UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 7 Examination in Engineering: March 2021

Module Number: EE7205 **Module Name: Object Oriented Design Patterns and Principles**

[Three Hours]

[Answer **all questions**, each question carries **10** marks]

Q1 a) List down 2 benefits of complying with SOLID principles when writing an object-oriented code.

[2.0 Marks]

b) What is Liskov Substitution in SOLID principles?

[3.0 Marks]

c) A student has identified the following classes and interfaces while refactoring the SalaryCalculator class (**Listing Q1.1**) to comply with SOLID principles.

- Employee – interface
- ContractEmployee – class
- PermanentEmployee – class
- EmployeeFactory – class

Write the code for the ContractEmployee and EmployeeFactory classes.

```
public class SalaryCalculator {

    public double getSalary(double basicSalary, String employmentType){

        if("Contract".equals(employmentType)){
            return basicSalary - calcTax(basicSalary, employmentType);
        }

        return basicSalary - calcTax(basicSalary, employmentType) -
(basicSalary * 8 / 100);

    }

    private double calcTax(double basicSalary, String employmentType) {

        if("Contract".equals(employmentType)){
            return basicSalary * 10 / 100;

        }
        return (basicSalary - 250000) * 6 / 100;
    }
}
```

Listing Q1.1

[5.0 Marks]

Q2 a) A developer has decided to follow the "interpreter" design pattern to develop a program to check whether a person is male and older than 18 years. Explain the interpreter pattern and its benefits using this scenario.

[4.0 Marks]

b) Implement one terminal and one non-terminal expression identified in the scenario mentioned in the question **Q2 (a)** using java.

[6.0 Marks]

Q3 a) Does high cohesion decrease coupling? - justify your answer.

[3.0 Marks]

b) Answer the (i) and (ii) based on the code in **Listing Q3.1**.

```
class LeaderSelector{
    private SelectionStrategy strategy = new RandomSelectionStrategy();

    public String select(String[] names) {
        return "Leader " + strategy.select(names);
    }
}

interface SelectionStrategy{
    String select(String[] names);
}

class RandomSelectionStrategy implements SelectionStrategy{

    @Override
    public String select(String[] names) {
        int index = new Random().nextInt(names.length);
        return names[index];
    }
}
```

Listing Q3.1

i) What is the SOLID principle violated in the above code?

[2.0 Marks]

ii) Solve the above mentioned issue by refactoring the code.

[5.0 Marks]

Q4 a) Briefly explain the benefits of the observer pattern.

[2.0 Marks]

b) Explain the difference between the observer pattern and the mediator pattern.

[2.0 Marks]

c)
```
class Aggregator {
    void showAggregates(int[] numbers){
        // implementation ....
    }
}
```

You are supposed to implement the above **showAggregates(int[] numbers)** method which will show the Total and Average of the given numbers in the command line. There is a high chance of adding many other operations like Mean, median and mode like aggregates in the future.

Implement the method using a suitable design pattern.

[6.0 Marks]

Q5 a) What is the importance of mocking in unit testing?

[2.0 Marks]

b) What is the benefit of using "hamcrest" library for unit tests written in Java?

[2.0 Marks]

c) Implement two unit tests for the **validate(String word)** method in the code given below.

```
interface SpellChecker{
    boolean check(String word);
}

public class Input{

    private SpellChecker checker;

    public Input(SpellChecker checker) {
        this.checker = checker;
    }

    public boolean validate(String word){
        if(word.equals("")){
            return false;
        }
        return checker.check(word);

    }
}
```


[4.0 Marks]

d) Which code line in the following code makes unit testing harder and explain why?

```
public class Calculator{

    public void add(int x, int y){
        int output = x + y;
        System.out.println(output);
    }
}
```

[2.0 Marks]