Lab 9

Requirements:

- Create a Java project named yourStudentId_OOP_Lab9
- Read instructions and create classes needed. You are supposed to add 4 classes (*Employee, Supervisor, BankAccount and Tester*) to the project.
- All instance variables are private. Please use public interfaces to access private variables.

Following figure shows the inheritance, attributes and methods for each class. For setter and getter please refer to Figure 1.

Description: The Lab mainly asks you to simulate the company's salary payment process . A company consists of staffs and supervisors. When employees start working, they will be asked to set the wages and salaries account. In this case, the salary includes the basic salary, and sales bonus. However, the amount of earnings that employee has left is after tax. As supervisors, their performance will include the sales of their subordinate and their self-sales. The question can be completed by using the "Inheritance" concept taught in this week.

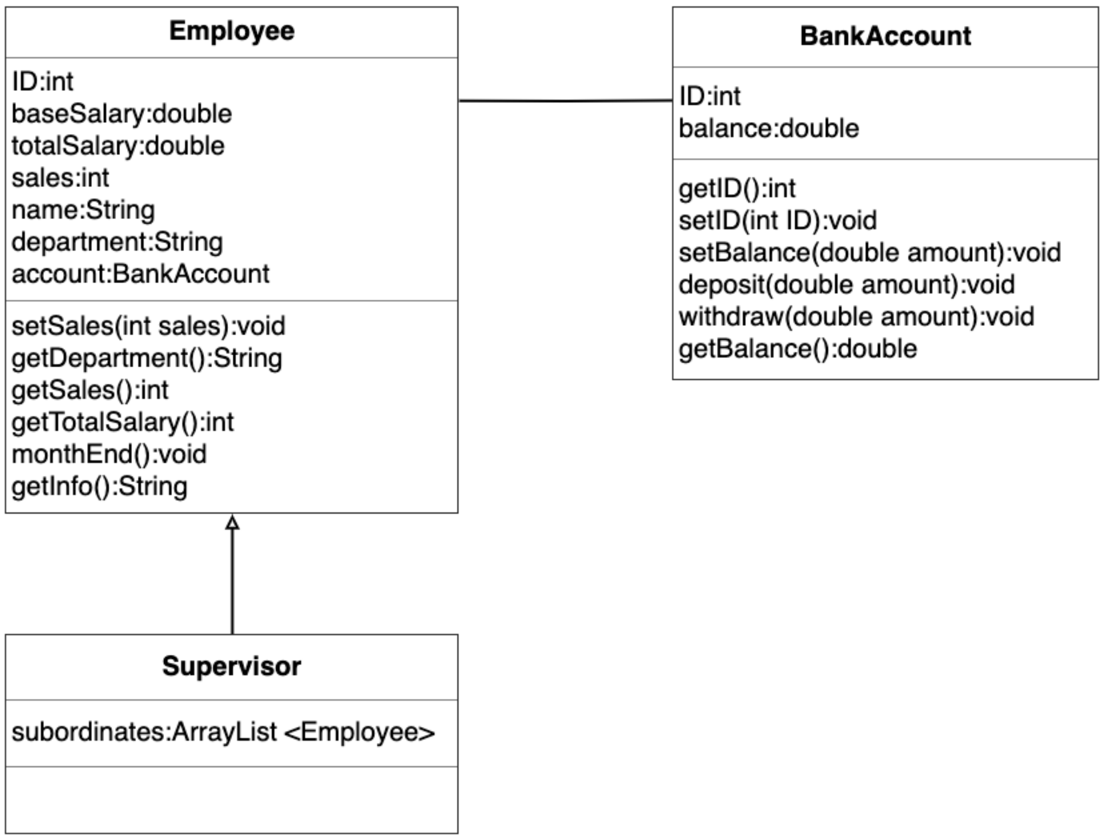

Figure 1.

1. **Create *Employee* class**

Employee	
Modifier and type	**Method (or Variable) and description**
Instance variable	
int	ID The employee's ID.
double	baseSalary The employee's base salary.
double	totalSalary The employee's total salary.
int	sales The employee's sales.
String	name The employee's name.
String	department The employee's department.
BankAccount	account The employee's pre-set account, the monthly salary will be transferred here.
Constructor	
Employee(int ID, String name, BankAccount account, String department, int baseSalary, int sales) Enable to instantiate an *Employee* object with given *ID, name, account, department, baseSalary*, and *sales*. The *totalSalary* should be set as 0.	
Instance methods	
-	3 getter for 3 attributes (getDepartment(), getSales(), and getTotalSalary()). 1 setter for 1 attribute (setSales(...)).
void	monthEnd() Calculate the employee's monthly salary and transfer it to the employee's account. ***Salary calculation formula:*** ***(Base salary + (Sales * Sales bonus))*(1 - tax rate)*** a. Set tax rate as 0.03 named *taxRate* and sales bonus as 500 named *salesBonus*. b. Use *salary calculation formula* to calculate employee's monthly total salary. c. Transfer monthly salary to the employee's account.
String	getInfo() a. Return a String object contains employee's ID, name, department, total sales, and total salary as example. **Example:** ID:109306201 Name:Alex Department:Sales Dept. Total sales:50 Total salary:58200

2. **Create *Supervisor* class**

Supervisor	
Modifier and type	**Method (or Variable) and description**
Instance variable	
ArrayList <Employee>	subordinates All subordinates of the supervisor.
Constructor	
Supervisor(int ID, String name, BankAccount account, String department, int baseSalary, ArrayList subordinates, int sales) Use *super(...)* keyword to instantiate the object of superclass by *ID, name, account, department, baseSalary, sales*. The subordinates with given *subordinates*. In addition to its own sales, Supervisor's sales also include subordinate's sales. ***<u>Hint:</u>*** Use *"for-each"* concept to read all subordinate's sales and then use *setSales()* to set the exactly sales of the supervisor.	

3. **Create *BankAccount* class**

BankAccount	
Modifier and type	**Method (or Variable) and description**
Instance variable	
int	ID The account's ID.
double	balance The account's existing balance.
Constructor	
BankAccount(int ID, double balance) Enable to instantiate a *BankAccount* object with given *ID, balance*.	
Instance methods	
-	1 getter for 1 attribute (getID ()). 2 setters for 2 attributes (setID(...), setBalance(...)).
void	deposit(double amount) a. Add amount to balance and update.
void	withdraw(double amount) a. Determine whether the user's balance is greater than the amount will be withdrawn. If it is "true", you should subtract the amount from the balance and update it. Instead, print out *"Your account does not have enough money."*.
double	getBalance() Return the balance of the user's account.

4. **The *main* method in *Tester* class**

   a. Use *ArrayList* named employees to store employee objects (i.e., Alex, Lily, and Tony).

   b. Use *"for-each"* concept to determine which are Bob's subordinates and record those people in *SalesSubOrdinates* ArrayList.

   c. Use "for-loop" statement and call responding method (i.e., monthEnd()) to calculate the current monthly salary of each employee and print as sample output.

Reference code

```
import java.util.ArrayList;

public class Tester {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

	BankAccount account1 = new BankAccount(102603901, 1000);
	BankAccount account2 = new BankAccount(102603902, 1500);
	BankAccount account3 = new BankAccount(102603903, 1400);
	BankAccount account4 = new BankAccount(102603904, 1800);

	Employee Alex = new Employee(109306201, "Alex",
	                           account1, "Sales Dept.", 35000, 50);
	Employee Lily = new Employee(109306203, "Lily",
	                           account3, "Sales Dept.", 27500, 8);
	Employee Tony = new Employee(109306204, "Tony",
	                           account4, "Marketing Dept.", 40000, 30);

	// a. Use ArrayList named employees to store employee object (i.e., Alex, Lily, and Tony).

	ArrayList <Employee> SalesSubOrdinates = new ArrayList <Employee>();


	// b. Use "for-each" concept to determine which are Bob's subordinates and record those people
	in SalesSubOrdinates ArrayList.

	// c. Use "for-loop" statement and call responding method (i.e., monthEnd()) to calculate the
	current monthly salary of each employee and print the following results.


	}
}
```

Sample output

```
ID:109306201
Name:Alex
Department:Sales Dept.
Total sales:50
Total salary:58200

ID:109306203
Name:Lily
Department:Sales Dept.
Total sales:8
Total salary:30555

ID:109306204
Name:Tony
Department:Marketing Dept.
Total sales:30
Total salary:53350

ID:110306202
Name:Bob
Department:Sales Dept.
Total sales:79
Total salary:86815
```

Submission: Submit your project as **".zip file"** via Moodle. No other submissions will be graded.

Reminder: Please zip the whole project.

Deadline: Tomorrow's midnight (for both Mon56 and Tue23)