|  |  |  |  |
| --- | --- | --- | --- |
| A picture containing drawing, stop, room  Description automatically generated | Advanced Web Programming  Practical #2 | | |
|  |  |  |  |
| **Name** | Sandeep Jain | **Roll Number** | 21302C0058 |
| **Subject/Course:** | **Advanced Web Programming** | | |
| **Topic** | **Working with Object Oriented C#** | | |
|  | | | |
| **1. Working with Object Oriented C#** | | | |
| 1. **Create simple application to perform following operations i. Finding factorial Value ii. Money Conversion iii. Quadratic Equation iv. Temperature Conversion**   **i. Finding factorial Value**  **Main class Code**  **using System;**  **using System.Collections.Generic;**  **using System.Linq;**  **using System.Text;**  **using System.Threading.Tasks;**    **namespace practical2a1**  **{**  **class Program**  **{**  **static void Main(string[] args)**  **{**  **Factorial f = new Factorial();**  **Console.WriteLine("enter a numbur");**  **int n = Convert.ToInt32(Console.ReadLine());**  **f.fact(n);**  **Console.ReadLine();**  **}**    **}**  **Class code**  **using System;**  **using System.Collections.Generic;**  **using System.Linq;**  **using System.Text;**  **using System.Threading.Tasks;**    **namespace practical2a1**  **{**  **class Factorial**  **{**  **public void fact(int n)**  **{**  **int f = 1;**  **for (int i = 1; i <= n; i++)**  **{**  **f= f\*i;**  **}**  **Console.WriteLine("Factorial =" + f);**  **}**  **}**  **}**  **Output**    **ii. Money Conversion**  **Code**  **using System;**    **namespace ConsoleApplication1**  **{**  **class MoneyConversion**  **{**  **static void Main(string[] args)**  **{**  **int choice;**  **Console.WriteLine("Enter your Choice:\n 1- Dollar to Rupee\n 2- Euro to Rupee \n 3 Malaysian Ringgit to Rupee ");**  **choice = int.Parse(Console.ReadLine());**  **switch (choice)**  **{**  **case 1:**  **Double dollar, rupee,val; Console.WriteLine("Enter the Dollar Amount :");**  **dollar = Double.Parse(Console.ReadLine());**  **Console.WriteLine("Enter the Dollar Value :"); val = double.Parse(Console.ReadLine());**  **rupee = dollar \* val;**  **Console.WriteLine("{0} Dollar Equals {1} Rupees", dollar, rupee);**  **break;**  **case 2:**  **Double Euro, rupe,valu; Console.WriteLine("Enter the Euro Amount :");**  **Euro = Double.Parse(Console.ReadLine());**  **Console.WriteLine("Enter the Euro Value :");**  **valu = double.Parse(Console.ReadLine());**  **rupe = Euro \*valu;**  **Console.WriteLine("{0} Euro Equals {1} Rupees", Euro, rupe);**  **break;**  **case 3:**  **Double ringit, rup,value; Console.WriteLine("Enter the Ringgit Amount :");**  **ringit = Double.Parse(Console.ReadLine());**  **Console.WriteLine("Enter the Ringgit Value :");**  **value = double.Parse(Console.ReadLine());**  **rup = ringit \* value;**  **Console.WriteLine("{0} Malaysian Ringgit Equals {1} Rupees", ringit, rup);**  **break;**  **}**  **Console.ReadLine();**  **}**  **}**  **}**      **Output**    **iii. Quadratic Equation**  **Code**  **using System;**  **namespace Practical2a1**  **{**  **class Quadraticroots**  **{**  **double a, b, c;**  **public void read()**  **{**  **Console.WriteLine(" \n To find the roots of a quadratic equation of the form a\*x\*x + b\*x + c = 0");**  **Console.Write("\n Enter value for a : ");**  **a = double.Parse(Console.ReadLine());**  **Console.Write("\n Enter value for b : ");**  **b = double.Parse(Console.ReadLine());**  **Console.Write("\n Enter value for c : ");**  **c = double.Parse(Console.ReadLine());**  **}**  **public void compute()**  **{**  **int m;**  **double r1, r2, d1;**  **d1 = b \* b - 4 \* a \* c;**  **if (a == 0)**  **m = 1;**  **else if (d1 > 0)**  **m = 2;**  **else if (d1 == 0)**  **m = 3;**  **else**  **m = 4;**  **switch (m)**  **{**  **case 1: Console.WriteLine("\n Not a Quadratic equation, Linear equation");**  **Console.ReadLine();**  **break;**  **case 2: Console.WriteLine("\n Roots are Real and Distinct");**  **r1 = (-b + Math.Sqrt(d1)) / (2 \* a);**  **r2 = (-b - Math.Sqrt(d1)) / (2 \* a);**  **Console.WriteLine("\n First root is {0:#.##}", r1);**  **Console.WriteLine("\n Second root is {0:#.##}", r2);**  **Console.ReadLine();**  **break;**  **case 3: Console.WriteLine("\n Roots are Real and Equal");**  **r1 = r2 = (-b) / (2 \* a);**  **Console.WriteLine("\n First root is {0:#.##}", r1);**  **Console.WriteLine("\n Second root is {0:#.##}", r2);**  **Console.ReadLine();**  **break;**  **case 4: Console.WriteLine("\n Roots are Imaginary");**  **r1 = (-b) / (2 \* a);**  **r2 = Math.Sqrt(-d1) / (2 \* a);**  **Console.WriteLine("\n First root is {0:#.##} + i {1:#.##}", r1, r2);**  **Console.WriteLine("\n Second root is {0:#.##} - i {1:#.##}", r1, r2);**  **Console.ReadLine();**  **break;**    **}**  **}**  **}**  **class QuadraticEquation**  **{**  **public static void Main()**  **{**  **Quadraticroots qr = new Quadraticroots();**  **qr.read();**  **qr.compute();**  **}**  **}**  **}**  **Output**    **iv. Temperature Conversion**  **Code**  **using System;**  **namespace Temperature\_Conversion**  **{**  **class TempConversion**  **{**  **public static float temp;**  **public static char tempUnit;**    **static void Main(string[] args)**  **{**  **//Getting user input**  **Console.WriteLine("Enter Temperature in to convert it into other i.e 30 k, 45 f, 50 c \*Put space between value and unit\* ");**  **string[] tempInput = Console.ReadLine().Split();**  **//parse element 0**  **temp = float.Parse(tempInput[0]);**  **//assinging tempUnit**  **tempUnit = char.Parse(tempInput[1]);**  **switch (tempUnit)**  **{**  **//Converting temp to F and K if tempUnit == c**  **case 'c':**  **Console.WriteLine("Celsius To Fahrenheit and Kelvin");**  **convertCelsiusToFahrenheit();**  **convertCelsiusToKelvin();**  **break;**  **//Converting temp to C and F if tempUnit == K**  **case 'k':**  **Console.WriteLine("Kelvin To Fahrenheit and Celsius");**  **convertKelvinToCelsius();**  **convertKelvinToFahrenheit();**  **break;**  **//Converting temp to C and K if tempUnit == F**  **case 'f':**  **Console.WriteLine("Fahrenheit to Celsius and kelvin");**  **convertFahrenheitToCelsius();**  **convertFahrenheitToKelvin();**  **break;**  **}**  **Console.ReadKey();**  **}**  **static void convertFahrenheitToCelsius()**  **{**  **Console.WriteLine((temp - 32) \* 0.5556 + "C");**    **}**  **static void convertFahrenheitToKelvin()**  **{**  **Console.WriteLine((temp + 459.67) \* 5 / 9 + "K");**  **}**  **static void convertCelsiusToFahrenheit()**  **{**  **Console.WriteLine((temp \* 1.8) + 32 + "F");**  **}**  **static void convertCelsiusToKelvin()**  **{**  **Console.WriteLine(temp + 273.15 + "K");**  **}**  **static void convertKelvinToCelsius()**  **{**  **Console.WriteLine(temp - 273.15 + "C");**  **}**  **static void convertKelvinToFahrenheit()**  **{**  **Console.WriteLine(temp - 459.67 + "F");**  **}**  **}**  **}**  **Output** | | | |
| **b. Create simple application to demonstrate use of following concepts**  **i. Function Overloading ii. Inheritance (all types)**  **iii. Constructor overloading iv. Interfaces**  **i. Function Overloading**  **Code**  **using System;**  **namespace Practical2b1**  **{**  **class shape**  **{**  **public void Area(int side)**  **{**  **int squarearea = side \* side;**  **Console.WriteLine("The Area of Square is :" + squarearea);**  **}**  **public void Area(int length, int breadth)**  **{**  **int rectarea = length \* breadth;**  **Console.WriteLine("The Area of Rectangle is :" + rectarea);**  **}**  **public void Area(double radius)**  **{**  **double circlearea = 3.14 \* radius \* radius;**  **Console.WriteLine("The Area of Circle is :" + circlearea);**  **}**    **}**  **class FunctionOverloading**  **{**  **static void Main(string[] args)**  **{**    **shape s = new shape();**  **s.Area(10);**  **s.Area(10, 20);**  **s.Area(10.8);**  **Console.ReadKey();**    **}**    **}**  **}**  **Output**    **ii. Inheritance (all types)**  **(a) Single Inheritance**  **Code**    **using System;**  **namespace Practical2b2a**  **{**  **class Shape**  **{**  **public void setWidth(int w)**  **{**  **width = w;**  **}**  **public void setHeight(int h)**  **{**  **height = h;**  **}**  **protected int width;**  **protected int height;**  **}**  **// Derived class**  **class Rectangle : Shape**  **{**  **public int getArea()**  **{**  **return (width \* height);**  **}**  **}**  **class SingleInheritance**  **{**  **static void Main(string[] args)**  **{**  **Rectangle Rect = new Rectangle();**  **Rect.setWidth(5);**  **Rect.setHeight(7);**  **// Print the area of the object.**  **Console.WriteLine("Total area: {0}", Rect.getArea());**  **Console.ReadKey();**  **}**  **}**  **}**  **Output**    **(a) Single Inheritance**  using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;      namespace ConsoleApp2 {     class furniture     {         public string material;         public double price;              public furniture(string material, double price)         {             this.material = material;             this.price = price;         }     }          class table : furniture     {         public int height;         public double surfacearea;              public table(int height, double surfacearea, string material, double price) : base(material, price)         {             this.height = height;             this.surfacearea = surfacearea;         }         public void display()         {             Console.WriteLine("The height is " + height);             Console.WriteLine("The Surface area is " + surfacearea);             Console.WriteLine("The Material is " + material);             Console.WriteLine("The Price is " + price);         }     }     class Program     {         static void Main(string[] args)         {              table t = new table(5, 52, "Chandan", 50000);             t.display();             Console.ReadKey();          }     } }  **(b) Multilevel Inheritance**  **Code**    **using System;**  **namespace Practical2b2b**  **{**  **class Student**  **{**  **int roll\_no;**  **string name;**  **public Student(int roll\_no, string name)**  **{**  **this.roll\_no = roll\_no;**  **this.name = name;**  **}**  **public void display()**  **{**  **Console.WriteLine("Roll No: " + roll\_no);**  **Console.WriteLine("Name: " + name);**  **}**  **}**  **class Test : Student**  **{**  **int marks1, marks2;**  **public Test(int roll\_no, string name, int marks1, int marks2)**  **: base(roll\_no, name)**  **{**  **this.marks1 = marks1;**  **this.marks2 = marks2;**  **}**  **public int getMarks1()**  **{**  **return marks1;**  **}**  **public int getMarks2()**  **{**  **return marks2;**  **}**  **public void display()**  **{**  **base.display();**  **Console.WriteLine("Marks1: " + marks1);**  **Console.WriteLine("Marks2: " + marks2);**  **}**  **}**    **class Result : Test**  **{**  **int total;**  **public Result(int roll\_no, string name, int marks1, int marks2)**  **: base(roll\_no, name, marks1, marks2)**  **{**  **total = getMarks1() + getMarks2();**  **}**  **public void display()**  **{**  **base.display();**  **Console.WriteLine("Total: " + total);**  **}**  **}**    **class MultilevelInheritance**  **{**  **static void Main(string[] args)**  **{**  **Result objResult = new Result(58, "Sandeep ", 98,99);**  **objResult.display();**  **Console.ReadKey();**  **}**  **}**  **}**  **Output**    **iii. Constructor overloading**  **Code**  **using System;**  **namespace Practical2b3**  **{**  **public class StudentData**  **{**  **private int stuID;**  **private string stuName;**  **private int stuAge;**  **public StudentData() //Default Constructor**  **{**  **stuID = 58;**  **stuName = "Sandeep";**  **stuAge = 21;**  **}**  **public StudentData(int num1, string str, int num2) //Parameterized Constructor**  **{**  **stuID = num1;**  **stuName = str;**  **stuAge = num2;**  **}**  **public StudentData(StudentData s) //Copy Constructor**  **{**  **stuID = s.stuID;**  **stuName = s.stuName;**  **stuAge = s.stuAge;**  **}**  **//Getter & Setter Methods**  **public int getStuID()**  **{**  **return stuID;**  **}**  **public void setStuID(int stuID)**  **{**  **this.stuID = stuID;**  **}**  **public string getStuName()**  **{**  **return stuName;**  **}**  **public void setStuID(string stuName)**  **{**  **this.stuName = stuName;**  **}**  **public int getStuAge()**  **{**  **return stuAge;**  **}**    **public void setStuAge(int stuAge)**  **{**  **this.stuAge = stuAge;**  **}**  **}**  **class OverloadConstructor**  **{**  **static void Main(string[] args)**  **{**  **StudentData myobj = new StudentData();//call to Default Constructor**  **Console.WriteLine("\nConstructor 1:Default Constructor");**  **Console.WriteLine("Student Name:" + myobj.getStuName());**  **Console.WriteLine("Student Age:" + myobj.getStuAge());**  **Console.WriteLine("Student ID:" + myobj.getStuID());**  **Console.WriteLine("\nConstructor 2:Parameterized Constructor");**  **//call to Parameterized Constructor**  **StudentData myobj2 = new StudentData(58, "Sandeep Jain", 20);**  **Console.WriteLine("Student Name:" + myobj2.getStuName());**  **Console.WriteLine("Student Age:" + myobj2.getStuAge());**  **Console.WriteLine("Student ID:" + myobj2.getStuID());**  **Console.WriteLine("\nConstructor 3:Copy Constructor");**  **//call to Copy Constructor**  **StudentData myobj3 = new StudentData(myobj2);**  **Console.WriteLine("Student Name:" + myobj3.getStuName());**  **Console.WriteLine("Student Age:" + myobj3.getStuAge());**  **Console.WriteLine("Student ID :" + myobj3.getStuID());**  **Console.ReadKey();**  **}**  **}**  **}**  **Output**    **iv. Interfaces (Multiple Inheritance using Interfaces)**  **Code**  **using System;**  **namespace Practical2b4**  **{**  **interface calc1**  **{**  **int add(int a, int b);**  **}**  **interface calc2**  **{**  **int sub(int x, int y);**  **}**  **interface calc3**  **{**  **int mul(int r, int s);**  **}**  **interface calc4**  **{**  **int div(int c, int d);**  **}**  **class Calculation : calc1, calc2, calc3, calc4**  **{**  **public int result1;**  **public int add(int a, int b)**  **{**  **return result1 = a + b;**  **}**  **public int result2;**  **public int sub(int x, int y)**  **{**  **return result2 = x - y;**  **}**  **public int result3;**  **public int mul(int r, int s)**  **{**  **return result3 = r \* s;**  **}**  **public int result4;**  **public int div(int c, int d)**  **{**  **return result4 = c / d;**  **}**  **}**  **class MultipleInheritance**  **{**  **static void Main(string[] args)**  **{**  **Calculation c = new Calculation();**    **c.add(8, 2);**  **c.sub(20, 10);**  **c.mul(5, 2);**  **c.div(20, 10);**  **Console.WriteLine("Multiple Inheritance concept Using Interfaces :\n ");**  **Console.WriteLine("Addition: " + c.result1);**  **Console.WriteLine("Subtraction: " + c.result2);**  **Console.WriteLine("Multiplication:" + c.result3);**  **Console.WriteLine("Division: " + c.result4);**  **Console.ReadKey();**  **}**  **}**  **}**  **Output**    using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp4  {  interface Gross  {  int TA  {  get;  set;  }  int DA  {  get;  set;  }  double Gross\_sal();  }  class Employee  {  public string name;  public Employee(string name)  {  this.name = name;  }  public double basic\_sal(double sal)  {  return sal;  }  }  class Salary : Employee, Gross  {  double hra;  int ta, da;  public Salary( string name,double hra ) : base(name)  {  this.hra = hra;  }  public double Gross\_sal()  {  double b = basic\_sal(300000);  return (TA + DA + b + hra);  }  public int TA  {  get { return ta; }  set { ta = value; }  }  public int DA  {  get { return da; }  set { da = value; }  }  public void display()  {  Console.WriteLine("Name: " + name);  string v = "Salary: " + Gross\_sal();  object p = v;  Console.WriteLine(p);  }  }  class Program  {  static void Main(string[] args)  {  Salary s = new Salary("Sandeep",5000000);  s.TA = 2000;  s.DA = 5000;  s.display();  Console.ReadKey();  }  }  } | | | |
| 1. **Create simple application to demonstrate use of Delegates and events**   **i .Using Delegates and events**  **Code**  **using System;**  **namespace Delegate**  **{**  **public delegate void EventDelegate(string str);//delegate declaration**  **public class EventClass**  **{**  **public event EventDelegate Status;//declaration of Event**  **public void TriggerEvent()**  **{**  **if (Status != null)**  **{**  **Status("Event Triggered");**  **}**  **else**  **{**  **Console.WriteLine("Not Registered");**  **}**  **}**  **}**  **class DelegateEvent**  **{**  **public static void CatchEvent(string str)//Delegate Method definition**  **{**  **Console.WriteLine(str);**  **}**    **static void Main(string[] args)**  **{**  **EventClass ec = new EventClass();**  **DelegateEvent et = new DelegateEvent();**  **ec.Status += new EventDelegate(CatchEvent);**  **ec.TriggerEvent();**  **Console.ReadLine();**  **}**  **}**  **}**  **Output**    **ii. Exception handling**  **Create user defined exception “bankexception” which will throw the exception when the**  **balance amount is less than 500 into the account. Then write a program to show the use of**  **“bank exception”.**  **Code:**  **using System;**  **namespace Exception\_handling**  **{**  **class BankException : Exception**  **{**  **public BankException(string msg)**  **: base(msg)**  **{**  **}**  **}**  **class BankAccount**  **{**  **public int AccountNo;**  **public double Balance;**  **public BankAccount(int AcctNo, double Balance)**  **{**  **try**  **{**  **AccountNo = AcctNo;**  **if (Balance < 500) throw new BankException("Oops!!!Lower Limit reached for Balance");**  **else**  **this.Balance= Balance;**  **}**  **catch (BankException e)**  **{**  **Console.WriteLine(e.Message);**  **}**  **finally**  **{**  **Console.WriteLine("Account no:{0} and Balance {1}", AccountNo, Balance);**  **Console.ReadKey();**  **}**  **}**  **}**  **class ExceptionHandling**  **{**  **static void Main(string[] args)**  **{**  **int a;**  **double b;**  **Console.Write("Enter Account Number:");**  **a = int.Parse(Console.ReadLine());**  **Console.Write("Enter Account Balance:");**  **b = double.Parse(Console.ReadLine());**  **BankAccount obj = new BankAccount(a, b);**  **}**  **}**  **}**  **Output** | | | |
|  | | | |
|  | | | |