FRESHER ACADEMY

JAVA BASICS

Contents

Objective

- Understand and practise with Classes, Objects, Inheritance, Encapsulation.
- Understand and practise with Polymorphism, Abstraction.
- Understanding the main difference between method overloading and overriding, between abstract class and interface in java

Business needs

Create a Java Console application bases on Java Classes/Objects, OOP, Exception Handing, IO, Java Collection to manage Human Resource (HR Management System). This assignment will cover all part of Java Basic. Class Diagram as below here:

Working requirements

- **Working environment:** Eclipse IDE, Java 7 or up.
- **Delivery:** Source code packaged in a compress archive.

Product architecture

- N/A

Technology

The product implements one or more technology:

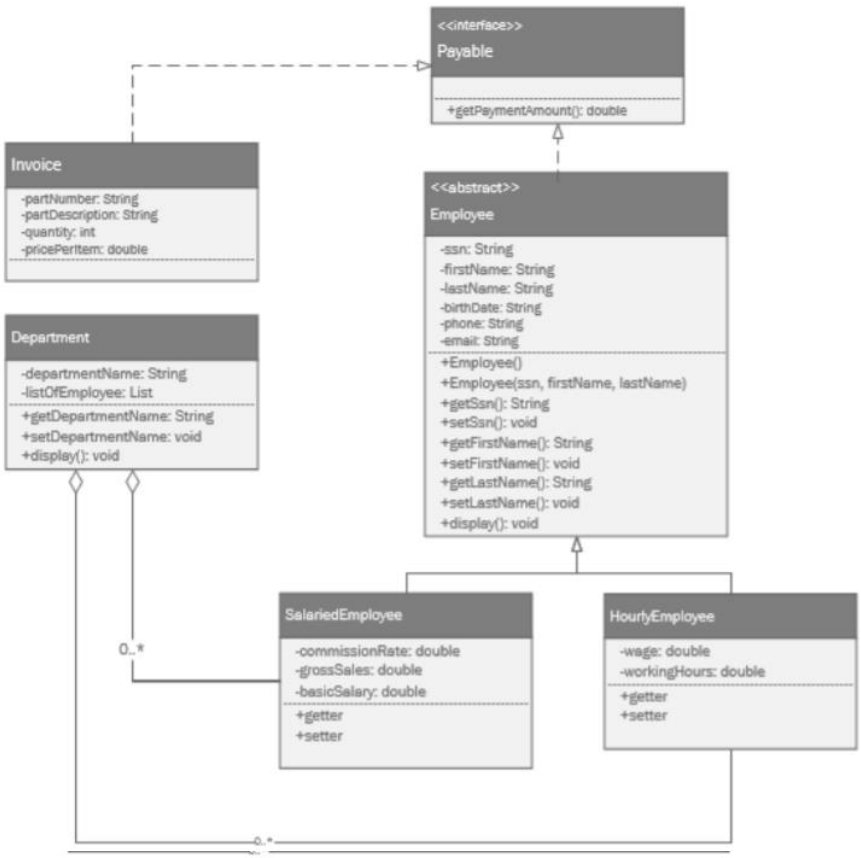
- Control of Flows
- OOP
- Exception Handing
- Java Collection

Stored Data

- Data stored in collection classes.

Project Descriptions

For the class hierarchy is as follows, the trainee let's create the java classes install this class diagram to be able to relationship between it.



- **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**;
- **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*;
- **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.

Program requirements must validate the properties:

- BirthDate : correct date format (dd/MM/yyyy);

- Phone: minimum 7 positive integers;
- Email: correct email format.

Functional/User Interface Requirement

The program has a screen console for UI

The DepartmentManage main method:

The main method in DepartmentManage class uses the Employee class and its subclasses, the main screen allows selecting the functions for:

1. Input data from the keyboard: create an employee list of all types as mentioned above and belong to several departments.

2. Display employees: displays information about each object polymorphically. The objects are stored in an Employee array.

3. Classify employees: the last for loop illustrates how to find out the specific class for each object.

4. Employee Search:

- Search for employees by enter the name of a specific department, display the list of employees by type to the screen;
- Enter the employee's name, display detailed information about the employee;

5. Report: display the list of departments and the number of employees for each.

--o--

The End!