**ASSIGNMENT 3**

1.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

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 EmpName, double Sal)

{

EmpNo = EmpNo;

EmpName = EmpName;

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 ClaculateSalary()

{

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 Gsal()

{

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 EmpNm, double Sal) : base(EmpNum, EmpNm, Sal)

{

PetrolAllowance = Salary \* 8 / 100;

FoodAllowance = Salary \* 13 / 100;

OtherAllowance = Salary \* 3 / 100;

}

public override void Gsal()

{

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

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

}

public override void ClaculateSalary()

{

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 EmpNm, double Sal, double klt) : base(EmpNum, EmpNm, Sal)

{

kilometertravel = klt;

TourAllowance = kilometertravel \* 5;

TelephoneAllowance = 1000;

}

public override void Gsal()

{

GrossSalary = (GrossSalary + TourAllowance + TelephoneAllowance);

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

}

public override void ClaculateSalary()

{

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, "chitti", 25000);

Manager M1 = new Manager(2, "Priya", 30000);

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

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

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

Console.WriteLine("salary:25000", Emp1);

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

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

Console.WriteLine("salary:30000", M1);

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

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

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

}

public static void Main(string[] args)

{

Program P = new Program();

P.IPrintableMethod();

Employee Emp1 = new Employee(1, "chitti", 25000);

Manager M1 = new Manager(2, "Priya", 30000);

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

Emp1.Gsal();

M1.Gsal();

MarkE1.Gsal();

M1.ClaculateSalary();

Emp1.ClaculateSalary();

MarkE1.ClaculateSalary();

}

}

}

}

2.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Stackprogram

{

class MyStack

{

private int[] arr;

private int top;

private int max;

//Constructor initialize the variables || it calls as soon as object get created.

public MyStack(int size)

{

arr = new int[size];

top = -1;

max = size;

}

public void push(int item)

{

if (top == max - 1)

{

throw new Exception("Stack Overflow, can't perform push");

}

else

{

arr[++top] = item;

}

}

public int pop()

{

if (top == -1)

{

throw new Exception("Stack is Empty");

}

else

{

Console.WriteLine("Poped element is " + arr[top]);

return arr[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) + "): " + arr[i]);

}

}

}

}

class program

{

public static void Main(string[] args)

{

//Stack is class || S is object

MyStack St = new MyStack(5);

//objectname.methodname() -> to call the method inside the class

St.push(10); // Last in First Out

St.push(20);

St.push(30);

St.push(40);

St.push(50);

Console.WriteLine("Items are : ");

St.printStack();

St.pop();

St.pop();

St.pop();

St.pop();

St.pop();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using StackImplementation;

using System.Collections;

namespace Iclonableinterface

{

class Client

{

public static void Main(string[] args)

{

try

{

StackImplementation.Stack mystack = new StackImplementation.Stack();

mystack.Push(10);

mystack.Push(20);

mystack.Push(30);

mystack.Push(40);

mystack.PrintStack();

mystack.Peek();

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

mystack.PrintStack();

StackImplementation.Stack mystack2 = (StackImplementation.Stack)mystack.Clone();

Console.WriteLine("Cloned Stack", mystack2);

mystack.Push(10);

mystack.Push(20);

mystack.Push(30);

mystack.Push(40);

mystack.PrintStack();

mystack.Peek();

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

mystack2.PrintStack();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

}

3.

using System;

using System.Collections;

namespace ExceptionHandling

{

public class Stack

{

static readonly int MAX = 1000;

int top;

int[] stack = new int[MAX];

public bool IsEmpty()

{

return (top < 0);

}

public Stack()

{

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.Stack();

}

}

class Program

{

public static void Main(string[] args)

{

Stack myStack = new Stack();

myStack.Push(10);

myStack.Push(20);

myStack.Push(30);

myStack.Push(40);

myStack.PrintStack();

myStack.Peek();

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

myStack.PrintStack();

}

}

}