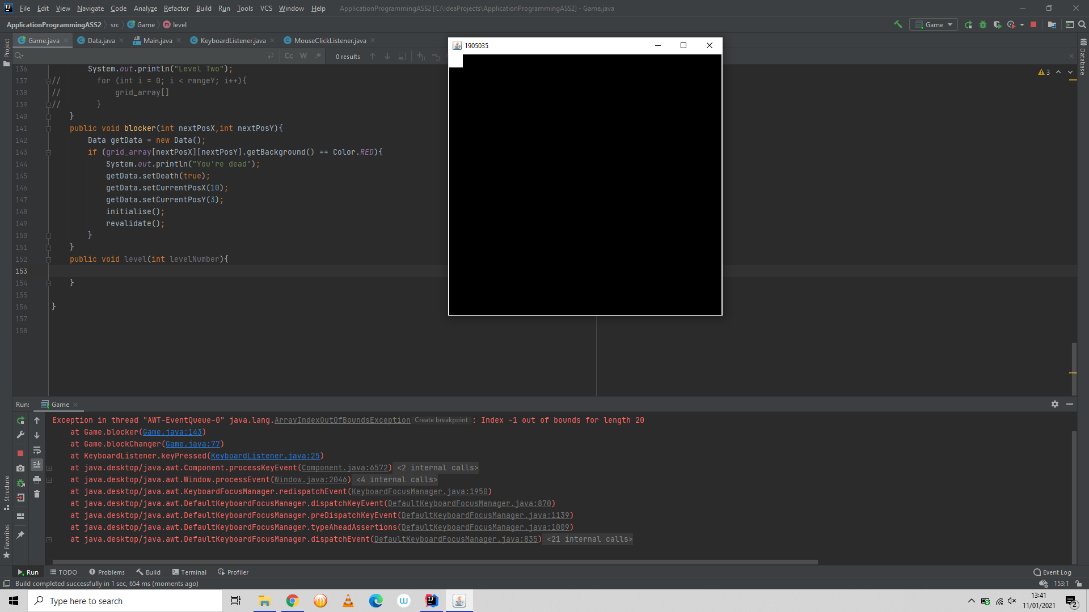
**Report**

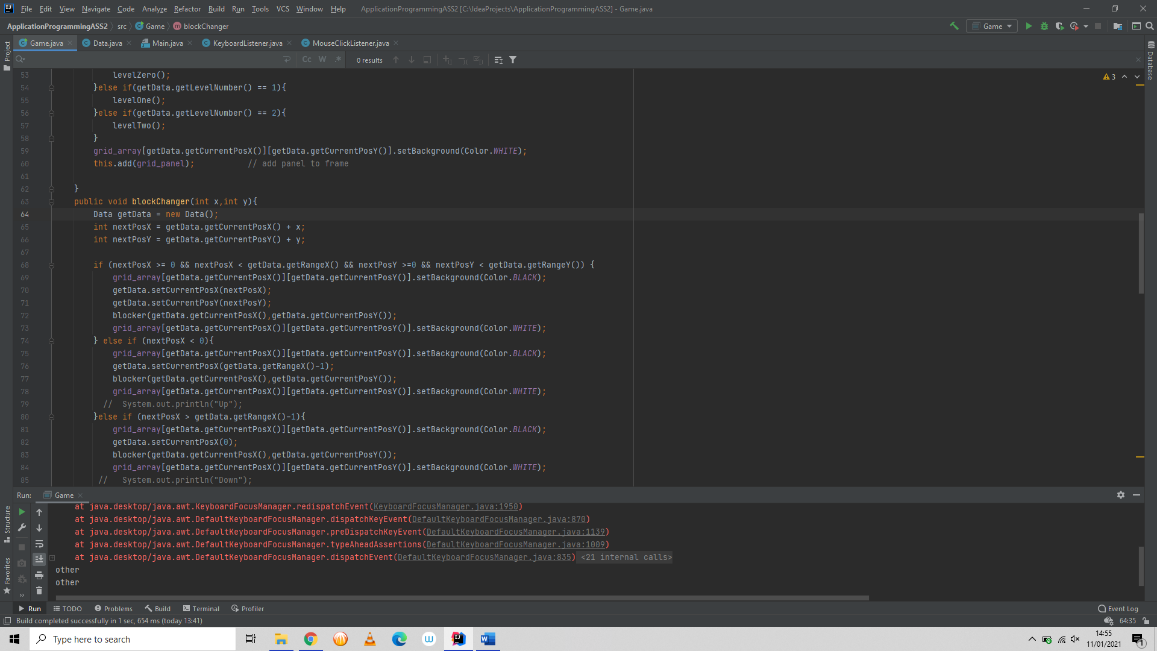
**By: Yusef Wohler(WOHLE42001)- 1905035**

**Bugs:**

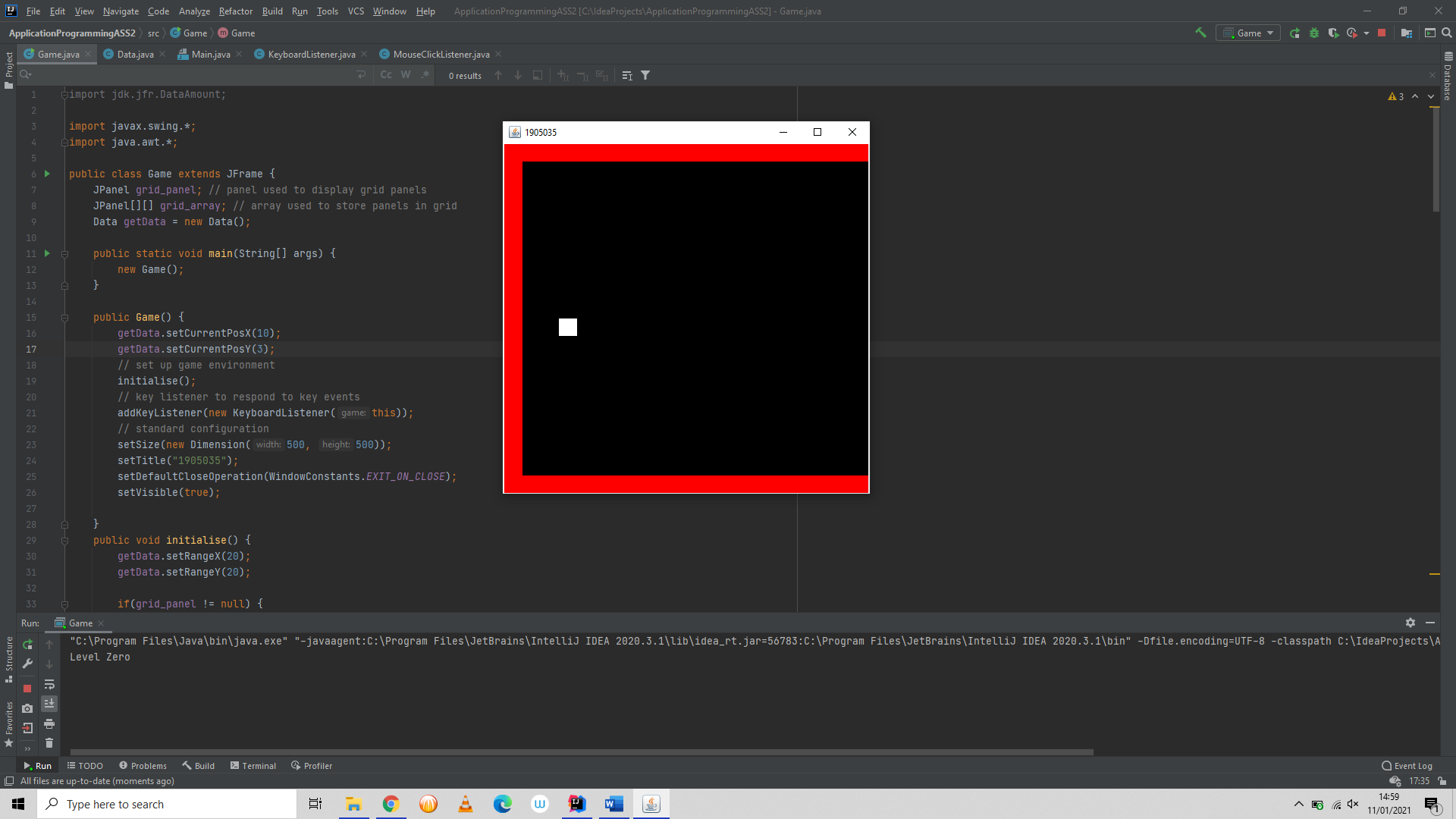
Bug1:

I came across this bug/error where I had to research deeply as to why it was causing this new error I never came across before. It made me lose my mind, so usually when I come across any advanced bug, I would debug it by the Rubber duck Debugging method. This helped me break down every single line of code, it finally helped me figure out the problem. The solution was simple, but the problem was difficult. What caused the problem was that I continuously called a new object at every function, such that it created new variables at every function instead of using the current or recently set ones by other functions.

How it looked before the solution:



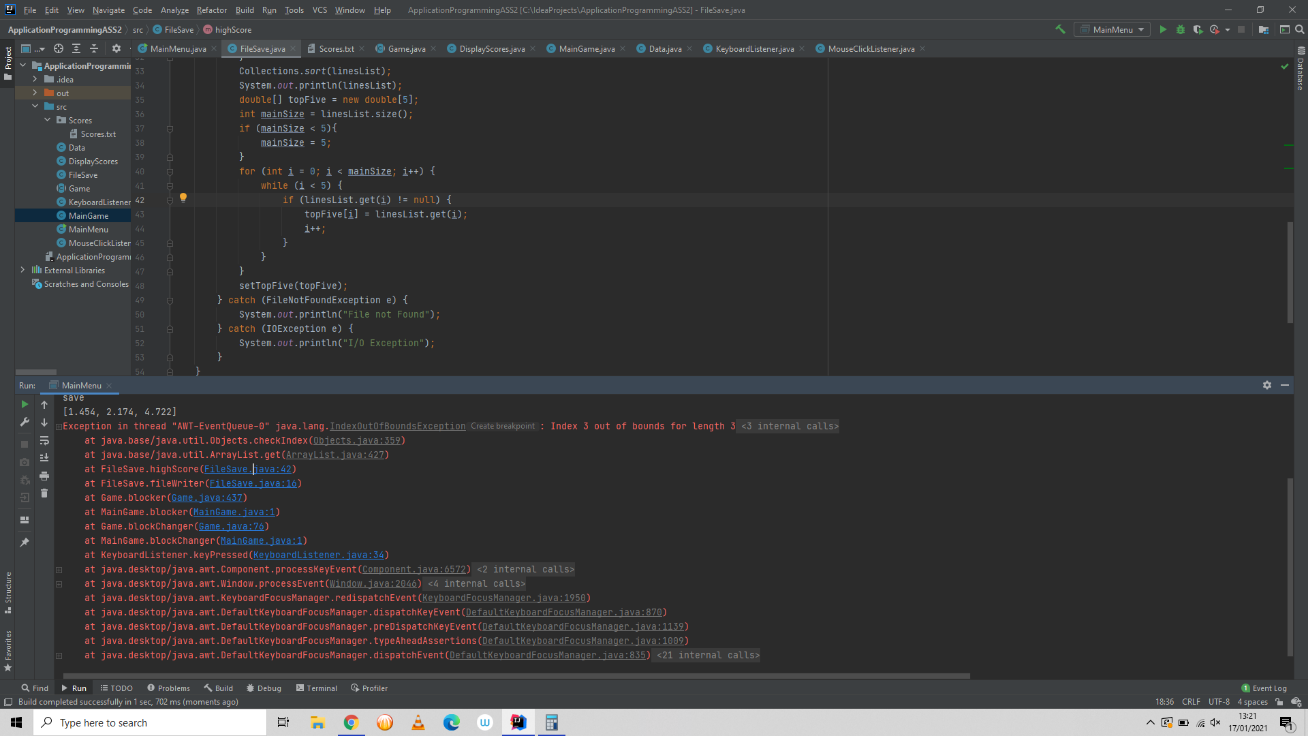
What it was supposed to look like and how it looked like after the solution:

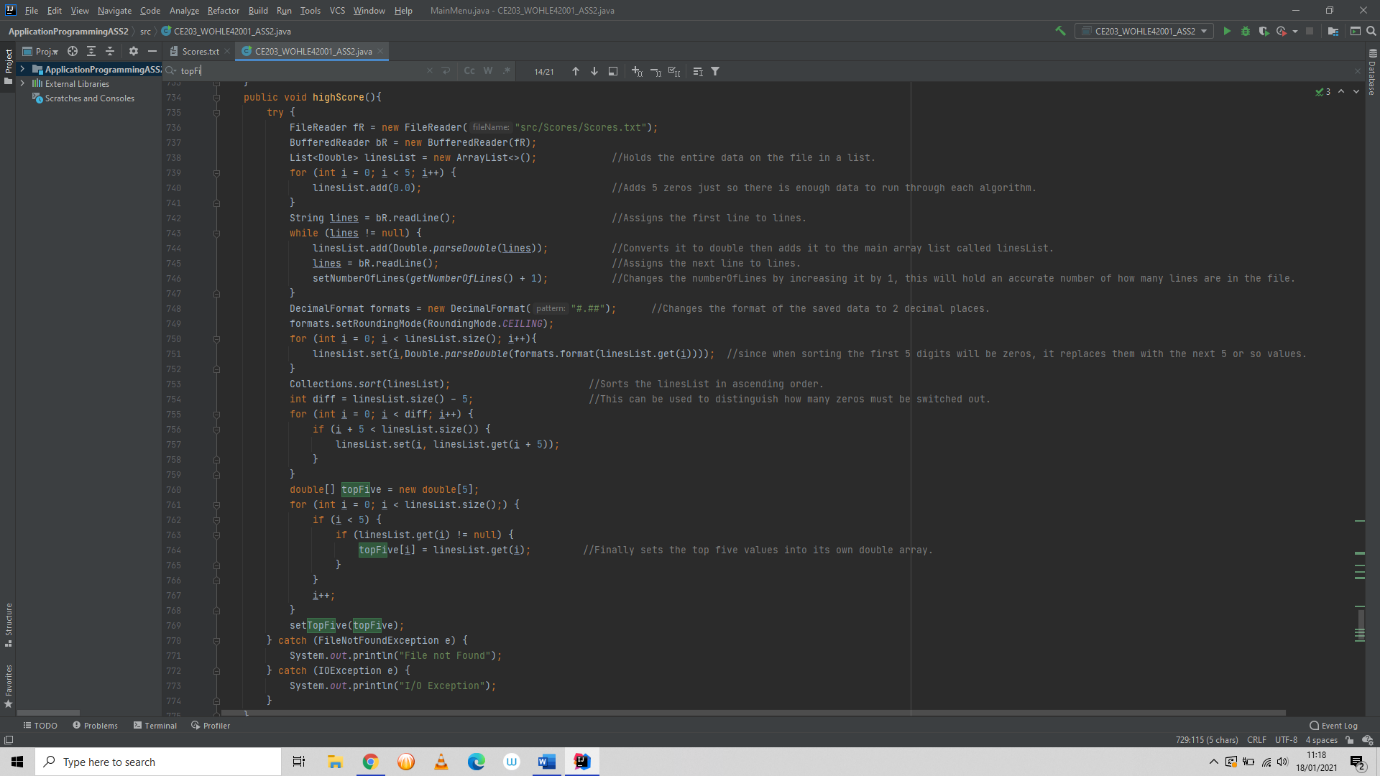


Solution: The solution was that I created a global variable called getData which is an object of the java class Data, which will use the currently set variables instead of creating a new set for each function.

Bug2:

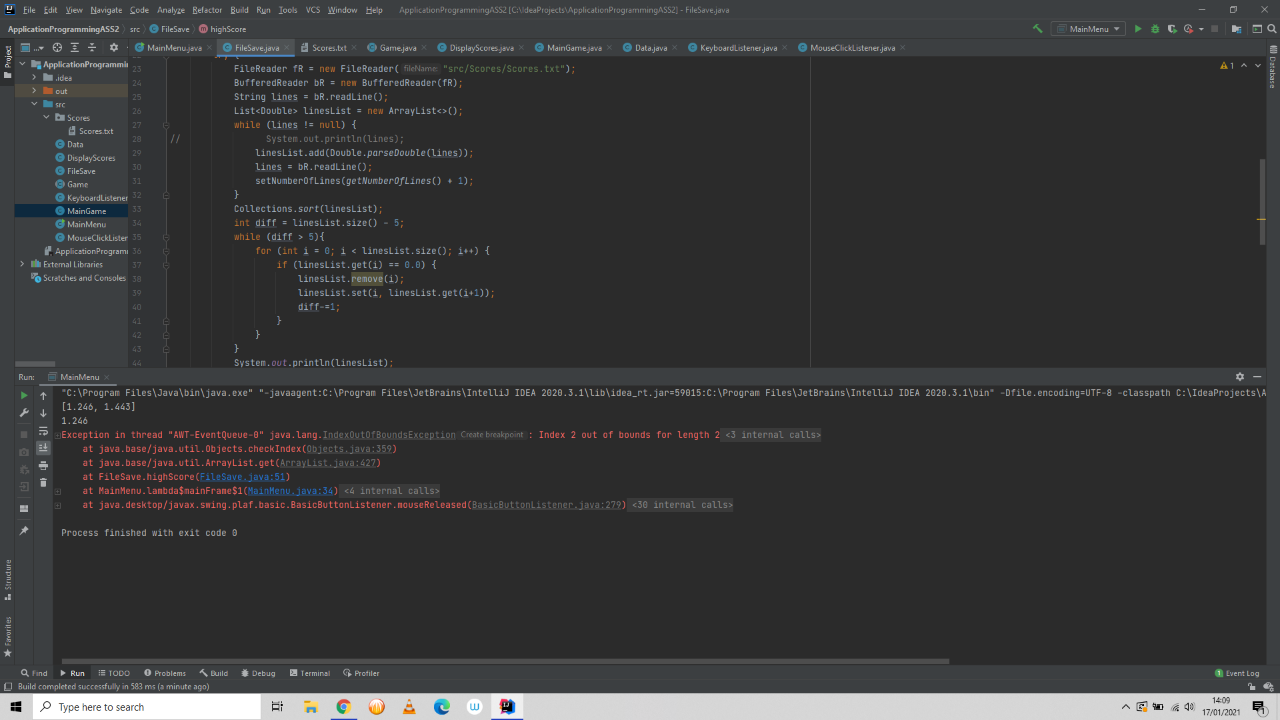
There was an issue with holding top five high scores only when the text file has less than 5 values. This was because I set the array size to how long the length of the file was, so my solution was to add 5 temporary zeros which will be replaced with the values in the file.

How it looked before the solution:

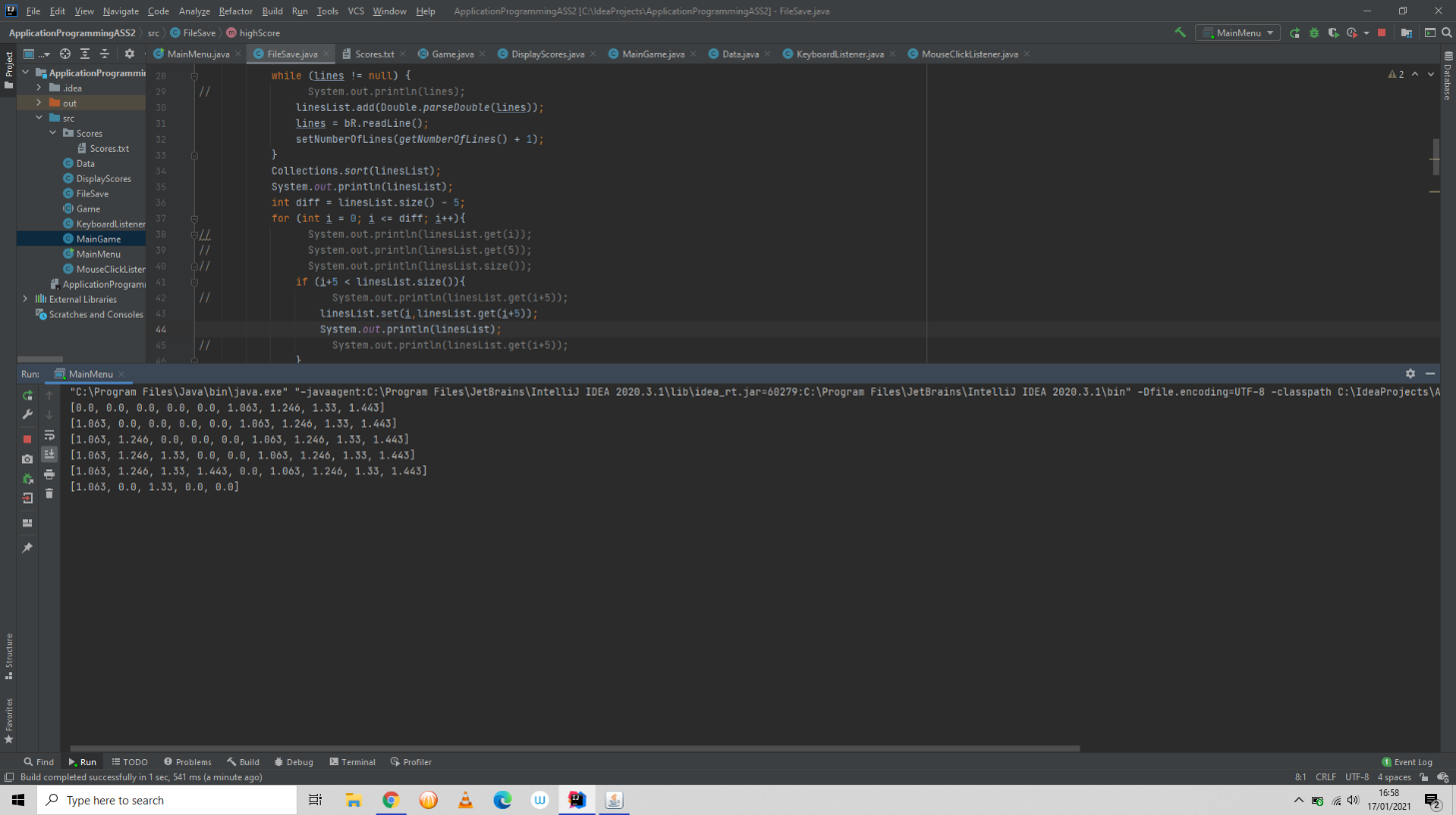
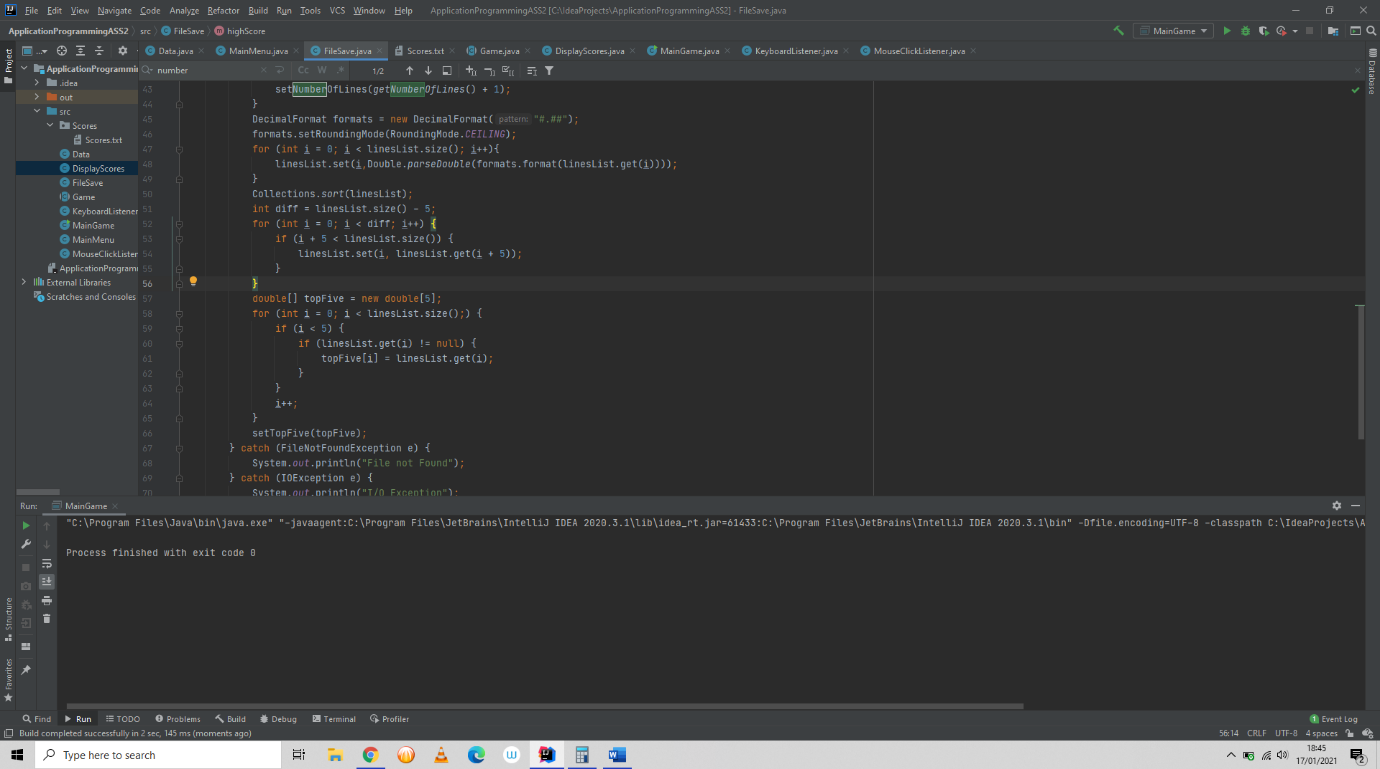
How it looked after the solution:

Solution: The solution was to add data to the main lines list such that the length of the topFive will not be greater than the length of the linesList. The temporary will be switched out as the user starts to add scores into the save text file.

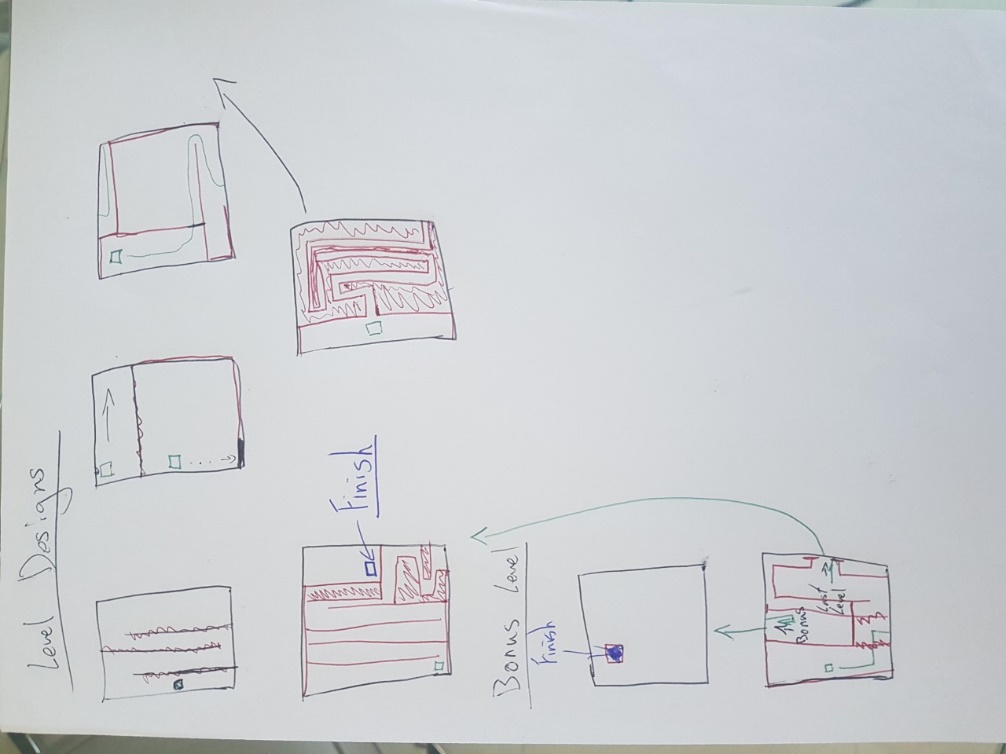
Bug3:

The third bug I have recorded from my program, when I tried to remove the zeros when the file has data. It gave me out of range index which in my eyes it seemed like this should not have been an issue, so as I researched to why I am getting this error. I came across some articles stating I should just use an iterator, but I thought to myself that it would not be my own work if I just copied this random persons’ iterator and it seemed too complicated for something simple, I wanted my algorithm to do. The solution was also simple, I was just another logical error I caused.

Solution: I added 2 to the index every iteration.

As you see in this screenshot, the output is not how it is supposed to appear as. The final array should have been “[1.063, 1.246, 1.33, 1.443, 0.0]”, however, the values “1.246” and “1.443” appear to be missing, the reason to this was that it was switched back to 0 when it was initially switched from 0, this is because at the end of the for loop I added the line “i++;” eventhough it was already specified in the for loop to add 1 to i, such that I added 2 to the index every iteration which caused to take away the second and fourth values back to 0.

**Pattern Design:**

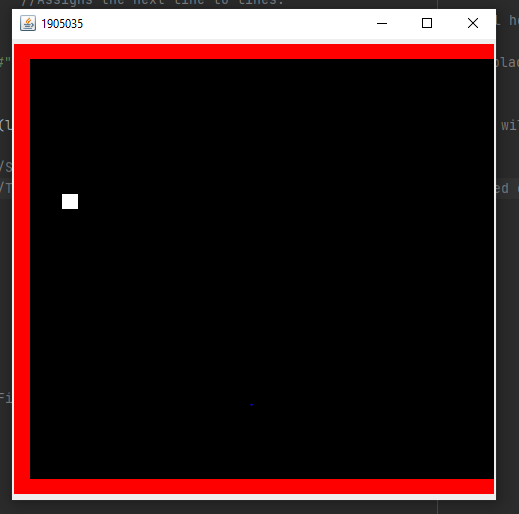
****I used various patterns in my sessions of coding, it sets my mind to a specific task I must complete before proceeding to the next task. However, some of the methods I could not stop going off the main task because I had to, some methods require other small methods before they were able to run. For example, my High Score class will not work properly if it was the only method to be called, it required the method fileWriter in the FileSave class to be called first, since it applies the data into the file first. While programming I definitely used more than one pattern and design technique, such as, I used coupling between classes by inheritance and by making object a global variable, to which would connect the first class with the second.

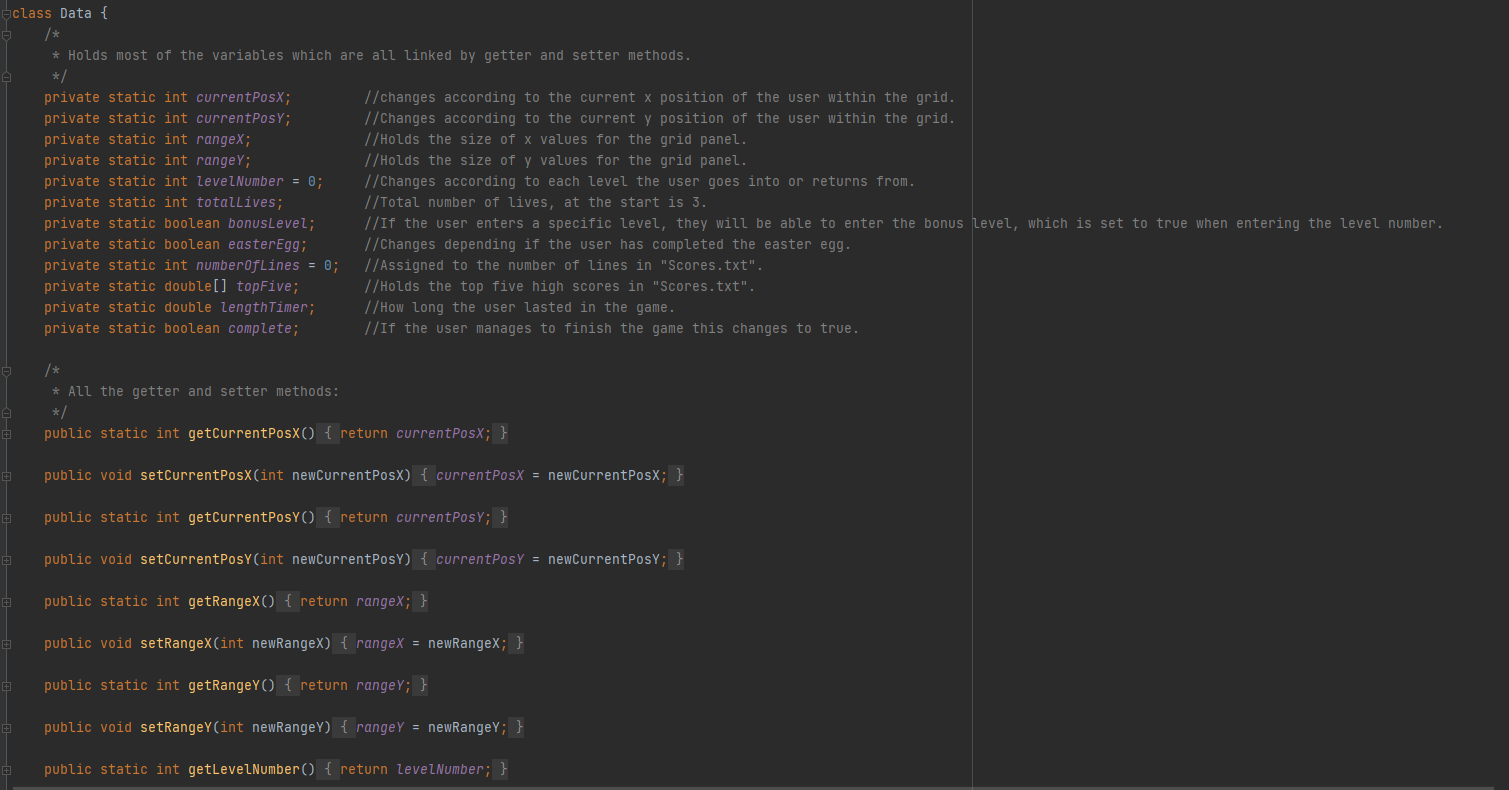
This was just a simple draft of how I will design most of the levels for the game. It helped me visualise how I want to display the game to the user.

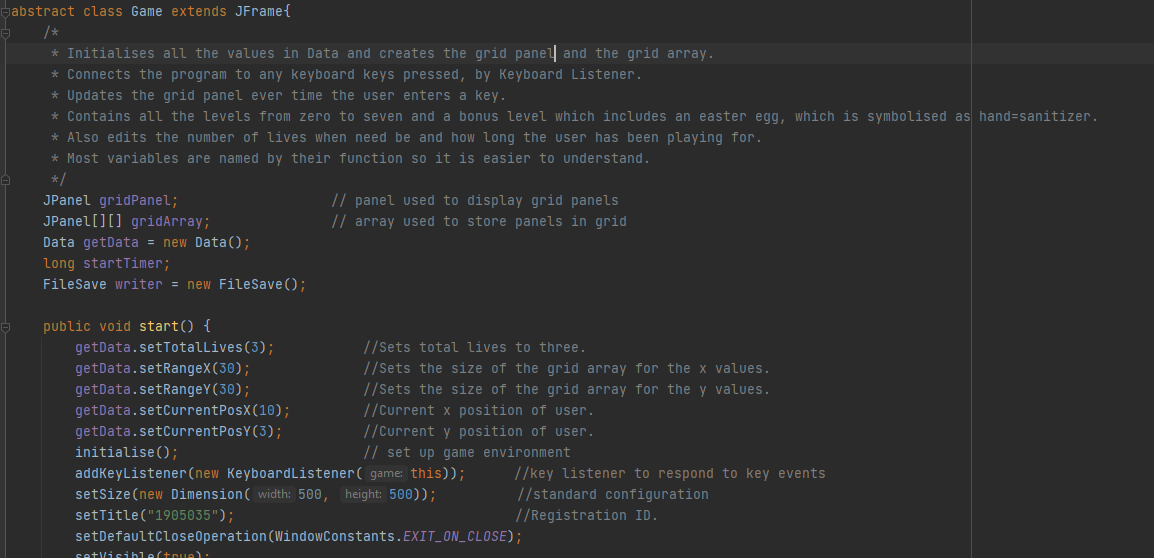
**SQL Extension:**

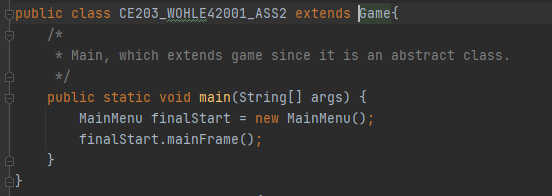
1. SQL can be implemented by connecting all the users around the world together, I will be able to hold a world high score, and possibly an only friend’s scoreboard, this can be used for the users to compare their scores with anyone around the world.
2. Right now, it is easy for the user just to input any score in the text file and say that was his score. Which would be known as cheating, such that with a SQL extension it would be quite harder and by law impossible. The reason I said by law is because there can be hackers who can create SQL injections in order to set their score differently, such that basically means the user is unable to edit his score.
3. It also may save space for the users’ PC, since currently the text file that holds the scores is on the users’ computer, with SQL databases the game would be smaller in size since the file would not be on their computer but on my database.

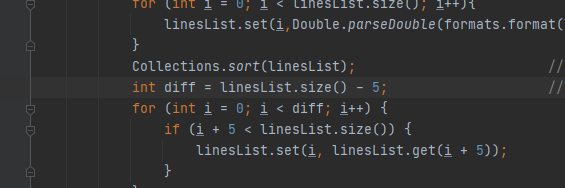
**Game (a to d):**

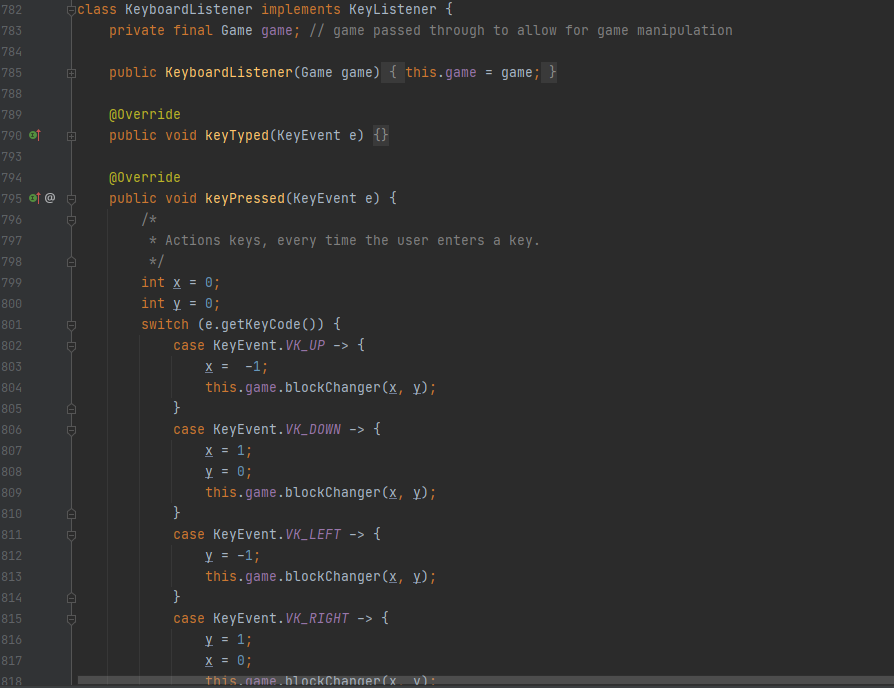




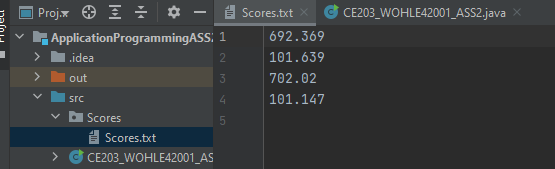


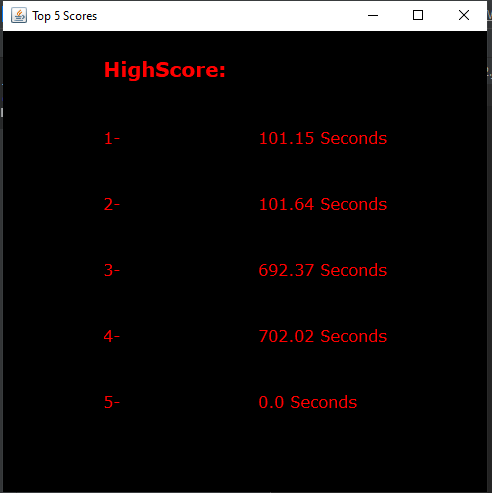


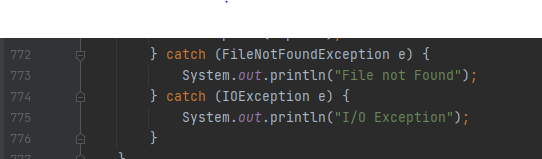


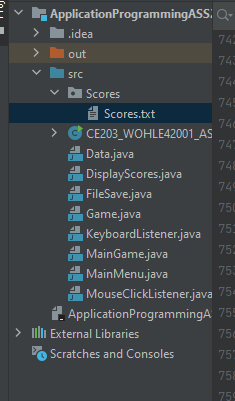










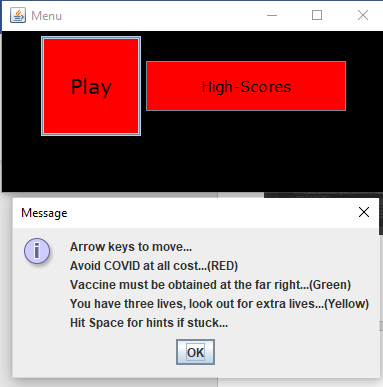


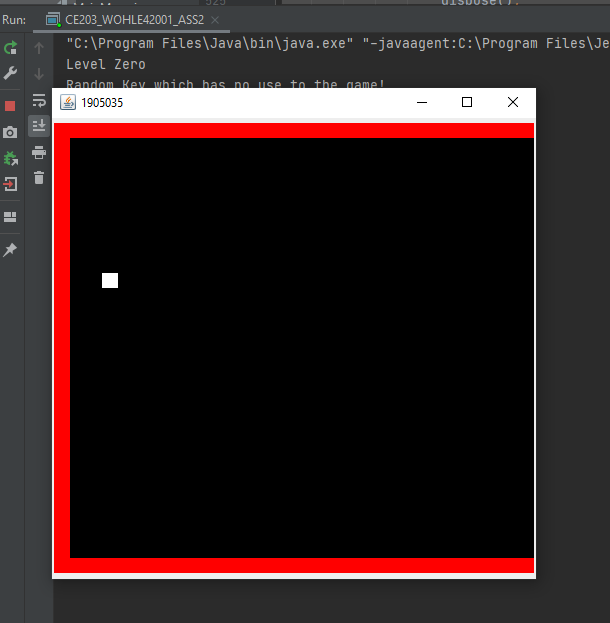
1. This is the report.
2. My game demonstrates how important social distancing can protect you from COVID (RED), but also how Vitamin C and Lemons can help your immune system. And to be playful I have also added Hand-Sanitizer to encourage users that sanitizer is in fact very good for you well being and to protect you from COVID.

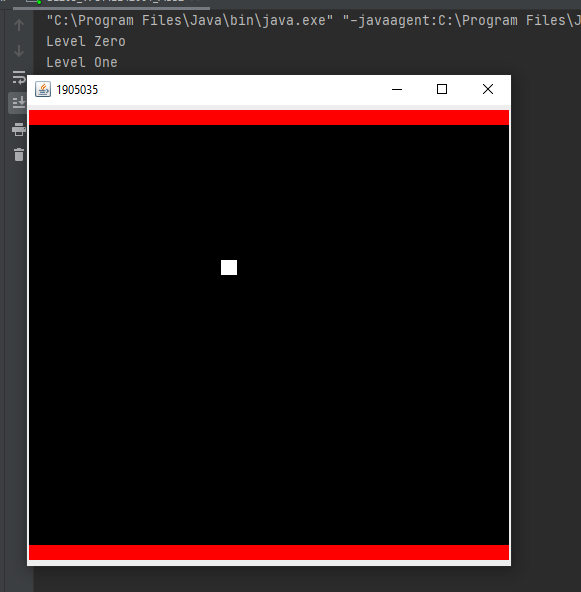
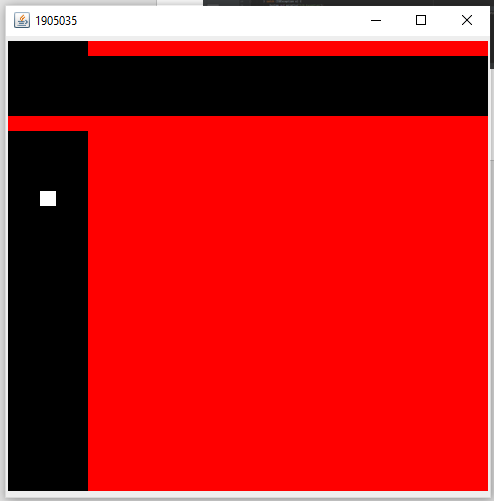
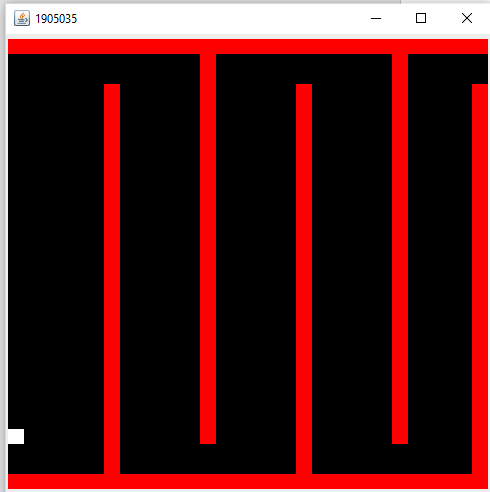
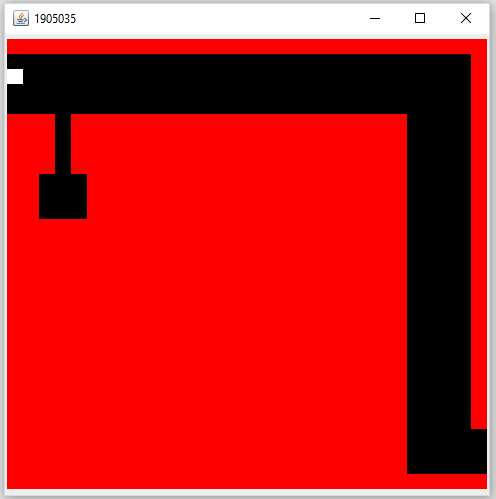
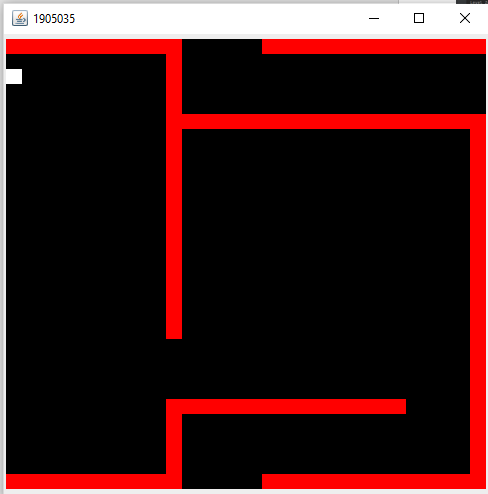
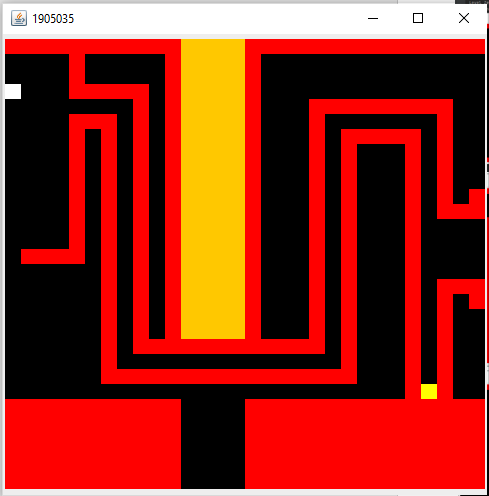
The High-Score screen is shown at the end of every round with the new user value if less than the current 5th value of scores. The points system works by how fast you complete the game, the easter egg will subtract time from your score which is an incentive to retrieve it, if users fail to complete the game 100 seconds gets added to their score, which is also an incentive to always try to finish the game and be precautious.

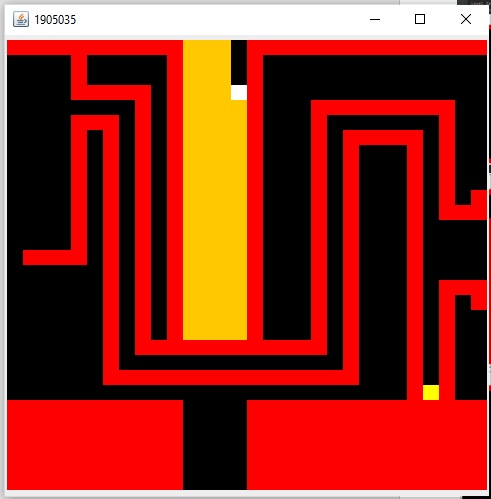
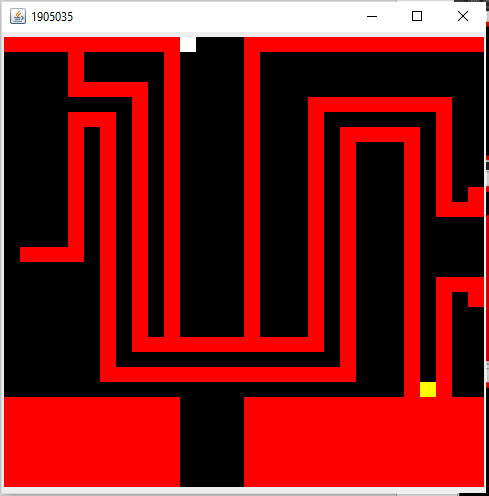
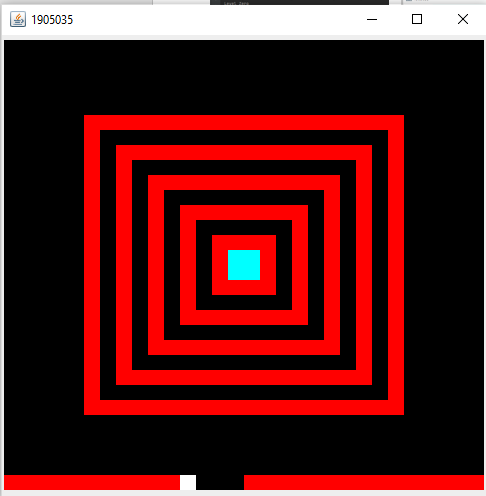
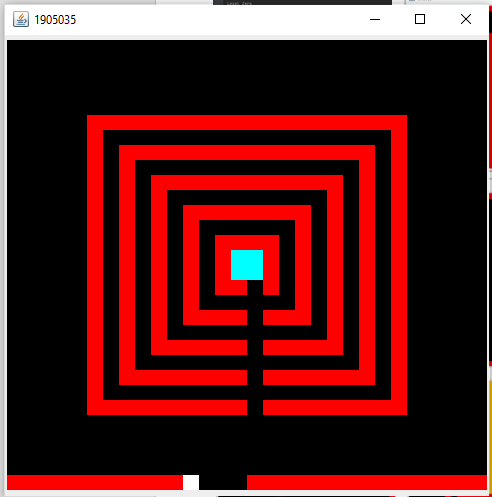
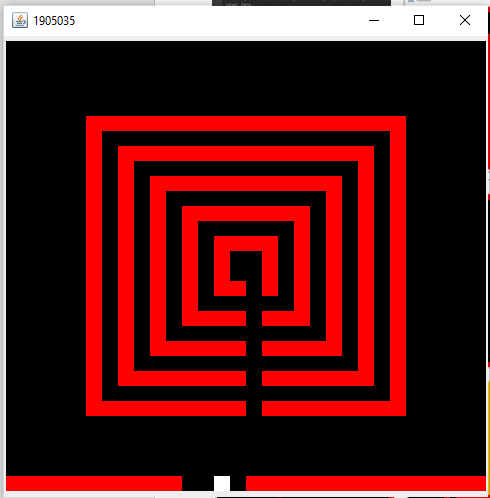
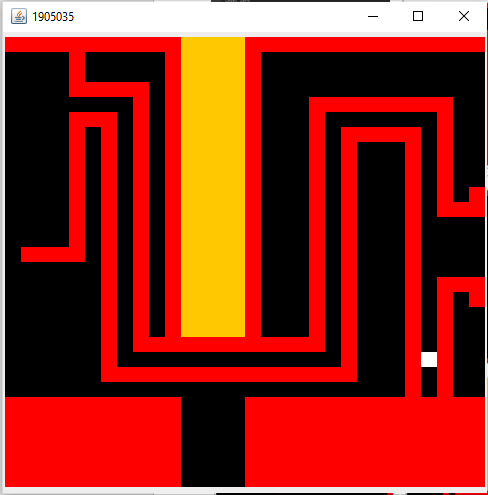
**GUI displays:**

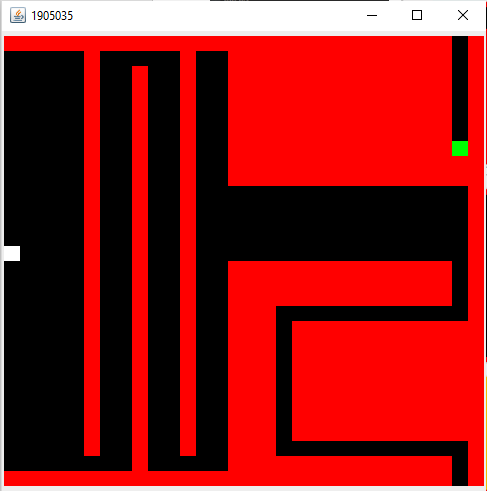
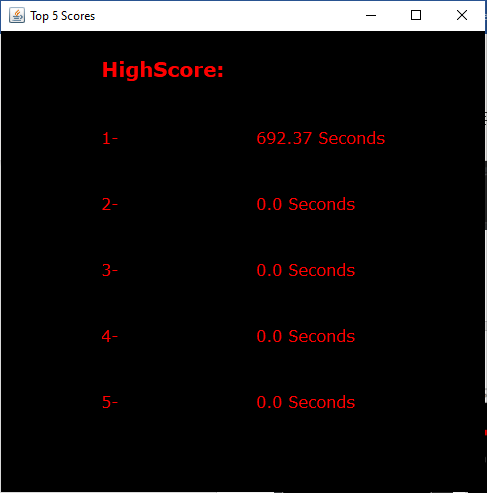


**Play Button:**

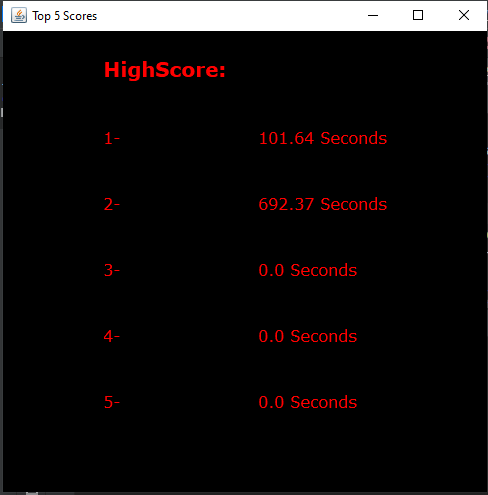








**High-Score Button:**



**Output from System:**



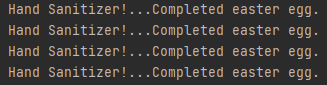








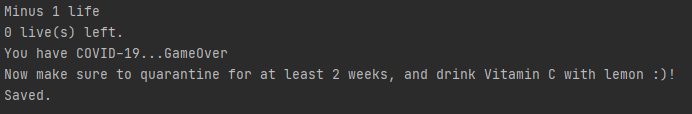














**Hints (Spacebar) JOptionPane:**

