<u>Assignment: 2</u>

1. Find an application of Singleton design pattern in your application and draw its corresponding class diagram and a collaboration diagram and Implement it in Java. Explain in a text document why singleton should be applied in the scenario and how it improves your design.

The Singleton Design pattern is a creational design pattern that states that one and only one instance of a class can be created per JVM and that particular single object will be shared across all the classes or throughout the application.

Scenario implementing Singleton Design Pattern in our application:

In our application Singleton design pattern is used to build the '**Admin**' class that performs the functionalities of the authorized admin of the application. We need only one Admin (i.e., only one object of Admin) to exist throughout the application.

Why Singleton:

This application requires there to be a single admin. An admin can login using ID: ***'adminID@marketplace.com'*** and Password: ***'abc@123'***. This will ensure access control to resources to a single entity that controls the application.

How Singleton pattern will improve the design:

Once an admin is logged in, no other admins can login even with the same ID and Password. Hence, implementing the Admin class as singleton enhances the security of Online Buying, Selling and Bidding Application.

Implementation:

The object of the Admin class has been created. The static property *getInstance(...)* can be used to retrieve a reference to the instance of the Admin class. The Admin class has a private constructor so that no instance of it can be created. The concept of Lazy Loading is also implemented using the *getInstance(...)* method and the object is created on its demand.

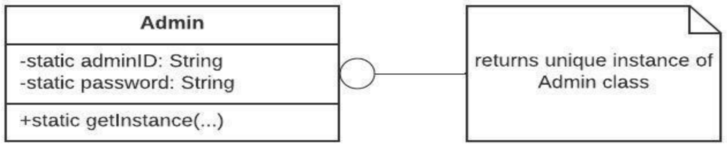
We have implemented Double Checked Locking in the getInstance(...) method. This method will only acquire the lock once, when the *admin* object is null.

We have declared the *admin* object volatile which ensures that multiple threads offer the *admin* variable correctly when it is being initialized to the singleton class *Admin* instance. This method reduces the overhead of calling the synchronized method every time.

Below are the corresponding class and collaboration diagrams:

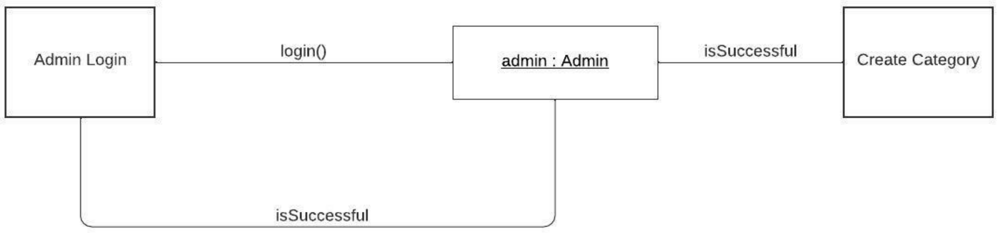
Singleton Design Pattern:

Class Diagram:

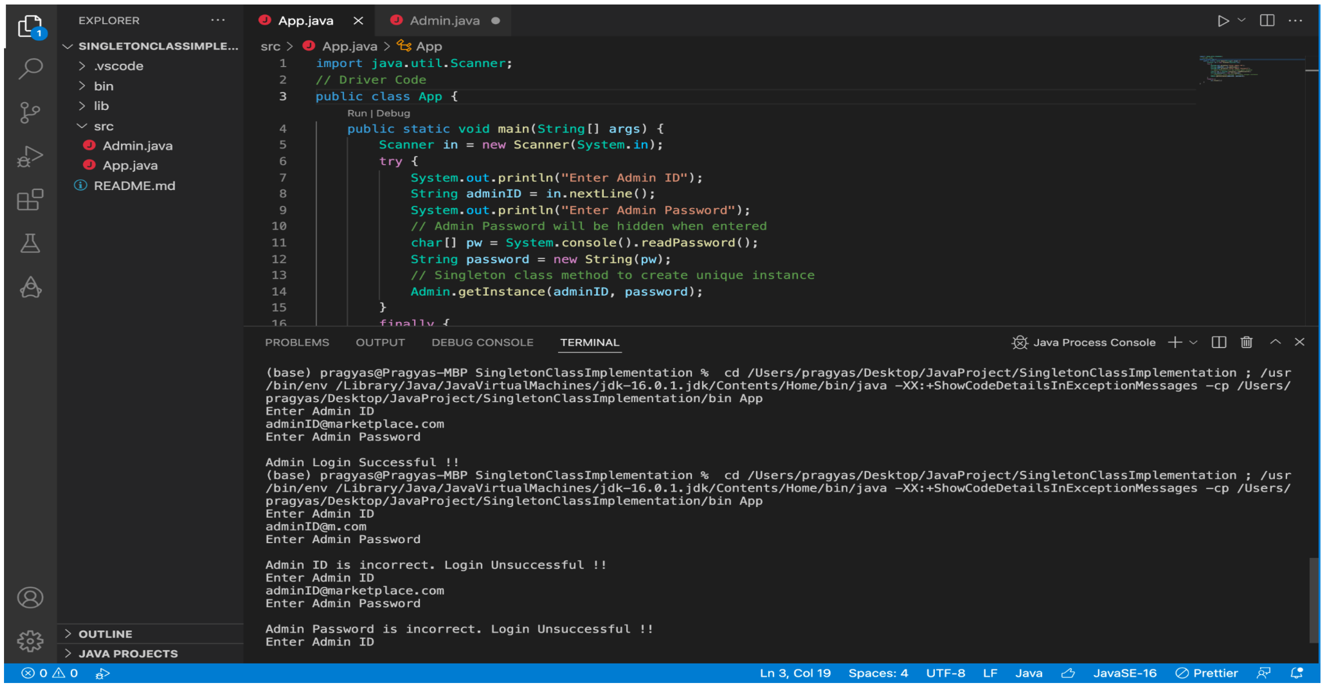


Singleton Design Pattern:

Collaboration Diagram:



Output Screenshot:



```java
import java.util.Scanner;
// Driver Code
public class App {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        try {
            System.out.println("Enter Admin ID");
            String adminID = in.nextLine();
            System.out.println("Enter Admin Password");
            // Admin Password will be hidden when entered
            char[] pw = System.console().readPassword();
            String password = new String(pw);
            // Singleton class method to create unique instance
            Admin.getInstance(adminID, password);
        }
        finally {
```

```
(base) pragyas@Pragyas-MBP SingletonClassImplementation %  cd /Users/pragyas/Desktop/JavaProject/SingletonClassImplementation ; /usr
/bin/env /Library/Java/JavaVirtualMachines/jdk-16.0.1.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/
pragyas/Desktop/JavaProject/SingletonClassImplementation/bin App
Enter Admin ID
adminID@marketplace.com
Enter Admin Password

Admin Login Successful !!
(base) pragyas@Pragyas-MBP SingletonClassImplementation %  cd /Users/pragyas/Desktop/JavaProject/SingletonClassImplementation ; /usr
/bin/env /Library/Java/JavaVirtualMachines/jdk-16.0.1.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/
pragyas/Desktop/JavaProject/SingletonClassImplementation/bin App
Enter Admin ID
adminID@m.com
Enter Admin Password

Admin ID is incorrect. Login Unsuccessful !!
Enter Admin ID
adminID@marketplace.com
Enter Admin Password

Admin Password is incorrect. Login Unsuccessful !!
Enter Admin ID
```