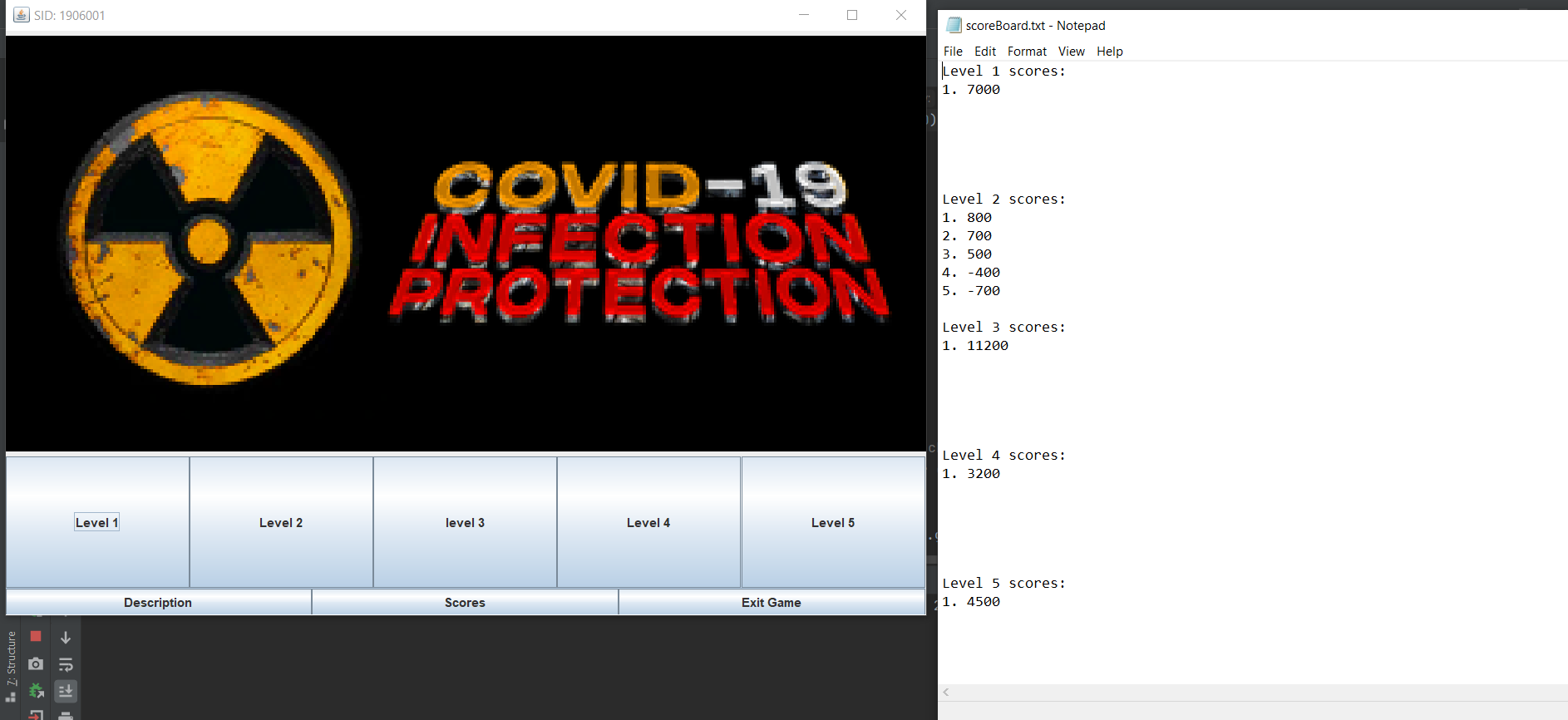
**CE203 Assignment 2 Report**

**Student ID: 1906001**

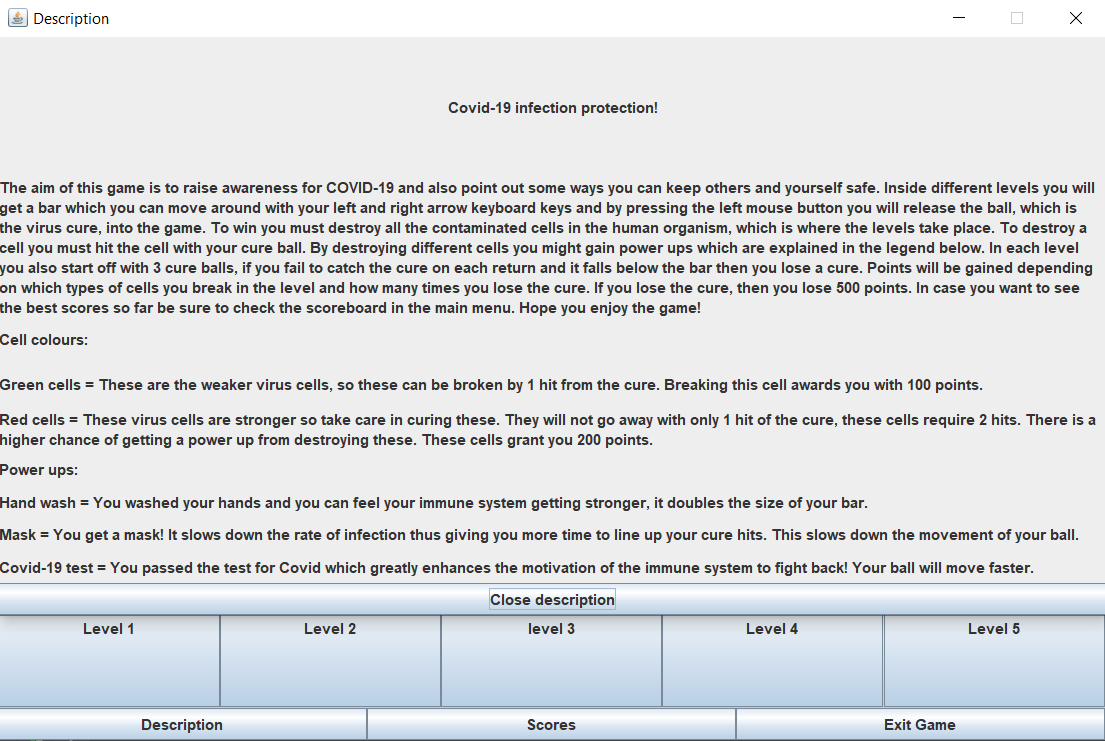
Before going into the testing phase I am also going to point out some things about the files associated with this program. The 3 picture files inside the zip folder need to be put into the same folder as the .java file goes into. As far as the scoreboard.txt is concerned the file will be created by itself but if there is a want to use the included scoreboard.txt file then it would need to be put into the main folder of the project where this .java is launched from. So this means that in the same folder where the projects src and out are.

**Testing**

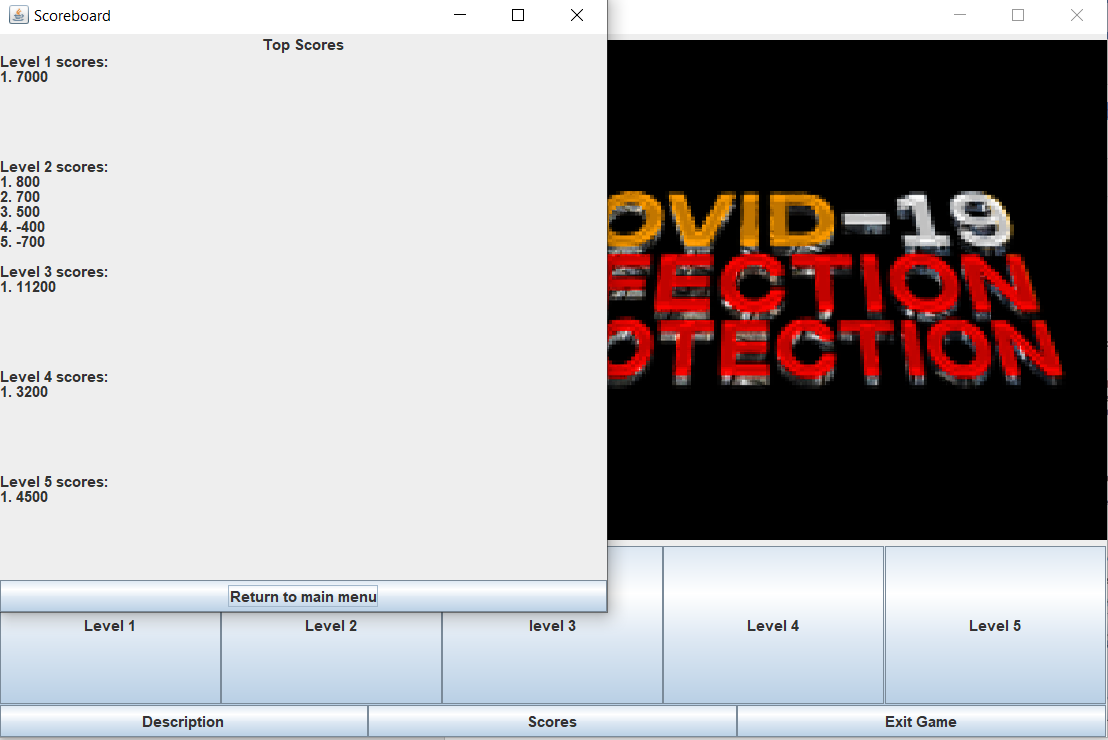
The first picture shows the .txt from the start of the test and also shows the main menu of the game. My SID can also be seen from the top of the main menu frame title, as the assignment guidelines task a) required.



If the player were to click on the “Description” button then a Description frame will show up. It describes the game and the rules. If the player were to press on “Close Description”, then the Description window would close.



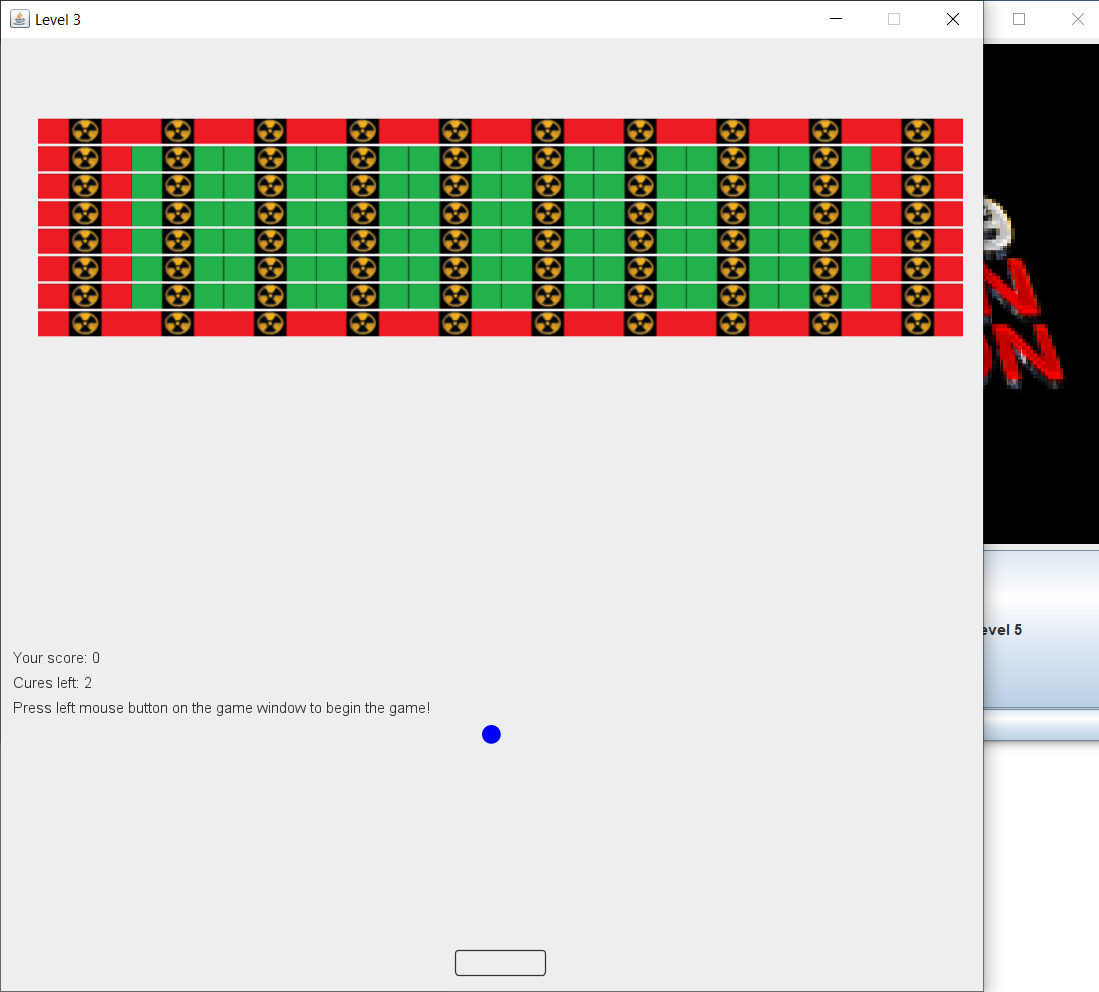
Going back to the main menu, if the player were to click on the “Scores” button then a new window would open which shows all the best scores in all the levels right now, if there were no scores(the .txt file did not exist) then it would ask the player to play the game to have a score put into the leaderboard. Once the button in the frame is pressed, the frame is closed.

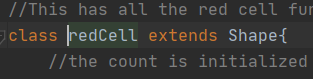


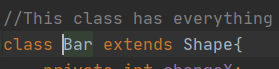
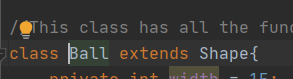
The last lower button in the main menu, “Exit Game”, gives the player a small window where it asks for reassurance on the players want to exit the game. If “Yes” is pressed then the whole game closes, if “No” is pressed then only the small window closes. For this test run I will be pressing “No” to show off a level next.



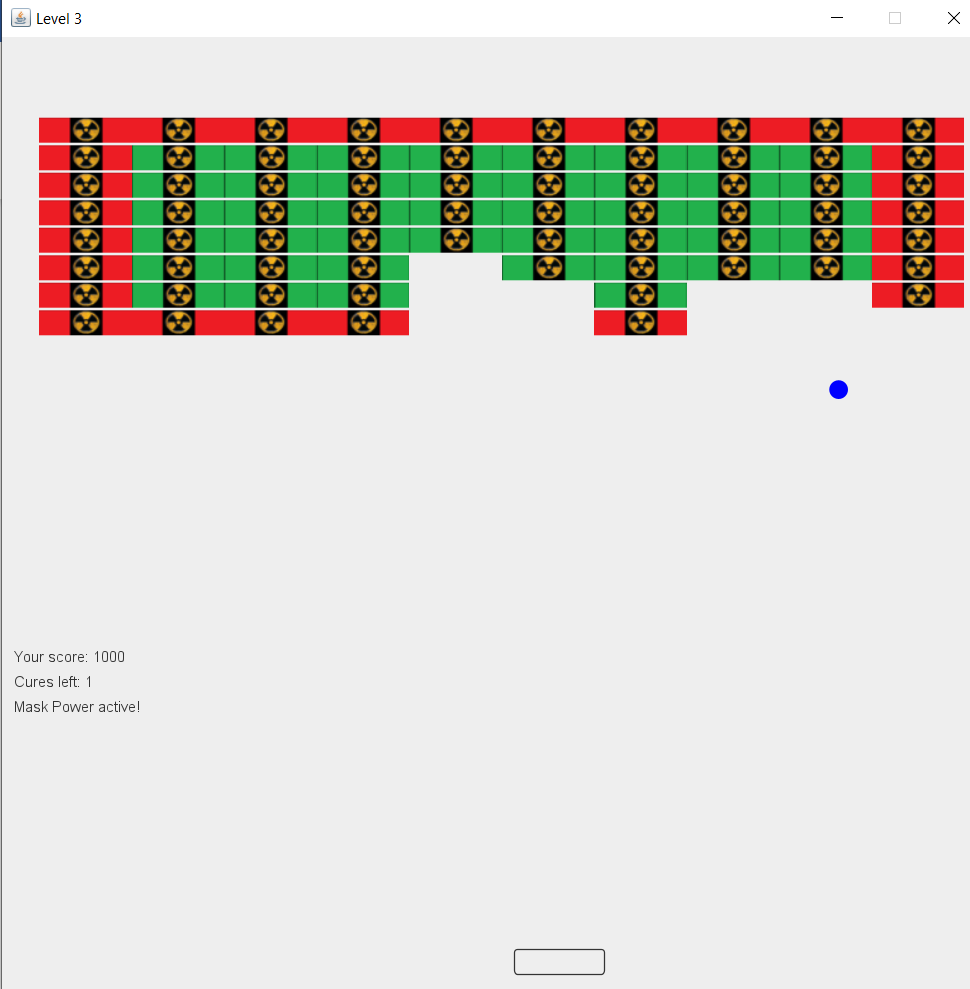
Next up, the main menu has 5 bigger buttons which point to different levels in the game. Since the only difference between the levels is the virus cell setup, meaning that the game physics and power ups are the same, then I will be using “Level 3” for this test run to show the game in a working state. On a new window, the player bar, the cure ball and all the necessary virus cells will be drawn. On the left side of the screen 3 lines of text will also appear, one pointing to game score, next one to the the cure balls left and final one lets the player know of needed information, for example what to do or what power up is active. This screenshot also presents that the point a) from the assignment 2 report guidelines has been done. All of these objects run through my abstract class called Shape in my code, so Shape knows the position of every virus cell, the player bar and the cure ball, as requested by task b). To play the game the player has to click the left mouse button on the window, and to move the bar the player has to use the right and left arrow keys, as requested by the task c).



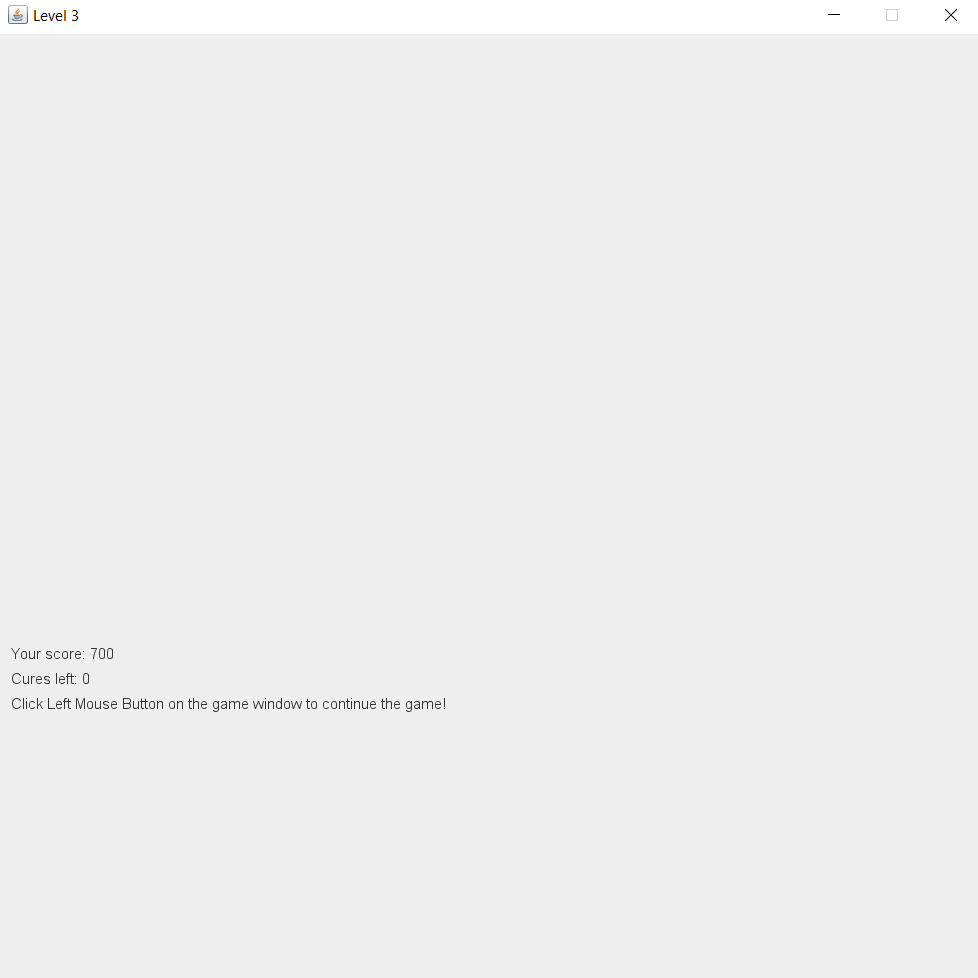


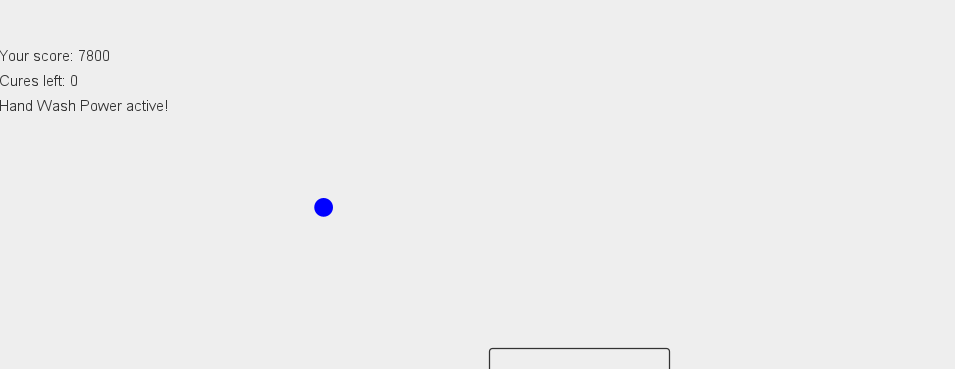


When the player uses the left mouses left click on the window the ball starts moving, and once it touches the red cell nothing happens to that because red cells take 2 hits to destroy. If a green cell is destroyed then it goes down in 1 hit. It can also be seen that the mask power up has been activated, sadly I cannot show it on the picture that well but this slows down the balls movement.

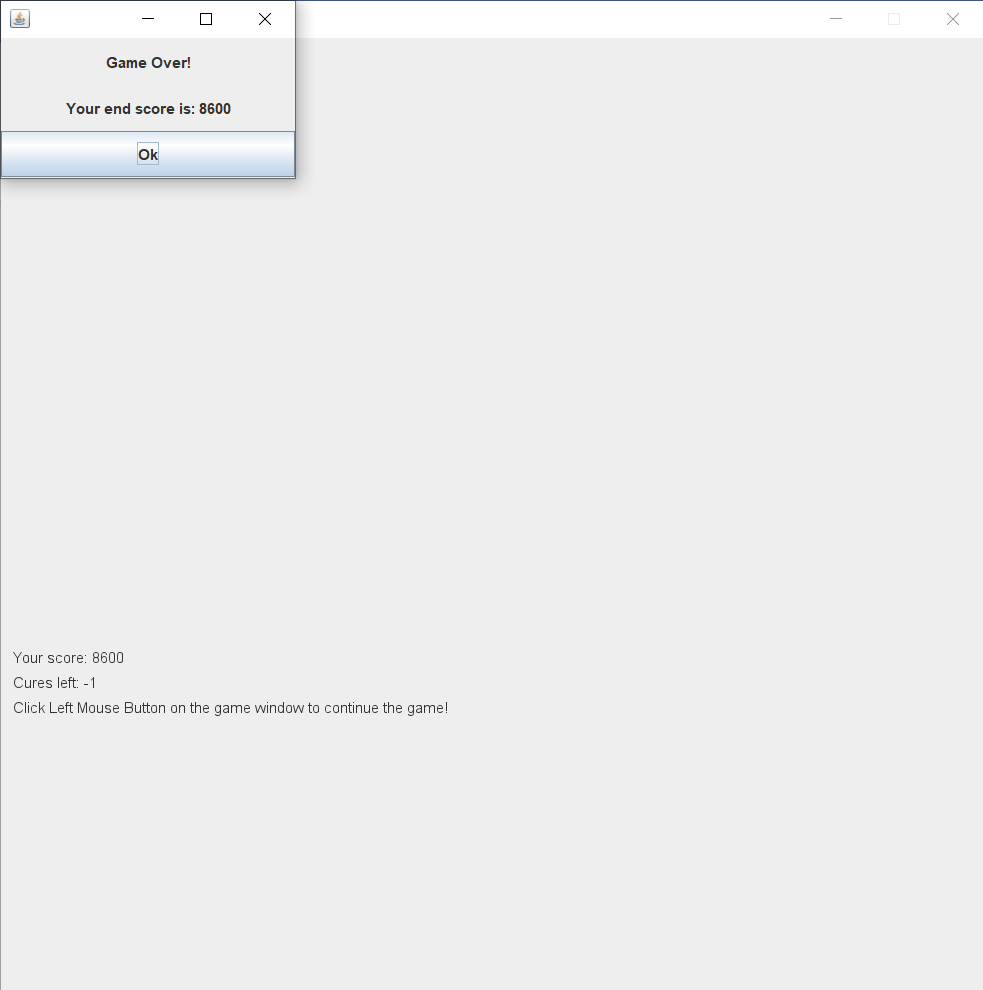


If the ball were to fly below the bar then the screen would go blank until the player clicks the window again. Also the score goes down by 500 and a life I lost.

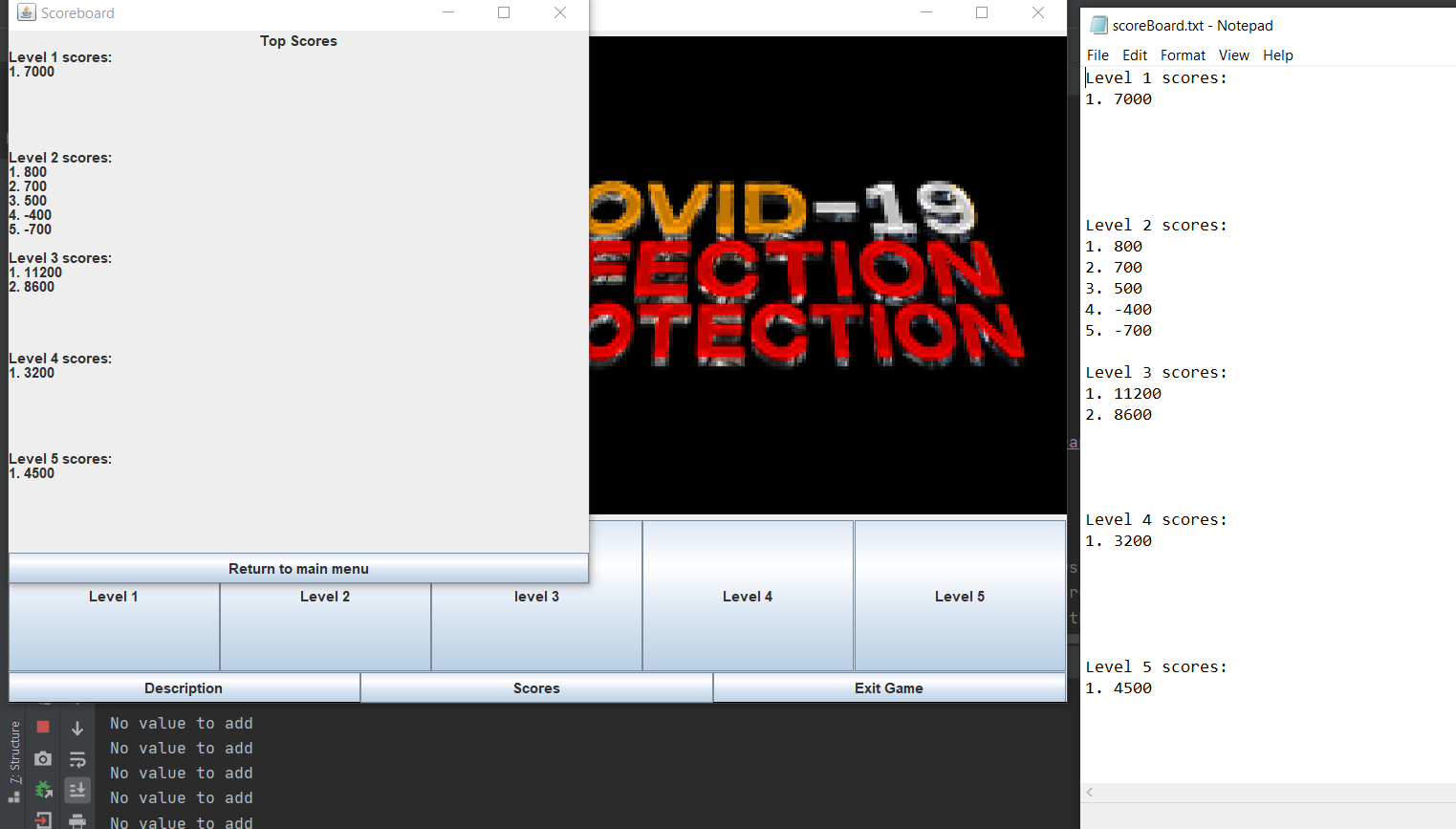


If the player were to receive the hand wash power up then the width of the bar would be doubled.  


So in this instance I lose the game so I get the losing screen. The only difference between the winning and losing screen is that right now it shows “Game Over!” while if I were to win it would show “You Won!”. It also shows the players end score and has a OK button. If that button is pressed or the small window is closed then both of the windows close, leaving only the main menu to be open.



Now since I played level 3 then the score board should have changed. If I go to the score board now I can see that the new score has been added and also in the right place. Next to it I also have the scoreboard.txt so the change can be seen there as well. This also covers the guidelines’ task d), of having a score system in the game.

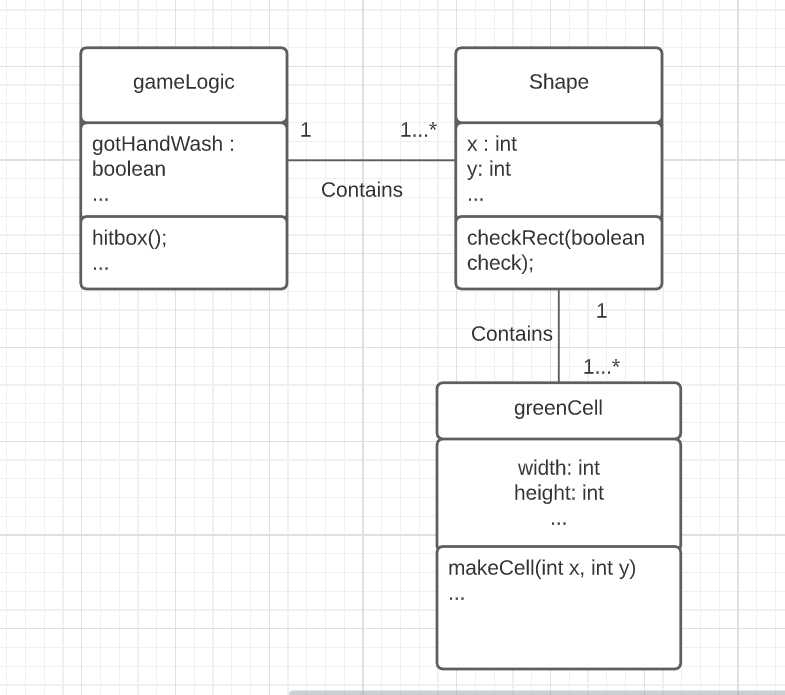


This should cover my game for this assignment. I will add that there was also the Covid Test Power Up but since it doubles the ball speed then I could not show it well enough on the pictures. Also the ball movement itself can only be seen by how the ball is currently situated on every picture, for the full movement and the power up showcase the game itself would need to be played. I also had one shortcoming as I was not able to fully make the ball bounce off the right border of the frame, as it goes a bit out of the frame and then bounces back, although not too much so it would break the game, same for the player bar.

**Design Patterns**

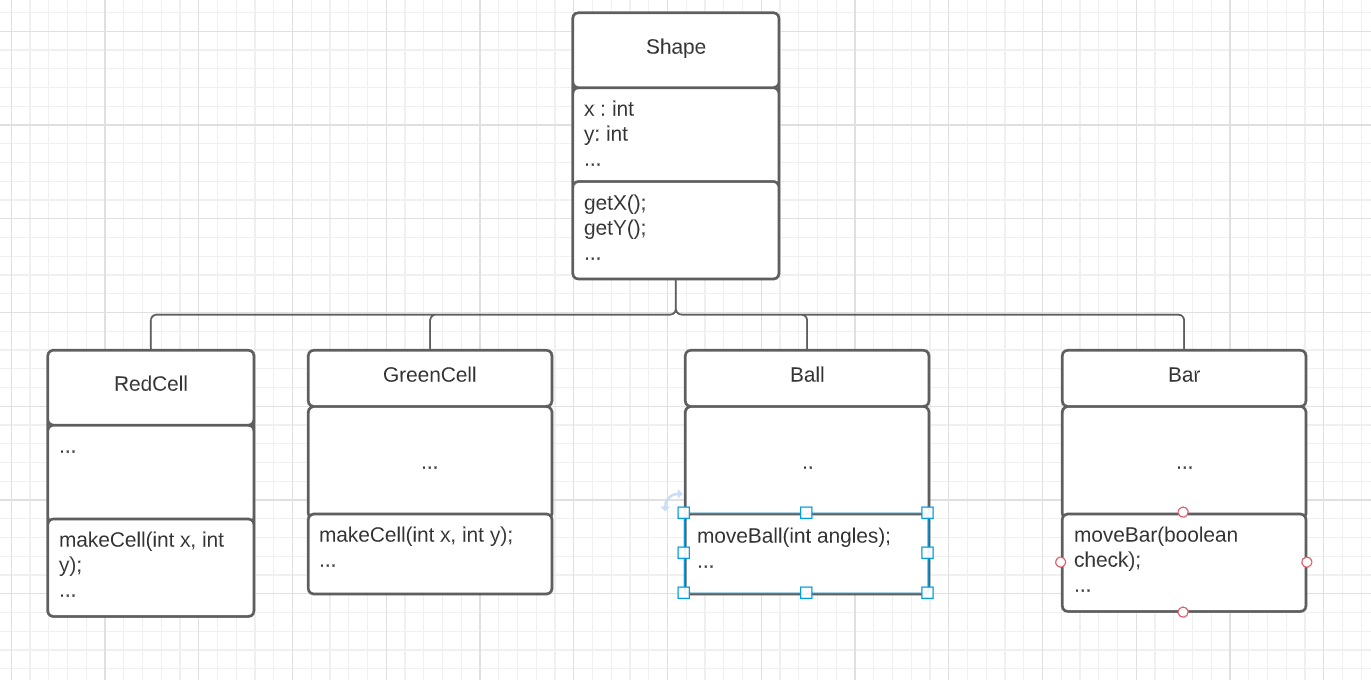
In this section I will bring 3 different examples of different design patterns used in my code.

1. **Grasp Pattern: Expert**



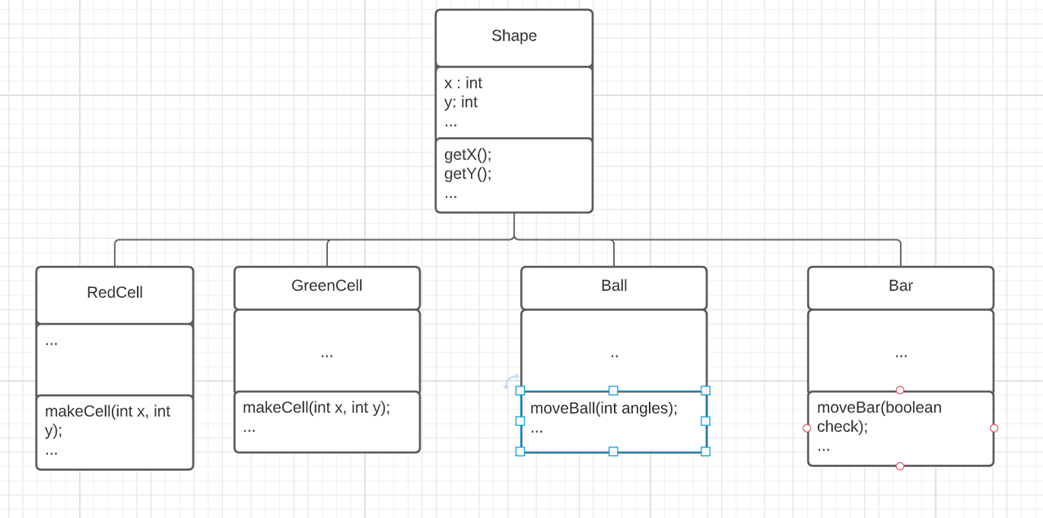
In this example the expert pattern of GRASP has been used. For the hitbox() to work it needs to know if the ball and the green cell have intersected with each other location-wise, and to do that it needs to perform a checkRect() on the shape of a green cell. Therefore the Shape class is the expert of that information, the Shape class knows where to shapes current rectangle is thanks to the checkRect(). In order to perform the function inside Shape, the class needs to know the green cells specific width and height, and these get assigned to every green cell inside the greenCell() class’s makeCell() function. So greenCell is the expert on what is the size of a single green cell.

1. **Grasp Pattern: Creator**



The Shape class is the Creator in this scenario, since, among other things, Shape contains and initializes the values of all the other classes in the picture. All of those classes need to know for example their x and y coordinate which the Shape has, to either create themselves or not where they are moving.

1. **Grasp Pattern: High Cohesion**

****

There is high cohesion between these Classes because even though they are all Shapes, instead of being thrown into the Shape class, each object has been made into their own class. For example a Ball does not need to know if a cell is destroyed or a Bar does not need to know where a ball is moving to move itself. They are all still connected to the Shape class because they all need similar functions from the Shape class.

**SQL Database Explanation**

If the scoring system would be done with the SQL database then the first thing I would do is import java.sql.\*, then I get access to the sql commands and functions. At the beginning of any function, which handles scores in my code, I would need to do the initialisation, and this is how it would look like if the class in question was for example LevelHandler:  
Connection conn = null;   
try {   
 LevelHandler.forName( "com.mysql.jdbc.Driver“ );  
 conn = DriverManager.getConnection( "jdbc:mysql://localhost:1065/scoreDatabase");   
} catch (ClassNotFoundException e) {   
 System.out.println("driver not found");   
} catch (SQLException e) {   
 System.out.println("connection failed");   
}

Next up I would make a Statement object to get access to the database values.  
Statement state = null;  
try{  
 state = conn.createStatement();  
} catch (SQLException e){  
 System.out.println("failed to access database");  
}

From here it would be specific to the class since LevelHandler, gameLogic, and ScoreHandler all access the database for different reasons. First up is going to be the LevelHandler. This class uses a method called fileCheck() that checks if the file exists and if it does not, then creates it, with all the necessary statements. So in this case I would instead need to create 5 tables for every level, so the scores would be placed inside, but before that also check if the tables already exist. So as an example checking only one level in the function fileCheck() would look like this(it would be put into a loop or repeated for other levels):  
DatabaseMetaData dbm = con.getMetaData();  
ResultSet tables = dbm.getTables(null, null, "level 1", null);  
if (tables.next()) {  
 System.out.println(“No new table is needed”);  
}  
else {  
 tables.state.executeQuery(“CREATE TABLE Level 1 { int levelNumber, int score}”);  
}

gameLogics fileUpdate() takes in the level number and also the score which was taken from the last level. What fileUpdate() does is update the file with the new score while also sorting the list, so for this I would need to access only 1 levels table, which is decided by the level parameter, and from there I will take all the values into an array, add the new score to that, sort it, and make sure there is only 5 values there, and then put them all in the right order into the database. For level 1, for example, the code would look like this:  
String levelReq = “Level “ + levelNum  
Integer scores1[] = new Integer[6];  
int specCount1 = 1;  
ResultSet valueReq = state.ExecuteQuery(“SELECT score FROM level 1”);  
while(valueReq.next()) {  
 scores1[specCount1] = valueReq;  
 specCount1++;  
}  
if(level == 1){  
 scores1[specCount1] = gameScore;  
 for(int i = 0 ; i < scores1.length;i++)  
 {  
 for(int j = i+1 ; j< scores1.length;j++)  
 {  
 try{  
 if(scores1[i] < scores1[j]) {  
 int temp = scores1[i];  
 scores1[i] = scores1[j];  
 scores1[j] = temp;  
 }  
 } catch (NullPointerException e){  
 continue;  
 }  
 }  
 }  
}  
try{  
 scores1 = ArrayUtils.remove(scores1, 5)  
} catch{NullPointerException e){  
 System.out.println(“No value to remove”);  
}  
int rowNum = 1;  
for(int I = 0; i<scores1.length; i++){  
 int newScore = scores1[i];  
 ResultSet valueUpdate = state.ExecuteQuery(“UPDATE “ + levelReq + “ SET score = “ + newScore + “ WHERE levelNumber = “ + rowNum);  
 rowNum++;

}

Finally, ScoreHandler needs to access the tables to display all of the top scores on all levels. For that I would need to display the level title and then put all the best scores below it, level by level So I think the code for that would be this:  
JLabel whitespace = new JLabel(“ “)  
for(int i = 1; i<6; i++){  
 String addLevel = “Level “ + i + “ scores: “;  
 gridPanel.add(addLevel)  
 for(int j = 1; j<6; j++){  
 ResultSet valueReq = state.ExecuteQuery(“SELECT score FROM level ” + i + “ WHERE levelNumber = “ + j );  
 gridPanel.add(valueReq);  
 }  
 gridPanel.add(whitespace);  
}

After those methods in all the functions, I would end each function off with these lines, to close the connection to the database:  
try {   
 state.close();   
 conn.close();   
} catch (SQLException e) {   
 System.out.println( "problems closing database");   
}