CST8132 Object-Oriented Programming

Lab 3: Store Management System I

Due Date: Week 4 – in own lab hours

Marks: 10 marks (worth 4% of term mark)

Demo: Demo your code and output to your lab professor during your own lab hours.

Recommended Reading: Chapter 9 of Deitel and Deitel, Java How to Program book

Exercise

In the next few labs, we will create a few classes to represent a Store Management system. A store has employees and customers in their system. In this lab, we are creating a few classes namely Person, Employee, Regular, Contractor, Store. All personal attributes like first name, last name, email, phone number should be in Person class. Employee should have an employee number attribute and an Employee object named emp. As we know that the store has regular and contractor employees, we need to create two classes – Regular and Contractor, both extends Employee class. Regular class should have a salary attribute that represents the monthly salary. Contractor class should have attributes to represent hourly rate and number of hours worked. Store class will manipulate the store employees. More classes like Customer, different types of customer classes etc. will be added later.

As the first step, you need to create the following classes:

`Person` class

Instance variables: `firstName`(String), `lastName`(String), `email`(String), `phoneNumber` (long).

Constructor: as required

Methods:

- getters that return name, email and phone number. Name should be returned as one string. If first name is "John" and last name is "Doe", getter should return "John Doe".
- `readPersonalDetails`() : Accepts nothing, returns nothing. Reads first name, last name, email and phone number and stores in corresponding variables.

Employee class (extends Person)

Instance variables: `employeeNumber` (int), `emp` (Person)

Constructor: parameterized constructor that initializes an employee with employee number and all personal properties. As you have a parameterized constructor for Person class, use that to set personal properties (you need to create emp as a Person object from this constructor).

Methods:

1. `readEmployee()`: accepts nothing, returns nothing. Reads all employee information.
2. `printEmployee()`: accepts nothing, returns nothing. This method prints details of an employee using formatted output (use printf).

Regular class (extends Employee)

Instance variables: `salary` (double)

Methods:

1. `readEmployee`(): accepts nothing, returns nothing. Make a call to the readEmployee() method of the parent class. Then, reads annual salary from the user, converts it to monthly salary and store it in salary instance variable.
2. `printEmployee`(): accepts nothing, returns nothing. Make a call to the printEmployee() of the parent class. Then, prints salary info (formatted output).

Contractor class (extends Employee)

Instance variables: `hourlyRate` (double), `numHours` (double)

Methods:

1. `readEmployee`(): accepts nothing, returns nothing. Make a call to the readEmployee() method of the parent class. Then, reads hourly rate and number of hours worked.
2. `printEmployee`(): accepts nothing, returns nothing. Make a call to the printEmployee() of the parent class. Then, prints salary, which is the product of hourly rate and the number of hours worked (formatted output).

Store class

Instance variables: an array of Employee named `employees`

Constructor: parameterized constructor that creates the array of employees with the given size (this size will be read in main(), and will be sent here when creating the Store object)

Methods:

1. `readEmployeeDetails`(): accepts nothing, returns nothing. In a for loop, read details of all employees. First, read the type of the employee. Based on the type of the employee, corresponding array object needs to be created (Polymorphism). Then, call readEmployee() method.

2. `printEmployeeDetails`(): accepts nothing, returns nothing. In a for loop, call printEmployee() to print details of all employees.
3. `printLine`(): *static* method that prints a line using "="
4. `printTitle`(): *static* method that prints the title of the output. This method gets the name of the store as a parameter, which will be used in the formatted print statement. printLine() method will be called from this method to print lines.

Lab3 class

This is the driver class (test class), which means this class has the main method.

Main method

- This method read the name of the store (example: "Quality") and the number of employees (stored in num).
- A Store object will be created with the 'num'.
- Call readEmployeeDetails() method to read details of all employees
- Print the title and the header row
- Call printEmployeeDetails() method to print details of all employees.

Format your code with proper indentation and formatting. Your code should be properly commented. Test plan and external documentation are not required for this exercise, but in future labs they will be required.

Grading Scheme

Item	Marks
Person class (correct access specifiers, constructors, 3 method)	1
Employee class (correct access specifiers, constructors, 2 methods)	1
Regular class (correct access specifiers, 2 methods)	1
Contractor class (correct access specifiers, 2 methods)	1
Store class (correct access specifiers, constructors, 4 methods)	2
Lab3 class (main method)	2
Comments (class header, provide comments wherever required)	2

Submission

Zip your code in a folder named <LastName><FirstName>Lab3.zip and submit it to Brightspace before the due date. Demonstrate your work to your lab professor during your own lab hours. Both submission and demo are required to get grades.

Getting ready for next lab

Once you are done with this lab, think about abstraction.

Expected Output (blue – user input)

```
Enter name of the store: Quality
How many employees do you have?3
Enter details of employee 1
1.  Regular
2.  Contractor
Enter type of employee: 1
Enter Employee Number: 110120
Enter first name: John
Enter last name: Doe
Enter email: doe@test.com
Enter phone number: 123456789
Enter annual salary: 98000
Enter details of employee 2
1.  Regular
2.  Contractor
Enter type of employee: 2
Enter Employee Number: 110125
Enter first name: Matt
Enter last name: James
Enter email: matt@test.com
Enter phone number: 456123789
Enter hourly rate: 25
Enter number of hours worked: 40
Enter details of employee 3
1.  Regular
2.  Contractor
Enter type of employee: 1
Enter Employee Number: 110128
Enter first name: Tim
Enter last name: Thomas
Enter email: tim@test.com
Enter phone number: 789456123
Enter annual salary: 95700
================================================================================
                Quality Store Management System
================================================================================
      Emp# |                 Name |            Email |        Phone |     Salary|
================================================================================
    110120 |             John Doe |    doe@test.com |    123456789 |   8166.67 |
    110125 |           Matt James |   matt@test.com |    456123789 |   1000.00 |
    110128 |           Tim Thomas |    tim@test.com |    789456123 |   7975.00 |
```