Example Program: UnSynchronized vs Synchronized

Unsynchronized Multithreaded Program:

In this example, two threads increment a shared counter without synchronization, leading to a race condition and incorrect results.

```
public class UnsynchronizedCounter {
    private int count = 0;

    public void increment() {
        count++;
    }

    public int getCount() {
        return count;
    }

    public static void main(String[] args) {
        UnsynchronizedCounter counter = new UnsynchronizedCounter();

        Thread thread1 = new Thread(() -> {
            for (int i = 0; i < 1000; i++) {
                counter.increment();
            }
        });

        Thread thread2 = new Thread(() -> {
            for (int i = 0; i < 1000; i++) {
                counter.increment();
            }
        });

        thread1.start();
        thread2.start();

        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Unsynchronized Counter Value: " + counter.getCount());
```

```
    }
}
```

Synchronized Multithreaded Program:

In this example, the shared counter is accessed within synchronized blocks, ensuring that only one thread can increment it at a time, preventing a race condition.

```
public class SynchronizedCounter {
    private int count = 0;

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

    public synchronized int getCount() {
        return count;
    }

    public static void main(String[] args) {
        SynchronizedCounter counter = new SynchronizedCounter();

        Thread thread1 = new Thread(() -> {
            for (int i = 0; i < 1000; i++) {
                counter.increment();
            }
        });

        Thread thread2 = new Thread(() -> {
            for (int i = 0; i < 1000; i++) {
                counter.increment();
            }
        });

        thread1.start();
        thread2.start();

        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Synchronized Counter Value: " + counter.getCount());
    }
}
```

In the unsynchronized program, due to the absence of synchronization, both threads access and modify the counter concurrently, leading to a race condition and incorrect counter value. In the synchronized program, the synchronized keyword ensures that only one thread can access the shared counter at a time, preventing the race condition and resulting in the correct counter value.