National University of Computer and Emerging Sciences

Laboratory manual # 08
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	04-17-24 (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-lab08-BSE4D.

Objective:

- ☐ Creational Design Patterns – I I
- ☐ Abstract Factory Design Pattern
- ☐ Singleton Design Pattern

Software for this lab:
- Java netbeans
- StarUML
- Microsoft SQL Server and Management Studio

Create class diagrams and write Java code for the following:

1. **Exercise: Marks: 10**

You're tasked with designing a game character creation system for a fantasy role-playing game (RPG). The game offers two character classes: Warrior and Mage. Each class has different attributes and abilities. Additionally, players can choose between two character races: Human and Elf, which further influence the character's appearance and starting stats.
Instructions:
- Define an abstract class called Character with methods void displayInfo() and String getType().
- Implement two concrete classes Warrior and Mage, both inheriting from Character.
- Define an abstract factory interface called CharacterFactory with methods Character createHumanCharacter() and Character createElfCharacter().
- Implement two concrete factory classes HumanCharacterFactory and ElfCharacterFactory, both implementing the CharacterFactory interface.
- Implement a client class called CharacterCreator with a main method to demonstrate the usage of the abstract factory pattern.
-
- Each character class should have a method displayInfo() that prints out the character's class and attributes and a method getType() that returns a string indicating the class of the character.
- createHumanCharacter() and createElfCharacter() methods in the concrete factory classes should return instances of Character subclasses accordingly.

- In the main method of the CharacterCreator class, demonstrate creating different types of characters (Warrior and Mage) for both Human and Elf races.

2. **Exercise: Marks: 10**

You're developing a software system for a high-end restaurant chain that offers a diverse range of cuisines and drink pairings. The restaurant chain has two distinct types of establishments: Steakhouse and Seafood. There are two tiers of dining experiences: Standard and VIP. Standard dining experiences offer traditional Steakhouse and Margarita, while VIP experiences provide Seafood and Mocktails.
Instructions:

- Define an abstract class called Menu with methods void generateMenu() and String getDescription().
- Define an abstract class called Drink with methods void serveDrink() and String getDescription().
- Implement concrete classes for each menu and drink type (e.gSteakhouseMenu, SeafoodMenu, Margarita, Mocktails).
- Define abstract factory interfaces for MenuFactory.
- Implement concrete factory classes for each family of products (e.g., StandardMenuFactory, VIPMenuFactory) each with 2 methods addDish() and addDrink().
- Create a client class with a main method to demonstrate the usage of the abstract factory pattern.
-
- The concrete factory classes should return instances of the corresponding concrete product classes based on the specified dining experience (Standard or VIP).
- In the main method of the client class, demonstrate creating menus and drink pairings for each type of restaurant and dining experience.
- **Make sure that the concrete factory classes are following the Singleton design pattern.**

3. **Exercise: Marks: 10**

Java NetBeans and Microsoft SQL connection guide
You are developing a user login system for a school management application. The system should support different types of users, such as students and teachers. You need to implement the following classes:

1. User Class:
    - Create an abstract class named User with the following attributes:
        - username (String)
        - password (String)
    - Implement a constructor to initialize the username, password, and type attributes.
    - Implement a method login(String username, String password) to authenticate the user's login credentials.
2. Student Class:

- Extend the User class to create a subclass named Student.
- Implement a constructor that initializes the username, password attributes using the superclass constructor.
- Override the login method to perform authentication specific to students (e.g., checking against a student database).

3. Teacher Class:
    - Extend the User class to create a subclass named Teacher.
    - Implement a constructor that initializes the username, password attributes using the superclass constructor.
    - Override the login method to perform authentication specific to teachers (e.g., checking against a teacher database).
4. Database Class (Singleton):
    - Implement a singleton class named Database to manage user authentication.
    - The Database class should have a private constructor to prevent instantiation from outside the class.
    - Implement a static method getInstance() that returns the sole instance of the Database class.
