Software Design Patterns

CAT 1: Design Principles and Analysis Patterns (evaluated on 50 points)

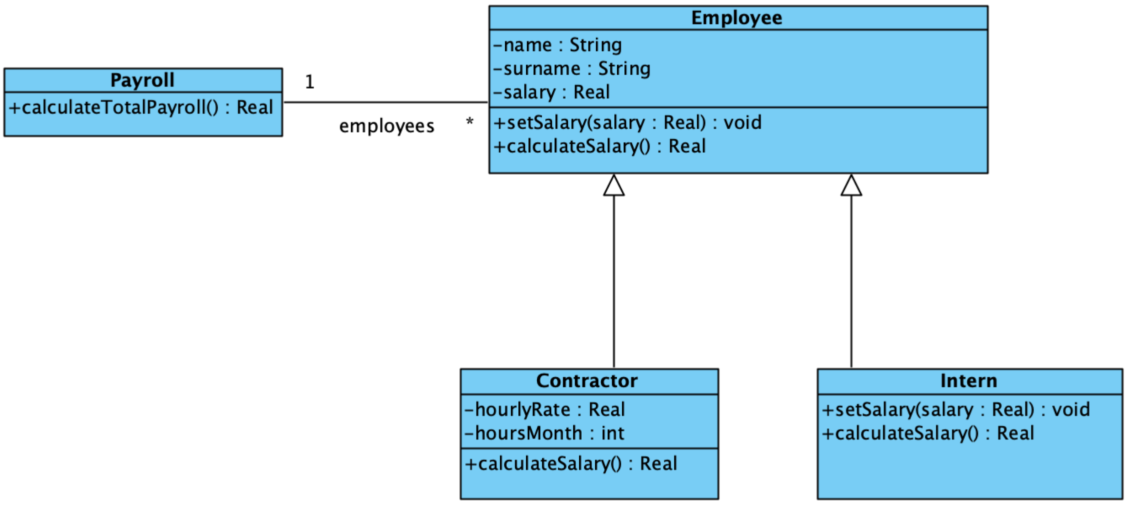
Question 1 (20 points)

Statement

We have developed a system for personnel management of a company. For each employee we will store their first name, last name and salary. In our company we have three types of employees: regular, interns and contractors. Regular workers receive their fixed salary at the end of the month, interns do not receive payment and contractor workers receive payment according to hours worked.

Our system also has a class that is responsible for calculating the total pay of all employees. To do this, it has a list of all employees and a *calculateTotalPayroll* function that goes through the list and adds up the salaries of all employees

Below is a possible model that represents the personnel management system described above:


Payroll
+calculateTotalPayroll() : Real
1
employees
*
Employee
-name : String
-surname : String
-salary : Real
+setSalary(salary : Real) : void
+calculateSalary() : Real
Contractor
-hourlyRate : Real
-hoursMonth : int
+calculateSalary() : Real
Intern
+setSalary(salary : Real) : void
+calculateSalary() : Real

NOTE: The constructors of the different classes have been omitted from the diagram.

In this model, *Employee* represents each of the company's workers. This class has two special employee subtypes *Intern* and *Contractor* that represent interns and freelancers respectively. Finally, *Payroll* manages the list of company workers and offers the calculateTotalPayroll operation that calculates the company's total personnel cost.

Integrity constraints:

- An employee is identified by first name and last name.

Pseudocode:

```
public class Employee {
    private String name;
    private String surname;
    private double salary;

    public Employee(String name, String surname, double salary) {
        this.name = name;
        this.surname = surname;
        this.salary = salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public double calculateSalary() {
        return salary;
    }
}

public class Intern extends Employee {
    public Intern(String name, String surname) {
        super(name, surname, 0); // Interns do not have regular salary
    }

    public void setSalary(double salary) {
        this.salary = 0; // Interns do not have regular salary
    }

    public double calculateSalary() {
        throw new UnsupportedOperationException("Interns do not have salary");
    }
}

public class Contractor extends Employee {
    private double hourlyRate;
    private int hoursMonth;

    public Contractor(String name, String surname, double hourlyRate, int
hoursMonth) {
        super(name, surname, 0); // Contractors do not have regular salary
        this.hourlyRate = hourlyRate;
        this.hoursMonth = hoursMonth;
    }

    public double calculateSalary() {
        return hourlyRate * hoursMonth;
    }
}

public class Payroll {
    private List<Employee> employees;

    public Payroll {
        employees = new List<Employee>();
    }

    public double calculateTotalPayroll() {
        double totalSalary = 0;
        if(employees!=null) {
            for (Employee employee : employees) {
                if(!employee instanceof Intern) {
                    totalSalary += employee.calculateSalary();
                }
            }
        }
        return totalSalary;
    }
}
```

It is requested:

a) (5 points) Indicate whether this design satisfies Open/Closed Principle (OCP) and justify your answer.

b) (5 points) Indicate whether this design satisfies Don't Repeat Yourself (DRY) Principle and justify your answer.

c) (5 points) Indicate whether this design satisfies High Cohesion Principle and justify your answer.

d) (5 points) Provide the software analysis and design to ensure compliance with the principles mentioned in the previous sections (if they were not already satisfied). The design must include the resulting class diagram and all modified or newly added pseudocode.

Question 2 (10 points)

Statement

Design principles play a crucial role in software development, as adhering to them ensures the quality of the design. Among them is the Liskov Substitution Principle (LSP), which states that a subclass should be able to replace its base class without altering the behavior of the program. To understand this in depth, research its meaning and application using the course materials and references available in the classroom.

Once the concept is understood, answer the following questions:

a) (2.5 points) Does the Liskov Substitution Principle hold in the design of Exercise 1 before applying your changes? Justify your answer.
b) (2.5 points) Does the Liskov Substitution Principle hold in the design of Exercise 1 after applying your changes? Justify your answer.
c) (2.5 points) Provide a code example (before or after the changes in Exercise 1) that clearly shows why the Liskov Substitution Principle is violated, and explain why it does so.
d) (2.5 points) What are the consequences of violating this principle?

Question 3 (10 points)

Statement

A company that manufactures medical equipment for hospitals and clinics wants to develop a system to record the specifications of the different types of medical devices it builds.

Each device has:

- An identifier name.
- A description that explains its function.
- A unit of measurement in which it operates.

For example, the device called “CARDIOMAX,” with the description “Digital heart rate monitor,” measures “Heart rate in beats per minute (bpm).”

Currently, the company manufactures devices that measure different physiological parameters such as heart rate, blood pressure (in mmHg), and oxygen saturation. For now, each device measures only one physiological parameter.

Each type of device, depending on its accuracy and technology, can operate within specific measurement values. These values must also be stored in our system.

For example, the “CARDIOMAX” device can measure heart rate from 40 bpm to 200 bpm.Se pide:

a) (2.5 puntos) Identificar qué patrones de análisis deben aplicarse para diseñar el sistema, justificando brevemente su utilidad.
b) (7.5 puntos) Proponer un diagrama estático de análisis, aplicando los patrones identificados y asegurando que se cumplan las restricciones de integridad del sistema.

Note: If necessary, adjustments can be made to the patterns explained in the materials to adapt the solution to the specific requirements.

Question 4 (10 points)

Statement

Once the company has developed the system to record the specifications of the different types of medical devices, they now want to add the ability to record the measurements taken by the devices.

Each measurement taken by a medical device has:

- A unique identifier, e.g., the serial number, of the device instance that performed the measurement.
- A date and time when the measurement was taken.
- The recorded measurement.

For example, the device “CARDIOMAX” with serial number “1234” recorded a heart rate measurement of 75 bpm on January 1, 2025, at 10:30 AM.

It is requested:

a) (2.5 points) Identify which analysis patterns should be applied to design the system, briefly justifying their usefulness.

b) (7.5 points) Propose a static analysis diagram, applying the identified patterns and ensuring that the system's integrity constraints are met.

Note: If necessary, adjustments can be made to the patterns explained in the materials to adapt the solution to the specific requirements.