# **Inheritance and Polymorphism**

## Objective

* To use inheritance (“is-a” relationship).
* To use polymorphism.
* To create an abstract class
* To use interface.

## Assignments to be done in this session

1. Create a hierarchy of Employee, Manager, MarketingExecutive in Employee Management System. They should have the following functionality.
   1. Manager with following private members.
      * Petrol Allowance: 8 % of Salary.
      * Food Allowance : 13 % of Salary.
      * Other Allowances : 3% of Salary.

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

* 1. MarketingExecutive with following private members.
     + Kilometer travel
     + Tour Allowances : Rs 5/- per Kilometer (Automatically generated).
     + Telephone Allowances : Rs.1000/-

Calculate GrossSalary by adding above allowances. Override CalculateSalary(). NetSalary,PF calculation should not consider above allowances.

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

**Employee.cs**

//Manager

public class Manager : Employee

{

double PetrolAllowance;

double FoodAllowance;

double OtherAllowance;

public double setPetrol(double \_Salary)

{

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

return this.PetrolAllowance;

}

public double setFood(double \_Salary)

{

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

return this.FoodAllowance;

}

public double setOther(double \_Salary)

{

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

return this.OtherAllowance;

}

public virtual double setAllowance(double PetrolAllowance, double FoodAllowance, double OtherAllowance)

{

this.\_GrossSalary = PetrolAllowance + FoodAllowance + OtherAllowance;

return this.\_GrossSalary;

}

public override void CalculateSalary(double GrossSalary)

{

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

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

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

}

public void showSalary()

{

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

public class MarketingExecutive : Manager

{

private double Kilometertravel;

private double TourAllowance = 5;

private double TelephoneAllowance = 1000;

public double setTravel(double Kilometertravel)

{

return this.Kilometertravel = Kilometertravel;

}

public void getTravel()

{

Console.WriteLine("You traveled for : " + Kilometertravel + " kms");

}

public double getTour(double TourAllowance)

{

this.TourAllowance = 5 \* this.Kilometertravel;

return this.TourAllowance;

}

public double setTour(double TourAllowance)

{

this.TourAllowance = 5 \* this.Kilometertravel;

return this.TourAllowance;

}

public void getTour()

{

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

}

public double setTelephone()

{

return this.TelephoneAllowance = 1000;

}

public override double setAllowance(double Kilometertravel, double TourAllowance, double TelephoneAllowance)

{

this.\_GrossSalary = TourAllowance + TelephoneAllowance;

return this.\_GrossSalary;

}

public override void CalculateSalary(double GrossSalary)

{

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

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

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

}

public void showSalary()

{

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

{

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

Employee.Manager obj1 = new Employee.Manager();

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);

double result\_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----------");

Employee.MarketingExecutive obj2 = new Employee.MarketingExecutive();

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);

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

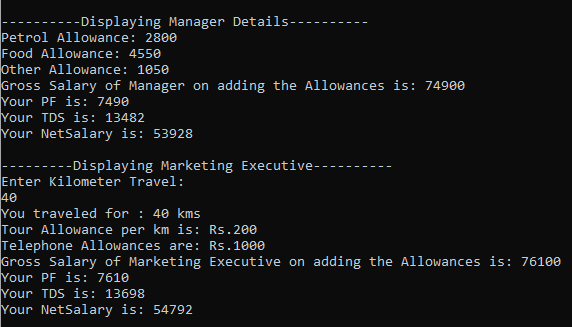
Console.WriteLine("Gross Salary of Marketing Executive on adding the Allowances is: {0}", result\_gross\_final);

obj2.CalculateSalary(result\_gross\_final);

obj2.showSalary();

Console.ReadLine();

}



1. Write a class called MyStack with following members.
   1. integer array
   2. integer variable to store top position
   3. size of the array.

Implement Push() and Pop() operation. Implement ICloneable interface 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 StackException when the stack is full or empty respectively.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class MyStack

{

private int[] StackArr;

private int top;

private int max;

public MyStack(int size)

{

StackArr = new int[size];

top = -1;

max = size;

}

public void push(int item)

{

if (top == max - 1)

{

throw new StackException("Stack OverFlow!!!");

}

else

{

StackArr[++top] = item;

}

}

public int pop()

{

if (top == -1)

{

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

}

else

{

Console.WriteLine("Popped Element : " + StackArr[top]);

return StackArr[top--];

}

}

public void display()

{

if (top == -1)

{

throw new StackException("Stack is Empty");

}

else

{

Console.WriteLine("Inserted items are: ");

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

{

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

}

}

}

}

class Program1

{

static void Main()

{

MyStack stack = new MyStack(5);

try

{

stack.push(155);

stack.push(228);

stack.push(980);

stack.push(682);

stack.push(64);

stack.push(5);

stack.push(19);

stack.push(35);

}

catch (StackException se)

{

Console.WriteLine(se);

}

try

{

//stack.pop();

}

catch (StackException se)

{

Console.WriteLine(se);

}

try

{

stack.display();

}

catch (StackException se)

{

Console.WriteLine(se);

}

Console.ReadLine();

}

}

public class StackException : Exception

{

public StackException(string s) : base(s)

{

}

}

}

