3. CSharp OOPs Assignment

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3\_CSharpOOPsAssign

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter Employee No.");

int emp\_no = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter Employee Name");

string emp\_name = Console.ReadLine();

Console.WriteLine("Enter the Salary");

double emp\_salary = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter Designation");

string emp\_desg = Console.ReadLine();

if(String.Equals(emp\_desg, "Manager"))

{

Employee emp = new Manager { EmpNo = emp\_no , EmpName = emp\_name , Salary = emp\_salary };

Console.WriteLine(emp.ToString());

}

else

{

Console.WriteLine("Enter the kilometer");

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

Employee emp = new MarketingExecutive(km);

emp.EmpNo = emp\_no; emp.EmpName = emp\_name; emp.Salary = emp\_salary;

Console.WriteLine(emp.ToString());

}

MyStack stack = new MyStack(5);

stack.Push(1);

stack.Push(2);

stack.Push(3);

int x = stack.Pop();

stack.Push(4);

int y = stack.Pop();

stack.Push(5);

MyStack stack1 = stack.Clone() as MyStack;

Console.WriteLine("Size of first stack "+stack);

Console.WriteLine("Size of second stack after cloning from first " + stack1);

Console.ReadKey();

}

}

}

**Employee.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3\_CSharpOOPsAssign

{

internal class Employee

{

private double HRA, TA, PF, DA, TDS, NetSalary, GrossSalary;

public int EmpNo { get; set; }

public string EmpName { get; set; }

public double Salary { get; set; }

public virtual double calculateSalary(double Salary)

{

try

{

if (Salary < 5000)

{

HRA = Salary \* 10 / 100; TA = Salary \* 5 / 100; DA = Salary \* 15 / 100;

}

else if (Salary > 5000 && Salary < 10000)

{

HRA = Salary \* 15 / 100; TA = Salary \* 10 / 100; DA = Salary \* 20 / 100;

}

else if (Salary > 10000 && Salary < 15000)

{

HRA = Salary \* 20 / 100; TA = Salary \* 15 / 100; DA = Salary \* 25 / 100;

}

else if (Salary > 15000 && Salary < 20000)

{

HRA = Salary \* 25 / 100; TA = Salary \* 20 / 100; DA = Salary \* 30 / 100;

}

else

{

HRA = Salary \* 30 / 100; TA = Salary \* 25 / 100; DA = Salary \* 35 / 100;

}

GrossSalary = Salary + HRA + TA + DA;

}

catch (DivideByZeroException)

{

Console.WriteLine("Salary cannot be zero");

}

catch (Exception ex)

{

Console.WriteLine("Give a proper salary ");

}

PF = GrossSalary \* 10 / 100;

TDS = GrossSalary \* 18 / 100;

NetSalary = GrossSalary - (PF + TDS);

return GrossSalary;

}

}

}

**Manager.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3\_CSharpOOPsAssign

{

internal class Manager:Employee

{

private double PA,FA,OA,PF,TDS,netSalary;

public override double calculateSalary(double Salary)

{

double baseSalary = base.calculateSalary(Salary);

PA = 0.08 \* Salary;

FA = 0.13 \* Salary;

OA = 0.03 \* Salary;

double grossSalary = baseSalary + PA + FA + OA;

PF = grossSalary \* 10 / 100;

TDS = grossSalary \* 18 / 100;

netSalary = grossSalary - (PF + TDS);

return grossSalary;

}

public override string ToString()

{

double grossSalary = calculateSalary(Salary);

return "\nEmployee No. : " + EmpNo + "\nEmployee Name : " + EmpName + "\nEmployee Salary : " + Salary +

"\nGross Salary : " + grossSalary + "\nPF : " + PF + "\nTDS : " + TDS +

"\nNet Salary : " + netSalary;

}

}

}

**MaeketingExecutive.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3\_CSharpOOPsAssign

{

internal class MarketingExecutive:Employee

{

private double km, TA, TelA,PF, TDS, netSalary;

public MarketingExecutive(double \_km) { km = \_km;}

public override double calculateSalary(double Salary)

{

double baseSalary = base.calculateSalary(Salary);

TA = km \* 5;

TelA = 1000;

double grossSalary = baseSalary + TA + TelA;

PF = grossSalary \* 10 / 100;

TDS = grossSalary \* 18 / 100;

netSalary = grossSalary - (PF + TDS);

return grossSalary;

}

public override string ToString()

{

double grossSalary = calculateSalary(Salary);

return "\nEmployee No. : " + EmpNo + "\nEmployee Name : " + EmpName + "\nEmployee Salary : " + Salary +

"\nGross Salary : " + grossSalary + "\nPF : " + PF + "\nTDS : " + TDS +

"\nNet Salary : " + netSalary;

}

}

}

**MyStack.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3\_CSharpOOPsAssign

{

internal class MyStack:ICloneable

{

private int top = 0;

private int size;

private int[] stack;

public MyStack(int size)

{

this.size = size;

stack = new int[size];

}

public bool IsEmpty()

{

if (top == 0)

return true;

else

return false;

}

public void Push(int element)

{

if (top > size)

throw new Exception("Stack Overflow");

stack[top] = element;

top++;

}

public int Pop()

{

if(IsEmpty())

throw new Exception("Stack Underflow");

else

{

top--;

return stack[top];

}

}

public object Clone()

{

return new MyStack(size);

}

public override string ToString()

{

return size.ToString();

}

}

}