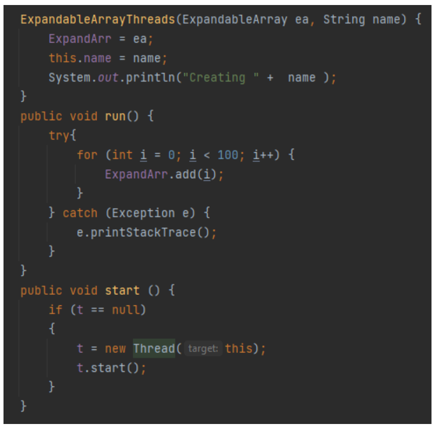
LAB04 COMP3712

Fun With Threads

Timothy Foong
Foon0020@flinders.edu.au

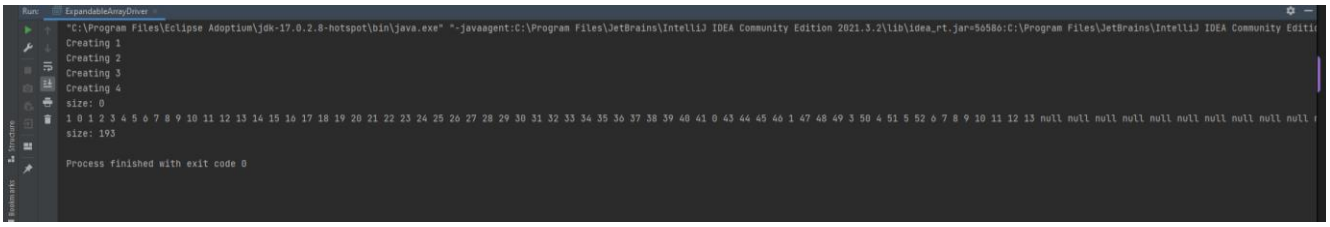
Checkpoint L4.1

I created the Expandable Array Threads class which extends the Thread class and copies some of the code from the ExpandableArray class. This class runs a for loop till 100 which adds the counted-up number to the ExpandableArray object.

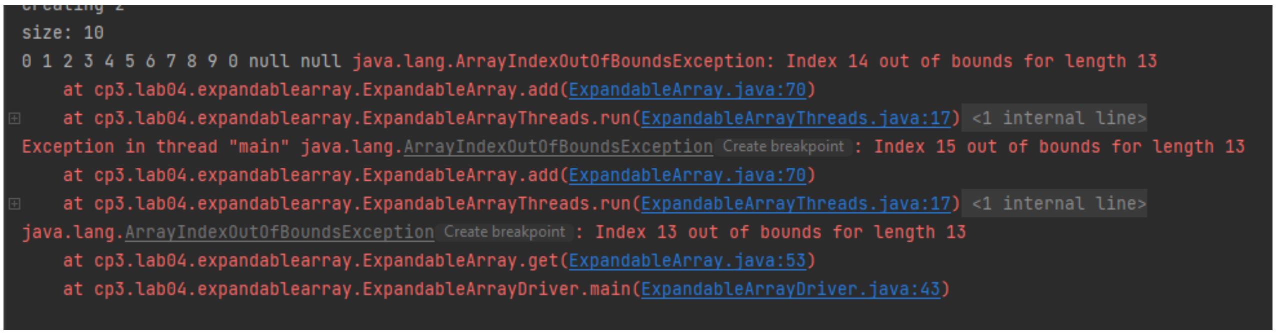


```java
ExpandableArrayThreads(ExpandableArray ea, String name) {
    ExpandArr = ea;
    this.name = name;
    System.out.println("Creating " +  name );
}
public void run() {
    try{
        for (int i = 0; i < 100; i++) {
            ExpandArr.add(i);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
public void start () {
    if (t == null)
    {
        t = new Thread( target: this);
        t.start();
    }
}
```

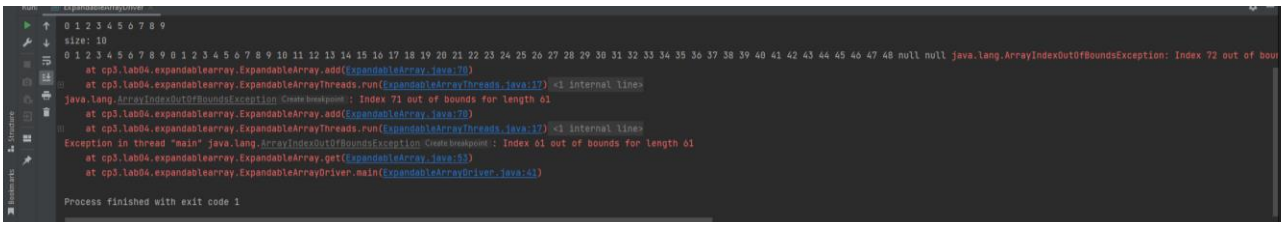
When this class runs, but does not crash, it adds null integers into the array. When it runs but crashes it will give multiple out of bounds errors for the array. Ill be completely honest I have no idea what causes it to add a null integer into the array, I think it may be because the add function checks the size of the object to add to the end of the array. Since multiple threads are accessing the array before it gets expanded, it calls the expand function multiple times, meaning that there are empty spaces between entries.



```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.3.2\lib\idea_rt.jar=56586:C:\Program Files\JetBrains\IntelliJ IDEA Community Editi
Creating 1
Creating 2
Creating 3
Creating 4
size: 0
1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 0 43 44 45 46 1 47 48 49 3 50 4 51 5 52 6 7 8 9 10 11 12 13 null null null null null null null null null null
size: 193

Process finished with exit code 0
```



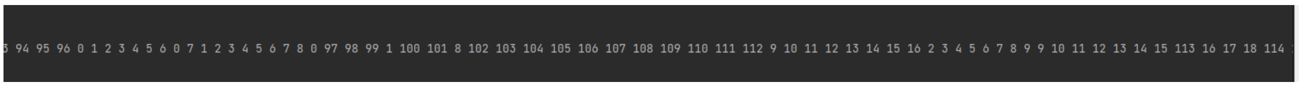
```
size: 10
0 1 2 3 4 5 6 7 8 9 0 null null java.lang.ArrayIndexOutOfBoundsException: Index 14 out of bounds for length 13
    at cp3.lab04.expandablearray.ExpandableArray.add(ExpandableArray.java:70)
    at cp3.lab04.expandablearray.ExpandableArrayThreads.run(ExpandableArrayThreads.java:17) <1 internal line>
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException Create breakpoint : Index 15 out of bounds for length 13
    at cp3.lab04.expandablearray.ExpandableArray.add(ExpandableArray.java:70)
    at cp3.lab04.expandablearray.ExpandableArrayThreads.run(ExpandableArrayThreads.java:17) <1 internal line>
java.lang.ArrayIndexOutOfBoundsException Create breakpoint : Index 13 out of bounds for length 13
    at cp3.lab04.expandablearray.ExpandableArray.get(ExpandableArray.java:53)
    at cp3.lab04.expandablearray.ExpandableArrayDriver.main(ExpandableArrayDriver.java:43)
```



```
0 1 2 3 4 5 6 7 8 9
size: 10
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 null null java.lang.ArrayIndexOutOfBoundsException: Index 72 out of bou
    at cp3.lab04.expandablearray.ExpandableArray.add(ExpandableArray.java:70)
    at cp3.lab04.expandablearray.ExpandableArrayThreads.run(ExpandableArrayThreads.java:17) <1 internal line>
java.lang.ArrayIndexOutOfBoundsException Create breakpoint : Index 71 out of bounds for length 61
    at cp3.lab04.expandablearray.ExpandableArray.add(ExpandableArray.java:70)
    at cp3.lab04.expandablearray.ExpandableArrayThreads.run(ExpandableArrayThreads.java:17) <1 internal line>
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException Create breakpoint : Index 61 out of bounds for length 61
    at cp3.lab04.expandablearray.ExpandableArray.get(ExpandableArray.java:53)
    at cp3.lab04.expandablearray.ExpandableArrayDriver.main(ExpandableArrayDriver.java:41)

Process finished with exit code 1
```

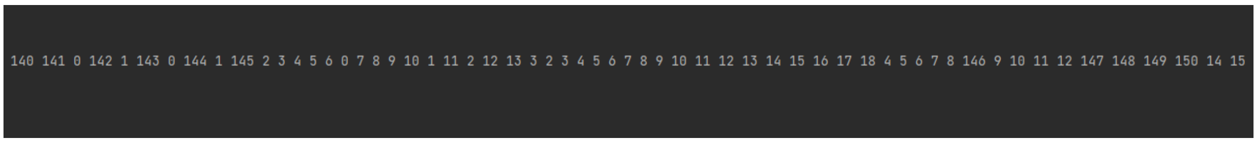
Checkpoint L4.2

To get threads working on the array, I used the ReadWriteLock on the expandable array to ensure only one thread could be writing to the array at once. The lock is locked at the start of the add function and the unlock function is added to the end of the function. This ensures that the threads will not be able to write to the array at the same time and thus avoids both OOB errors and null entry errors. Below are 2 examples of the lock working without issues.



```
3 94 95 96 0 1 2 3 4 5 6 0 7 1 2 3 4 5 6 7 8 0 97 98 99 1 100 101 8 102 103 104 105 106 107 108 109 110 111 112 9 10 11 12 13 14 15 16 2 3 4 5 6 7 8 9 9 10 11 12 13 14 15 113 16 17 18 114
```

Different output but still working.



```
140 141 0 142 1 143 0 144 1 145 2 3 4 5 6 0 7 8 9 10 1 11 2 12 13 3 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 4 5 6 7 8 146 9 10 11 12 147 148 149 150 14 15
```

Below is a code snippet showing where I lock the lock and unlock the lock.

```java
public void add(Object x)
{
    writeLock.lock();
    if (size == data.length) // too small
    {
        Object[] od = data;
        data = new Object[3 * (size + 1)];
        //data = new Object[size + 2];
        System.arraycopy(od, 0, data, 0, od.length);
    }
    data[size++] = x;
    writeLock.unlock();
}
```

Checkpoint L4.3

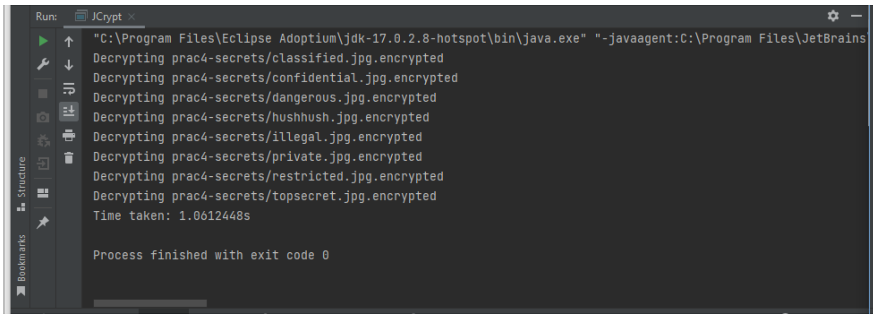
To implement multithreading on the JCrypt file I used the ExecutorService implementation. I used ExecutorService as I was looking up how to dynamically create a certain number of threads and it was the first option that came up. Using the opts.filenames.length variable I was able to ensure that the number of threads made was equal to the number of files being processed. I created a custom class called "JCryptThread" that would take in a JCryptUtil.Options variable and an int, it would save them to the class and run the process function. This ensured that the function was thread safe as it would not be accessing any shared resources. The original for loop in JCrypt will create a new JCryptThread object, assign the variables as the original did, and then submit it to the ExecutorService thread pool. Once they have all been submitted to the pool, the shutdown function is called which will allow all threads to finish running before terminating the pool. The if function at the bottom will ensure that all threads have been terminated before counting the time it took for the program to run. Below is my code, showing how I passed function arguments into the thread.

```java
ExecutorService pool = Executors.newFixedThreadPool(opts.filenames.length);
long starttime = System.nanoTime();

for (int i = 0; i < opts.filenames.length; i++) {
    JCryptThread t = new JCryptThread(opts,i);
    pool.submit(t);
}
```

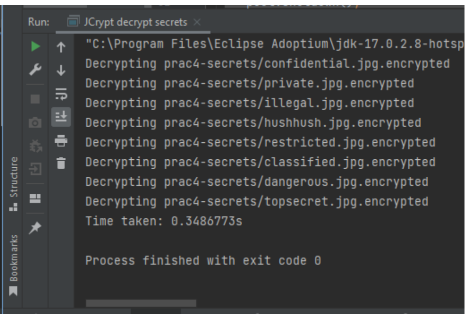
This is how those arguments are saved into the object to prevent multiple threads editing the same objects.

```java
JCryptUtil.Options opts;
int index;

public JCryptThread(JCryptUtil.Options opts, int index)
{
    this.opts = opts;
    this.index = index;
}
```

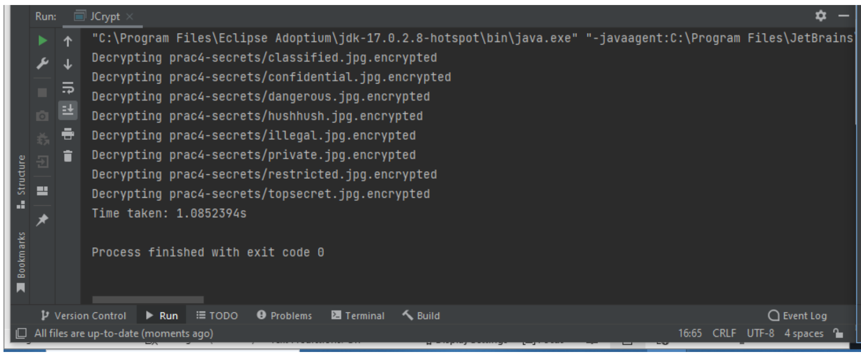
Below is a collection of screenshots showing the speed difference between the original implementation, on the left, and the multithreading implementation, on the right. As can be seen it is much faster, on average .7 seconds faster.



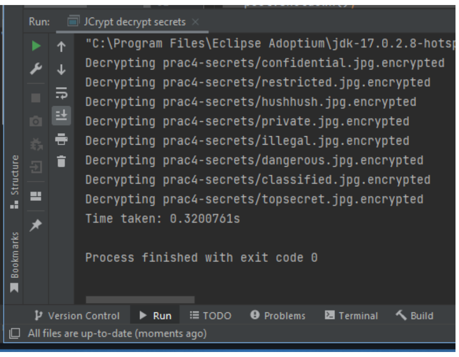
```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains
Decrypting prac4-secrets/classified.jpg.encrypted
Decrypting prac4-secrets/confidential.jpg.encrypted
Decrypting prac4-secrets/dangerous.jpg.encrypted
Decrypting prac4-secrets/hushhush.jpg.encrypted
Decrypting prac4-secrets/illegal.jpg.encrypted
Decrypting prac4-secrets/private.jpg.encrypted
Decrypting prac4-secrets/restricted.jpg.encrypted
Decrypting prac4-secrets/topsecret.jpg.encrypted
Time taken: 1.0612448s

Process finished with exit code 0
```



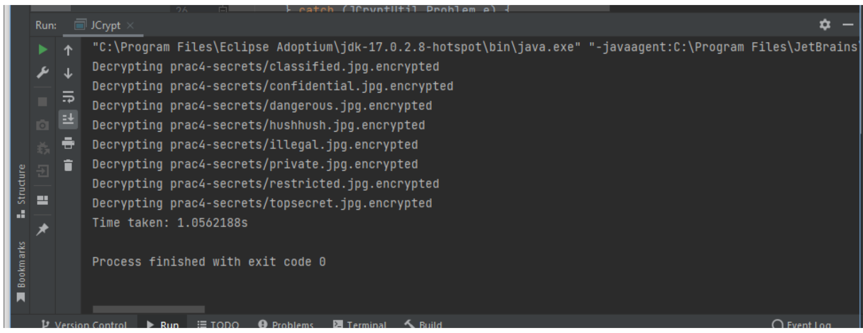
```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotsp
Decrypting prac4-secrets/confidential.jpg.encrypted
Decrypting prac4-secrets/private.jpg.encrypted
Decrypting prac4-secrets/illegal.jpg.encrypted
Decrypting prac4-secrets/hushhush.jpg.encrypted
Decrypting prac4-secrets/restricted.jpg.encrypted
Decrypting prac4-secrets/classified.jpg.encrypted
Decrypting prac4-secrets/dangerous.jpg.encrypted
Decrypting prac4-secrets/topsecret.jpg.encrypted
Time taken: 0.3486773s

Process finished with exit code 0
```


Run:
JCrypt
Version Control
Run
TODO
Problems
Terminal
Build
Event Log
All files are up-to-date (moments ago)
16:65 CRLF UTF-8 4 spaces

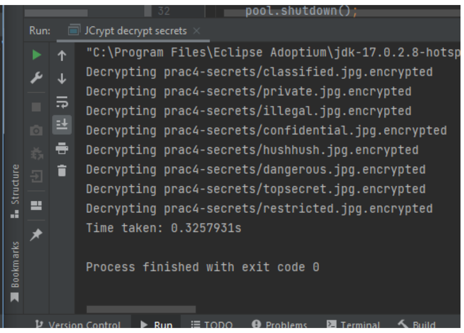
```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains
Decrypting prac4-secrets/classified.jpg.encrypted
Decrypting prac4-secrets/confidential.jpg.encrypted
Decrypting prac4-secrets/dangerous.jpg.encrypted
Decrypting prac4-secrets/hushhush.jpg.encrypted
Decrypting prac4-secrets/illegal.jpg.encrypted
Decrypting prac4-secrets/private.jpg.encrypted
Decrypting prac4-secrets/restricted.jpg.encrypted
Decrypting prac4-secrets/topsecret.jpg.encrypted
Time taken: 1.0852394s

Process finished with exit code 0
```



```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotsp
Decrypting prac4-secrets/confidential.jpg.encrypted
Decrypting prac4-secrets/restricted.jpg.encrypted
Decrypting prac4-secrets/hushhush.jpg.encrypted
Decrypting prac4-secrets/private.jpg.encrypted
Decrypting prac4-secrets/illegal.jpg.encrypted
Decrypting prac4-secrets/dangerous.jpg.encrypted
Decrypting prac4-secrets/classified.jpg.encrypted
Decrypting prac4-secrets/topsecret.jpg.encrypted
Time taken: 0.3200761s

Process finished with exit code 0
```



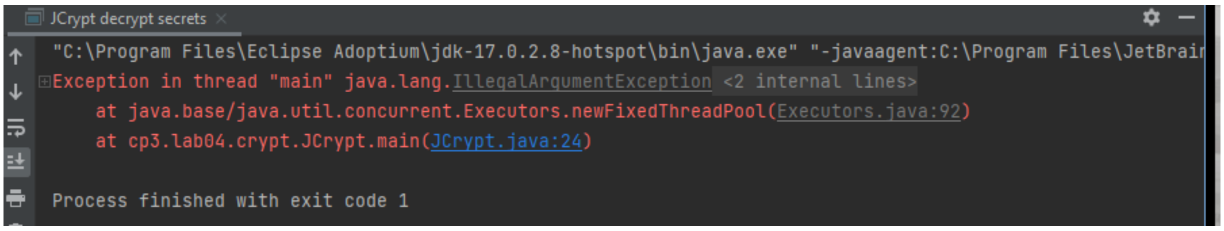
```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains
Decrypting prac4-secrets/classified.jpg.encrypted
Decrypting prac4-secrets/confidential.jpg.encrypted
Decrypting prac4-secrets/dangerous.jpg.encrypted
Decrypting prac4-secrets/hushhush.jpg.encrypted
Decrypting prac4-secrets/illegal.jpg.encrypted
Decrypting prac4-secrets/private.jpg.encrypted
Decrypting prac4-secrets/restricted.jpg.encrypted
Decrypting prac4-secrets/topsecret.jpg.encrypted
Time taken: 1.0562188s

Process finished with exit code 0
```



```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotsp
Decrypting prac4-secrets/classified.jpg.encrypted
Decrypting prac4-secrets/private.jpg.encrypted
Decrypting prac4-secrets/illegal.jpg.encrypted
Decrypting prac4-secrets/confidential.jpg.encrypted
Decrypting prac4-secrets/hushhush.jpg.encrypted
Decrypting prac4-secrets/dangerous.jpg.encrypted
Decrypting prac4-secrets/topsecret.jpg.encrypted
Decrypting prac4-secrets/restricted.jpg.encrypted
Time taken: 0.3257931s

Process finished with exit code 0
```

Checkpoint L4.4

As I have used the ExecutorService implementation for my code, it was very easy to alter the newFixedThreadPool function to take in the opts.threads variable instead of the opts.filenames.length variable. One snag I found was that if opts.threads was not initialised i.e. there was no -t argument in the command line, then it would crash. To fix this error I simply checked if the opts.threads variable was set to its default value, 0. If it was then I would set it to the opts.filenames.length to ensure that the error below did not occur.



```
JCrypt decrypt secrets
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrai
Exception in thread "main" java.lang.IllegalArgumentException <2 internal lines>
    at java.base/java.util.concurrent.Executors.newFixedThreadPool(Executors.java:92)
    at cp3.lab04.crypt.JCrypt.main(JCrypt.java:24)

Process finished with exit code 1
```

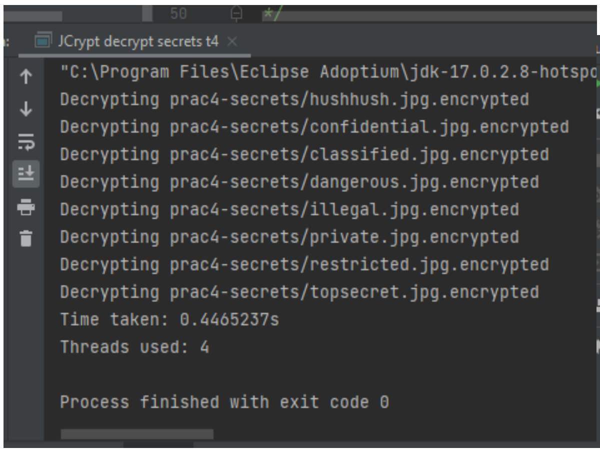
Swapping from this implementation.

```
ExecutorService pool = Executors.newFixedThreadPool(opts.filenames.length);
```

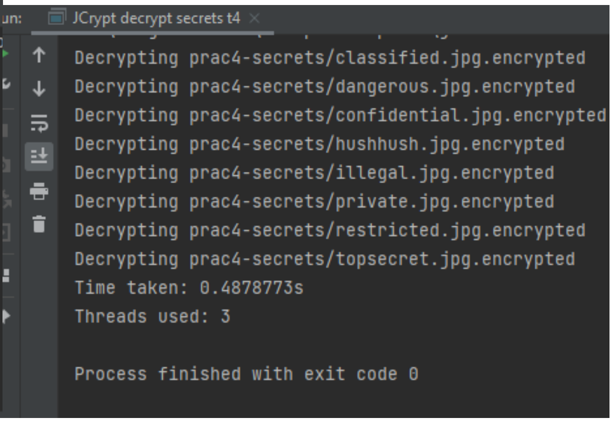
To this implementation

```
ExecutorService pool = Executors.newFixedThreadPool(opts.threads);
```

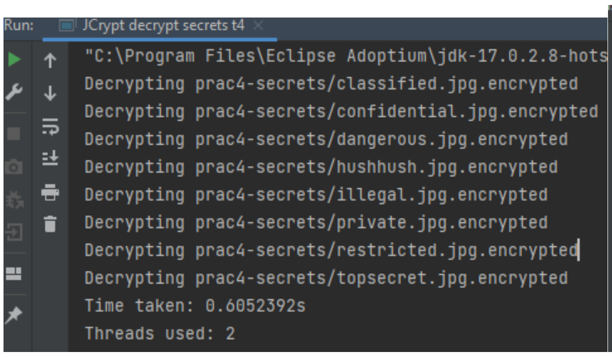
When using the -t 4 argument in the command line, the average time to finish goes from 0.3 to 0.4. This shows that the thread limiting is working as intended. Below is a collection of screenshots that prove the thread limiting is working.



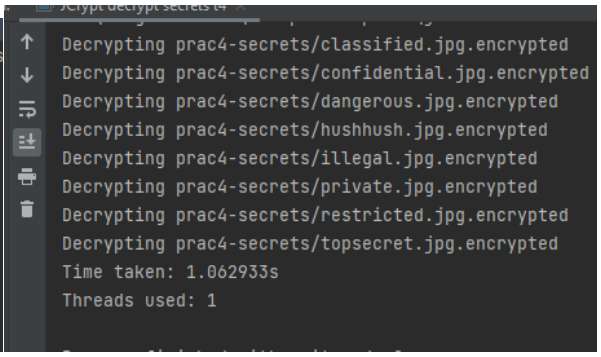
```
JCrypt decrypt secrets t4
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotspo
Decrypting prac4-secrets/hushhush.jpg.encrypted
Decrypting prac4-secrets/confidential.jpg.encrypted
Decrypting prac4-secrets/classified.jpg.encrypted
Decrypting prac4-secrets/dangerous.jpg.encrypted
Decrypting prac4-secrets/illegal.jpg.encrypted
Decrypting prac4-secrets/private.jpg.encrypted
Decrypting prac4-secrets/restricted.jpg.encrypted
Decrypting prac4-secrets/topsecret.jpg.encrypted
Time taken: 0.4465237s
Threads used: 4

Process finished with exit code 0
```



```
JCrypt decrypt secrets t4
Decrypting prac4-secrets/classified.jpg.encrypted
Decrypting prac4-secrets/dangerous.jpg.encrypted
Decrypting prac4-secrets/confidential.jpg.encrypted
Decrypting prac4-secrets/hushhush.jpg.encrypted
Decrypting prac4-secrets/illegal.jpg.encrypted
Decrypting prac4-secrets/private.jpg.encrypted
Decrypting prac4-secrets/restricted.jpg.encrypted
Decrypting prac4-secrets/topsecret.jpg.encrypted
Time taken: 0.4878773s
Threads used: 3

Process finished with exit code 0
```

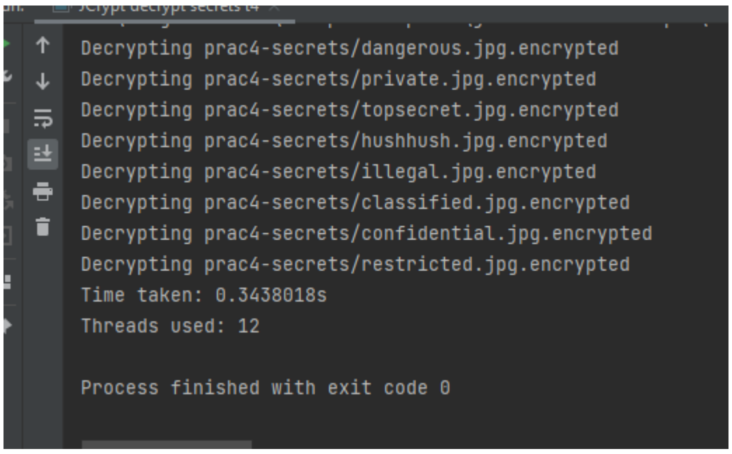


```
Run: JCrypt decrypt secrets t4
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hots
Decrypting prac4-secrets/classified.jpg.encrypted
Decrypting prac4-secrets/confidential.jpg.encrypted
Decrypting prac4-secrets/dangerous.jpg.encrypted
Decrypting prac4-secrets/hushhush.jpg.encrypted
Decrypting prac4-secrets/illegal.jpg.encrypted
Decrypting prac4-secrets/private.jpg.encrypted
Decrypting prac4-secrets/restricted.jpg.encrypted
Decrypting prac4-secrets/topsecret.jpg.encrypted
Time taken: 0.6052392s
Threads used: 2
```



```
Decrypting prac4-secrets/classified.jpg.encrypted
Decrypting prac4-secrets/confidential.jpg.encrypted
Decrypting prac4-secrets/dangerous.jpg.encrypted
Decrypting prac4-secrets/hushhush.jpg.encrypted
Decrypting prac4-secrets/illegal.jpg.encrypted
Decrypting prac4-secrets/private.jpg.encrypted
Decrypting prac4-secrets/restricted.jpg.encrypted
Decrypting prac4-secrets/topsecret.jpg.encrypted
Time taken: 1.062933s
Threads used: 1
```

As I was using thread pools from the start, and only using as many threads as files I was processing, this implementation was not any faster than my previous implementation. Below is a screenshot showing the program running with 12 threads, and as can be seen it is not any faster than my implementation.



```
Decrypting prac4-secrets/dangerous.jpg.encrypted
Decrypting prac4-secrets/private.jpg.encrypted
Decrypting prac4-secrets/topsecret.jpg.encrypted
Decrypting prac4-secrets/hushhush.jpg.encrypted
Decrypting prac4-secrets/illegal.jpg.encrypted
Decrypting prac4-secrets/classified.jpg.encrypted
Decrypting prac4-secrets/confidential.jpg.encrypted
Decrypting prac4-secrets/restricted.jpg.encrypted
Time taken: 0.3438018s
Threads used: 12

Process finished with exit code 0
```