**Collections**

**Workshop 7**

**In this workshop, you’ll learn:**

* Collections framework
* ArrayList collection
* Set collection

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

<<interface>>  
IEmployee

+salary() : double

EmployeeList

-list : ArrayList<Employee>

+getList() : ArrayList<Employee>  
+getEmployee(i : int) : Employee  
+addEmployees(): void  
+managerList() : ArrayList<Employee>  
+technicalSalaryAverage() :double  
+searchID(id : String) : Employee  
+searchName(name :String):ArrayList<Employee>  
+searchGender(gender:boolean):ArrayList<Employee>  
+searchDate(date :Date) : ArrayList<Employee>  
+removeID(id : String) : void  
+removeDate(date : Date) : void  
+lastSalaryMax():Employee  
+updateEmployee():void

*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  
+setPositionAllowance (pa : double) : void  
+toString() : String

Technician

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

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

**a.** Declare ageneral 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

and 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.** Create a class named EmployeeList. This class manages a ArrayList of Employee objects, consisting of methods:

- Add new employees to the ArrayList

- List all managers in the ArrayList

- Compute the average salary of all technicians

- Search employee:

+ by employee index  
+ by fullname  
+ by gender  
+ by working start date

- Remove employee having specified employee index

- Remove employees with working start date after the specified date.

- Search the last occurrence of the employee having maximum salary

- Update employee information

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

**Additional proposed exercises:**

**2. Create a class named DictionaryWord as:**

|  |
| --- |
| DictionaryWord |
| - word: String                                                              - meanings: String |
| + DictionaryWord (String word, String meanings) + getWord(): String + setWord (String word): void + getMeanings(): String + setMeanings(String meanings): void |

Write a program with the following requirements:  
Creates 8 DictionaryWord objects with:

* Word and meanings as the table:

|  |  |
| --- | --- |
| **Word** | **Meanings** |
| Bank robber | Steals money from a bank |
| Burglar | Breaks into a home to steal things |
| Forger | Makes an illegal copy of something |
| Hacker | Breaks into a computer system |
| Hijacker | Takes control of an airplane |
| Kidnapper | Holds someone for ransom money |
| Mugger | Attacks and steals money from someone |
| Murderer | Kills another person |

* Ensure that there is no duplicate DictionaryWord objects (02 DictionaryWord objects a and b are equal when a.word=b.word).

Displays all DictionaryWord in ascending order of word with the format as:

<<no>.<<word>>  
<<meanings>>

<<no>.<<word>>  
<<meanings>>

Where: <<no>>=1,2…

Hint:

* class DictionaryWord implements Comparable to order 2 DictionaryWord objects.
* override equals(..) method to compare 2 DictionaryWord objects.
* override toString()
* use Set to ensure no duplicate.
* use support class Collections to sort DictionaryWord objects.