = "Index";

}

<h1>Index</h1>

<div>

<h4>User</h4>

<hr />

<dl class="row">

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Name)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Age)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Age)

</dd>

</dl>

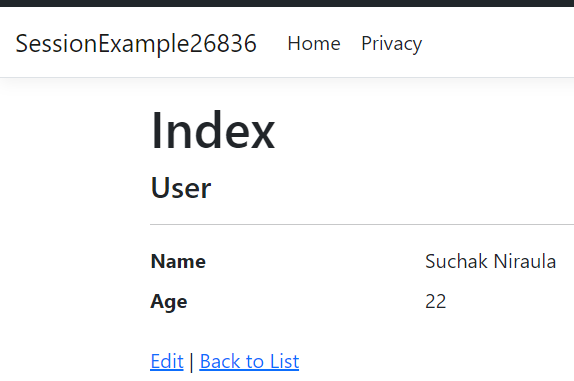
</div>

<div>

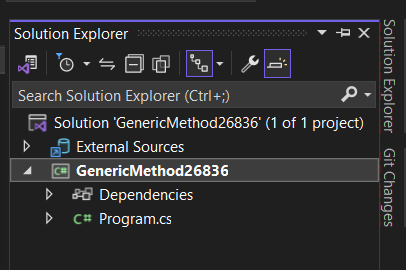
@Html.ActionLink("Edit", "Edit", new { /\* id = Model.PrimaryKey \*/ }) |

<**a** **asp-action**="Index">Back to List</**a**>

</div>

****

**LAB 25: Write a program in C# to** **implement a generic list data structure.**

****

**Source Code:**

using System;

using System.Collections.Generic;

namespace Generic\_method\_26808

{

internal class Program

{

static void Main(string[] args)

{

MyList<int> myList = new MyList<int>();

myList.Add(1);

myList.Add(2);

myList.Add(3);

Console.WriteLine("Count: " + myList.Count); // Output: 3

Console.WriteLine("Contains 2: " + myList.Contains(2)); // Output: True

Console.WriteLine("Contains 5: " + myList.Contains(5)); // Output: False

myList.Remove(2);

Console.WriteLine("Count after removal: " + myList.Count); // Output: 2

Console.ReadLine();

}

}

public class MyList<T>

{

private List<T> items = new List<T>();

public void Add(T item)

{

items.Add(item);

}

public bool Remove(T item)

{

return items.Remove(item);

}

public bool Contains(T item)

{

return items.Contains(item);

}

public int Count

{

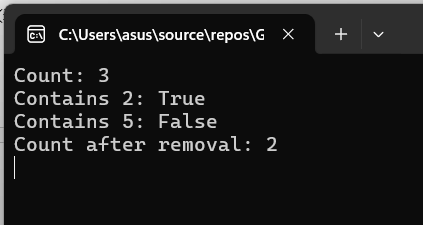
get { return items.Count; }

}

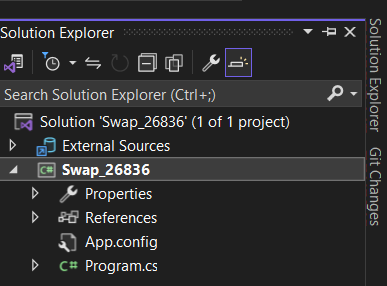
}

}

**Output:**

****

**Lab 26: Write a generic method called ‘Swap’ that takes two parameters of the same type and swaps their values.**

****

**Source Code:**

using System;

namespace Swap\_26808

{

internal class Program

{

static void Main(string[] args)

{

int a = 7, b = 23;

Console.WriteLine($"Before swap: a = {a}, b = {b}");

GenericExample.Swap(ref a, ref b);

Console.WriteLine($"After swap: a = {a}, b = {b}");

Console.ReadLine();

}

}

public class GenericExample

{

public static void Swap<T>(ref T a, ref T b)

{

T temp = a;

a = b;

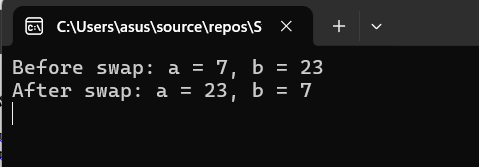
b = temp;

}

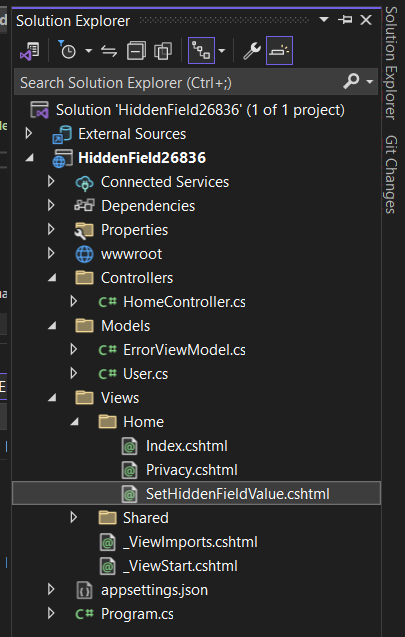
}

}

**Output:**

****

**Lab 27: Write an ASP.NET Core program to demonstrate use of hidden fields.**

****

**HomeController.cs**

[HttpGet]

public IActionResult SetHiddenFieldValue()

{

User newUser = new User()

{

Id = 101,

Name = "John",

};

return View(newUser);

}

[HttpPost]

public IActionResult SetHiddenFieldValue(User us)

{

var id = us.Id;

ViewBag.ID = id;

return View(us);

}

**User.cs(Models)**

namespace HiddenField26836.Models

{

public class User

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**SetHiddenFieldValue.cshtml(Views)**

@model HiddenField26836.Models.User

<h2>SetHiddenFieldValue</h2>

<h3>User</h3>

@using (Html.BeginForm("SetHiddenFieldValue", "Home", FormMethod.Post))

{

@Html.HiddenFor(model => model.Id)

<div>

@Html.LabelFor(model => model.Name)

@Html.DisplayFor(model => model.Name)

</div>

<input type="submit" value="Update" class="btn btn-primary" />

}

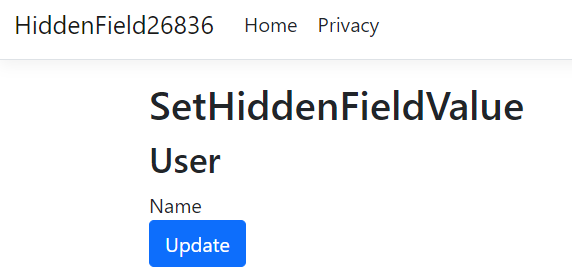
**Program.cs**

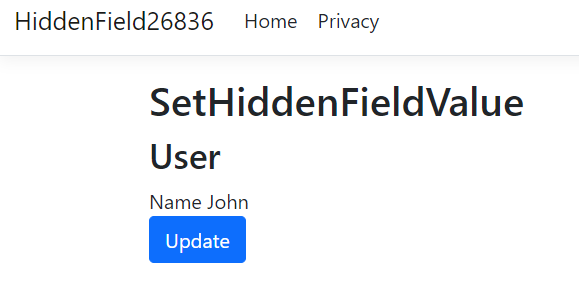
app.MapControllerRoute(

name: "default",

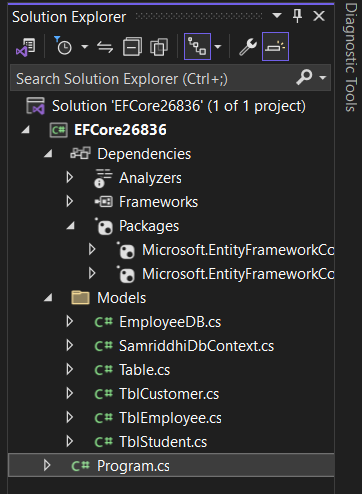
pattern: "{controller=Home}/{action=SetHiddenFieldValue}/{id?}");

**Output:**

****

****

**LAB-28: Write a C# Program to insert and display records in tablename tblEmployee (id int, name nvarchar(50), address nvarchar(50), salary decimal(18,0), gender nvarchar50) using EntityFramework.**

****

**EmployeeDB.cs**

using EFCore26836.Models;

using System;

namespace Entity\_Framework\_26836.Models

{

internal class EmployeeDB

{

SamriddhiDbContext db = new SamriddhiDbContext();

public void CreateEmployee(TblEmployee tb)

{

db.TblEmployees.Add(tb);

db.SaveChanges();

Console.WriteLine("Data Saved");

}

public List<TblEmployee> GetAllEmployees()

{

List<TblEmployee> lst = db.TblEmployees.ToList();

return lst;

}

}

}

**Program.cs**

using EFCore26836.Models;

using Entity\_Framework\_26836.Models;

EmployeeDB emp = new EmployeeDB();

TblEmployee tb = new TblEmployee();

tb.Name = "Ram Shah";

tb.Address = "Bhaktapur";

tb.Salary = 20000;

tb.Gender = "Male";

emp.CreateEmployee(tb);

List<TblEmployee> lst = emp.GetAllEmployees();

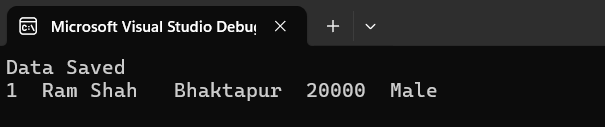
foreach (TblEmployee item in lst)

{

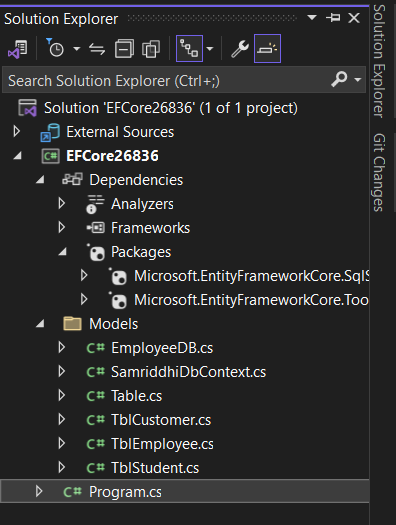
Console.WriteLine("{0} {1} {2} {3} {4}", item.Id, item.Name, item.Address, item.Salary, item.Gender);

}

Console.ReadLine();

****

**Lab 29: Write a C# program to compute aggregate salary of 5 employee and then display employee record in descending order with respect to employee salary using EntityFramework.**

****

**Program.cs**

using EFCore26836.Models;

SamriddhiDbContext db = new SamriddhiDbContext();

string totalsalary = db.TblEmployees.Take(5).Sum(a => a.Salary).ToString();

decimal averagesalary = Convert.ToDecimal(totalsalary) / 5;

Console.WriteLine("Average Salary:" + averagesalary);

List<TblEmployee> emp = db.TblEmployees.OrderByDescending(a => a.Salary).ToList();

foreach (TblEmployee item in emp)

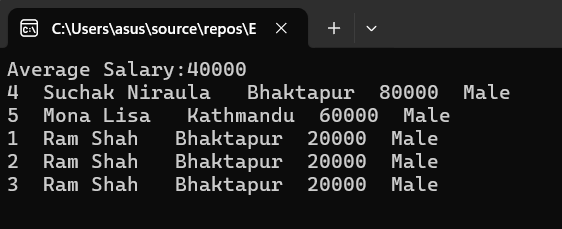
{

Console.WriteLine("{0} {1} {2} {3} {4}", item.Id, item.Name, item.Address, item.Salary, item.Gender);

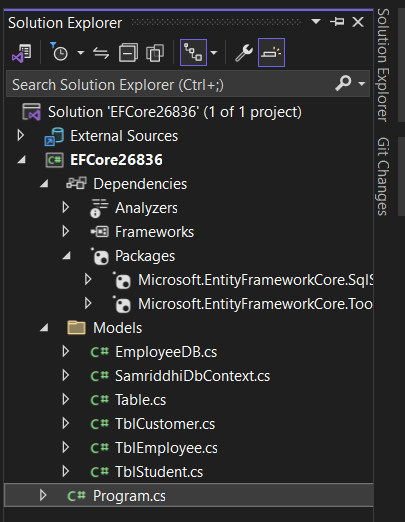
}

Console.ReadLine();

**Output:**

****

**Lab 30: Write a program to select employees whose salary is greater than 20000 and whose address is Kathmandu using EntityFramework.**

****

**Program.cs**

using EFCore26836.Models;

SamriddhiDbContext db = new SamriddhiDbContext();

List<TblEmployee> emp = db.TblEmployees.Where(a => a.Salary > 20000 && a.Address == "Kathmandu").ToList();

foreach (TblEmployee item in emp)

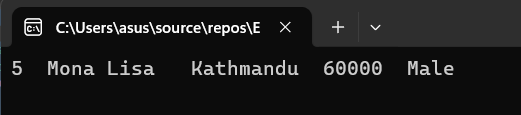
{

Console.WriteLine("{0} {1} {2} {3} {4}", item.Id, item.Name, item.Address, item.Salary, item.Gender);

}

Console.ReadLine();

**Output:**

****