IllegalMonitorStateException

Learn the reasons that cause IllegalMonitorStateException to be thrown.

Explanation

The `IllegalMonitorStateException` is a common programming error that can show up in concurrent programs. Depending on the structure of the program, the exception may occur consistently or only occasionally. The `IllegalMonitorStateException` exception class extends the `RuntimeException` class and according to the official documentation is **thrown to indicate that a thread has attempted to wait on an object's monitor or to notify other threads waiting on an object's monitor without owning the specified monitor**. In other words if you invoke `wait()` or `notify()`/`notifyAll()` without synchronizing on the object i.e. outside of a `synchronized` method or block (with the object as the synchronization target) then `IllegalMonitorStateException` will be thrown. Similarly, the exception is thrown when you invoke these methods on an instance of `Condition` class without acquiring the associated lock with the condition. The class is part of the `java.lang` package and not `java.util.concurrent.*`.

Repro using `Lock`

Repro using `Lock`

The program below demonstrates generating `IllegalMonitorStateException` using a `Condition` object that was instantiated from a `Lock` object.

Java — Press ⇧ + ↵ to interact — main.java — Saved

```java
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class Demonstration {
    public static void main( String args[]) throws Exception {
        Lock lock = new ReentrantLock();
        Condition condition = lock.newCondition();

        // throws exception because we didn't lock the associated
        // lock object with the condition variable before invoking
        // await() on the condition object.
```

Run

The fix for the above program appears below:

```java
// acquire the associated lock
lock.lock();
try {
    while (/* some condition */) {
        // always invoke await or wait in a loop to cater for spurious wake-ups
        // and after synchronizing on the associated lock
        condition.await();
    }
} finally {
    // remember to unlock in a finally block
    lock.unlock();
}
```

Repro using object

The program in the widget below causes `IllegalMonitorStateException` by invoking `notifyAll()` on an object without synchronizing on it.

Java — main.java — Saved

```java
class Demonstration {
    public static void main( String args[] ) throws Exception {
        Object myObject = new Object();

        // throws exception because we didn't synchronize
        // on myObject before invoking the wait() method
        myObject.notifyAll();
    }
}
```

Run

The fix for the above program appears below:

```
Object myObject = new Object();

synchronized (myObject) {
    // invoking notifyAll() in a block synchronized on myObject
    myObject.notifyAll();
}
```

Technical Quiz

Consider the program below and state the outcome of executing the `main()` method.

1.
```
public class IllegalMonitorStateQuiz {

    synchronized void someFunction() throws InterruptedException {
        this.wait();
    }

    public static void main(String[] args) throws Exception {

        (new IllegalMonitorStateQuiz()).someFunction();
    }
```

- A. `IllegateMonitorStateException` is thrown.
- B. `InterruptedException` is thrown.
- C. Program is either blocked on `wait()` forever or exits because of a spurious wakeup.
- D. The target of `synchronized` and the object invoking `wait()` are different and the program doesn't compile.

Reset Quiz | Q1 / Q2 | Submit Answer