Name: -Aditya Raj Roll No: - 11212714 Section: - A1

Design Patterns
W4A1

Topic: - Implement Singleton Design Pattern.

Github link: - https://github.com/RajAditya01/Design-And-Pattern

The singleton pattern is one of the simplest design patterns. Sometimes we need to have only one instance of our class for example a single DB connection shared by multiple objects as creating a separate DB connection for every object may be costly. Similarly, there can be a single configuration manager or error manager in an application that handles all problems instead of creating multiple managers.

Definition:

The singleton pattern is a design pattern that restricts the instantiation of a class to one object.
Let's see various design options for implementing such a class. If you have a good handle on static class variables and access modifiers this should not be a difficult task.

Code: -

```
class SingletonEagar {
  private static SingletonEagar instance = new SingletonEagar();

  private SingletonEagar(){}

  public static SingletonEagar getInstance() {
    return instance;
  }
}

class Singleton {
  private static Singleton instance;

  private Singleton(){}

  public static Singleton getInstance() {
    if(instance == null) {
```

```
    instance = new Singleton();
   }

   return instance;
  }
}

class SingletonSynchronizedMethod {
  private static SingletonSynchronizedMethod instance;

  private SingletonSynchronizedMethod(){}

  public static synchronized SingletonSynchronizedMethod getInstance() {
   if(instance == null) {
    instance = new SingletonSynchronizedMethod();
   }
   return instance;
  }
}

class SingletonSynchronized {
  private static SingletonSynchronized instance;

  private SingletonSynchronized(){}

  public static SingletonSynchronized getInstance() {
   if(instance == null) {
    synchronized (SingletonSynchronized.class) {
     if(instance == null) {
      instance = new SingletonSynchronized();
     }
    }
   }
   return instance;
  }
}

public class SingletonExample {

  public static void main(String[] args) {
   SingletonSynchronized instance = SingletonSynchronized.getInstance();

   System.out.println(instance);
```

```
    SingletonSynchronized instance1 = SingletonSynchronized.getInstance();

    System.out.println(instance1);
  }
}
```

Output: -

```
PS D:\Design-And-Pattern> cd "d:\Design-And-Pattern\" ;
SingletonSynchronized@5d22bbb7
SingletonSynchronized@5d22bbb7
PS D:\Design-And-Pattern>
```