Java SE 8 Programming Language

Training Assignments

Document Code	25e-BM/HR/HDCV/FSOFT
Version	1.1
Effective Date	20/11/2012

Hanoi, 04/2019

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	08/Oct/2018	Create new	Add the new assignment	DieuNT1	VinhNV
2	01/Mar/2019	Update	Apply fsoft template	DieuNT1	VinhNV

Contents

CODE:	JPL.L.A202
TYPE:	LONG
LOC:	N/A
DURATION:	180 MINUTES

Long Assignment 2 – Option 2: Java I/O Fudamentals

Objectives:

After finishing the following exercises, trainees will:

- Understand and practise with Advanced OOP in java.
- Understand and apply with Collection classes in java: List, Set, Map.
- Understand and practise with Exception handing, JavaIO.

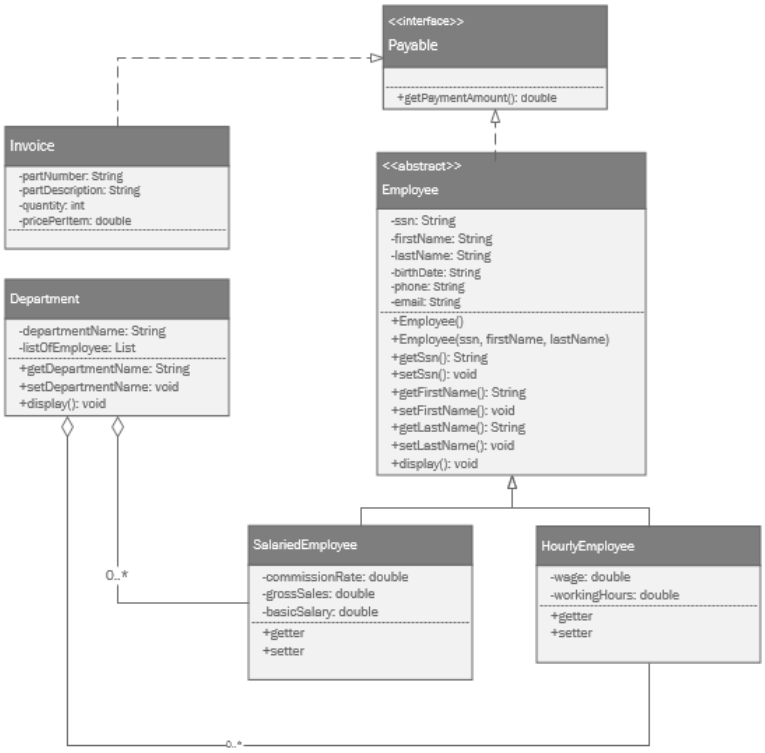
Working Environments:

- JDK 1.8
- Eclipse

Assignment Descriptions:

Create a Java Console application bases on Java Classes/Objects, OOP, Exception Handing, IO, Java Collection to manage Human Resource (HR Management System.

Refer to the class hierarchy which is decribed below, trainees need to create Java classes in order to implement the entities and the relationship between them.



The **Employee** is an abstract superclass and has six fields: *ssn*, *firstName*, *lastName, birthDate, phone, email*;

A **Salaried Employee** is paid annually. Salaried employees are usually supervisory, managerial, or professional employees who work on an annual basis and are not paid an hourly rate. **SalariedEmployee** is

a concrete class that is a subclass of Employee and adds 3 fields: **commisstionRate**, **grossSales**, **basicSalary**;

The **HourlyEmployee**: Unlike a salaried employee who is paid a flat salary regardless of how many hours worked during a work month, an hourly employee is paid an hourly wage for each hour worked. This is a concrete class that is a subclass of Employee and adds two fields: *rate*, *workingHours*;

The **Department**: Each department will have a list of employee (contains: salaried and hourly);

Noting that: each abstract class and concrete class has a constructor and methods for getting and setting its fields (getters and setters) and a toString method.

Validation Rules:

Program requirements must validate the properties:

- » BirthDate : correct date format (dd/MM/yyyy);
- » Phone: minimum 7 positive integers;
- » Email: correct email format.

Functional Requirements:

1. The program has the function to create an employee list of all types as mentioned above and belong to several departments.
2. Display employees: displays information about each object polymorphically.
3. Classify employees: the last for loop illustrates how to find out the specific class for each object.
4. Employee Search:
   a. Search employee by department name, return a list of all employees
   b. Search employee by employee name, return a list of all employee that same name
5. Report: display the list of departments and the number of employees for each.

User Interface Requirements:

Create a new class named **EmployeeManagement** that contains a main() method to display user interface.

The main screen allows user to select the following functions:

Menu

```
====== EMPLOYEE MANAGEMENT SYSTEM ======
1. Add an employee
2. Display employees
3. Classify employees
4. Search book by (department, emp's name)

Please choose function you'd like to do:
```

Storage Data:

a. The user inputs data from the keyboard.

b. Data is stored in a text file.

c. Output data is displayed on the console.

Guildelines:

- Create a project named JPL.L.A202, create package *fa.training.entities* that contains classes/interfaces: Payable, Employee, HourlyEmployee, SalariedEmplpyee, Department.
- Create other package ***fa.training.services*** that contains classes to implement Functional Requirements. The package ***fa.training.utils*** to implement Validation Rules requirements.

  Note, the functional requirements related to any entity, you have to create the service class corresponding to that entity. Ex: EmployeeService, …

- Create a package named ***fa.training.main*** contains **EmployeeManagement** class.

-- THE END --