National University of Computer and Emerging Sciences

Lab 10 Manual

for

SCD Lab

Course Instructor	Sir Waqas Ali
Lab Instructor(s)	Zain Ali Nasir, Hira Tayyab
Section	BSE 5C & B
Date	Wednesday, Oct 30, 2024
Semester	Fall 2024

Department of Software Engineering
FAST-NU, Lahore, Pakistan

In this lab we cover:

- Design Pattern and Java Unit Testing

What are Design Patterns?

Design patterns are typical solutions to commonly occurring problems in software design. They are like pre-made blueprints that you can customize to solve a recurring design problem in your code.

You can't just find a pattern and copy it into your program, the way you can with off-the-shelf functions or libraries. The pattern is not a specific piece of code, but a general concept for solving a particular problem. You can follow the pattern details and implement a solution that suits the realities of your own program.

Patterns are often confused with algorithms, because both concepts describe typical solutions to some known problems. While an algorithm always defines a clear set of actions that can achieve some goal, a pattern is a more high-level description of a solution. The code of the same pattern applied to two different programs may be different.

An analogy to an algorithm is a cooking recipe: both have clear steps to achieve a goal. On the other hand, a pattern is more like a blueprint: you can see what the result and its features are, but the exact order of implementation is up to you.

Classification of Design Patterns:

In addition, all patterns can be categorized by their *intent*, or purpose. Here we covers three main groups of patterns:

- **Creational patterns** provide object creation mechanisms that increase flexibility and reuse of existing code.

- **Structural patterns** explain how to assemble objects and classes into larger structures, while keeping these structures flexible and efficient.

- **Behavioral patterns** take care of effective communication and the assignment of responsibilities between objects.

Source : https://refactoring.guru/design-patterns/what-is-pattern

What is JUnit Testing?

JUnit is a unit testing open-source framework for the Java programming language. Java Developers use this framework to write and execute automated tests. In Java, there are test cases that have to be re-executed every time a new code is added. This is done to make sure that nothing in the code is broken.

Lab 10 Manual

Question 1: Using the Builder Design Pattern, write a Java program to display a `JFrame` with the following properties:

- The `JFrame` should have a title passed as a parameter.
- It should display two buttons: a positive button labeled "**Yes**" and a negative button labeled "No."
- The height and the width of the frame should also be customizable through a parameter.

Use the Builder pattern to create the `JFrame` instance with the specified title, height,width, and buttons. The final builder should be able to create a frame with any title, height,width, and button configurations as needed.

Required Builder Interface implements by JFrameBuilder class.

Question 2: You have been tasked with developing a CGPA (Cumulative Grade Point Average) calculator in Java. This calculator should accept an array of course grades (each grade between 0.0 and 4.0) and calculate the CGPA as the average of these grades. If the input array is empty, the calculator should return 0.0 as the CGPA.

Write JUnit tests to verify the correctness of your `CGPACalculator` class. Ensure your tests cover:

- Normal Input: Calculating CGPA for an array of valid grade points (e.g., `[3.0, 3.5, 4.0, 2.5]`).
- Boundary Conditions: Calculating CGPA for grades at the lower and upper limits (e.g., `[0.0, 4.0, 4.0, 0.0]`).
- Empty Input: Calculating CGPA when the input array is empty.
- Invalid Input: Handling of grades outside the 0.0 to 4.0 range. (Assume any invalid input should throw an `IllegalArgumentException`.)

Write JUnit test cases to validate each of the above scenarios.