# Inheritance and Polymorphism

## Objective

* To use inheritance (“is-a”relationship).
* To usepolymorphism.
* To create an abstractclass
* To useinterface.

## Assignments to be done in this session

1. Create a hierarchy of Employee, Manager, MarketingExecutive in 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.

Implement IPrintableinterface for every Employee which will allow to print details of Employee on console.

1. Write a class called MyStackwith following members.
   1. integerarray
   2. integer variable to store topposition
   3. size of thearray.

Implement Push()and Pop()operation. Implement ICloneableinterface to perform cloning. Write a client application to perform cloning.

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

Program

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Threading.Tasks;

usingstaticSystem.Console;

namespaceEmployeeManagementSystem

{

publicinterfaceIPrintable

{

voidDisplayEmployeeDetails();

}

publicclassEmployee

{

publicintemp\_id;

publicstringemp\_name;

publicfloat Salary;

publicfloatGrossSalary;

publicvoidGetSalaryWithEmployeeDetails()

{

WriteLine("Enter Employee Details Like emp\_id,emp\_name");

emp\_id = int.Parse(ReadLine());

emp\_name = ReadLine();

WriteLine("Enter Salary of an Employe .");

Salary = float.Parse(ReadLine());

}

publicvirtualvoidCalculateSalary()

{

}

}

publicclassManager : Employee,IPrintable

{

privatefloat PA;

privatefloat FA;

privatefloat OA;

publicoverridevoidCalculateSalary()

{

PA = 8 \* Salary / 100;

FA=13\*Salary / 100;

OA=3\*Salary / 100;

GrossSalary = Salary + PA + FA + OA;

}

publicvoidDisplayEmployeeDetails()

{

WriteLine("Manager Details...");

WriteLine("emp\_id: {0}\nemp\_name: {1}\nGrossSalary:{2}",emp\_id,emp\_name,GrossSalary);

}

}

publicclassMarketingExecutive : Employee,IPrintable

//inherited Employee class and interface IEmployee

{

privateint Distance;

privatefloatTourAllowances;

privatefloatTelephone\_Allowances;

publicoverridevoidCalculateSalary() //override method of Employee class

{

WriteLine("Enter Distance travelled during Marketing Executive in km");

Distance = int.Parse(ReadLine());//taking distance input

TourAllowances = 5 \* Distance; //5 rupees per km

Telephone\_Allowances = 1000;

GrossSalary=Salary + TourAllowances+Telephone\_Allowances;

}

publicvoidDisplayEmployeeDetails() //implementation of interface method

{

WriteLine("Marketing Executive Details....");

WriteLine("emp\_id: {0}\n emp\_name: {1}\nGross Salary:{2}", emp\_id, emp\_name,GrossSalary);

}

}

internalclassProgram

{

staticvoid Main(string[] args)

{

Manager mg = newManager();

MarketingExecutive me = newMarketingExecutive();

WriteLine("Manager Input");

mg.GetSalaryWithEmployeeDetails();

mg.CalculateSalary();

mg.DisplayEmployeeDetails();

WriteLine("Marketing Executive Input");

me.GetSalaryWithEmployeeDetails();

me.CalculateSalary();

me.DisplayEmployeeDetails();

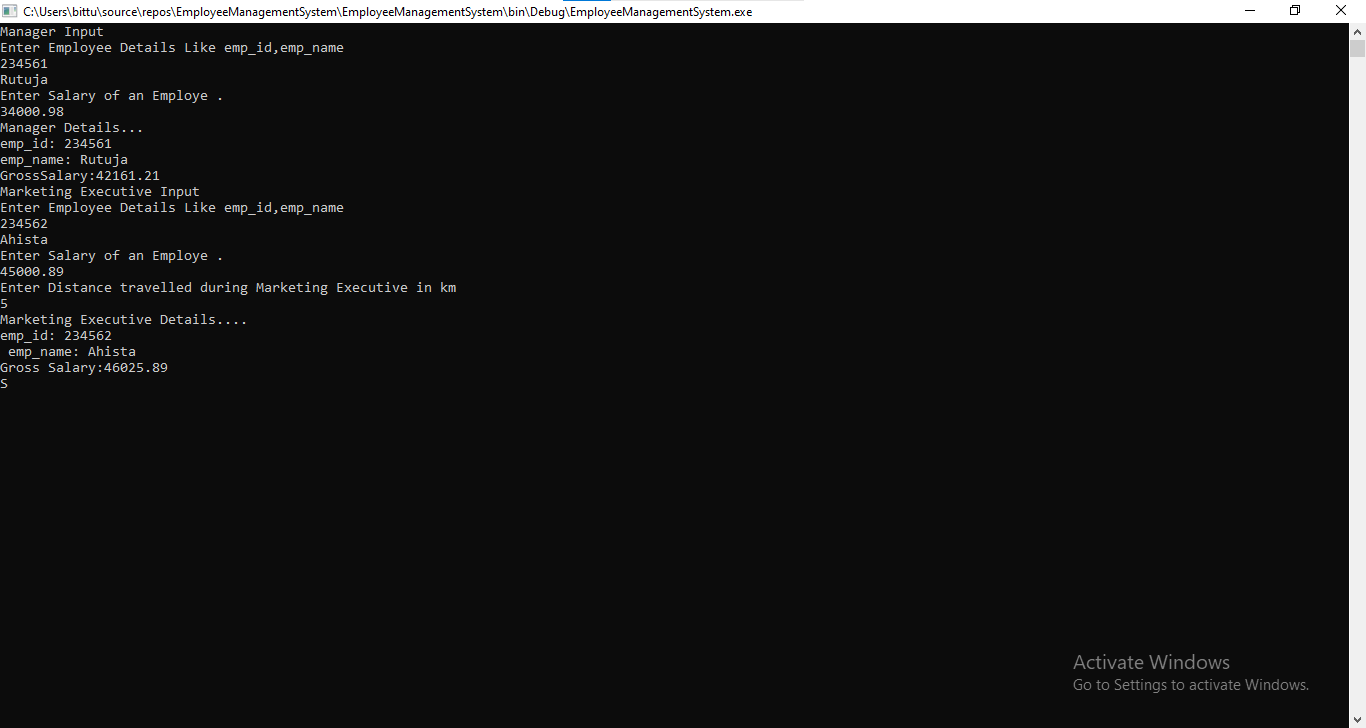
ReadLine();

}

}

}

Output



2.Write a class called MyStackwith following members.

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

Implement Push()and Pop()operation. Implement ICloneableinterface to perform cloning. Write a client application to perform cloning.

Program

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Threading.Tasks;

usingstaticSystem.Console;

usingstaticSystem.Convert;

namespaceMyStackPushPop

{

publicclassMyStack

{

intsize\_of\_stack;

inttop\_position = 0;

int[] array = newint[20];

publicvoidGetSizeOfStack()//for getting size of stack

{

WriteLine("Enter Size of Stack");

size\_of\_stack = int.Parse(ReadLine());

}

publicvoid Push()

{

if(top\_position==(size\_of\_stack-1))

{

WriteLine("Stack is Full");

}

else

{

for(int i=0;i<size\_of\_stack;i++)

{

WriteLine("Insert element in stack");

int element = int.Parse(ReadLine());

array[++top\_position] = element;

WriteLine("Item pushed Successfully!");

}

}

}

publicobject Pop()

{

if(top\_position== -1)

{

WriteLine("Stack is Empty");

return"No elements";

}

else

{

return array[top\_position--];

}

}

publicvoid Display()

{

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

{

WriteLine("Item {0}: |{1}| ", (i), array[i]);

}

ReadLine();

}

}

internalclassProgram

{

staticvoid Main(string[] args)

{

MyStack stack = newMyStack();

//stack.GetSizeOfStack();

while(true)

{

Clear();

WriteLine("Stack Menu Program");

WriteLine("1. Enter Size of Stack");

WriteLine("2. Push operation");

WriteLine("3. Pop Operation");

WriteLine("4. Display ");

WriteLine("5. Exit");

Write("Enter Your Choice");

int choice = ToInt32(ReadLine());

switch(choice)

{

case 1:

stack.GetSizeOfStack();

break;

case 2:

stack.Push();

break;

case 3:

WriteLine("Element removed: {0}", stack.Pop());

ReadLine();

break;

case 4:

stack.Display();

break;

case 5:

Environment.Exit(0);

break;

}

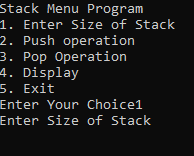
}

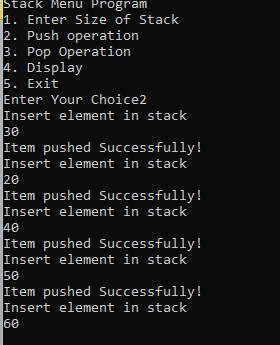
}

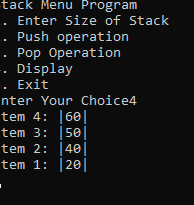
}

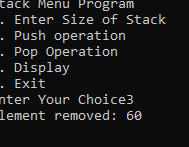
}

Output









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;

usingSystem.Collections;

usingSystem.Collections.Generic;

usingSystem.Data;

usingSystem.Diagnostics;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Threading.Tasks;

namespaceStackExceptionPro

{

interfaceIstack

{

Boolean IsEmpty();

void Push(object element);

object Pop();

object Peek();

void Display();

}

classPushPop : Istack

{

publicint top;

object[] item;

publicintstack\_size;

publicPushPop(intstack\_size) //parametrised constructor

{

this.stack\_size = stack\_size;

item = newobject[stack\_size];

top = -1;

}

publicPushPop() //default constructor

{

stack\_size = 10;

item = newobject[stack\_size];

top = -1;

}

publicboolIsEmpty() //Method to check is stack empty or full

{

if (top == (stack\_size - 1))

{

returntrue;

}

else

returnfalse;

}

publicvoid Push(object element)

{

if (top == (stack\_size - 1))

{

thrownew Exception("Exception Stack overflow because stack is full");

}

else

{

item[++top] = element;

Console.WriteLine("Item Pushed Successfully");

}

}

publicobject Pop()

{

if (IsEmpty())

{

thrownew Exception("Exception underflow because stack is empty");

returnnull;

}

else

{

return item[top--];

}

}

publicobject Peek()

{

if (IsEmpty())

{

thrownew Exception("Stack Underflow exception because stack is empty");

returnnull;

}

else

{

return item[top];

}

}

publicvoid Display()

{

for(int i = top; i>-1;i--)

{

Console.WriteLine(item[i]);

}

}

}

internalclassProgram

{

staticvoid Main(string[] args)

{

PushPopst = newPushPop();

object element;

while(true)

{

Console.Clear();

Console.WriteLine("1. IsEmpty()");

Console.WriteLine("2. Push()");

Console.WriteLine("3. Pop()");

Console.WriteLine("4. Peek()");

Console.WriteLine("5. Display()");

Console.WriteLine("6. Exit()");

Console.WriteLine("Enter Your Choice");

int choice;

choice = int.Parse(Console.ReadLine());

Console.WriteLine();

switch(choice)

{

case 1:

if(st.IsEmpty())

{

Console.WriteLine("Stack is Empty");

}

else

{

Console.WriteLine("Stack is not empty");

}

break;

case 2:

Console.WriteLine("Enter element");

element = Console.ReadLine();

try

{

st.Push(element);

}

catch(StackOverflowException e)

{

Console.WriteLine(e.Message);

}

break;

case 3:

try

{

element = st.Pop();

Console.WriteLine("Element removed is {0} ", element);

}

catch(Exception e)

{

Console.WriteLine(e.Message);

}

break;

case 4:

try

{

element = st.Peek();

Console.WriteLine("Element removed is {0} ", element);

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

break;

case 5:

st.Display();

break;

case 6:

Environment.Exit(0);

break;

default:

Console.WriteLine("Enter valid choice");

break;

}

Console.ReadLine();

}

}

}

OUTPUT

