Payroll Exercise

Case: Partson-Derpy-Shpecktur law firm would like you to create their payroll system. According to them, they have several kinds of employees. Payment schemes may vary depending on the position of the employee.

An **Accountant** always receives a **fixed** salary. A **Janitor's** salary on the other hand, may vary depending on the janitor's hourly rate and the number of hours he/she has worked. The computation for a janitor's salary is as follows: *Hourly Rate* * *Hours Worked***.** A **Lawyer** has a consultation fee and can have several clients at once. A lawyer also logs the number of hours he/she has provided consultation. A lawyer's salary is computed as: *Consultation fee* * *Hours Worked*.

All payments are subject to tax. The tax computation for each employee is as follows:

3% for the first ₱3200 of the salary
6.5% for the next ₱8800 of the salary
13.5% for the remaining value

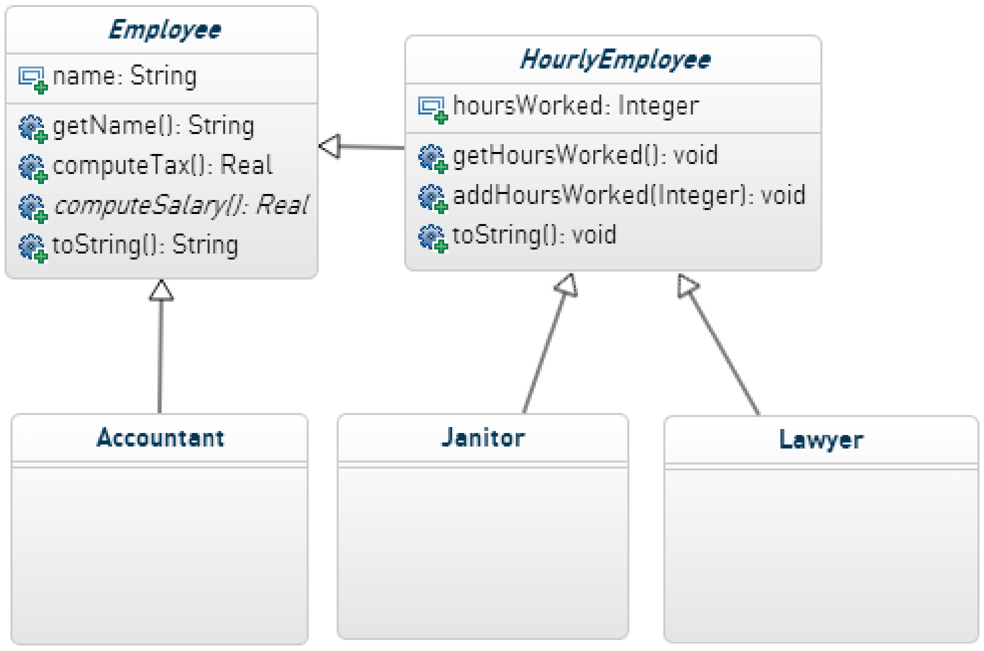
Phase 1

The driver class "PayrollMainPhase1.java" will be given. It contains the following code:

```
import java.util.List;
import java.util.ArrayList;


public class PayrollMainPhase1 {
        public static void main(String[] args){
                List<Employee> employees = new ArrayList<Employee>();

                HourlyEmployee louis = new Janitor("Louis Litt", 70); // hourlyRate = 70
                HourlyEmployee harvey = new Lawyer("Harvey Specter", 750);
                HourlyEmployee mike = new Lawyer("Mike Ross", 400); // consultationFee = 400
                louis.addHoursWorked(217.5);
                harvey.addHoursWorked(100);
                mike.addHoursWorked(150);

                employees.add(new Accountant("Rachel Zane", 19500));
                employees.add(new Accountant("Donna Paulsen", 8000));
                employees.add(louis);
                employees.add(harvey);
                employees.add(mike);

                /* the for-each loop - iterates over a list and assigns an element
                 * to e for every iteration. You can read it as:
                 * "for each employee e in the list employees" */
                for(Employee e: employees){
                        System.out.println(e.toString());
                        System.out.println("=====================================");
                }

        }
}
```

Tasks:

1. Complete the class diagram of the Employee hierarchy. Include the classes and subclasses for the *Employee* **abstract class**. The partial class diagram is shown below. Note that *Employee* and *HourlyEmployee* are both **abstract classes**. In this case, *HourlyEmployee* is an abstract class because it extends the *Employee* abstract class but does not provide an implementation for the abstract method of Employee (*computeSalary*). Therefore, the subclasses of *HourlyEmployee* are responsible for providing an implementation for *computeSalary*.



2. Create the classes for the *Employee* hierarchy. Make sure your classes are compatible with PayrollMainPhase1.java.

3. Test your program. The expected output is shown below:

```
Name: Rachel Zane
Salary: 19500.0
Tax: 1680.5
=====================================
Name: Donna Paulsen
Salary: 8000.0
Tax: 408.0
=====================================
Name: Louis Litt
Hours worked: 217.5
Hourly rate: 70.0
Salary: 15225.0
Tax: 1103.375
=====================================
Name: Harvey Specter
Hours worked: 100.0
Consultation Fee: 750.0
Salary: 75000.0
Tax: 9173.0
=====================================
Name: Mike Ross
Hours worked: 150.0
Consultation Fee: 400.0
Salary: 60000.0
Tax: 7148.0
=====================================
```

Phase 2

The driver class “PayrollMainPhase2.java” will be given. It contains the following code:

```java
public class PayrollMainPhase2 {
        public static void main(String[] args){
                PayrollSystem ps = new PayrollSystem();

                ps.addEmployee(new Accountant("Rachel Zane", 19500));
                ps.addEmployee(new Accountant("Donna Paulsen", 8000));
                ps.addEmployee(new Janitor("Louis Litt", 70)); // hourlyRate = 70
                ps.addEmployee(new Lawyer("Harvey Specter", 750));
                ps.addEmployee(new Lawyer("Mike Ross", 400)); // consultationFee = 400

                // increments Louis’ hours by 217.5, and Harvey's by 100, etc.
                ps.logHours("Louis Litt", 217.5);
                ps.logHours("Harvey Specter", 100);
                ps.logHours("Mike Ross", 150);

                System.out.println("Employee names: ");
                ps.displayAllEmployees();
                System.out.println();

                // Fire Mike Ross
                ps.removeEmployee("Mike Ross");
                System.out.println("Mike Ross was fired\n");

                System.out.println("Employee Salaries: ");
                ps.showAllSalaries();
        }
}
```

Tasks:

1. Revise your class diagram to include the PayrollSystem class. The PayrollSystem class includes the following functionalities:
    a. Add an employee in the payroll.
    b. Remove an employee from the payroll.
    c. Update the hours worked by an *HourlyEmployee*
    d. Display the information of all employees.
    e. Compute the salary of all employees.

2. Create the classes for the *Employee* hierarchy. Make sure your classes are compatible with PayrollMainPhase1.java.

3. Test your program. The expected output is shown below:

```
Employee names:
1.) Rachel Zane
=====================================
2.) Donna Paulsen
=====================================
3.) Louis Litt
=====================================
4.) Harvey Specter
=====================================
5.) Mike Ross
=====================================

Mike Ross was fired

Employee Salaries:
1.)
Name: Rachel Zane
Salary: 19500.0
Tax: 1680.5
=====================================
2.)
Name: Donna Paulsen
Salary: 8000.0
Tax: 408.0
=====================================
3.)
Name: Louis Litt
Hours worked: 217.5
Hourly rate: 70.0
Salary: 15225.0
Tax: 1103.375
=====================================
4.)
Name: Harvey Specter
Hours worked: 100.0
Consultation Fee: 750.0
Salary: 75000.0
Tax: 9173.0
=====================================
```