<https://stackoverflow.com/questions/11474769/trying-to-understand-why-a-thread-could-be-blocked-in-eclipse>

# [Trying to understand why a thread could be blocked in eclipse?](https://stackoverflow.com/questions/11474769/trying-to-understand-why-a-thread-could-be-blocked-in-eclipse)

Q:

I am trying to get to the bottom of a threading problem in a web application. I am using Eclipse. The problem is when one thread is a certain stage in a very long winded piece of code, another thread can't go any further even though it should. It has the appearance of a blocked thread. Eclipse has marked it as "stepping". It makes no sense why it has stopped as there are no monitors, semaphores, sychnronization blocks anywhere near the code.

Can I use anything in eclipse to understand any reason why the thread looks like it is being blocked?

I can hit the pause button, which suspends the thread. When I click the resume button, it goes back to "stepping", but can't complete this simple line of code:

long beginTime = System.currentTimeMillis();

A:

In Eclipse, marking a thread as "Stepping" means that you are stepping through the code, either with step-in or (more likely) step-over, and the thread is blocked somehow either doing some IO, waiting for a synchronization lock, calling wait(), etc.. Eclipse is not controlling the thread and is waiting for the thread to move past the block before it can continue with the "stepping". Once Eclipse has control back of the thread at the next line, your thread is changed to being "Suspended".

For example, while debugging the following test code:

// i put a break point here

System.out.println("Foo");

Object lock = new Object();

synchronized (lock) {

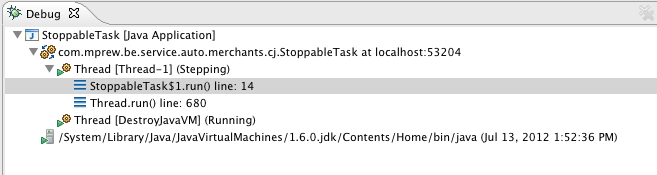
// when I step over this line, my thread is labeled as "Stepping"

lock.wait();

}

System.out.println("Boo");

Once I step over a method which blocks the thread, the thread gets labeled as "Stepping".



Best example: FizzBuzz problem

**import** java.util.function.IntConsumer;

**public** **class** FizzBuzz {

**private** **int** n;

**private** **int** counter = 1;

**public** FizzBuzz(**int** n) {

**this**.n = n;

}

// printFizz.run() outputs "fizz".

**public** **synchronized** **void** fizz(Runnable printFizz) **throws** InterruptedException {

**while**(counter <= n){

**while**(counter <= n && (counter % 3 != 0 || counter % 5 == 0)){

wait();

}

**if**(counter > n) **break**;

printFizz.run();

++counter;

notifyAll();

}

}

// printBuzz.run() outputs "buzz".

**public** **synchronized** **void** buzz(Runnable printBuzz) **throws** InterruptedException {

**while**(counter <= n){

**while**(counter <= n && (counter % 5 != 0 || counter % 3 == 0)){

wait();

}

**if**(counter > n) **break**;

printBuzz.run();

++counter;

notifyAll();

}

}

// printFizzBuzz.run() outputs "fizzbuzz".

**public** **synchronized** **void** fizzbuzz(Runnable printFizzBuzz) **throws** InterruptedException {

**while**(counter <= n){

**while**(counter <= n && !(counter % 3 == 0 && counter % 5 == 0)){

wait();

}

**if**(counter > n) **break**;

printFizzBuzz.run();

++counter;

notifyAll();

}

}

// printNumber.accept(x) outputs "x", where x is an integer.

**public** **synchronized** **void** number(IntConsumer printNumber) **throws** InterruptedException {

**while**(counter <= n){

**while**(counter <= n && (counter % 3 == 0 || counter % 5 == 0)){

wait();

}

**if**(counter > n) **break**;

printNumber.accept(counter);

++counter;

notifyAll();

}

}

**public** **static** **void** main(String[] args) {

FizzBuzz fizzBuzz = **new** FizzBuzz(15);

Runnable printFizz = () -> System.***out***.println("fizz");

Runnable printBuzz = () -> System.***out***.println("buzz");

Runnable printFizzBuzz = () -> System.***out***.println("fizzbuzz");

IntConsumer printNumber = number -> System.***out***.println(number);

Thread threadA = **new** Thread(() -> {

**try** {

fizzBuzz.fizz(printFizz);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

});

Thread threadB = **new** Thread(() -> {

**try** {

fizzBuzz.buzz(printBuzz);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

});

Thread threadC = **new** Thread(() -> {

**try** {

fizzBuzz.fizzbuzz(printFizzBuzz);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

});

Thread threadD = **new** Thread(() -> {

**try** {

fizzBuzz.number(printNumber);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

});

threadA.start();

threadB.start();

threadC.start();

threadD.start();

}

}