ASSIGMENT JAVA DAY10&11

Harshit Kushmakar| 16896

1. **What are the three different ways of creating Singleton Objects? Show a demo of all three.**

```
2. package assignment10_11;

   public class EagerSingleton {
       private static final EagerSingleton INSTANCE = new
   EagerSingleton();


       private EagerSingleton() {
       }

       public static EagerSingleton getInstance() {
           return INSTANCE;
       }
   }
```

```
package assignment10_11;

public class LazySingleton {
    private static LazySingleton INSTANCE;

    private LazySingleton() {
    }

    public static LazySingleton getInstance() {
        if (INSTANCE == null) {
            INSTANCE = new LazySingleton();
        }
        return INSTANCE;
    }
}
```

```
package assignment10_11;

public enum EnumSingleton {
    INSTANCE;

    public void doSomething() {
        // implementation goes here
    }
}
```

```
package assignment10_11;

public class SingletonDemo {
```

```
    public static void main(String[] args) {
        EagerSingleton eagerSingleton1 = EagerSingleton.getInstance();
        EagerSingleton eagerSingleton2 = EagerSingleton.getInstance();

        System.out.println(eagerSingleton1 == eagerSingleton2); // true

        LazySingleton lazySingleton1 = LazySingleton.getInstance();
        LazySingleton lazySingleton2 = LazySingleton.getInstance();

        System.out.println(lazySingleton1 == lazySingleton2); // true

        EnumSingleton enumSingleton1 = EnumSingleton.INSTANCE;
        EnumSingleton enumSingleton2 = EnumSingleton.INSTANCE;

        System.out.println(enumSingleton1 == enumSingleton2); // true
    }
}
```

```
package assignment10_11;

public enum EnumSingleton {
    INSTANCE;

    public void doSomething() {
        // implementation goes here
    }
}
```

2. Find out which of the SOLID principles are used in Java API and which part of API. Explain the same citing the references in a pdf file.

Single Responsibility Principle (SRP): According to this principle, a class should have only one reason to change. In other words, a class should have only one responsibility. One example of SRP in the Java API is the File class.

Open-Closed Principle (OCP): According to this principle, a class should be open for extension but closed for modification. In other words, a class should allow its behavior to be extended without modifying its source code.

Liskov Substitution Principle (LSP): According to this principle, objects of a superclass should be able to be replaced with objects of a subclass without affecting the correctness of the program.

Dependency Inversion Principle (DIP): According to this principle, high-level modules should not depend on low-level modules. Instead, both should depend on abstractions

Interface Segregation Principle (ISP): According to this principle, clients should not be forced to depend on methods they do not use. In other words, a class should not have to implement methods that it does not need

3. **Findout which part of Java API is based on Adapter pattern. Document your findings**

   The Adapter pattern is a design pattern that allows incompatible interfaces to work together. It involves creating an adapter class that translates the interface of one class into another interface that the client expects.

   In the Java API, one example of the Adapter pattern is the java.util.Arrays class. This class provides a set of static methods that can be used to manipulate arrays in various ways, such as sorting, searching, and copying. The methods in the Arrays class are designed to work with arrays, but they are not designed to work with other collection types, such as List or Set.