**Interface and Inheritance**  
**Workshop 5**

**In this workshop, you’ll learn:**

* Inheritance
* Method overriding
* Super keyword
* Abstract class
* Polymorphism

**1. We have the design for Phone number:**

|  |
| --- |
| **PhoneNumber** |
| # int area  # String number |
| + PhoneNumber()  + PhoneNumber(int a, String n);  + **void display();** |

|  |
| --- |
| **IntPhoneNumber** |
| - String countryCode |
| + IntPhoneNumber ()  + IntPhoneNumber (String cc, int a, String n);  + **void display();** |

**Implement the above classes and make your own main program to test all the above methods.**

You can using an array of base class PhoneNumber object

**PhoneNumber phonelist[] = new PhoneNumber[1000];**

**The output of your program something look like:**

|  |
| --- |
| Enter list of phone numbers  -----------------------------------------  Type of phone number ? (1 – local phone, 2 – Inter phone number, 0 - exit): 1  Enter area code: 111  Enter number: 111111  Type of phone number ? (1 – local phone, 2 – Inter phone number, 0 - exit): 2  Enter country code: 22  Enter area code: 222  Enter number: 222222  Type of phone number ? (1 – local phone, 2 – Inter phone number, 0 - exit): 1  Enter area code: 333  Enter number: 333333  Type of phone number ? (1 – local phone, 2 – Inter phone number, 0 - exit): 2  Enter country code: 44  Enter area code: 444  Enter number: 444444  Type of phone number ? (1 – local phone, 2 – Inter phone number, 0 - exit): 0  List of phone number:  ----------------------------------------  111 – 111111  22 – 222 – 222222  333 – 333333  44 - 444 – 444444 |

**Additional proposed exercises:**

**2. We have the design for employee management:**

<<interface>>  
IEmployee

+salary() : double

Employee *{abstract}*

#employeeID:String  
#fullName:String  
#salaryCoefficient: double  
#workStartDate: Date  
#gender: boolean  
+basicSalary: double =1490000 {static}  
+yearSeniorityAllowance: double=100000 {static}

+Employee(parameter)  
+getEmployeeID() : String  
+setEmployedID(id : String) : void  
…  
+toString() : String  
*+reward(): double {abstract}*

Manager

-position: String  
-positionAllowance: double  
+yearReward:double=100000 {static}

+Manager(parameter)  
+getPosition() : String  
+setPosition (position : String) : void  
+getPositionAllowance () : double  
+set PositionAllowance (pa : double) : void  
+toString() : String

Technician

-work: String  
+yearReward:double=50000 {static}

+Technician(parameter)  
+getWord() : String  
+setWord(work : String) : void  
+ toString() : String

**a.** Declare angeneral interface IEmployee for different types of Employee that consists of a method to compute employee salary.

**b.** The system is to keep track of the information for each employee as follows:

employee index

fullName  
salary coefficient  
working start date  
gender

class constants: basic salary = 1490000, seniority allowance per year = 100000

empoyee index is unique.

**c.** There are two types of employees, technician and manager.

**Technician** is added the attribute work.

**Manager** is added the attributes such as position, position allowance.

The formula to calcualte seniority allowance:

**Seniority allowance = seniority allowance per year \* number of years**

The formula to calcualte salary and reward is different for technician and manager:

Salary is calculated for technician as follows:

**Salary = salary** [**coefficient**](http://en.wikipedia.org/wiki/Coefficient) **\* basic salary + seniority allowance**

Salary is calculated for manager as follows:

**Salary = salary** [**coefficient**](http://en.wikipedia.org/wiki/Coefficient) **\* basic salary + seniority allowance + position allowance;**

Reward is calculated for technician as follows:

**Reward = technician reward per year \* Number of years**

Reward is calculated for manager as follows:

**Reward = manager reward per year \* number of years**

In particular, the **number of years** is calculated from the working start date to the present.

**d.** Implement the above classes and make your own main program to test all the above methods.