ShortQuestion_3

ShortQuestion_3

1. **What are the types of design patterns in Java?**

Desgin Pattern Type	Sub-type
Creational	Factory Pattern; Abstract Factory Pattern; Singleton Pattern; Prototype Pattern; Builder Pattern.
Structural	Adapter Pattern; Bridge Pattern; Composite Pattern; Decorator Pattern; Facade Pattern; Flyweight Pattern; Proxy Pattern
Behavioral	Chain Of Responsibility Pattern; Command Pattern; Interpreter Pattern; Iterator Pattern; Mediator Pattern; Memento Pattern; Observer Pattern; State Pattern; Strategy Pattern; Template Pattern; Visitor Pattern

2. **What are the SOLID Principles?**

SOLID Principles	Description
Single Responsibility	Each class should be responsible for a single part or functionality of the system.
Open-Closed	Software components should be open for extension, but not for modification.
Liskov Substitution	Objects of a superclass should be replaceable with objects of its subclasses without breaking the system.
Interface Segregation	No client should be forced to depend on methods that it does not use.
Dependency Inversion	High-level modules should not depend on low-level modules, both should depend on abstractions.

3. **How can you achieve thread-safe singleton patterns in Java?**
   We can use synchronized block inside the if loop and volatile variable, as shown below.

```
public class Singleton {
    private Singleton() {}

    private static volatile Singleton instance;

    public static Singleton getInstance() {
```

```
        if(instance == null) {
            synchronized (Singleton.class) {
                if (instance == null) {
                    instance = new Singleton();
                }
            }
        }
        return instance;
    }
}
```

4. **What do you understand by the Open-Closed Principle (OCP)?**
   The code should be written in a way that new functionality can be added without changing the existing code. The change to one of the classes don't affect other depending classes. In a nutshell, the developer must need to change only a specific part of the code (a class or a function) every time a requirement changes. The open/closed principle is generally achieved by using **inheritance** and **polymorphism**.