University of Ruhuna
Faculty of Technology
Department of Informtion and Communication Technology

Advanced Programming Practicum
(ICT3122)

Lab Sheet 2 22nd Jan 2024

Factory Design Pattern

Step 1: Understanding the Problem

The code we are going to construct is a simple implementation of the Factory Design Pattern for a cellular plan. The goal is to create different network plans with varying rates and calculate the bill based on the selected plan and usage.

Step 2: Create a Project Structure

1. Open your Java IDE (Integrated Development Environment) or a text editor to create a new Java project.
2. Create a package named **FactoryDesignPattern**.
3. Inside this package, create five files with the respective class names: **cellularplan**, **abcNetwork**, **pqrNetwork**, **xyzNetwork**, and **SelectNetworkFactory**.

Step 3: Define the Abstract Class cellularplan

1. In the **cellularplan** class file, declare a protected double variable **rate**.
2. Declare an abstract method **getRate()** with no implementation.
3. Define a method **processBill(int minutes)** to calculate and print the bill based on the given formula.
4. Save the file.

Step 4: Implement Concrete Classes

1. In the **abcNetwork**, **pqrNetwork**, and **xyzNetwork** class files, extend the **cellularplan** abstract class.
2. Implement the **getRate()** method in each class, setting the rate as specified.
3. Save each file.

Step 5: Create the SelectNetworkFactory Class

1. In the **SelectNetworkFactory** class file, declare a method **getPlan(String planType)** that takes a plan type as input and returns an instance of the corresponding network plan.
2. Use conditional statements to check the **planType** and return the appropriate plan instance (**abcNetwork**, **pqrNetwork**, or **xyzNetwork**).
3. Save the file.

Step 6: Implement the phoneBill Class

1. In the **phoneBill** class file, create a **main** method.
2. Instantiate a **SelectNetworkFactory** object.
3. Use **BufferedReader** to read the network name and minutes from the user.
4. Get the corresponding network plan using the factory and calculate the bill.
5. Print the bill amount.
6. Save the file.

Step 7: Compile and Run

1. Compile all the Java files.
2. Run the **phoneBill** class.
3. Enter the network name and minutes when prompted.
4. Verify that the bill amount is calculated correctly based on the selected network plan and usage.

Singleton Design Pattern _____

Pre-requisites:

- You must have MySQL server installed on your machines before starting this.

Step 1: Create a Project Structure

1. Open your Java IDE or a text editor to create a new Java project.
2. Create two files, naming them **JDBCSingleton.java** and **JDBCSingletonDemo.java**.

Step 2: Define the JDBCSingleton Class

1. In the **JDBCSingleton** class file, declare a private static instance of **JDBCSingleton** and a private constructor to make it a singleton.
2. Implement a public method **getInstance()** to get the singleton instance.
3. Implement a private method **getConnection()** to establish a database connection.
4. Implement methods for database operations: **insert**, **view**, **update**, and **delete**.
5. Save the file.

Step 3: Implement the JDBCSingletonDemo Class

1. In the **JDBCSingletonDemo** class file, declare a **JDBCSingleton** object.
2. Use a **BufferedReader** for user input.
3. Create a loop to display a menu for different database operations.
4. Implement cases for each operation: **insert**, **view**, **delete**, **update**, and **exit**.
5. Call the corresponding methods in the **JDBCSingleton** class for each operation.
6. Save the file.

Step 4: Compile and Run

1. Compile both Java files.
2. Run the **JDBCSingletonDemo** class.
3. Follow the menu prompts to perform database operations such as insertion, viewing, deletion, and updating.
4. Verify that the operations are executed successfully and the program responds as expected.