**Analysis of the Memory Leak in the Application.**

**Introduction:**

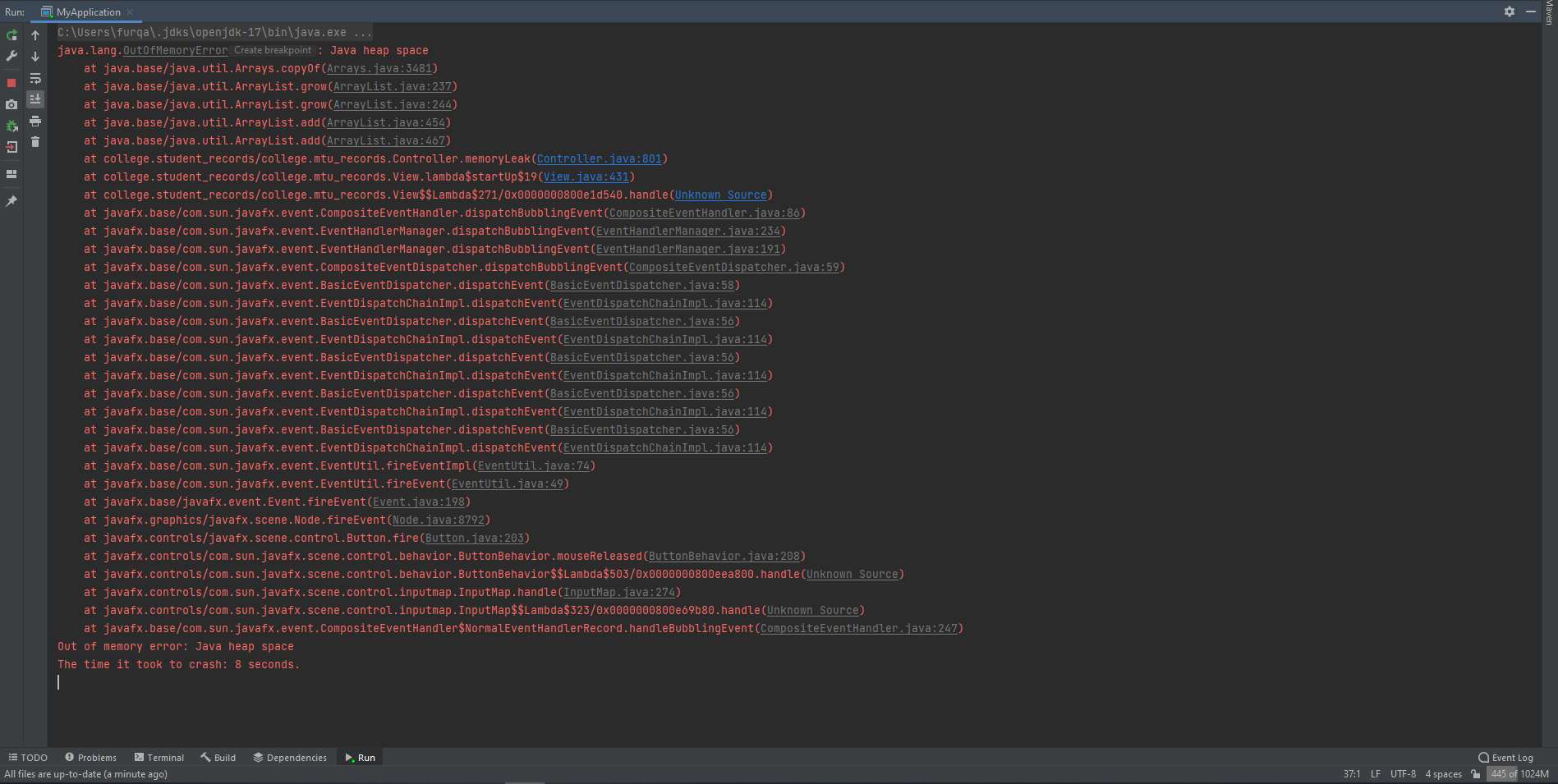
A memory leak happens when an application continues to use available memory until it runs out and crashes the program. Memory leaks can occur when there are unreleased objects in heap space that are not being used. In this report we purposely cause a memory leak by adding objects in an infinite loop to see how it affects the heap space.

**Setup:**

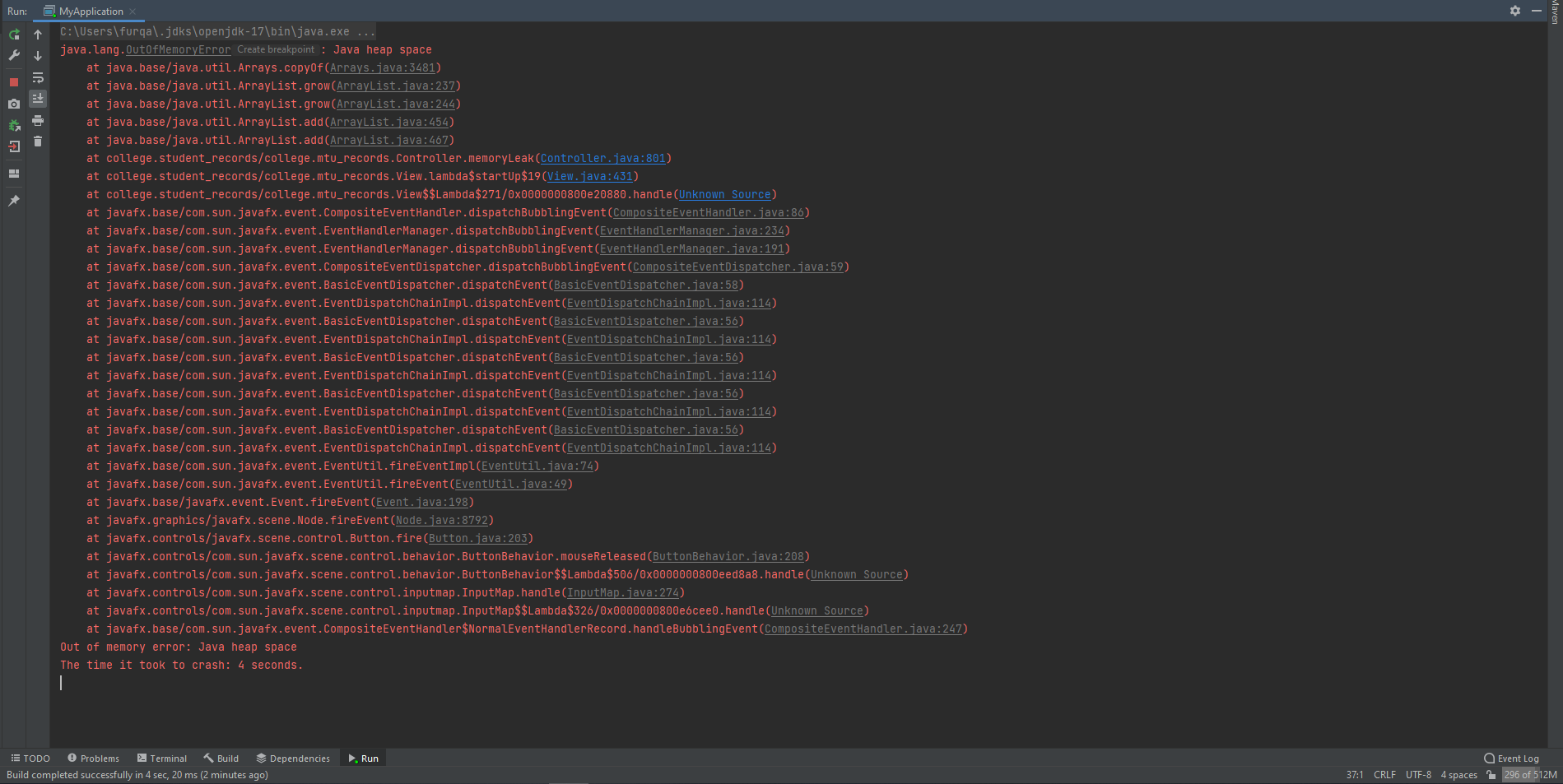
1. A button to cause the memory leak.
2. Heap size set to 1024 MB and the time to crash is recorded.
3. Heap size set to 512 MB and the time to crash is recorded.

**Results:**

1. With a heap size of 1024 MB, the application crashes in 8 seconds.



1. With a heap size of 512 MB, the application crashes in 4 seconds.



**Conclusion:**

As seen above with the results the time taken for 512 MB is half of what it takes for when it is 1024 MB. This is because the lower the memory the faster the infinite loop crashes the application. The higher the allocated memory the longer it takes for the application to crash as the application has more space to store these objects. However, when testing it is not always the case that the result is half in terms of time. Sometimes the result was much closer with only a difference of 2 seconds. This might happen because before causing the memory to leak the application could have been taking more memory than usual which caused the difference to be much less. The best way I found to test was to close all background processes and restart the IDE, this ensured that for both allocated memories the application started with taking the minimum amount of memory.

**Report By Furqan Ali | R00224511**