**Source Code**

Runner.java

// Thread class

public class Runner implements Runnable {

final private String direction;

public Runner(String direction) {

this.direction = direction;

}

@Override

public void run() {

if (this.direction.equals("up")) {

System.out.println("Counting up\n");

for (int i = 1; i < 21; i++) {

System.out.println(i);

}

}

else if (this.direction.equals("down")) {

System.out.println("\nCounting down\n");

for (int i = 20; i > 0; i--) {

System.out.println(i);

}

}

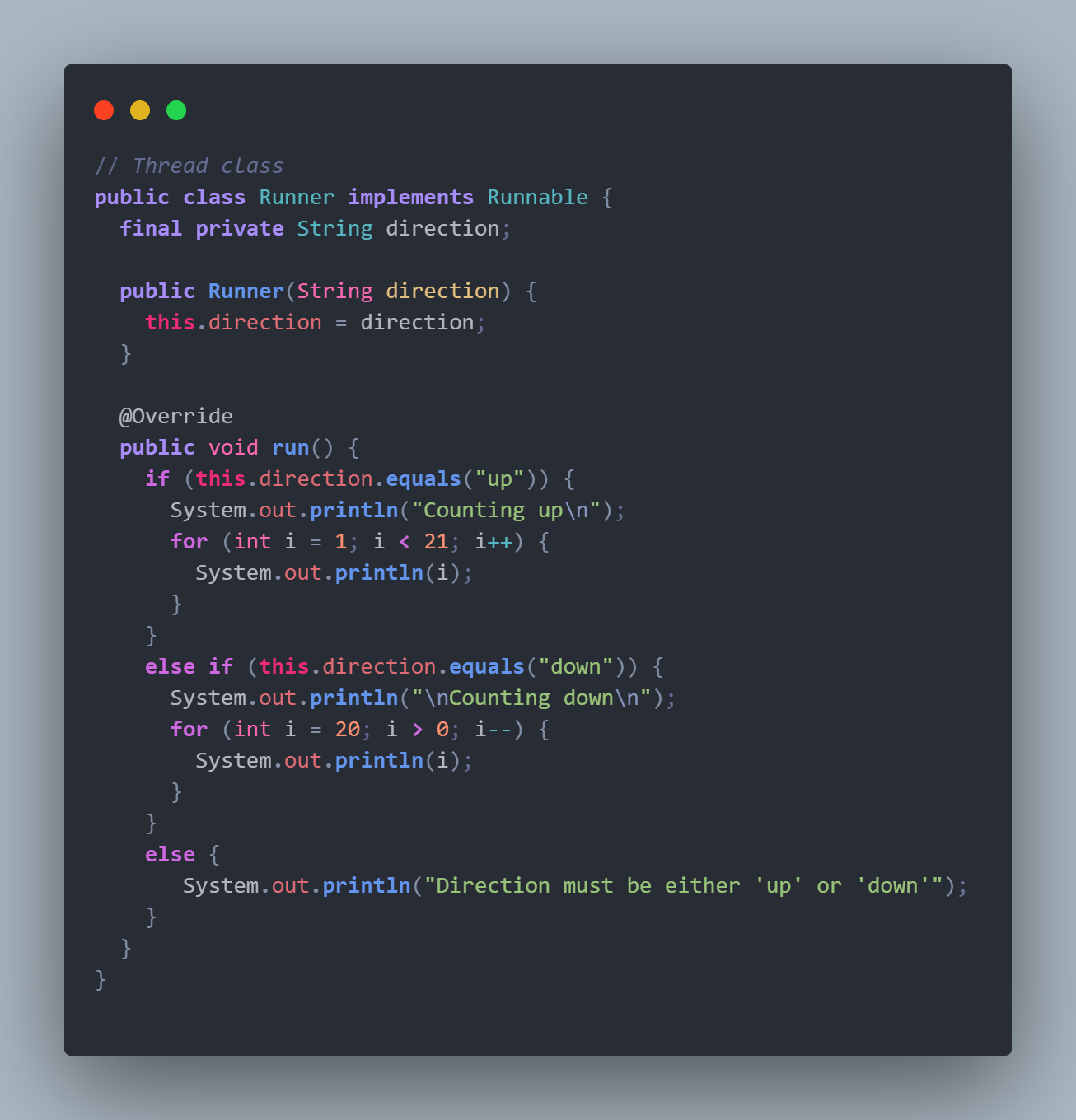
else {

System.out.println("Direction must be either 'up' or 'down'");

}

}

}



App.java

public class App {

public static void main(String[] args) {

Thread t1 = new Thread(new Runner("up"));

Thread t2 = new Thread(new Runner("down"));

try {

t1.start();

t1.join();

t2.start();

t2.join();

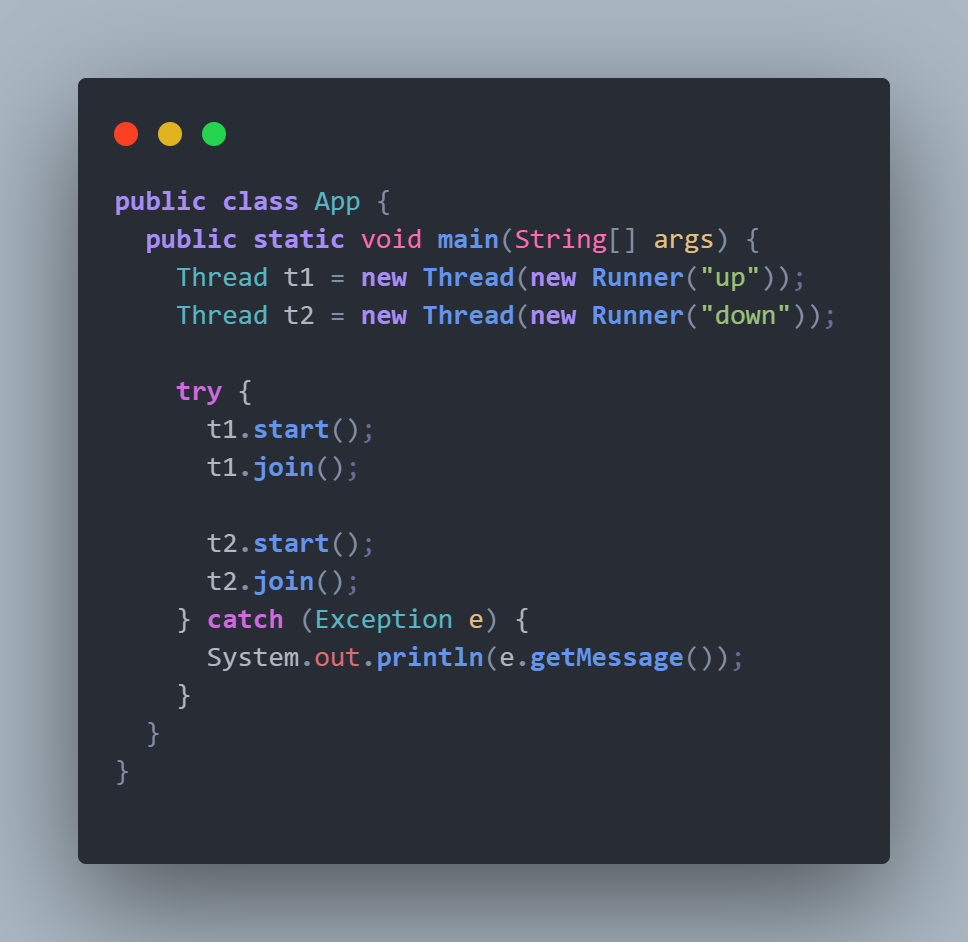
} catch (Exception e) {

System.out.println(e.getMessage());

}

}

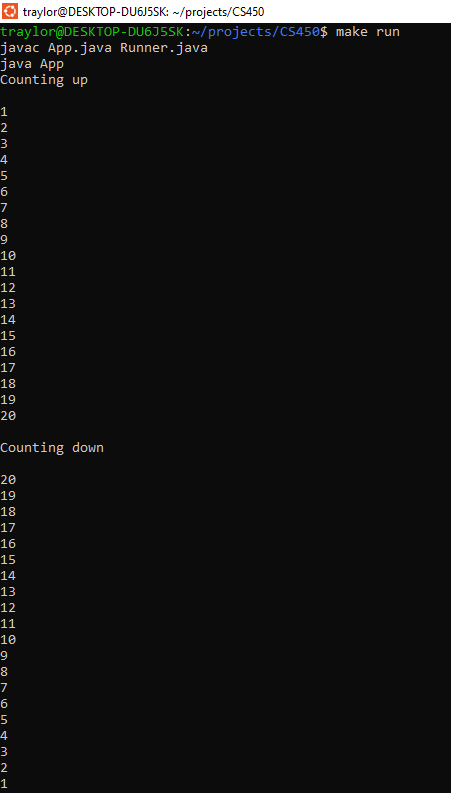
}



**Analysis**

This application does not contain many vulnerabilities since it does not utilize user input nor does it access memory. The only issue that can arise is from performance. However, since the threads do not access any shared resources, the performance of the application should not be affected as much. The program is able to utilize two threads that operate using a parameter called direction. This can be an issue since it does not properly handle exceptions if an incorrect parameter is passed. This can be seen as a vulnerability, however, a message is shown if a value or than “up” or “down” is passed.

**Output**

****