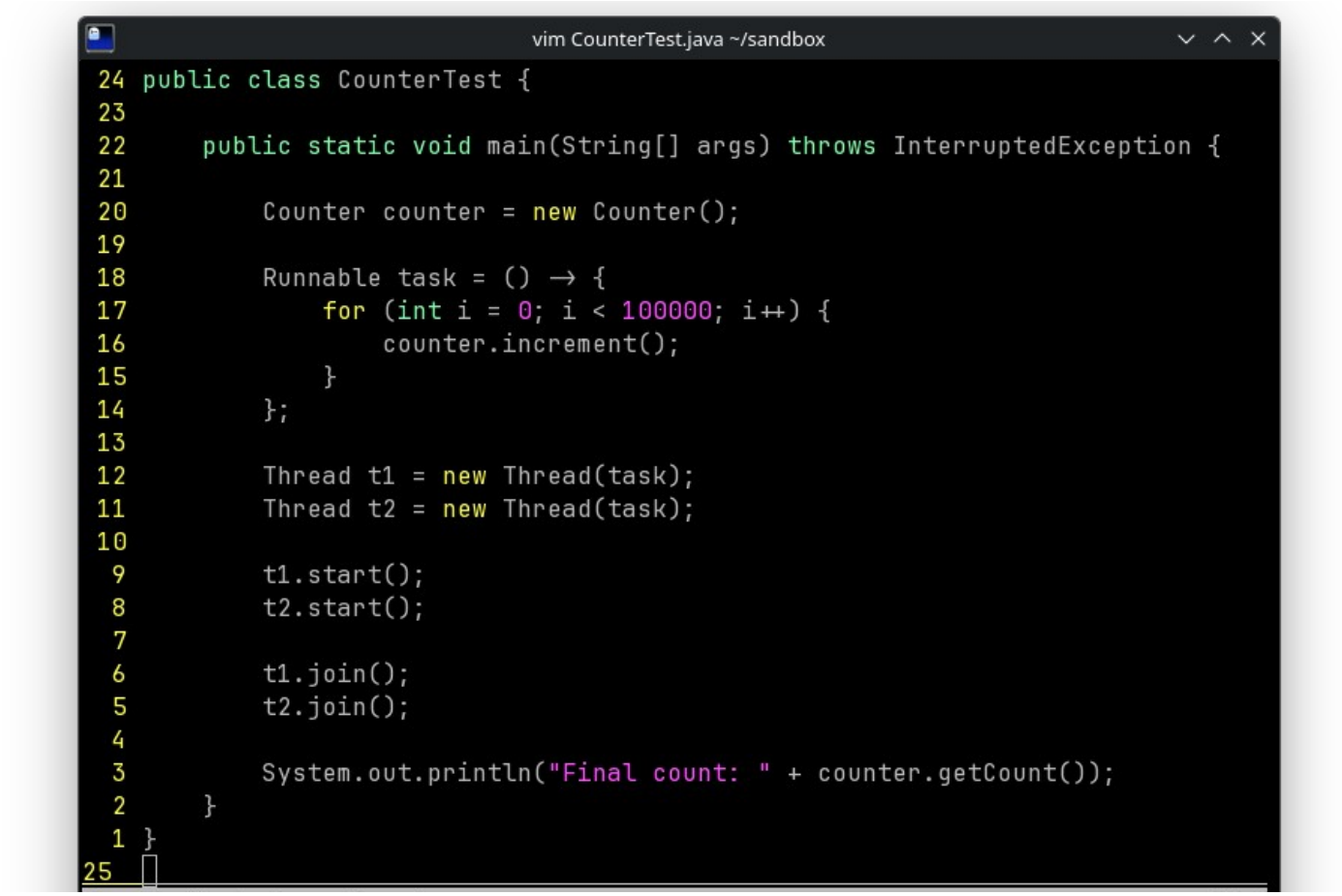
Testing Threads in Linux with Jshell

Part 1 – Simple Threading

A. In your Linux environment (either the Xubuntu VM or your own Linux box), create the following Java class file Counter.java. While it is possible to do this in Eclipse, I recommend doing this in a simple text editor. I am using vim on my Arch Linux box, so yours may appear different than mine.

B. Now create the main method to run the counter as a thread.



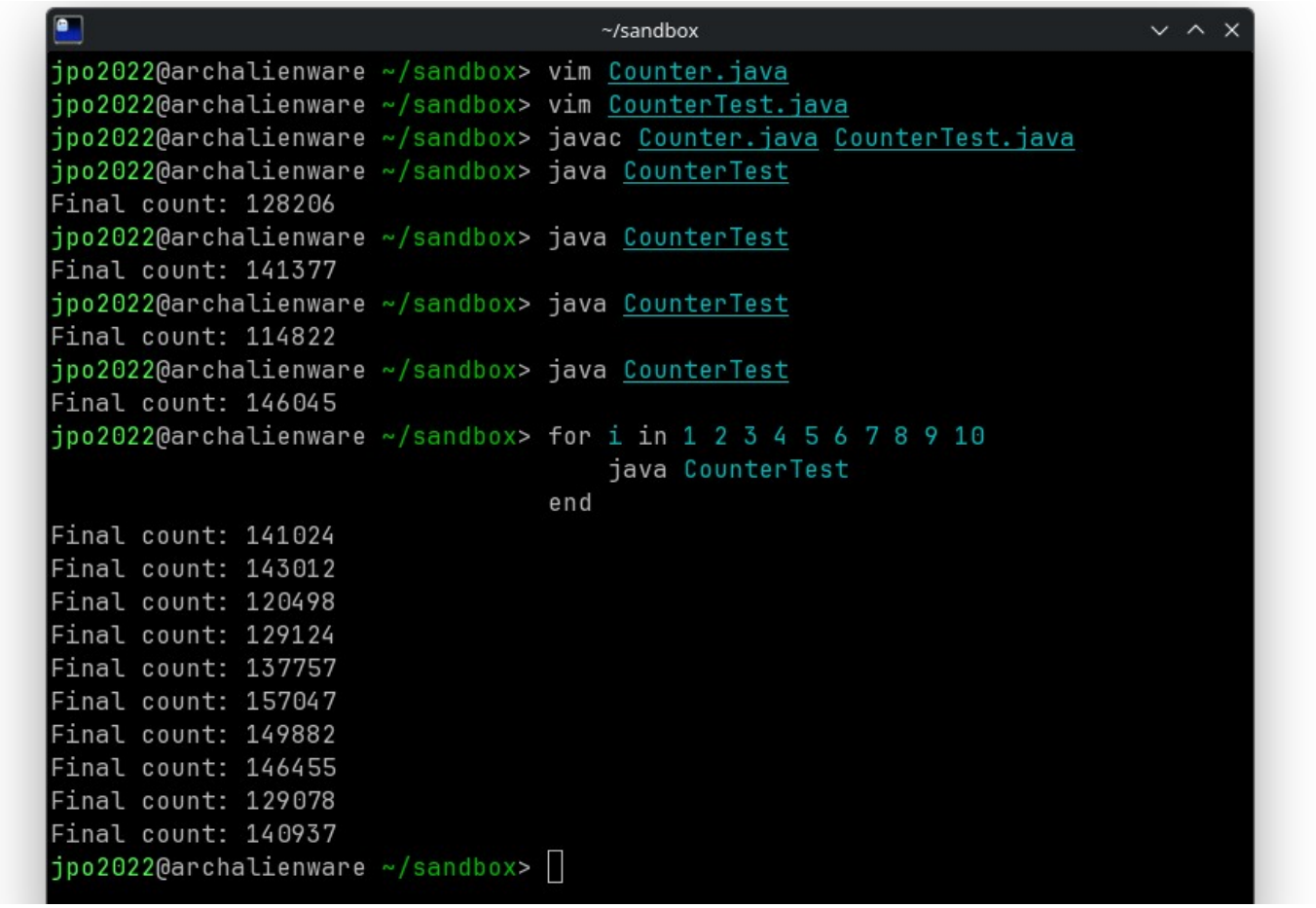
C. On the command line, compile and run the class using javac and java multiple times like shown in the image below (including writing the bash script).

Note: I am using the script syntax for fish in a ghostty terminal, which is not bash compliant.

- If you are in Xubuntu, this should work:

```
for i in {1..10}; do java CounterTest; done
```

- If you are in another setup, you may have to find an alternate syntax for the script.
- You can also write the script into file instead of doing it directly on the command line, but be sure to include the file or a screenshot so I can see your script code.



Answer these questions in a separate document (docx, odt, or a text file is okay), e. g., lastnametaskw07.odt.

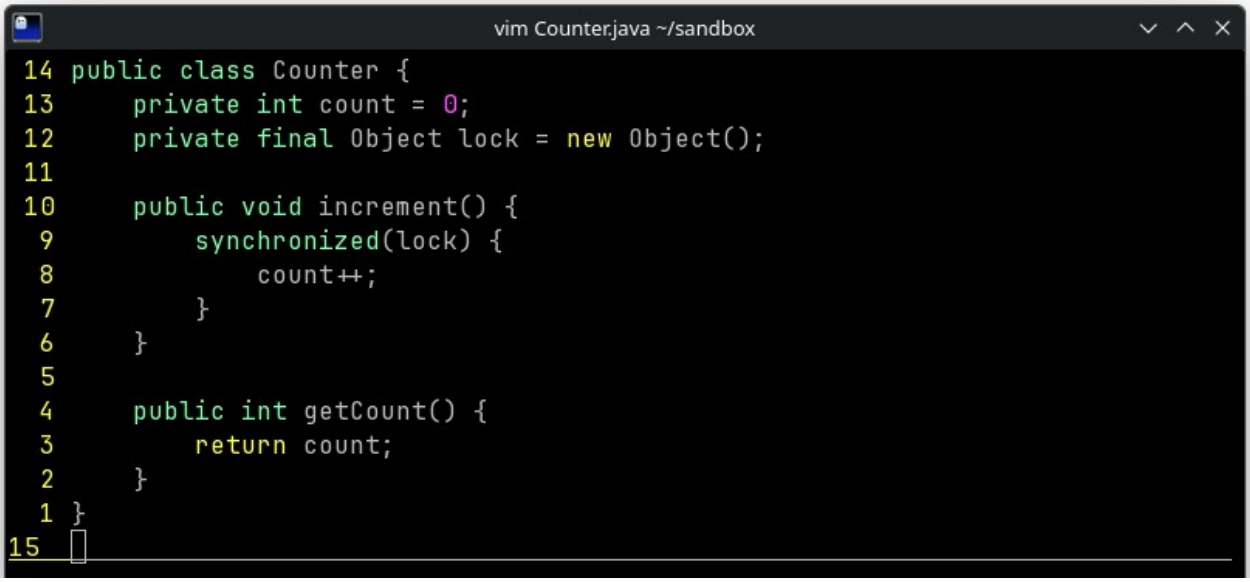
- What value should be the result?
- Why are you not always getting the correct result?
- Why are your results different than mine?
- List all the programming concepts being demonstrated by these classes. There are several here, not just the one we are currently learning this week—think back to everything we have learned about Java)

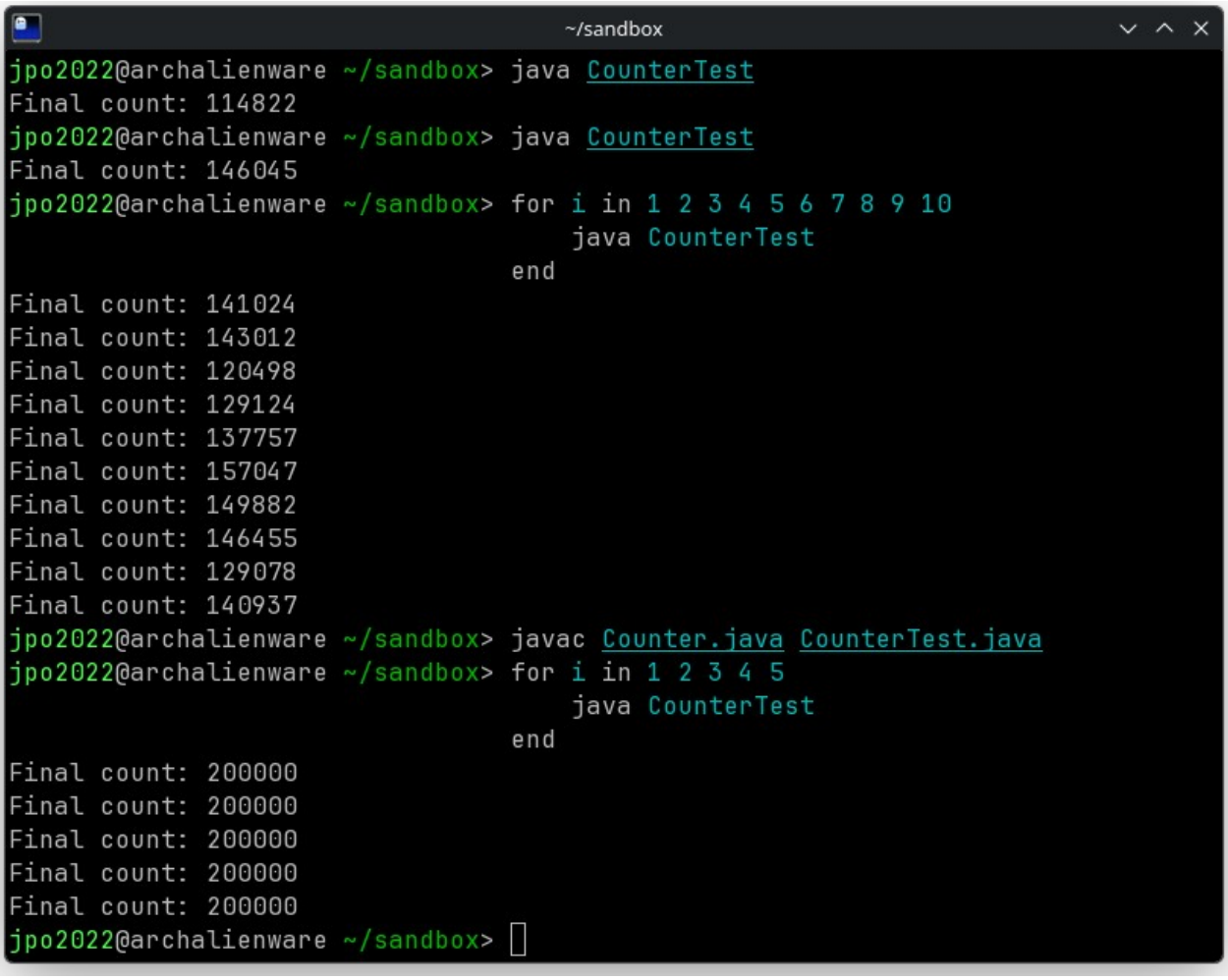
Take screenshots of your code and the results (like my three screenshots), to show you have completed this step, and place your answers document and the screenshots in a “taskw07” folder in your work repository. Please name your screenshot files so I can easily identify them, e. g., p1A.png.

Part 2 – Fixing the Problem.

A. Go to increment method and add synchronized: `public synchronized void increment()`

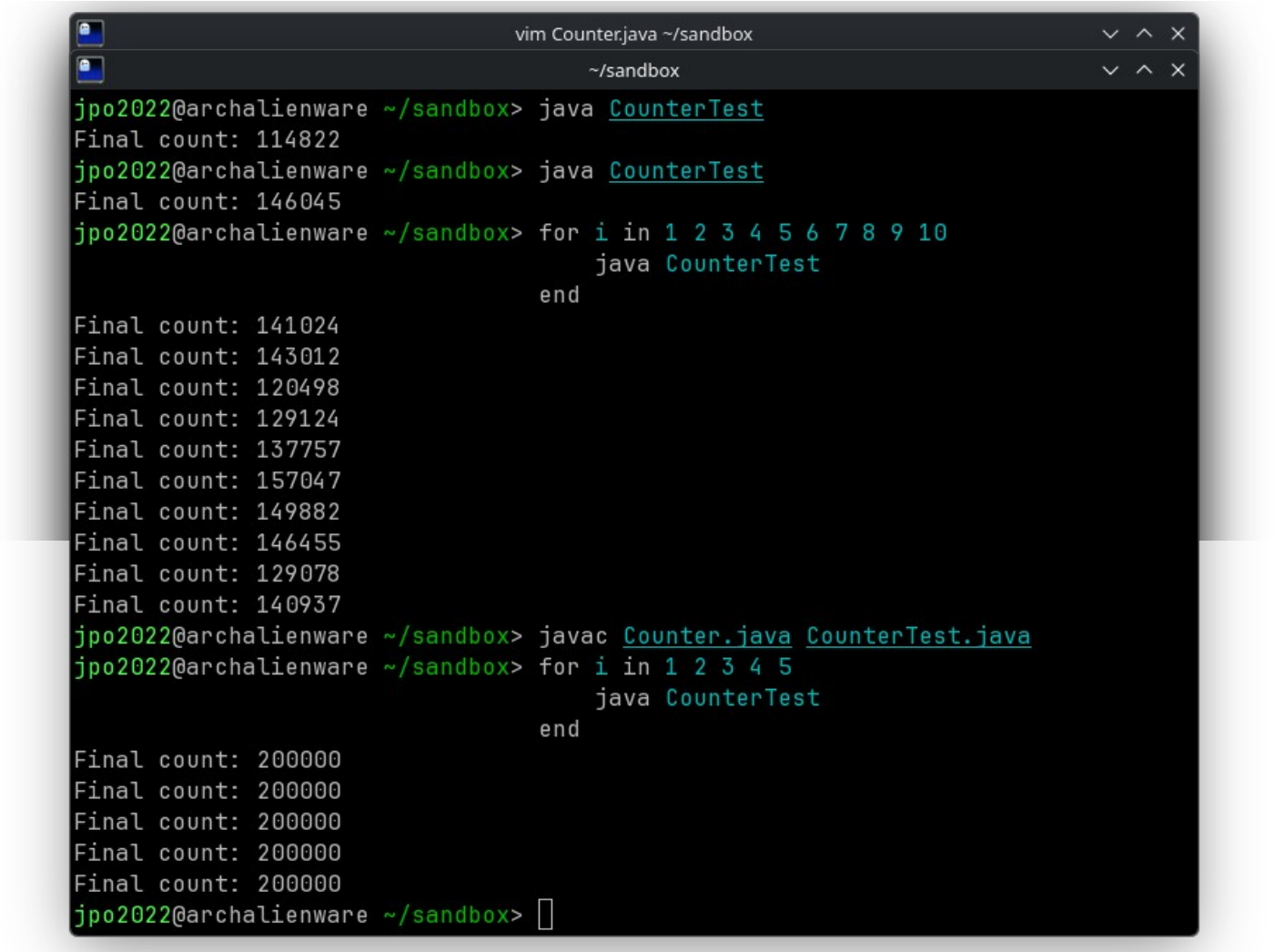
Recompile and run the loop script again. Takes screenshots of your edited code and the results and add them to your taskw07 folder.





B. Now go back into your code and remove “synchronized” and adjust code like the image below. Compile and run the code with the loop script again. Take screenshots of the code and your results, and answer the following questions in your answers document.

- What is the key word “synchronized” in Java
- Explain the similarities and differences between the two approaches we just used to solve the problem.
- What are the advantages and disadvantages of the two approaches?



Part 3 – Using JShell (Interactive Exploration)

Lets explore the issue with JShell. The site https://www.codejava.net/java-core/tools/java-9-the-java-shell-jshell-tutorial provided a good tutorial on it’s purpose and how to get started.

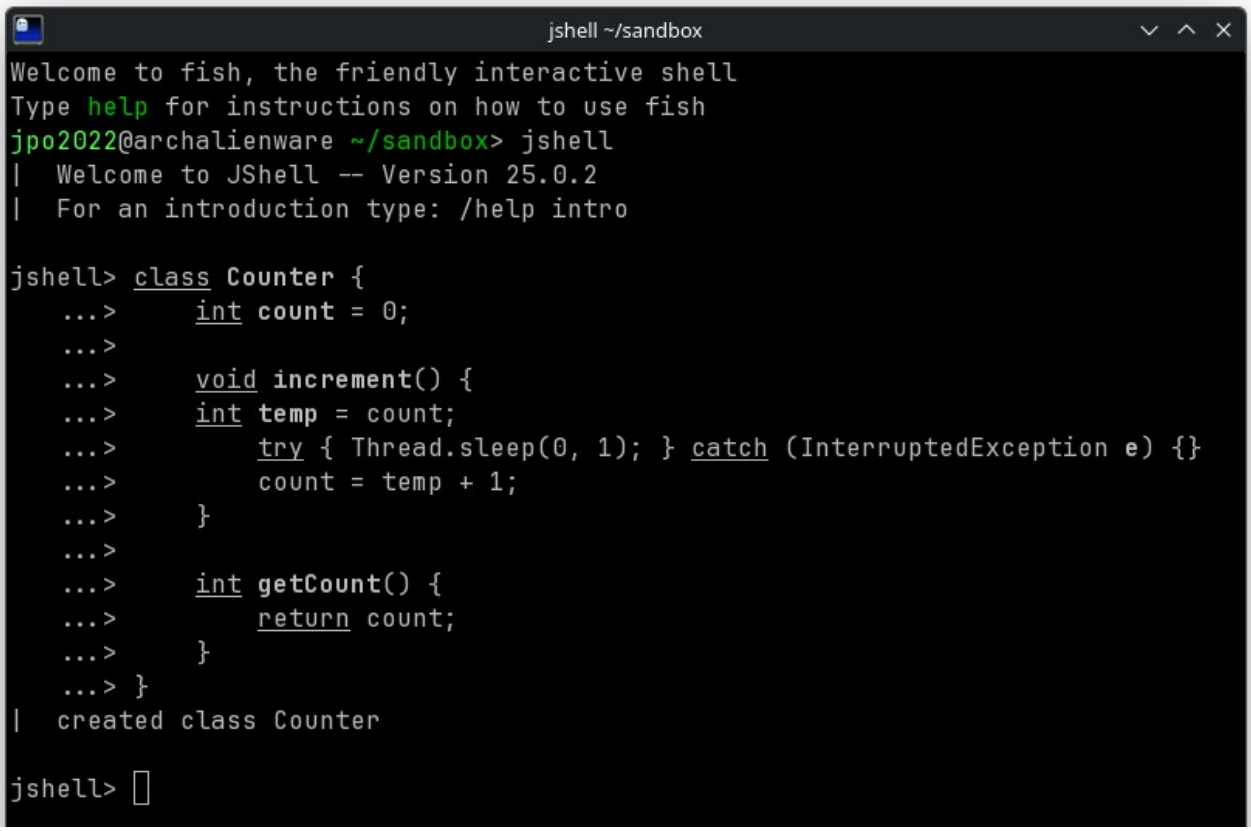
A. On the Linux command line, run jshell. For the first block of code, we need to set up the Counter class. You should be able to copy and paste the code from this PDF, but you may have to be careful. For Part A, you do not need to take screenshots because you will have similar screenshots for Part B after correcting the code.

```
class Counter {
```

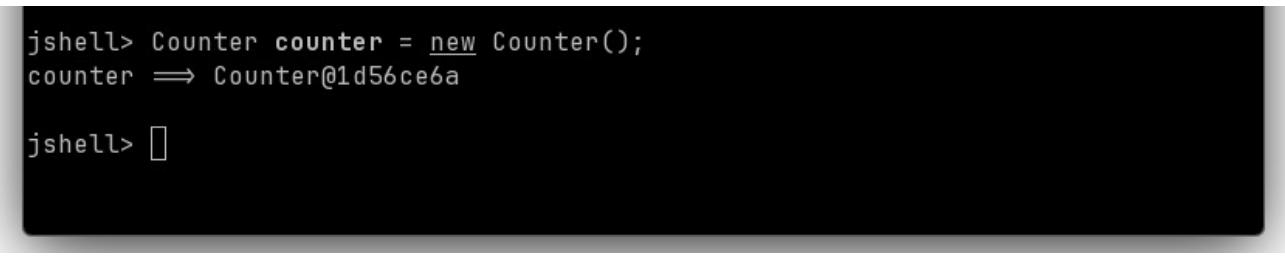
```
    int count = 0;

    void increment() {
      int temp = count;
      try { Thread.sleep(0, 1); } catch (InterruptedException e) {}
      count = temp + 1;
    }

    int getCount() {
        return count;
    }
}
```



Now instantiate a counter: `Counter counter = new Counter();`



Now set up the task:

```
Runnable task = () -> {
    for (int i = 0; i < 100000; i++) {
        counter.increment();
    }
};
```

```
jshell> Runnable task = () -> {
   ...>     for (int i = 0; i < 100000; i++) {
   ...>         counter.increment();
   ...>     }
   ...> };
task ==> $Lambda/0x0000000080048630@5010be6

jshell>
```

Now create, start, and join the threads:

```
Thread t1 = new Thread(task);
Thread t2 = new Thread(task);

t1.start();
t2.start();

t1.join();
t2.join();
```

```
jshell> Thread t1 = new Thread(task);
   ...> Thread t2 = new Thread(task);
   ...>
   ...> t1.start();
   ...> t2.start();
   ...>
   ...> t1.join();
   ...> t2.join();
t1 ==> Thread[#38,Thread-0,5,main]
t2 ==> Thread[#39,Thread-1,5,main]

jshell>
```

And get the result: `counter.getCount();`

```
jshell> counter.getCount();
$10 ==> 103460

jshell>
```

B. As you can see, the results show we have the same issue as before. Now, work out a solution using the AtomicInteger instead of the primitive int, and in a step-wise manner like we just walked through (i.e., do not code it all out and paste it in—break it down into steps like we did), take screenshots or a video that shows you walking through it. Hint: You may need to import the Concurrent package while inside JShell.

The screenshot(s) or video does not have to be perfect. The goal is to demonstrate yourself walking through the steps.

Submission: Make sure to submit your document with all the answers and screenshots in your 3970 work repository.