1. Create a hierarchy of Employee, Manager, MarketingExecutivein Employee Management System. They should have the followingfunctionality.

* 1. Manager with following privatemembers.
     + Petrol Allowance: 8 % of Salary.
     + Food Allowance : 13 % ofSalary.
     + Other Allowances : 3% ofSalary.

Calculate GrossSalaryby adding above allowances. Override CalculateSalary() method to calculate Net Salary. NetSalary.PF calculation should not consider above allowances.

* 1. MarketingExecutivewith following private members.
     + Kilometertravel
     + Tour Allowances : Rs 5/- per Kilometer (Automaticallygenerated).
     + Telephone Allowances :Rs.1000/-

Calculate GrossSalaryby adding above allowances. Override CalculateSalary(). NetSalary,PFcalculation should not consider above allowances.

**Employee.cs**

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Threading.Tasks;

namespaceLitwareLib

{

publicclassEmployee

{

intEmpNo;

stringEmpName;

double Salary;

double HRA;

double TA;

double DA;

double PF;

double TDS;

doubleNetSalary;

doubleGrossSalary;

publicdouble \_GrossSalary

{

get

{

returnGrossSalary;

}

set

{

GrossSalary = value;

}

}

publicdouble \_Salary

{

get

{

return Salary;

}

set

{

Salary = value;

}

}

publicdouble \_PF

{

get

{

return PF;

}

set

{

PF = value;

}

}

publicdouble \_TDS

{

get

{

return TDS;

}

set

{

TDS = value;

}

}

publicdouble \_NetSalary

{

get

{

returnNetSalary;

}

set

{

NetSalary = value;

}

}

publicvoidsetEmpDetails(intEmpNo, stringEmpName, double Salary)

{

this.EmpNo = EmpNo;

this.EmpName = EmpName;

this.Salary = Salary;

}

publicvoidgetEmpDetails()

{

Console.WriteLine("Employee Number is: " + this.EmpNo);

Console.WriteLine("Employee Name is: " + this.EmpName);

Console.WriteLine("Employee Salary is: " + this.Salary);

}

publicdoublesetHRA(double HRA)

{

if (Salary > 0 & Salary < 5000)

{

HRA = (10 \* Salary) / 100;

returnthis.HRA = HRA;

}

elseif (Salary > 5000 & Salary < 10000)

{

HRA = (15 \* Salary) / 100;

returnthis.HRA = HRA;

}

elseif (Salary > 10000 & Salary < 15000)

{

HRA = (20 \* Salary) / 100;

returnthis.HRA = HRA;

}

elseif (Salary > 15000 & Salary < 20000)

{

HRA = (25 \* Salary) / 100;

returnthis.HRA = HRA;

}

else

{

HRA = (30 \* Salary) / 100;

returnthis.HRA = HRA;

}

}

publicdoublesetTA(double TA)

{

this.TA = TA;

if (Salary < 5000)

{

TA = (5 \* Salary) / 100;

returnthis.TA = TA;

}

elseif (Salary < 10000)

{

TA = (10 \* Salary) / 100;

returnthis.TA = TA;

}

elseif (Salary < 15000)

{

TA = (15 \* Salary) / 100;

returnthis.TA = TA;

}

elseif (Salary < 20000)

{

TA = (20 \* Salary) / 100;

returnthis.TA = TA;

}

else

{

TA = (25 \* Salary) / 100;

returnthis.TA = TA;

}

}

publicdoublesetDA(double DA)

{

this.DA = DA;

if (Salary < 5000)

{

DA = (15 \* Salary) / 100;

returnthis.DA = DA;

}

elseif (Salary < 10000)

{

DA = (20 \* Salary) / 100;

returnthis.DA = DA;

}

elseif (Salary < 15000)

{

DA = (25 \* Salary) / 100;

returnthis.DA = DA;

}

elseif (Salary < 20000)

{

DA = (30 \* Salary) / 100;

returnthis.DA = DA;

}

else

{

DA = (35 \* Salary) / 100;

returnthis.DA = DA;

}

}

publicdoublegetGrossSalary(double Salary, double HRA, double TA, double DA)

{

this.GrossSalary = Salary + HRA + TA + DA;

returnthis.GrossSalary;

}

publicvirtualvoidCalculateSalary(doubleGrossSalary)

{

this.PF = (10 \* GrossSalary) / 100;

this.TDS = (18 \* GrossSalary) / 100;

this.NetSalary = GrossSalary - (this.PF + this.TDS);

}

publicvoidshowSalary()

{

Console.WriteLine("Your PF is: {0}", this.PF);

Console.WriteLine("Your TDS is: {0}", this.TDS);

Console.WriteLine("Your NetSalary is: {0}", this.NetSalary);

}

}

//Manager

publicclassManager : Employee

{

doublePetrol\_Allowance;

doubleFood\_Allowance;

doubleOther\_Allowance;

publicdoublesetPetrol(double \_Salary)

{

this.Petrol\_Allowance = (8 \* \_Salary) / 100;

returnthis.Petrol\_Allowance;

}

publicdoublesetFood(double \_Salary)

{

this.Food\_Allowance = (13 \* \_Salary) / 100;

returnthis.Food\_Allowance;

}

publicdoublesetOther(double \_Salary)

{

this.Other\_Allowance = (3 \* \_Salary) / 100;

returnthis.Other\_Allowance;

}

publicvirtualdoublesetAllowance(doublePetrol\_Allowance, doubleFood\_Allowance, doubleOther\_Allowance)

{

this.\_GrossSalary = Petrol\_Allowance + Food\_Allowance + Other\_Allowance;

returnthis.\_GrossSalary;

}

publicoverridevoidCalculateSalary(doubleGrossSalary)

{

this.\_PF = (10 \* GrossSalary) / 100;

this.\_TDS = (18 \* \_GrossSalary) / 100;

this.\_NetSalary = \_GrossSalary - (this.\_PF + this.\_TDS);

}

publicvoidshowSalary()

{

Console.WriteLine("Your PF is: {0}", this.\_PF);

Console.WriteLine("Your TDS is: {0}", this.\_TDS);

Console.WriteLine("Your NetSalary is: {0}", this.\_NetSalary);

}

}

//MarketingExecutive

publicclassMarketingExecutive : Manager

{

privatedoubleKilometer\_travel;

privatedoubleTour\_Allowance = 5;

privatedoubleTelephone\_Allowance = 1000;

publicdoublesetTravel(doubleKilometer\_travel)

{

returnthis.Kilometer\_travel = Kilometer\_travel;

}

publicvoidgetTravel()

{

Console.WriteLine("You traveledfor : " + Kilometer\_travel + " kms");

}

publicdoublegetTour(doubleTour\_Allowance)

{

this.Tour\_Allowance = 5 \* this.Kilometer\_travel;

returnthis.Tour\_Allowance;

}

publicdoublesetTour(doubleTour\_Allowance)

{

this.Tour\_Allowance = 5 \* this.Kilometer\_travel;

returnthis.Tour\_Allowance;

}

publicvoidgetTour()

{

Console.WriteLine("Tour Allowance per km is: Rs.{0}", this.Tour\_Allowance);

}

publicdoublesetTelephone()

{

returnthis.Telephone\_Allowance = 1000;

}

publicoverridedoublesetAllowance(doubleKilometer\_travel, doubleTour\_Allowance, doubleTelephone\_Allowance)

{

this.\_GrossSalary = Tour\_Allowance + Telephone\_Allowance;

returnthis.\_GrossSalary;

}

publicoverridevoidCalculateSalary(doubleGrossSalary)

{

this.\_PF = (10 \* GrossSalary) / 100;

this.\_TDS = (18 \* GrossSalary) / 100;

this.\_NetSalary = GrossSalary - (this.\_PF + this.\_TDS);

}

publicvoidshowSalary()

{

Console.WriteLine("Your PF is: {0}", this.\_PF);

Console.WriteLine("Your TDS is: {0}", this.\_TDS);

Console.WriteLine("Your NetSalary is: {0}", this.\_NetSalary);

}

}

}

Program.cs

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Threading.Tasks;

usingLitwareLib;

namespace Assignment2

{

publicclassProgram

{

staticvoid Main(string[] args)

{

Employee obj = newEmployee();

Console.WriteLine("----------Accepting Employee Details----------");

Console.WriteLine("Enter Employee Number:");

int no = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Employee Name:");

string name = Console.ReadLine();

Console.WriteLine("Enter Employee Salary:");

double salary = double.Parse(Console.ReadLine());

obj.setEmpDetails(no, name, salary);

Console.WriteLine("\n----------Displaying Employee Details----------");

obj.getEmpDetails();

doubleresultHRA = obj.setHRA(salary);

Console.WriteLine("Your HRA is: {0}", resultHRA);

doubleresultTA = obj.setTA(salary);

Console.WriteLine("Your TA is: {0}", resultTA);

doubleresultDA = obj.setDA(salary);

Console.WriteLine("Your DA is: {0}", resultDA);

doublegrossSalary = obj.getGrossSalary(salary, resultHRA, resultDA, resultTA);

Console.WriteLine("Your Gross Salary is: {0}", grossSalary);

obj.CalculateSalary(grossSalary);

obj.showSalary();

Console.WriteLine("\n----------Displaying Manager Details----------");

Manager obj1 = newManager();

double petrol = obj1.setPetrol(salary);

Console.WriteLine("Petrol Allowance: {0}", petrol);

double food = obj1.setFood(salary);

Console.WriteLine("Food Allowance: {0}", food);

double other = obj1.setOther(salary);

Console.WriteLine("Other Allowance: {0}", other);

double gross1 = obj1.getGrossSalary(salary, resultHRA, resultDA, resultTA);

double gross2 = obj1.setAllowance(petrol, food, other);

doubleresult\_gross = gross1 + gross2;

Console.WriteLine("Gross Salary of Manager on adding the Allowances is: {0}", result\_gross);

obj.CalculateSalary(result\_gross);

obj.showSalary();

Console.WriteLine("\n---------Displaying Marketing Executive----------");

MarketingExecutive obj2 = newMarketingExecutive();

Console.WriteLine("Enter Kilometer Travel: ");

double travel = double.Parse(Console.ReadLine());

obj2.setTravel(travel);

obj2.getTravel();

double tour = obj2.setTour(travel);

obj2.getTour();

double tele = obj2.setTelephone();

Console.WriteLine("Telephone Allowances are: Rs.{0}", tele);

double gross3 = obj2.setAllowance(travel, tour, tele);

doubleresult\_gross\_final = result\_gross + gross3;

Console.WriteLine("Gross Salary of Marketing Executive on adding the Allowances is: {0}", result\_gross\_final);

obj2.CalculateSalary(result\_gross\_final);

obj2.showSalary();

Console.ReadLine();

}

}

}

2. of Employee on console.

1)Write a class called MyStackwith following members.

* 1. integerarray
  2. integer variable to store topposition
  3. size of thearray.

//<summary>

//Stacks are abstract data types

//LIFO - Last In First Out

//Push Pop and Peek operations

//Push refers to inserting an item at the top of the Stack

//Pop refers to removing an item from the top of the Satck

//Peek refers to the top element of the Stack

//Stack can be imlemented in multiple ways

//Typically they are implemented using arrays or linked lists

//arrays are more efficient because they stored items in a sequential memory

//</summary>

using System;

namespace Assignment3

{

class MyStack

{

private int[] StackArray;

private int top; //index of top of the stack

private int max;

public MyStack(int size)

{

StackArray = new int[size];

top = -1; //empty stack

max = size; // size = 3 max = 3

}

public void push(int item)

{

if (top == max - 1) // max - 1 = index of the top

{

throw new StackException("Stack OverFlow !! You are trying to add items to the Stack that is full.");

}

else

{

StackArray[++top] = item; //increment index

}

}

public int pop()

{

if (top == -1)

{

throw new StackException("Error: No element to pop");

}

else

{

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

return StackArray[top--]; // decrement the index

}

}

public void displayStack()

{

if (top == -1)

{

throw new StackException("Stack is Empty");

}

else

{

Console.WriteLine("Items inserted are: ");

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

{

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

}

}

}

}

class Program

{

static void Main()

{

MyStack S = new MyStack(3);

try

{

S.push(133);

S.push(213);

S.push(768);

S.push(140);

S.push(15);

S.push(10);

}

catch (StackException se)

{

Console.WriteLine(se);

}

try

{

//S.pop();

//S.pop();

}

catch (StackException se)

{

Console.WriteLine(se);

}

try

{

S.displayStack();

}

catch (StackException se)

{

Console.WriteLine(se);

}

Console.ReadLine();

}

}

public class StackException: Exception

{

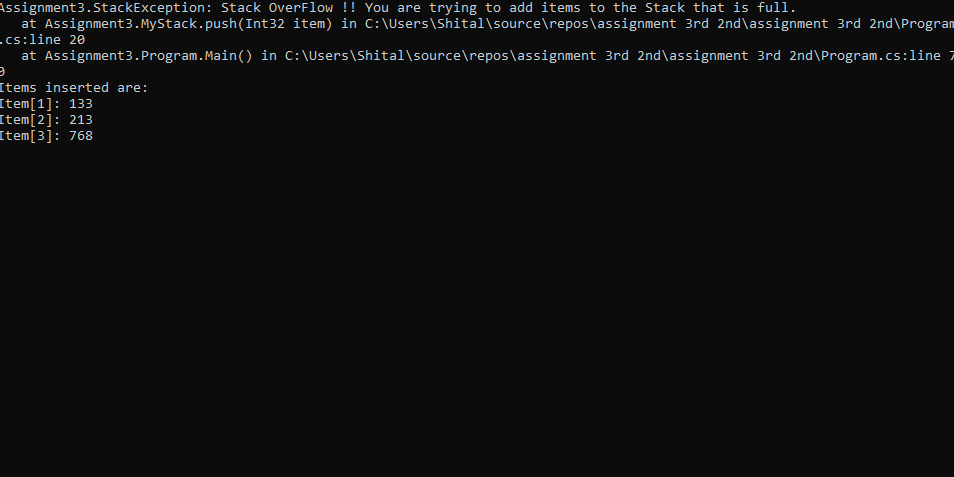
public StackException(string s) : base(s)

{

}

}

}



3.Create a custom exception class named StackException. The Push()and Pop() method should throw object of StackExceptionwhen the stack is full or empty respectively.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication1

{

class Stack : ICloneable

{

private int[] ele;

private int top;

private int max;

public Stack(int size)

{

ele = new int[size];

top = -1;

max = size;

}

public void push(int item)

{

try

{

ele[++top] = item;

}

catch (Exception Stackexception)

{

throw new StackOverflowException(GetType().Name);

}

}

public int pop()

{

try

{

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

return ele[top--];

}

catch (Exception Stackexception)

{

throw new InsufficientExecutionStackException("stack underflow exception");

}

}

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]);

}

}

}

public object Clone()

{

return this;

}

}

class Program

{

public static void Main()

{

try

{

Console.WriteLine("enter length of the Stack");

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

Stack S = new Stack(length);

again:

Console.WriteLine("do you want to push or pop");

string choice = Console.ReadLine();

{

if (choice.ToUpper() == "POP")

{

S.pop();

goto again;

}

else if (choice.ToUpper() == "PUSH")

{

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

S.push(num);

goto again;

}

else

{

S.printStack();

}

}

S.printStack();

S.Clone();

}

catch (Exception ex)

{

Console.WriteLine(ex.GetType().Name);

}

}

}

}

