Java Synchronization: Method, Block, and Static (with Clarifications)

1. synchronized Method (Instance-level Lock)

- Syntax: public synchronized void methodName() {}

- Lock is on 'this' object (instance).

- If two threads access the method using the SAME object, then synchronization works correctly.

- If two threads access the method using DIFFERENT objects, then each gets a separate lock, so synchronization DOES NOT WORK between them.

Example:

```
Counter obj1 = new Counter();
Counter obj2 = new Counter();

Thread t1 = new Thread(() -> obj1.increment());
Thread t2 = new Thread(() -> obj2.increment());
```

Both threads run simultaneously because obj1 and obj2 are different objects, and locks are not shared.

2. synchronized Block

- More flexible. You can synchronize only a specific block of code.

- Lock can be on any object (e.g., this, a shared object, or a static object).

Example:

```
class Counter {
    int count = 0;

    public void increment() {
        synchronized(this) {
            count++;
        }
    }
}
```

3. static synchronized Method (Class-level Lock)

- Lock is on the Class object (i.e., ClassName.class).
- No matter how many objects are created, all threads use the same lock.
- Ensures true synchronization when dealing with static (shared) variables.

Example:

```
class Counter {
    static int count = 0;

    public static synchronized void increment() {
        count++;
    }
}
```

Summary:

- synchronized method -> object-level lock

- static synchronized method -> class-level lock

- synchronized block -> custom control on locking object