National University of Computer and Emerging Sciences

Laboratory manual # 06
For
Software Design and Architecture

Course Instructor	Amir Iqbal
Lab Instructor	Muhammad Hashir Mohsineen, Syeda Aina Batool
Email	hashir.mohsineen@lhr.nu.edu.pk ainnie.batool275@gmail.com
Section	BSE-4D
Date	3-13-24 (MM/DD/YY)
Semester	Spring 24

Instructions for lab submission:
You have to submit source files along with a word document. In the word document you have to give the heading of each exercise/question, then paste your code. Save your word document in the following format: roll number-lab no-section i.e. 21I-0008-lab06-BSE4D.

Objective:

- ☐ Liskov Substitution Principle (LSP)
- ☐ Interface Segregation Principle (ISP)
- ☐ Dependency Inversion Principle (DIP)

Software for this lab:

- Java netbeans
- (You can also use other Java IDEs i.e. Eclipse)

1. **Exercise: Marks: 10**

BankTech Solutions" is developing a banking application to manage various account types, such as savings and checking accounts. The company prioritizes adherence to Liskov's Substitution Principle to maintain a flexible and scalable system architecture.

Implement a banking system according to the following key components:

- Superclass Definition: Create a superclass named **Account** with common functionalities such as *deposit()* and *withdraw()* methods.
- Subclass Implementation: Implement subclasses like **SavingsAccount** and **CheckingAccount**, each extending the Account superclass and providing specific implementations of *deposit()* and *withdraw()* methods.
- Make sure that the behavior of the *withdraw()* method is appropriate for overdraft protection in CheckingAccount.
- Client Utilization: In main,Use the **BankOperations** class to perform banking operations such as transferring funds between accounts without directly referencing the subclasses.
- Analyze whether your implementation follows the Liskov Substitution Principle. If not, identify the issues and refactor your code accordingly.

2. **Exercise:** **Marks: 10**

Develop a system for managing different types of devices in a smart home. You have an interface Device that is implemented by various classes representing different types of devices. Your task is to ensure that the code adheres to the Interface Segregation Principle.

Instructions:

- Define an interface **Device** with methods *turnOn()* and *turnOff()*.
- Implement two additional interfaces: **Light** and **SmartSpeaker**.
- Ensure that the Light interface contains methods specific to controlling lights, such as *dim()* and *brighten()*.
- Ensure that the SmartSpeaker interface contains methods specific to controlling smart speakers, such as *playMusic()* and *stopMusic()*.
- Implement classes representing different types of devices such as **SmartBulb**, **SmartSwitch**, and **SmartSpeakerDevice**. Make sure these classes implement the necessary interfaces.
- Assess whether your implementation follows the Interface Segregation Principle. If not, refactor your code to adhere to it.

3. **Exercise:** **Marks: 10**

Develop a system to manage tasks in a project management tool. You have a **Project** class responsible for managing tasks, and a **NotificationService** class responsible for sending notifications to users. Your task is to ensure that the code adheres to the Dependency Inversion Principle.

Instructions:

- Define an interface **TaskRepository** with methods *addTask()* and *getTasks()*.
- Project class will depend on the TaskRepository interface rather than concrete implementations.
- Implement two concrete classes: **DatabaseTaskRepository** and **FileTaskRepository**, both implementing TaskRepository. DatabaseTaskRepository stores tasks in a database, while FileTaskRepository stores tasks in files.
- The **NotificationService** class will accept a TaskRepository instance in its constructor, rather than directly accessing the task repository.

- Assess whether your implementation follows the Dependency Inversion Principle. If not, refactor your code to adhere to it.
- Test the behavior of the system by adding tasks and retrieving them.
- In this question your main focus should be on the architecture of the program rather than small details like implementing function to save in database (You can just print some statement in function implementation).
