**ASSIGNMENT 3**

1) using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

public class Inheritence

{

class Employee

{

protected int EmpNo { get; set; }

protected string EmpName { get; set; }

protected double Salary { get; set; }

protected double HRA { get; set; }

protected double TA { get; set; }

protected double DA { get; set; }

public double PF { get; set; }

protected double TDS { get; set; }

protected double NetSalary { get; set; }

protected double GrossSalary { get; set; }

public Employee(int EmpNum, string Emp\_Name, double Sal)

{

EmpNo = EmpNum;

EmpName = Emp\_Name;

Salary = Sal;

if (Salary < 5000)

{

HRA = Salary \* 10 / 100;

TA = Salary \* 5 / 100;

DA = Salary \* 15 / 100;

GrossSalary = Salary + HRA + TA + DA;

}

else if (Salary < 10000)

{

HRA = Salary \* 15 / 100;

TA = Salary \* 10 / 100;

DA = Salary \* 20 / 100;

GrossSalary = Salary + HRA + TA + DA;

}

else if (Salary < 15000)

{

HRA = Salary \* 20 / 100;

TA = Salary \* 15 / 100;

DA = Salary \* 25 / 100;

GrossSalary = Salary + HRA + TA + DA;

}

else if (Salary < 20000)

{

HRA = Salary \* 25 / 100;

TA = Salary \* 20 / 100;

DA = Salary \* 30 / 100;

GrossSalary = Salary + HRA + TA + DA;

}

else

{

HRA = Salary + 30 / 100;

TA = Salary + 25 / 100;

DA = Salary + 35 / 100;

GrossSalary = Salary + HRA + TA + DA;

}

}

public virtual void CalculateSalary()

{

PF = GrossSalary \* (0.1);

TDS = GrossSalary \* (0.18);

NetSalary = GrossSalary - (PF + TDS);

Console.WriteLine("NetSalary of Employee:{0}", NetSalary);

Console.WriteLine("PF of Employee:{0}", PF);

}

public virtual void Grosssal()

{

Console.WriteLine("GrossSalary of Employee:{0}", GrossSalary);

}

}

class Manager : Employee

{

private double PetrolAllowance { get; set; }

private double FoodAllowance { get; set; }

private double OtherAllowance { get; set; }

public Manager(int EmpNum, string Emp\_Name, double Sal) : base(EmpNum, Emp\_Name, Sal)

{

PetrolAllowance = Salary \* 8 / 100;

FoodAllowance = Salary \* 13 / 100;

OtherAllowance = Salary \* 3 / 100;

}

public override void Grosssal()

{

GrossSalary = (GrossSalary + PetrolAllowance + FoodAllowance + OtherAllowance);

Console.WriteLine("GrossSalary of Manager :{0}", GrossSalary);

}

public override void CalculateSalary()

{

PF = GrossSalary \* (0.1);

TDS = GrossSalary \* (0.18);

NetSalary = GrossSalary - (PF + TDS);

Console.WriteLine("NetSalary of Manager :{0}", NetSalary);

Console.WriteLine("PF of Manager:{0}", PF);

}

}

class MarketingExecutive : Employee

{

private double kilometertravel { get; set; }

private double TourAllowance { get; set; }

private double TelephoneAllowance { get; set; }

public MarketingExecutive(int EmpNum, string Emp\_Name, double Sal, double klt) : base(EmpNum, Emp\_Name, Sal)

{

kilometertravel = klt;

TourAllowance = kilometertravel \* 5;

TelephoneAllowance = 1000;

}

public override void Grosssal()

{

GrossSalary = (GrossSalary + TourAllowance + TelephoneAllowance);

Console.WriteLine("GrossSalary of MarketingExecutive:{0}", GrossSalary);

}

public override void CalculateSalary()

{

PF = GrossSalary \* (0.1);

TDS = GrossSalary \* (0.18);

NetSalary = GrossSalary - (PF + TDS);

Console.WriteLine("NetSalary of MarketingExecutive:{0}", NetSalary);

Console.WriteLine("PF of MarketingExecutive:{0}", PF);

}

}

public interface IPrintableinterface

{

void IPrintableMethod();

}

class Program : IPrintableinterface

{

public void IPrintableMethod()

{

Employee Emp1 = new Employee(1, "Abhi", 14000);

Manager M1 = new Manager(2, "Amit", 25000);

MarketingExecutive MarkE1 = new MarketingExecutive(3, "Ronit ", 40000, 10);

Console.WriteLine("empNum:1", Emp1);

Console.WriteLine("EmpName:Abhi", Emp1);

Console.WriteLine("salary:14000", Emp1);

Console.WriteLine("empNum:2", M1);

Console.WriteLine("EmpName:Amit", M1);

Console.WriteLine("salary:25000", M1);

Console.WriteLine("empNum:3", MarkE1);

Console.WriteLine("EmpName:Ronit", MarkE1);

Console.WriteLine("salary:40000", MarkE1);

}

public static void Main(string[] args)

{

Program P = new Program();

P.IPrintableMethod();

Employee Emp1 = new Employee(1, "Abhi", 14000);

Manager M1 = new Manager(2, "Amit", 25000);

MarketingExecutive MarkE1 = new MarketingExecutive(3, " Ronit ", 40000, 10);

Emp1.Grosssal();

M1.Grosssal();

MarkE1.Grosssal();

M1.CalculateSalary();

Emp1.CalculateSalary();

MarkE1.CalculateSalary();

Console.ReadKey();

}

}

}

2) using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StackImplementation

{

class MyStack

{

static readonly int MAX = 1000;

int top;

int[] stack = new int[MAX];

public bool IsEmpty()

{

return (top < 0);

}

public MyStack()

{

top = -1;

}

public bool Push(int data)

{

if (top >= MAX)

{

Console.WriteLine("Stack Overflow");

return false;

}

else

{

stack[++top] = data;

return true;

}

}

public int Pop()

{

if (top < 0)

{

Console.WriteLine("Stack Underflow");

return 0;

}

else

{

int value = stack[top--];

return value;

}

}

public void Peek()

{

if (top < 0)

{

Console.WriteLine("Stack Underflow");

return;

}

else

Console.WriteLine("The topmost element of Stack is : {0}", stack[top]);

}

public void PrintStack()

{

if (top < 0)

{

Console.WriteLine("Stack Underflow");

return;

}

else

{

Console.WriteLine("Items in the Stack are :");

for (int i = top; i >= 0; i--)

{

Console.WriteLine(stack[i]);

}

}

}

public virtual object Clone()

{

return new StackImplementation.MyStack();

}

}

class Program

{

public static void Main(string[] args)

{

MyStack ms= new MyStack();

ms.Push(10);

ms.Push(20);

ms.Push(30);

ms.Push(40);

ms.PrintStack();

ms.Peek();

Console.WriteLine("Item popped from Stack : {0}", ms.Pop());

ms.PrintStack();

}

}

}

3) using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

class Stack\_Q3

{

private int[] ele;

private int top;

private int max;

public class StackException\_push:Exception

{

public override string Message

{

get { return "Stack Overflow, can't perform push operation further";

}

}

}

public class StackException\_pop : Exception

{

public override string Message

{

get

{

return "Stack is Empty, can't perform pop operation further";

}

}

}

public Stack\_Q3(int size)

{

ele = new int[size];

top = -1;

max = size;

}

public void push(int item)

{

if (top == max - 1)

{

throw new StackException\_push();

}

else

{

ele[++top] = item;

}

}

public int pop()

{

if (top == -1)

{

throw new StackException\_pop();

}

else

{

Console.WriteLine("Popped element is " + ele[top]);

return ele[top--];

}

}

public void printStack()

{

if (top == -1)

{

Console.WriteLine("Stack is Empty");

return;

}

else

{

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

{

Console.WriteLine("Item[" + (i + 1) + "]: " + ele[i]);

}

}

}

}

class Program

{

public static void Main(string[] args)

{

try

{

Stack\_Q3 S = new Stack\_Q3(7);

S.push(15);

S.push(16);

S.push(17);

S.push(18);

S.push(19);

S.push(20);

S.push(21);

// S.push(22);

Console.WriteLine("Items are : ");

S.printStack();

S.pop();

S.pop();

S.pop();

S.pop();

S.pop();

S.pop();

S.pop();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Console.ReadKey();

}

}