**Synchronize and scheduling rule**

**1.synchronize**

A java synchronized block marks a method or a block of code as synchronized. Java synchronized blocks can be used to avoid **race conditions.**

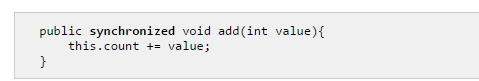
Synchronized blocks in Java are marked with the synchronized keyword. A synchronized block in Java is synchronized on some object. All synchronized blocks synchronized on the same object can only have one thread executing inside them at the same time. All other threads attempting to enter the synchronized block are blocked until the thread inside the synchronized block exits the block.

The *synchronized* keyword can be used to mark 4 types of blocks:

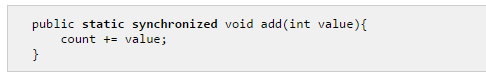
* Instance methods
* Static methods
* Code blocks inside instance methods
* Code blocks inside static methods

1. Synchronized Instance methods

Here is a synchronized instance method:

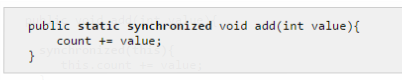


1. Synchronized static methods



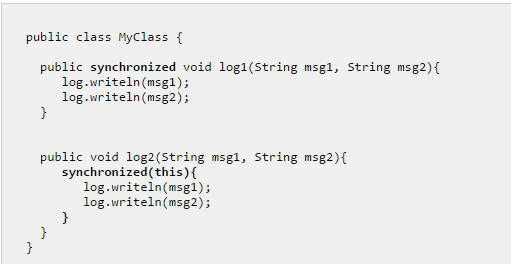
1. Synchronized Blocks in instance methods

You do not have to synchronize a whole method. Sometimes it is preferable to synchronize oly part of a method. Java synchronized blocks inside methods makes this possible.

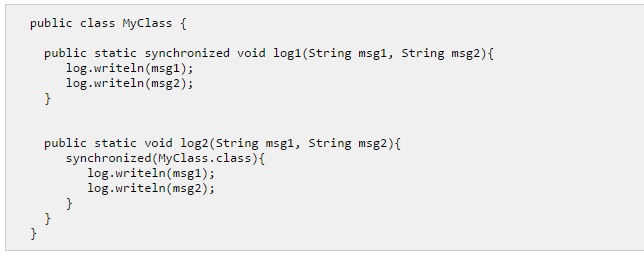


Notice how the Java synchronized block construct takes an object in parentheses. In the example “this” is used, which is the instance the add method is called on.

The following 2 examples are both synchronized on the instance they are called on. They are equivalent with respect to synchronization:



1. Synchronized Blocks in static methods



**2. Scheduling Rule**

First, we have to create a rule:

class Mutex implements ISchedulingRule {

public boolean isConflicting(ISchedulingRule rule) {

return rule == this;

}

public boolean contains(ISchedulingRule rule) {

return rule == this;

}

}

Then, this rule is then added