Singleton Pattern

Definition

- Singleton pattern is one of the twenty-three design pattern that are introduced by “Gang of four”
- It restricts the instantiation of a class to one single instance while providing a global access to this instance.
- This is useful when exactly one object is enough for entire application.
- The most common reason for this is to control access to some shared resource—for example, a database or a file.

Creation Method - 1

```
public final class Singleton {

    private static final Singleton INSTANCE = new Singleton();

    private Singleton() {}

    public static Singleton getInstance() {

        return INSTANCE;

    }

}
```

- Private constructor restricts creation of new objects outside of class
- Static getInstance() method returns pointer to INSTANCE w/o creating new object
- The object is created only once(final) and accessible using static method

Creation Method - 2(Lazy)

```
public final class Singleton {

    // volatile makes read and write atomic

    private static volatile Singleton instance = null;

    private Singleton() {}

    public static Singleton getInstance() {

        if (instance == null) {

            synchronized(Singleton.class) {

                if (instance == null) {

                    instance = new Singleton();

                }

            }

        }

        return instance;

    }

}
```

- Same as before except the instance is created when the static method is first invoked
- The creation part is atomized using synchronized block, ensuring single object creation

Applicability

- Use when a class in your program should have just a single instance available to all clients.
- When you need stricter control over global variables.

Problems

- Introduces global state into an application
- Violates the Single Responsibility Principle. The pattern solves two problems at the time(single object creation and controlled global access)
- The pattern requires special treatment in a multithreaded environment so that multiple threads won’t create a singleton object several times.

Thanks