THE SINGLETON PATTERN

SYNCHRONIZED METHODS

ANY METHOD IN JAVA CAN BE MARKED AS SYNCHRONIZED

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

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

    public synchronized void decrement() {
        c--;
    }

    public synchronized int value() {
        return c;
    }
}
```

REMEMBER HOWEVER THAT THE "ONLY-1-THREAD-AT-A-TIME" ONLY APPLIES TO THE SAME METHOD OF THE SAME OBJECT

DOING SO MEANS THAT ONLY ONE THREAD CAN BE EXECUTING THIS MEMBER FUNCTION ON THIS OBJECT AT A GIVEN POINT IN TIME

SO, IF FOR INSTANCE THE METHOD DOES SOMETHING TO A STATIC CLASS VARIABLE (NOT AN OBJECT VARIABLE), ERRORS CAN STILL RESULT

USED RIGHT, MARKING A METHOD AS SYNCHRONIZED CAN HELP ELIMINATE THREAD INTERFERENCE AND MEMORY CONSISTENCY ERRORS

SYNCHRONIZED BLOCKS OF CODE

SINCE EVERY OBJECT IN JAVA HAS AN INTRINSIC LOCK ASSOCIATED WITH IT, IT IS ACTUALLY POSSIBLE TO LOCK ANY SECTION OF CODE BY MARKING IT AS SYNCHRONIZED

IN GENERAL, ANY OBJECT CAN BE USED AS A LOCK USING THE SYNCHRONIZED STATEMENT

```
public void addName(String name) {
    synchronized(this) {
        lastName = name;
        nameCount++;
    }
    nameList.add(name);
}
```

IN FACT MARKING A METHOD AS SYNCHRONIZED IS MERELY A SHORTCUT TO MARKING THE ENTIRE BODY OF THE METHOD AS SYNCHRONIZED ON 'THIS', I.E. ON THE OBJECT IN QUESTION

BTW, A THREAD NEVER GETS BLOCKED ON ITSELF, WHICH MEANS THAT ONE SYNCHRONIZED METHOD OF AN OBJECT CAN ALWAYS CALL ANOTHER SYNCHRONIZED METHOD OF THE SAME OBJECT WITHOUT BLOCKING

THREAD CONTENTION

SYNCHRONIZATION AND LOCKS ARE POWERFUL
AND ANY POWER CAN BE MISUSED

DEADLOCK

TWO THREADS, EACH IS BLOCKED
ON A LOCK HELD BY THE OTHER

LIVELOCK

TWO THREADS DON'T DEADLOCK,
BUT KEEP BLOCKING ON LOCKS
HELD BY EACH OTHER. NEITHER
REALLY CAN PROGRESS

STARVATION

SOME THREADS KEEP ACQUIRING
LOCKS GREEDILY, AND CAUSE
OTHER THREADS TO BE UNABLE
TO GET ANYTHING DONE

CODE IS TRULY LIKE CORPORATE LIFE

MAKE SURE YOUR SINGLETON OBJECTS CAN'T BE CLONED!

1. THE .CLONE METHOD BELONGS IN THE OBJECT CLASS (I.E. EVERY OBJECT HAS THIS METHOD), WHEN IT OUGHT TO HAVE BELONGED IN THE CLONEABLE INTERFACE

2. THE IMPLICATION OF (1) IS THAT ALL OBJECTS HAVE A CLONE METHOD, BUT IF YOU TRY TO CLONE AN OBJECT THAT DOES NOT IMPLEMENT CLONEABLE, A NOTCLONEABLE EXCEPTION IS THROWN

SO MAKE SURE THAT YOUR SINGLETON CLASS DOES NOT IMPLEMENT CLONEABLE - OR IF FOR SOME STRANGE REASON IT DOES -

OVERRIDE THE CLONE() METHOD TO THROW AN EXCEPTION