**Practical 1**

1. Write a program to demonstrate class, constructor, properties and method

using System;

// Define a class named "Car"

public class Car

{

// Properties

public string Make { get; set; }

public string Model { get; set; }

public int Year { get; set; }

// Constructor

public Car(string make, string model, int year)

{

Make = make;

Model = model;

Year = year;

}

// Method to display car details

public void DisplayCarDetails()

{

Console.WriteLine($"Car Details: {Year} {Make} {Model}");

}

}

public class Program

{

public static void Main()

{

// Create an object of the Car class using the constructor

Car myCar = new Car("Toyota", "Corolla", 2020);

// Call the method to display car details

myCar.DisplayCarDetails();

}

}

1. Write a program to demonstrate method overloading?

using System;

public class Calculator

{

// Method to add two integers

public int Add(int a, int b)

{

return a + b;

}

// Overloaded method to add three integers

public int Add(int a, int b, int c)

{

return a + b + c;

}

// Overloaded method to add two doubles

public double Add(double a, double b)

{

return a + b;

}

}

public class Program

{

public static void Main()

{

// Create an object of the Calculator class

Calculator calc = new Calculator();

// Calling the Add method with two integers

int sum1 = calc.Add(5, 10);

Console.WriteLine("Sum of 5 and 10: " + sum1);

// Calling the Add method with three integers

int sum2 = calc.Add(5, 10, 15);

Console.WriteLine("Sum of 5, 10, and 15: " + sum2);

// Calling the Add method with two doubles

double sum3 = calc.Add(5.5, 10.5);

Console.WriteLine("Sum of 5.5 and 10.5: " + sum3);

}

}

1. Create a class Calculate which contains data member num1 and num2 both in integer and methods setCalc() to set the data, calcSum() that calculate the sum of num1 and num2 and display the result, calcMulti() that calculate the multiplication of num1 and num2 and returns the result, calcDifference that calculate the difference between num1 and num2 and display the result. Now, create some instance of Calculate and invoke all the methods.

using System;

namespace Lab1

{

internal class Calculate

{

// Data members

private int num1, num2;

// Method to set values

public void SetCalc(int a, int b)

{

num1 = a;

num2 = b;

}

// Method to calculate and display the sum

public void CalcSum()

{

Console.WriteLine("Sum: " + (num1 + num2));

}

// Method to calculate multiplication and return the result

public int CalcMulti()

{

return num1 \* num2;

}

// Method to calculate and display the difference

public void CalcDifference()

{

Console.WriteLine("Difference: " + (num1 - num2));

}

// Main method to test the Calculate class

public static void Main(string[] args)

{

// Create instances of Calculate

Calculate calc1 = new Calculate();

Calculate calc2 = new Calculate();

// Set values for calc1

calc1.SetCalc(10, 5);

Console.WriteLine("Operations for calc1:");

calc1.CalcSum();

Console.WriteLine("Multiplication: " + calc1.CalcMulti());

calc1.CalcDifference();

Console.WriteLine(); // Just for spacing

// Set values for calc2

calc2.SetCalc(20, 8);

Console.WriteLine("Operations for calc2:");

calc2.CalcSum();

Console.WriteLine("Multiplication: " + calc2.CalcMulti());

calc2.CalcDifference();

}

}

}

1. Create a class Number having instance variable x and y both in integer, default constructor that set the value of x and y to 0, parameterized constructor that sets the value of x and y, method findOdd() that calculates the even no. occurring between x and y and display the result, findEven() that calculates the odd no. occurring between x and y and display the results. Now, create some instance of Number and invoke all the methods.

using System;

namespace Lab1

{

internal class Number

{

private int x, y;

public Number() { x = y = 0; } // Default constructor

public Number(int a, int b) { x = a; y = b; } // Parameterized constructor

public void FindEven()

{

Console.Write("Even: ");

for (int i = x; i <= y; i++) if (i % 2 == 0) Console.Write(i + " ");

Console.WriteLine();

}

public void FindOdd()

{

Console.Write("Odd: ");

for (int i = x; i <= y; i++) if (i % 2 != 0) Console.Write(i + " ");

Console.WriteLine();

}

public static void Main()

{

Number num1 = new Number(5, 15);

num1.FindEven();

num1.FindOdd();

Console.WriteLine();

Number num2 = new Number(10, 20);

num2.FindEven();

num2.FindOdd();

}

}

}

1. Create a class Shape that contains instance variable length, breadth and height. Create a default constructor that sets the value of instance variable to zero, constructor with two parameter that will sets the value of length and breadth only and constructor with three parameter that will sets the value of length, breadth and height. After this create calcAreaRectangle() that calculates the area of rectangle, calcVolumeBox() that calculates volume of box and display the result. Now create first object of Shape wihich will have name rectangle and calls constructor with two parameter and calAreaRectangle() method, create second object of Shape that will have name Box which will call constructor with three parameter and calcVolumeBox() method.

using System;

namespace Lab1

{

internal class Shape

{

private int length, breadth, height;

// Default constructor

public Shape() { length = breadth = height = 0; }

// Constructor for rectangle

public Shape(int l, int b) { length = l; breadth = b; height = 0; }

// Constructor for box

public Shape(int l, int b, int h) { length = l; breadth = b; height = h; }

// Method to calculate area of rectangle

public void CalcAreaRectangle()

{

Console.WriteLine("Rectangle Area: " + (length \* breadth));

}

// Method to calculate volume of a box

public void CalcVolumeBox()

{

Console.WriteLine("Box Volume: " + (length \* breadth \* height));

}

public static void Main()

{

// Creating a rectangle object using 2-parameter constructor

Shape rectangle = new Shape(5, 10);

rectangle.CalcAreaRectangle();

Console.WriteLine();

// Creating a box object using 3-parameter constructor

Shape box = new Shape(4, 6, 8);

box.CalcVolumeBox();

}

}

}

1. Create a class EmployeeDetails having data member empId, empName, empGender, empAddress, and empPosition, constructor to set the details and display method to display the details. Create a subclass SalaryInfo that will inherit EmployeeDetails having own data member salary which will record salary per year, constructor to set the value of salary and method calcTax() that deduct the tax on salary and display the final salary. Tax rate is given as (if salary <= 400000 tax is 1%, salary between 400001 to 800000 tax is 10% and salary > 800000 tax 20%). Now create the object of Salary info and demonstrate the scenario.

using System;

namespace Lab1

{

// Base class

internal class EmployeeDetails

{

protected int empId;

protected string empName, empGender, empAddress, empPosition;

// Constructor to initialize employee details

public EmployeeDetails(int id, string name, string gender, string address, string position)

{

empId = id;

empName = name;

empGender = gender;

empAddress = address;

empPosition = position;

}

// Method to display employee details

public void DisplayDetails()

{

Console.WriteLine("Employee ID: " + empId);

Console.WriteLine("Name: " + empName);

Console.WriteLine("Gender: " + empGender);

Console.WriteLine("Address: " + empAddress);

Console.WriteLine("Position: " + empPosition);

}

}

// Derived class

internal class SalaryInfo : EmployeeDetails

{

private double salary;

// Constructor to initialize salary details

public SalaryInfo(int id, string name, string gender, string address, string position, double sal)

: base(id, name, gender, address, position)

{

salary = sal;

}

// Method to calculate tax and display final salary

public void CalcTax()

{

double taxRate = salary <= 400000 ? 0.01 : (salary <= 800000 ? 0.10 : 0.20);

double taxAmount = salary \* taxRate;

double finalSalary = salary - taxAmount;

Console.WriteLine("Salary: " + salary);

Console.WriteLine("Tax Deducted: " + taxAmount);

Console.WriteLine("Final Salary After Tax: " + finalSalary);

}

}

// Main method

internal class Program

{

public static void Main()

{

// Creating an employee object with salary

SalaryInfo emp1 = new SalaryInfo(101, "Alice Smith", "Female", "123 Main St", "Software Engineer", 750000);

// Display employee details

emp1.DisplayDetails();

Console.WriteLine();

// Calculate and display salary after tax

emp1.CalcTax();

}

}

}

1. Write a program to demonstrate single level, multilevel inheritance?

using System;

namespace YK\_IT\_Department

{

// Single-Level Inheritance

class Employee

{

protected string Name, Position;

public Employee(string name, string position)

{

Name = name;

Position = position;

}

public void DisplayEmployee()

{

Console.WriteLine($"Name: {Name}");

Console.WriteLine($"Position: {Position}");

}

}

// Multilevel Inheritance

class Department : Employee

{

protected string DepartmentName;

public Department(string name, string position, string dept)

: base(name, position)

{

DepartmentName = dept;

}

public void DisplayDepartment()

{

DisplayEmployee();

Console.WriteLine($"Department: {DepartmentName}");

}

}

class YK\_IT : Department

{

public YK\_IT(string name, string position)

: base(name, position, "IT Department") { }

public void Work()

{

Console.WriteLine($"{Name} is working as a {Position} in {DepartmentName}.");

}

}

class Program

{

static void Main()

{

YK\_IT emp1 = new YK\_IT("Ayusha Adhikari", "Front-End Developer");

emp1.DisplayDepartment();

emp1.Work();

}

}

}

1. Write a program to demonstrate use of base keyword

using System;

namespace FoodDemo

{

// Base class

class Food

{

public string Name;

// Constructor to initialize food name

public Food(string name)

{

Name = name;

}

// Method to display food name

public void DisplayFood()

{

Console.WriteLine("Food: " + Name);

}

}

// Derived class

class Dessert : Food

{

public string Type;

// Constructor to initialize food and type of dessert

public Dessert(string name, string type) : base(name)

{

Type = type;

}

// Method to display dessert details

public void DisplayDessert()

{

base.DisplayFood(); // Using base keyword to call parent class method

Console.WriteLine("Dessert Type: " + Type);

}

}

class Program

{

static void Main()

{

Dessert myDessert = new Dessert("Cake", "Chocolate");

myDessert.DisplayDessert(); // Demonstrating the use of base keyword

}

}

}

1. Write a program to demonstrate method overriding (dynamic polymorphism)?

using System;

class Country

{

// Virtual method allows it to be overridden in derived classes

public virtual void ShowInfo()

{

Console.WriteLine("Country: Nepal");

}

}

class City : Country

{

// Overriding the ShowInfo method to provide a different implementation

public override void ShowInfo()

{

base.ShowInfo(); // Optionally call the base class method

Console.WriteLine("Capital City: Kathmandu");

}

}

class Program

{

static void Main(string[] args)

{

Country country = new Country();

country.ShowInfo(); // Calls base class method

Console.WriteLine();

Country cityAsCountry = new City(); // Polymorphism in action

cityAsCountry.ShowInfo(); // Calls derived class method

}

}

1. Write a program to demonstrate multiple inheritance using interface

using System;

interface IProgrammingSkills

{

void ShowProgrammingSkills();

}

interface IMathSkills

{

void ShowMathSkills();

}

class CSMajor : IProgrammingSkills, IMathSkills

{

// Implementing method from IProgrammingSkills

public void ShowProgrammingSkills()

{

Console.WriteLine("CS Major: Strong programming skills in C#, Python, and Java.");

}

// Implementing method from IMathSkills

public void ShowMathSkills()

{

Console.WriteLine("CS Major: Proficient in Discrete Math and Algorithms.");

}

}

class Program

{

static void Main(string[] args)

{

CSMajor csMajor = new CSMajor();

csMajor.ShowProgrammingSkills(); // Calls method from IProgrammingSkills

csMajor.ShowMathSkills(); // Calls method from IMathSkills

}

}

1. Write a program to demonstrate abstract class

using System;

abstract class Animal

{

// Abstract method with no implementation in the base class

public abstract void Sound();

// Regular method

public void Sleep()

{

Console.WriteLine("The animal is sleeping.");

}

}

class Dog : Animal

{

// Implementing the abstract method

public override void Sound()

{

Console.WriteLine("The dog barks.");

}

}

class Program

{

static void Main(string[] args)

{

Dog dog = new Dog();

dog.Sound(); // Calls the implemented method in Dog class

dog.Sleep(); // Calls the regular method from the base class

}

}

1. Write a program to demonstrate exception handline (try, catch, throw throws)

using System;

class ExceptionHandlingDemo

{

// Method that throws an exception

public static void Divide(int a, int b)

{

if (b == 0)

{

// Throwing an exception when dividing by zero

throw new DivideByZeroException("Division by zero is not allowed.");

}

else

{

Console.WriteLine("Result: " + (a / b));

}

}

static void Main(string[] args)

{

try

{

Console.WriteLine("Enter two numbers:");

int num1 = Convert.ToInt32(Console.ReadLine());

int num2 = Convert.ToInt32(Console.ReadLine());

// Calling the method that may throw an exception

Divide(num1, num2);

}

catch (DivideByZeroException ex)

{

// Catching and handling divide by zero exception

Console.WriteLine("Error: " + ex.Message);

}

catch (FormatException ex)

{

// Catching and handling format exception (invalid input)

Console.WriteLine("Error: Invalid input, please enter numeric values.");

}

finally

{

// Finally block to clean up, runs regardless of exception

Console.WriteLine("Program has ended.");

}

}

}

1. Write a program to demonstrate interface

using System;

interface IFurniture

{

void Material(); // Declaring a method in the interface

}

class Table : IFurniture

{

public void Material()

{

Console.WriteLine("Table is made of wood.");

}

}

class Chair : IFurniture

{

public void Material()

{

Console.WriteLine("Chair is made of plastic.");

}

}

class Program

{

static void Main(string[] args)

{

IFurniture table = new Table();

IFurniture chair = new Chair();

table.Material(); // Calls Table's Material method

chair.Material(); // Calls Chair's Material method

}

}

1. Write a program to demonstrate lamda expression

using System;

class LambdaDemo

{

delegate int Square(int x); // Declare a delegate

static void Main(string[] args)

{

// Lambda expression to square a number

Square square = x => x \* x;

int number = 5;

Console.WriteLine($"The square of {number} is {square(number)}");

}

}

**Practical 2**

1. Create a web form that contains two label that display Enter first number and enter second number, two text box for taking an input, third text box for output and three button add, subtract and find prime. Add proper validation like text box should not be empty, value of first field should be greater than value of second field. If add button is clicked display the addition of two number given in textboxes, if subtract button is clicked display the subtraction of two number given in textboxes and if findprime is clicked then display the prime number from first value to second value given in textboxes.
2. Write a console program (ADO.net) to create a table tbl\_registration that have fields (id int primary key, username, password, repassword, gender, course and country). After this perform the following operation

* Insert any 5 data into tbl\_registration. All the required input should be taken from user
* Display all the record of database table
* Update the name and course of a person to data given by user according to id given by user
* Delete the record of person whose id is given by user
* Display all the record of person who are male and also from country Nepal

1. For the table created in question no. 3, create a web form for registration which should contains username, password, repassword, gender (radio button), course (checkbox) and country (dropdown) and submit button. When submit is pressed insert the value given by user into database table. Use proper validation: username, password and repassword should not be empty, item of radio button, checkbox and dropdown menu should be selected.

**Practical 3**

1. Demonstrate model, view and controller by showing different action method, views, model, accessing controller, model and view.
2. Demonstrate use of razor syntax
3. Demonstrate use of html tag helper
4. Using Entity framework create a table tbl\_officer having field (id, name, gender, phone, department and position) after this perform complete CRUDE operation (insert, update, display and delete). User proper validation.
5. Demonstrate different state management technique like SessionState, TempData, HttpContext
6. Demonstrate different client-side state management like cookies, Query string and hidden fields
7. Write a program to create complete form and validate using jquery and react
8. Write a program to demonstrate authentication and authorization (Role, claim and policies) by create a complete form in asp.net core
9. Write a program to prevent SQLInjectionAttack, Cross Site Request forgery (CSRF) and open redirect attack