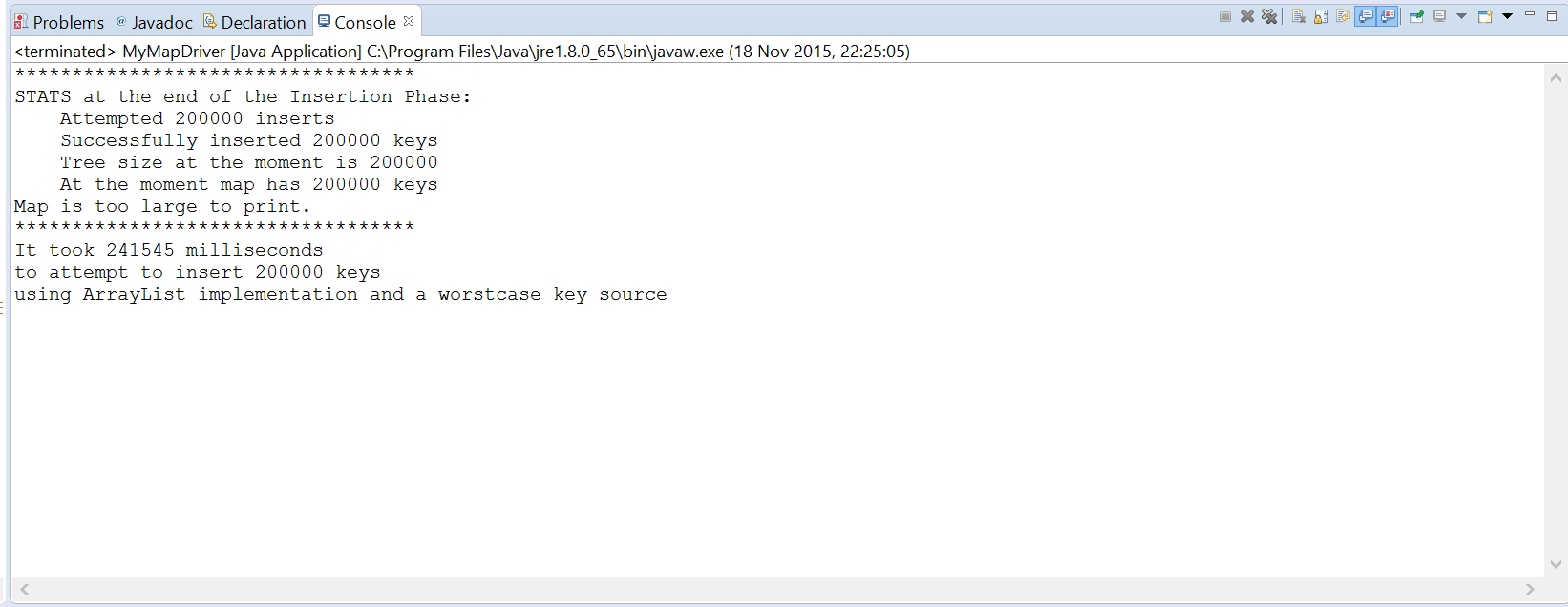
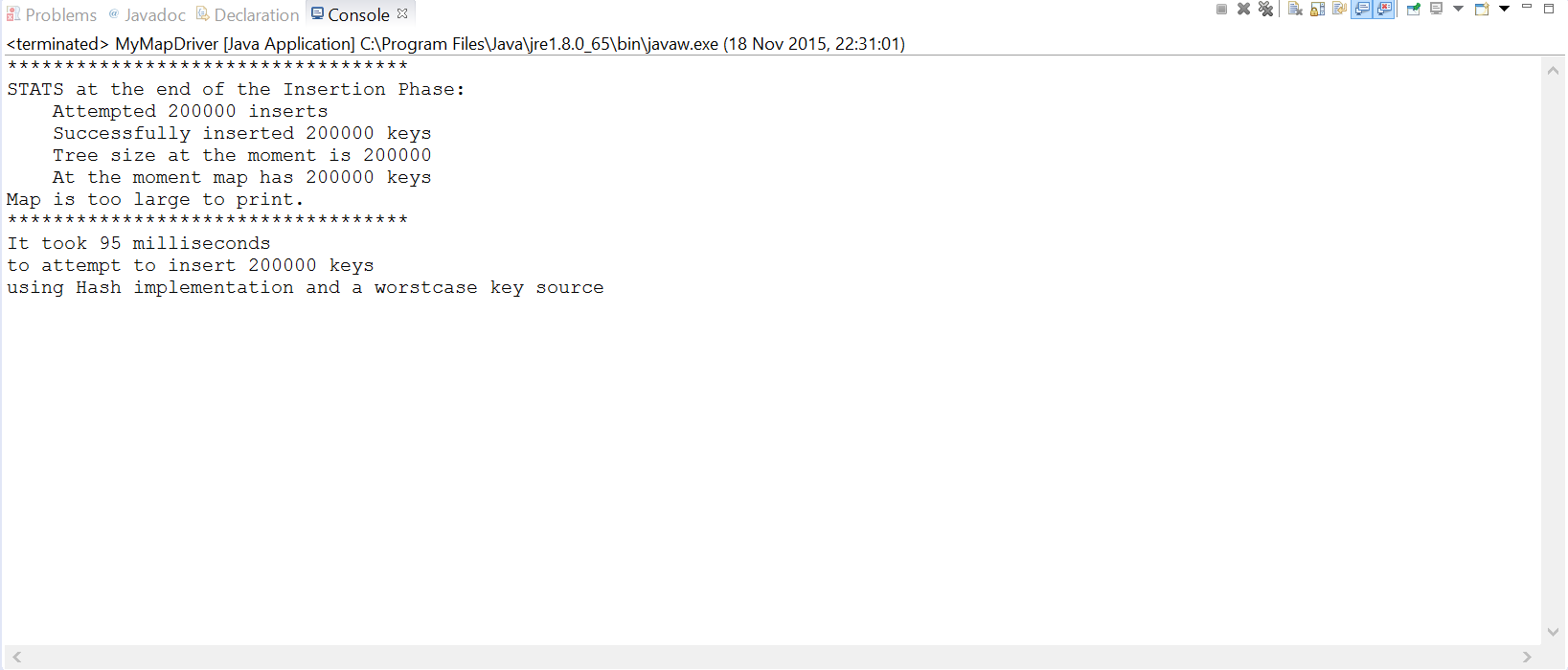
**ANALYSIS FOR LAB 5:**

**“Worst case” Scenario:**

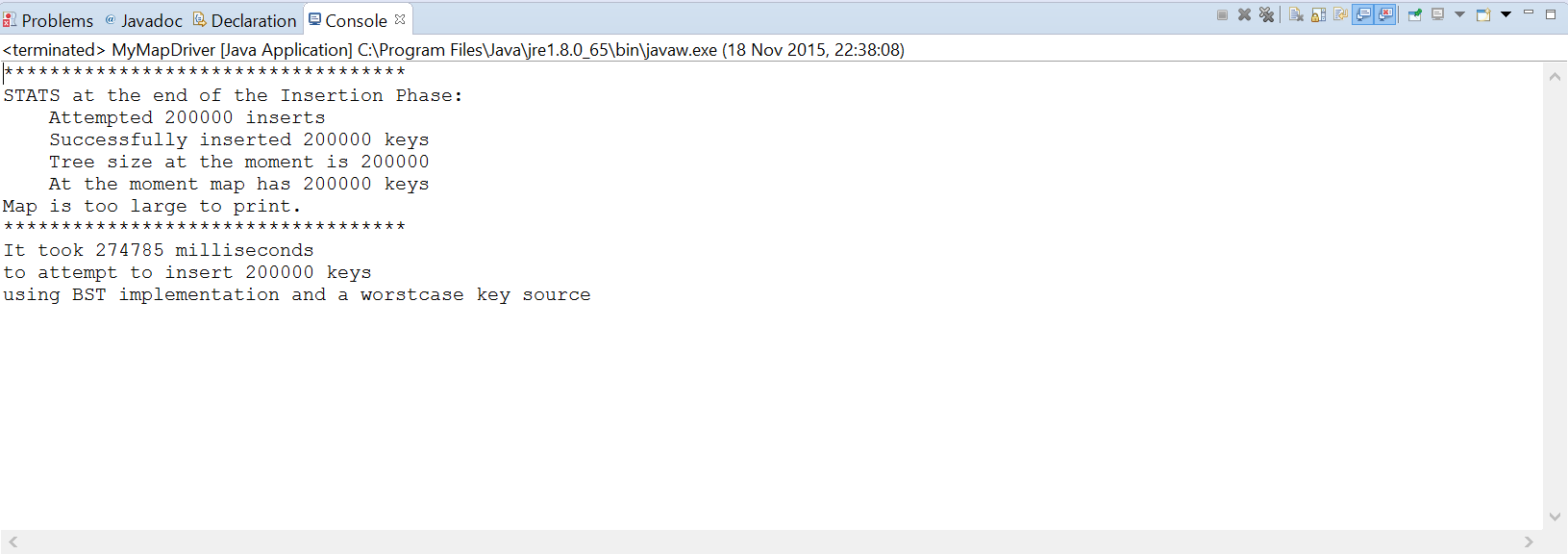
|  |  |
| --- | --- |
| For Array List | |
| Number of Keys | Run time/milliseconds |
| 3125 | 29 |
| 6250 | 98 |
| 12500 | 400 |
| 25000 | 1504 |
| 50000 | 7080 |
| 100000 | 50488 |
| 200000 | 241545 |



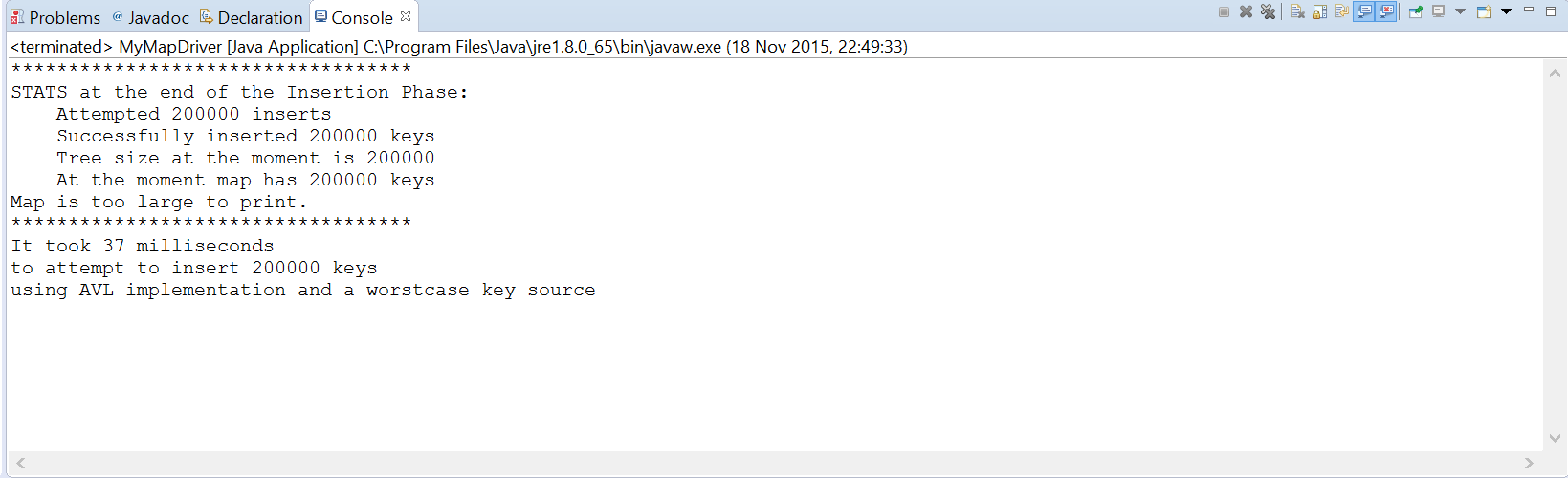
|  |  |
| --- | --- |
| For Hash Maps | |
| Number of Keys | Run time/milliseconds |
| 3125 | 1 |
| 6250 | 3 |
| 12500 | 6 |
| 25000 | 18 |
| 50000 | 25 |
| 100000 | 55 |
| 200000 | 95 |



|  |  |
| --- | --- |
| For BSTs | |
| Number of Keys | Run time/milliseconds |
| 3125 | 49 |
| 6250 | 166 |
| 12500 | 838 |
| 25000 | 2911 |
| 50000 | 11963 |
| 100000 | 57341 |
| 200000 | 274785 |

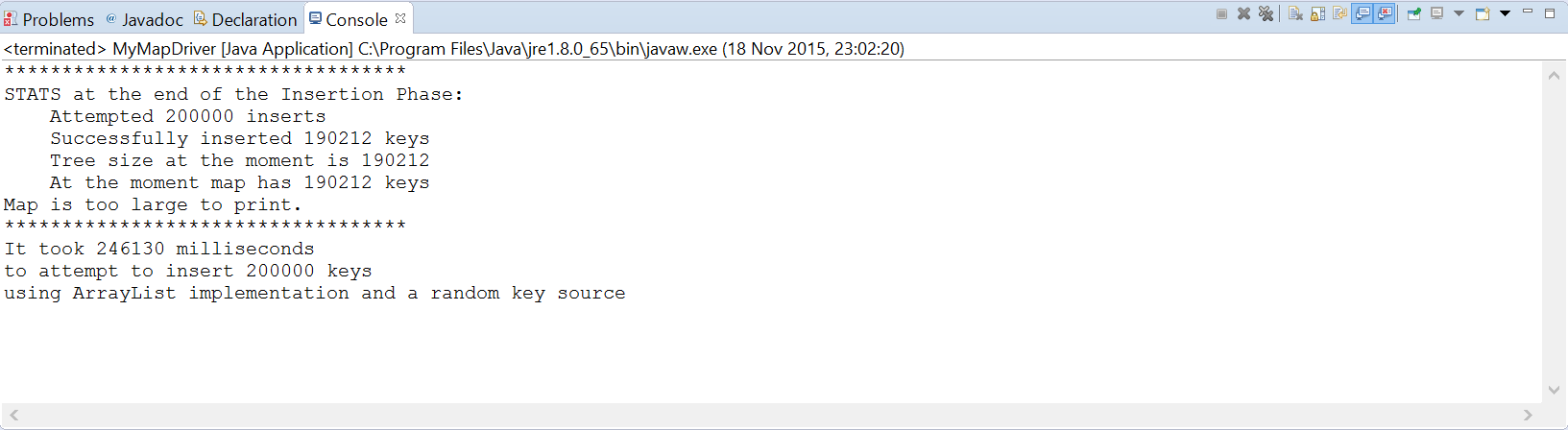


|  |  |
| --- | --- |
| For AVL trees | |
| Number of Keys | Run time/milliseconds |
| 3125 | 5 |
| 6250 | 8 |
| 12500 | 10 |
| 25000 | 15 |
| 50000 | 31 |
| 100000 | 35 |
| 200000 | 37 |



**“Random” Scenario:**

|  |  |
| --- | --- |
| For Array List | |
| Number of Keys | Run time/milliseconds |
| 25000 | 1442 |
| 50000 | 7294 |
| 100000 | 49083 |
| 200000 | 246130 |

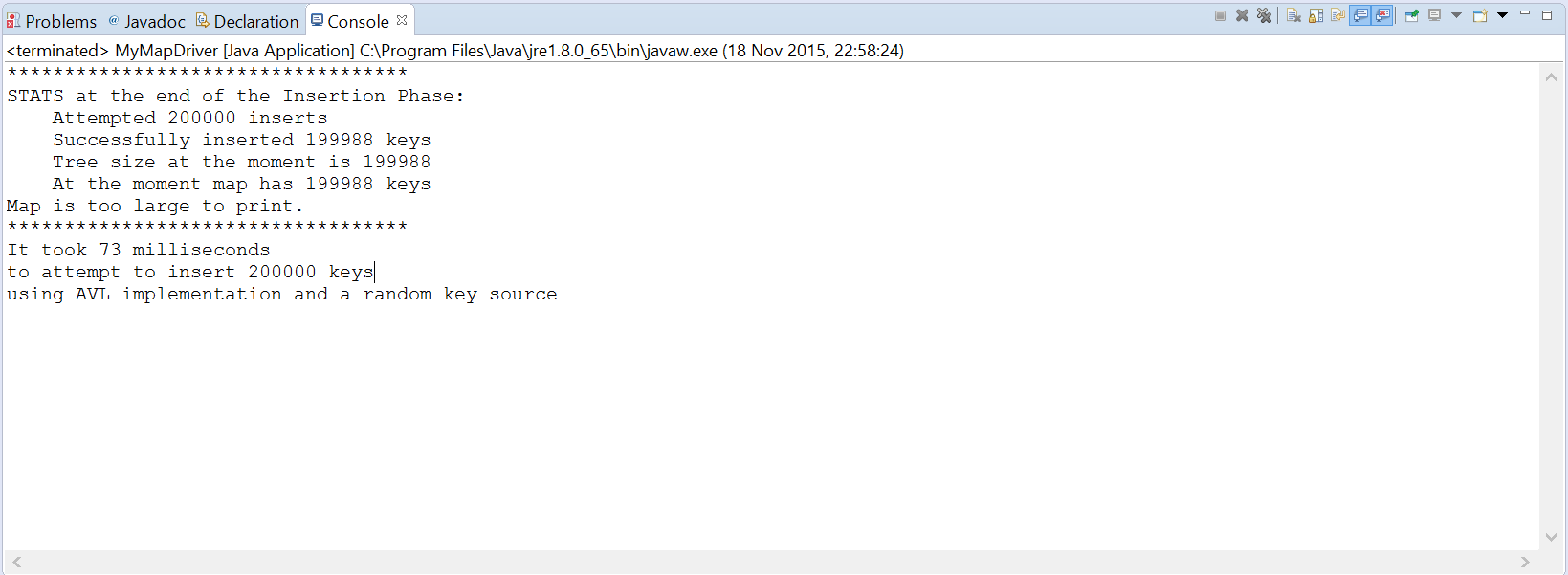


|  |  |
| --- | --- |
| For Hash Maps | |
| Number of Keys | Run time/milliseconds |
| 25000 | 14 |
| 50000 | 27 |
| 100000 | 44 |
| 200000 | 76 |



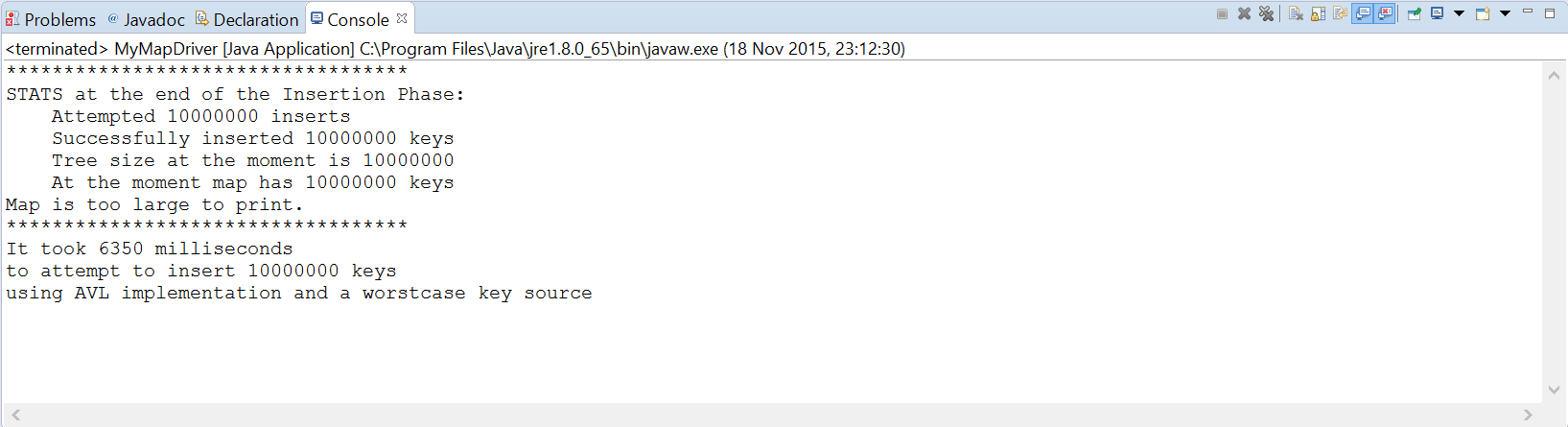
|  |  |
| --- | --- |
| For BSTs | |
| Number of Keys | Run time/milliseconds |
| 25000 | 18 |
| 50000 | 35 |
| 100000 | 61 |
| 200000 | 129 |

|  |  |
| --- | --- |
| For AVL trees | |
| Number of Keys | Run time/milliseconds |
| 25000 | 21 |
| 50000 | 38 |
| 100000 | 51 |
| 200000 | 73 |



**AVL with huge number of entries**

|  |  |
| --- | --- |
| **Number of Keys** | **Time/milliseconds** |
| 100000 | 29 |
| 1000000 | 135 |
| 10000000 | 6350 |



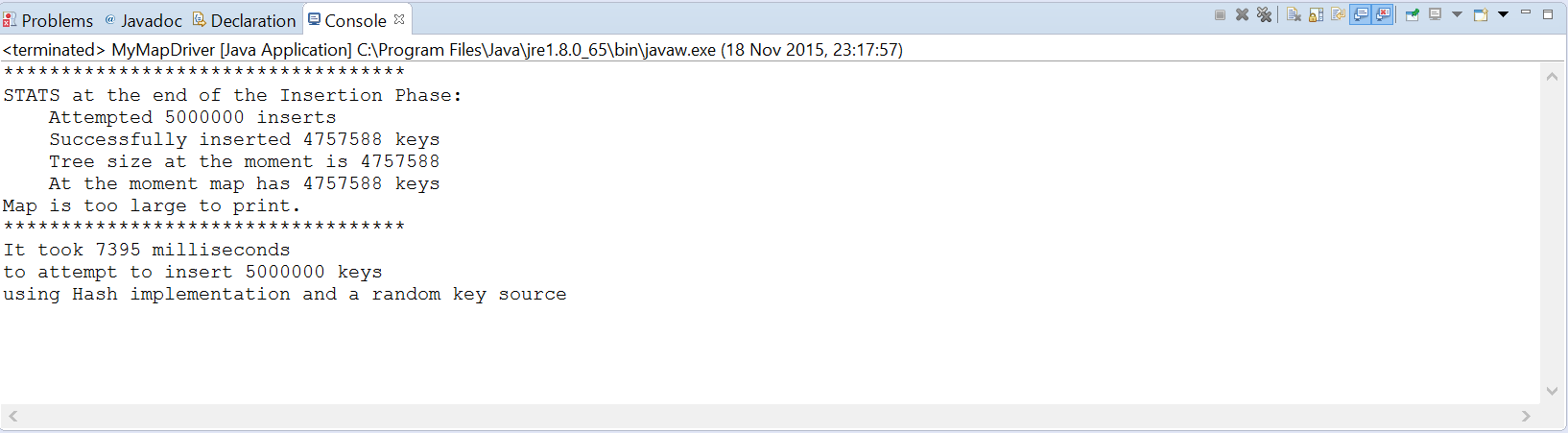
**How would large number of entries for BST differ:**

Time for insertion of 10,000,000 keys to a BST would be 100 times the ‘worstcase’ scenario time from above for 100 000 key entries. And that was 57 s approximately. This would take 5700 s. It is proportional to the number of keys because it’s the worst case where the keys are being entered in one long chain.

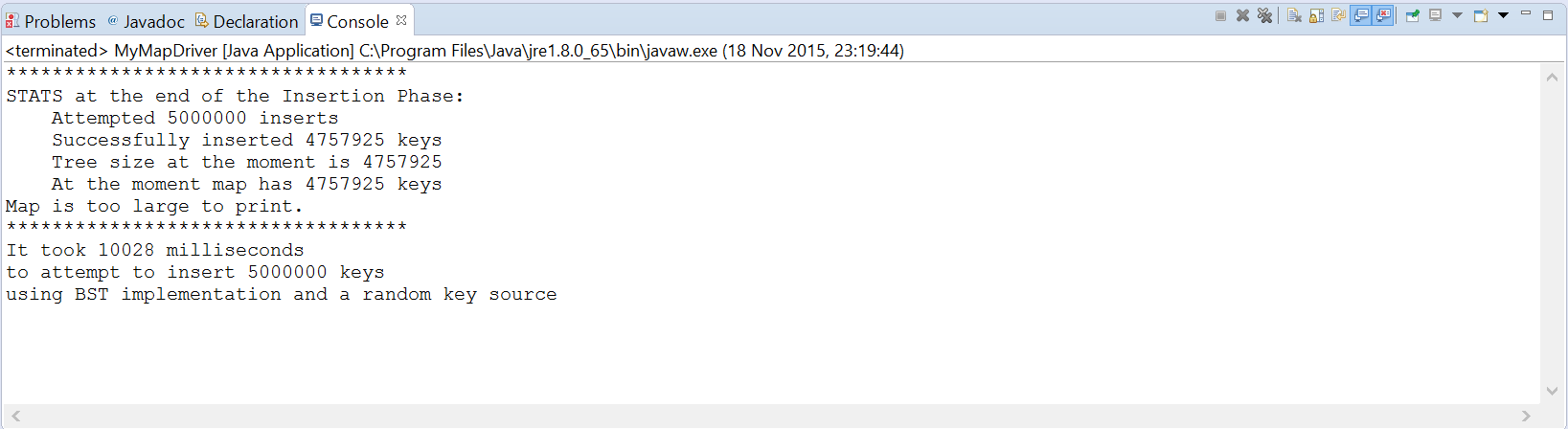
**REALLY LARGE NUMBER OF ENTRIES!**

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of Keys** | **Hash** | **BST** | **AVL** |
| 100000 | 36 ms | 54 | 61 |
| 1000000 | 502 ms | 1213 | 277 |
| 5000000 | 7395 ms | 10028 | 3244 |
|  |  |  |  |

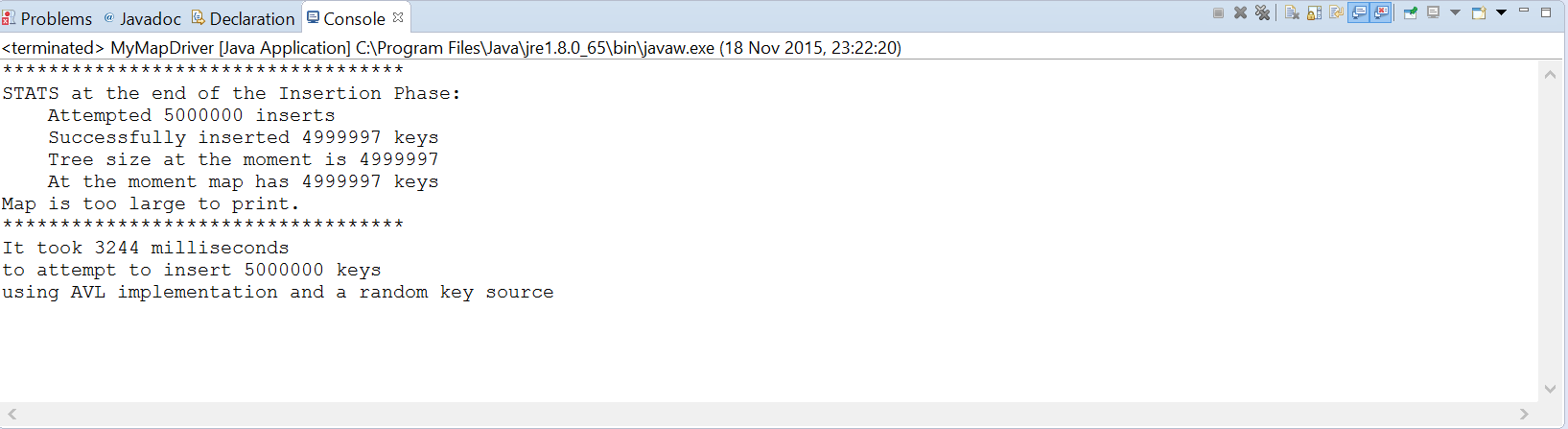
**Hash (5,000,000):**

****

**BST (5,000,000):**

****

**AVL (5,000,000):**

****

**Expectation for BST:**

I would expect random BST to run quicker than AVL from discussion in class since random additions won’t make one chain and also the time taken to fix heights would not be needed. But that is not the case in my running of my algorithm so I am assuming I did something wrong somewhere.