**Difference between Synchronized Method and Block**

1) One significant difference between synchronized method and block is that, Synchronized block generally **reduce scope of lock**. As scope of lock is inversely proportional to performance, its always better to lock only critical section of code. One of the best example of using synchronized block is [double checked locking in Singleton pattern](http://javarevisited.blogspot.com/2012/07/why-enum-singleton-are-better-in-java.html) where instead of locking whole getInstance() method we only lock critical section of code which is used to create Singleton instance. This improves performance drastically because locking is only required one or two times.

2) Synchronized block provide **granular control over lock**, as you can use arbitrary any lock to provide mutual exclusion to critical section code. On the other hand synchronized method always lock either on current object represented by [this keyword](http://javarevisited.blogspot.com/2012/01/this-keyword-java-example-tutorial.html) or class level lock, if its static synchronized method.

3) Synchronized block can throw throw [java.lang.NullPointerException](http://java67.blogspot.sg/2012/09/what-is-nullpointerexception-in-java.html) if expression provided to block as parameter evaluates to null, which is not the case with synchronized methods.

4) In case of synchronized method, lock is acquired by thread when it enter method and released when it leaves method, either normally or by throwing Exception. On the other hand in case of synchronized block, thread acquires lock when they enter synchronized block and release when they leave synchronized block.

**What are the differences between synchronized method and synchronized block (statement)?**

The synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared resource while another thread is in the process of using or updating that resource.

There two synchronization syntax in Java Language. The practical differences are in controlling scope and the monitor. With a synchronized method, the lock is obtained for the duration of the entire method. With synchronized blocks you can specify exactly when the lock is needed.

Basically, synchronized blocks are more general, and synchronized methods can be rewritten to use synchronized blocks:

class Program {  
 public **synchronized** void f() {  
 .........  
 }  
}

is equivalent to

class Program {  
 public void f() {  
 **synchronized(this){  
 ...  
 }**  
 }  
}

For example, You have a method with some parts that need synchronized and others don't.

The synchronized block lets you synchronize only the partial line codes that really need it.

public class Program {  
 private static Object locker1 = new Object();  
 private static Object locker1 = new Object();  
 public void doSomething1() {  
 ...  
 synchronized(locker1) {  
 ......... //do something protected;  
 }  
 ....  
 }  
 public void doSomething2() {  
 ...  
 synchronized(locker2) {  
 ......... //do something protected;  
 }  
 ....  
 }  
}

The synchronized block can only be executed after a thread has acquired the lock for the object or class referenced, for example the "locker1" or "locker2" in above code, in the synchronized statement.

The above code shows that synchronized block can be holding different object monitors. Maybe it's necessary to protect doSomething1() method and doSomething2() method from multiple threads, but it's fine if one thread is in the doSomething1() method and another is in the doSomething2() method. But the synchronized method can not do it.

A synchronized method synchronizes on the object instance or the class. A thread only executes a synchronized method after it has acquired the lock for the method's object or class.

* *static* synchronized methods synchronize on the class object. If one thread is executing a static synchronized method, all other threads trying to execute any static synchronized methods, in the same class, will block.
* *non-static* synchronized methods synchronize on "this" (the instance object). If one thread is executing a synchronized method, all other threads trying to execute any synchronized methods, in the same class, will block.

These are very public monitors, meaning some other thread could synchronize on them for the wrong reason, leading to slowdowns or deadlocks.