# Report and Testing

## Structure and Functionality:

Classes:

1. Window
2. Game
3. KeyboardListener
4. MouseClickListener
5. Results
6. GameObject (Abstract)
7. Snake\_player
8. Apple
9. Border

My program consists of 8 classes, 1 of which is an abstract class GameObject which is inherited by each game object within Snake (each of these are separate classes which extend the GameObject class): Snake\_player, Apple and Border are all the game objects required. Each class has a public method (same name as class) as well as a draw method. Each of these classes are instantiated within the Game class and stored within a List (Collection).

Another class called Window is used as a container for the Game class. It extends and instantiates the JFrame class. This class is the main class used to run the game. It does this by instantiating the Game class which contains all the methods that run the game. The Game class extends and instantiates the JPanel class – this panel contains all the game components to be loaded and spawned and calls the repaint method which allows the game to run (transition to each state). Also, the game class includes methods which allow movement within the game – the user can control the Snake\_player object by pressing the up, down, left and right arrow keys – this in turn calls a key listener class which contains methods to allow the changes in direction of the player. The aim of the game is to ‘eat’ the red apple, which will in turn increment the snake length and score. Each time the snake eats the apple, it spawns in a new random location on the board. The MouseListener class allows the user to pause the game by clicking twice – the game will resume if clicked once.

As well as including the functionality of the game, the Game class also instantiates the Results class when the round is over. This class creates a scores.txt file and outputs the current score to it. Each round, the score is appended to the file. Two Lists are used to store scores – one which is used to store the scores within the file, sorted by highest-to-lowest values. The other is used to store a maximum of 5 high scores – these are drawn on to the JPanel when the game is over. The user gets to compare their score with the other high scores in the game.

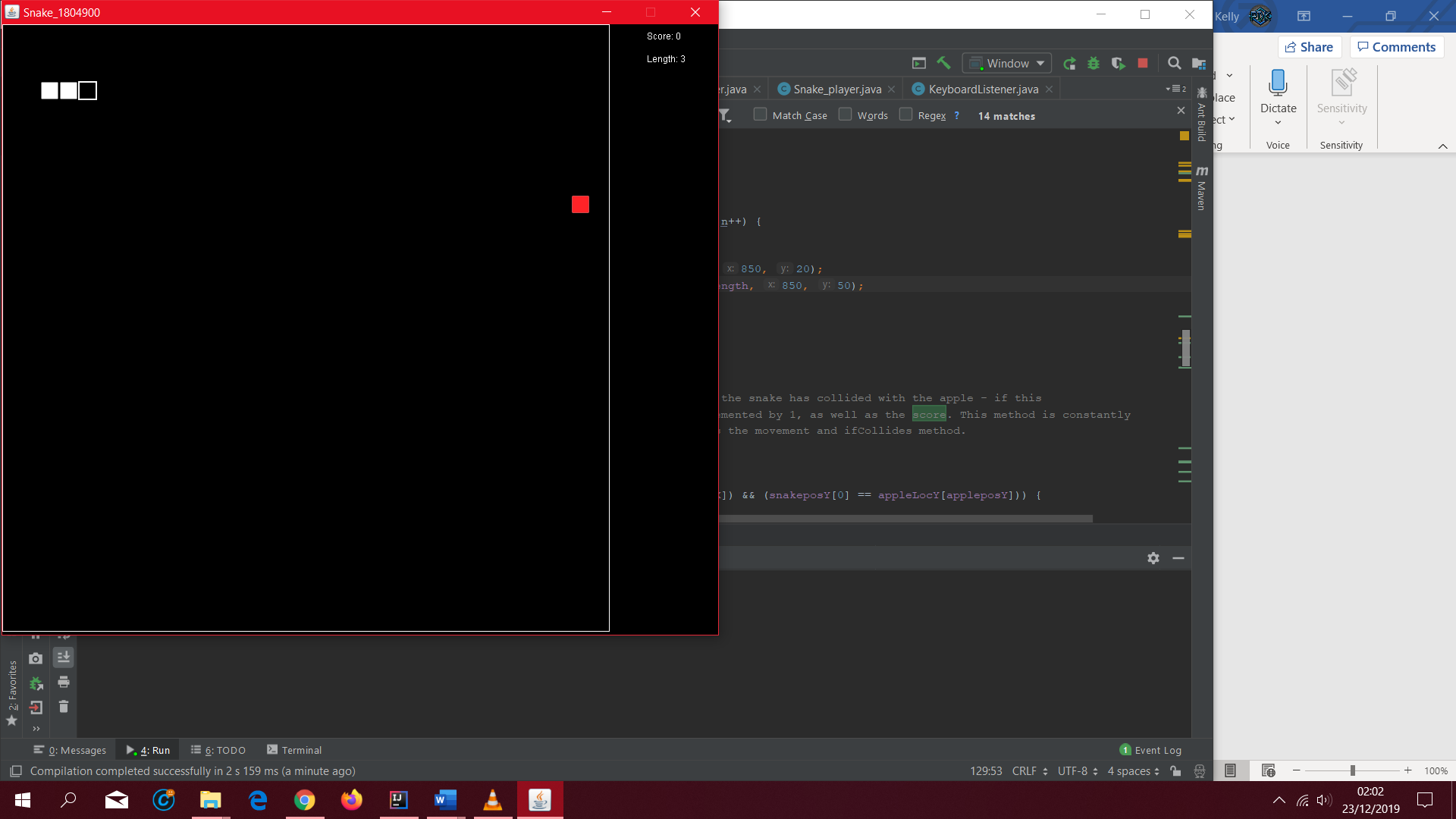
I have included comments within each class of my program, which explains the variables used and what each method does.

## Testing

### Task A

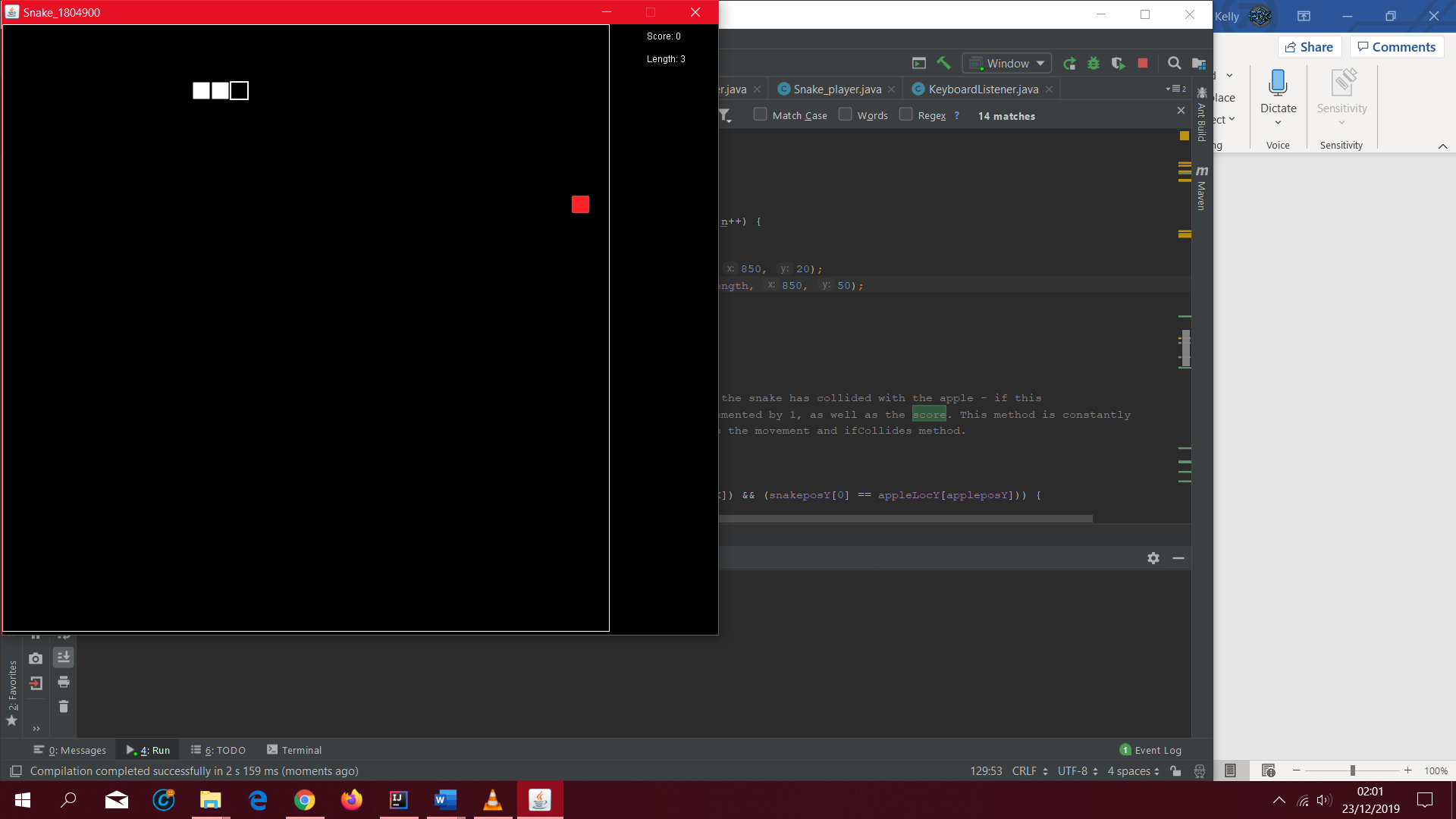
Screenshot 1.

Shows starting position of Snake



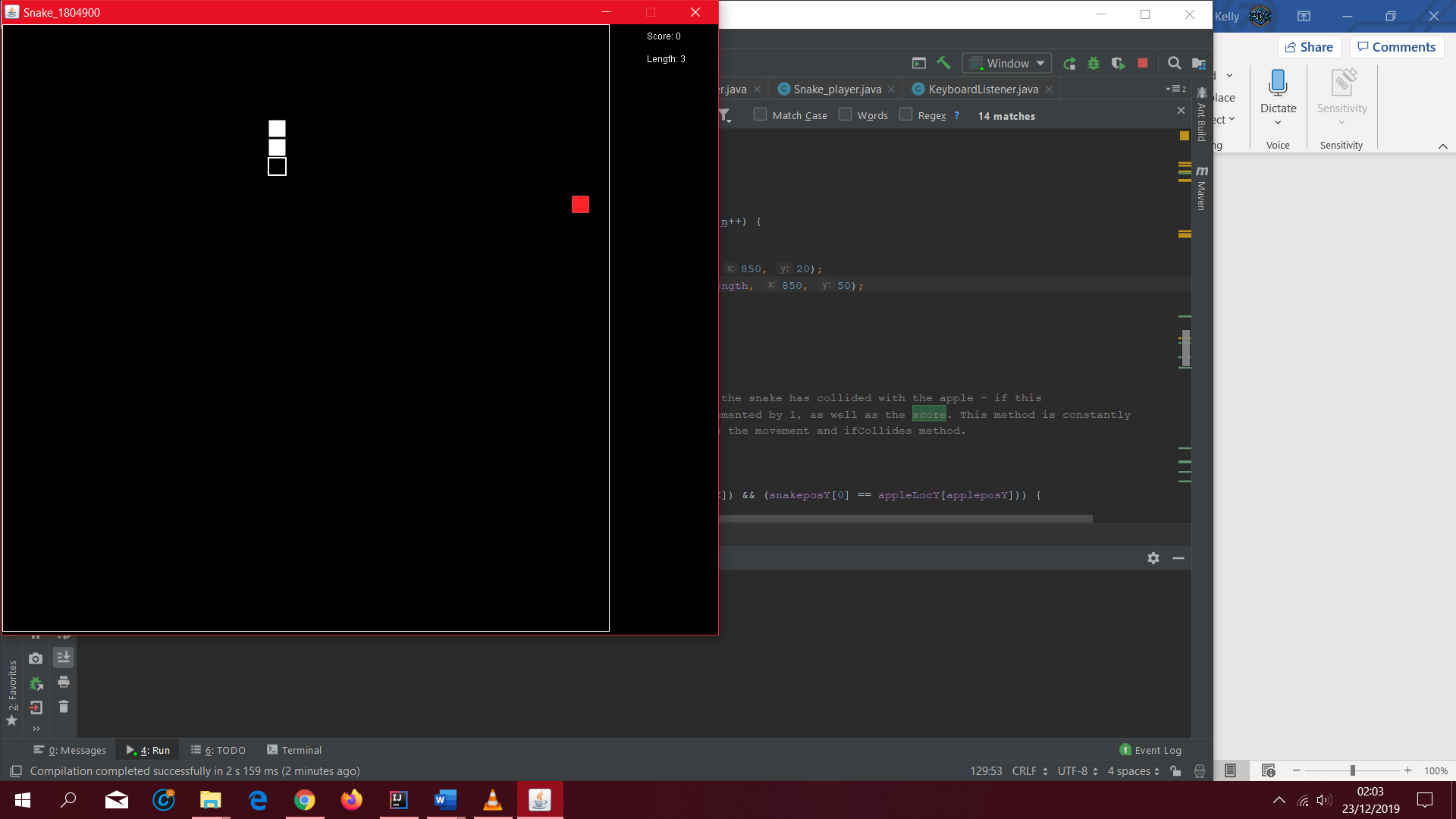
Screenshot 2.

Shows Snake moving to the right (starting direction / movement)



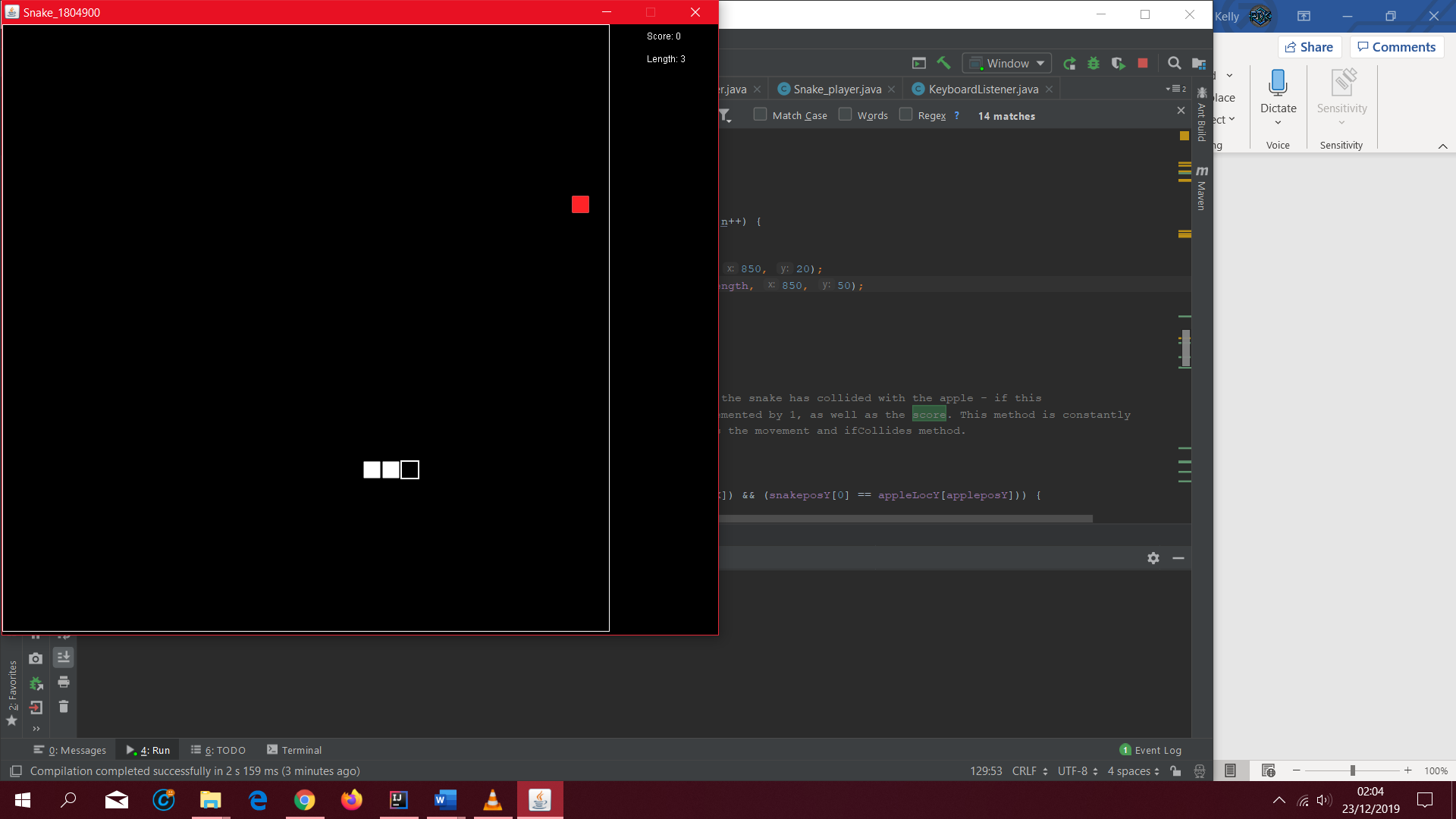
Screenshot 3.

Snake moving in downwards direction (User presses DOWN arrow key)



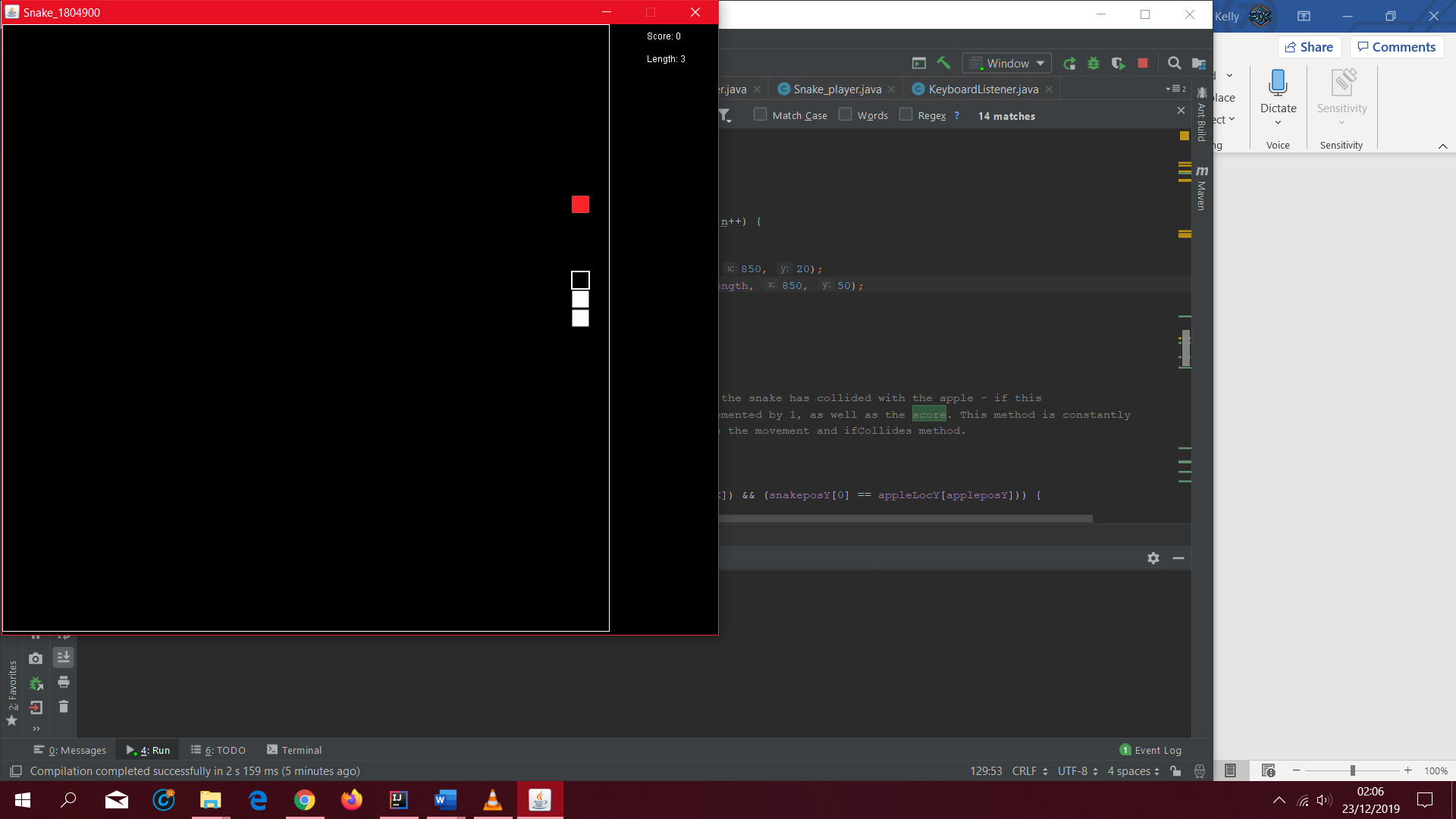
Screenshot 4.

Shows Snake moving to the right, after moving downwards (user presses RIGHT arrow key)



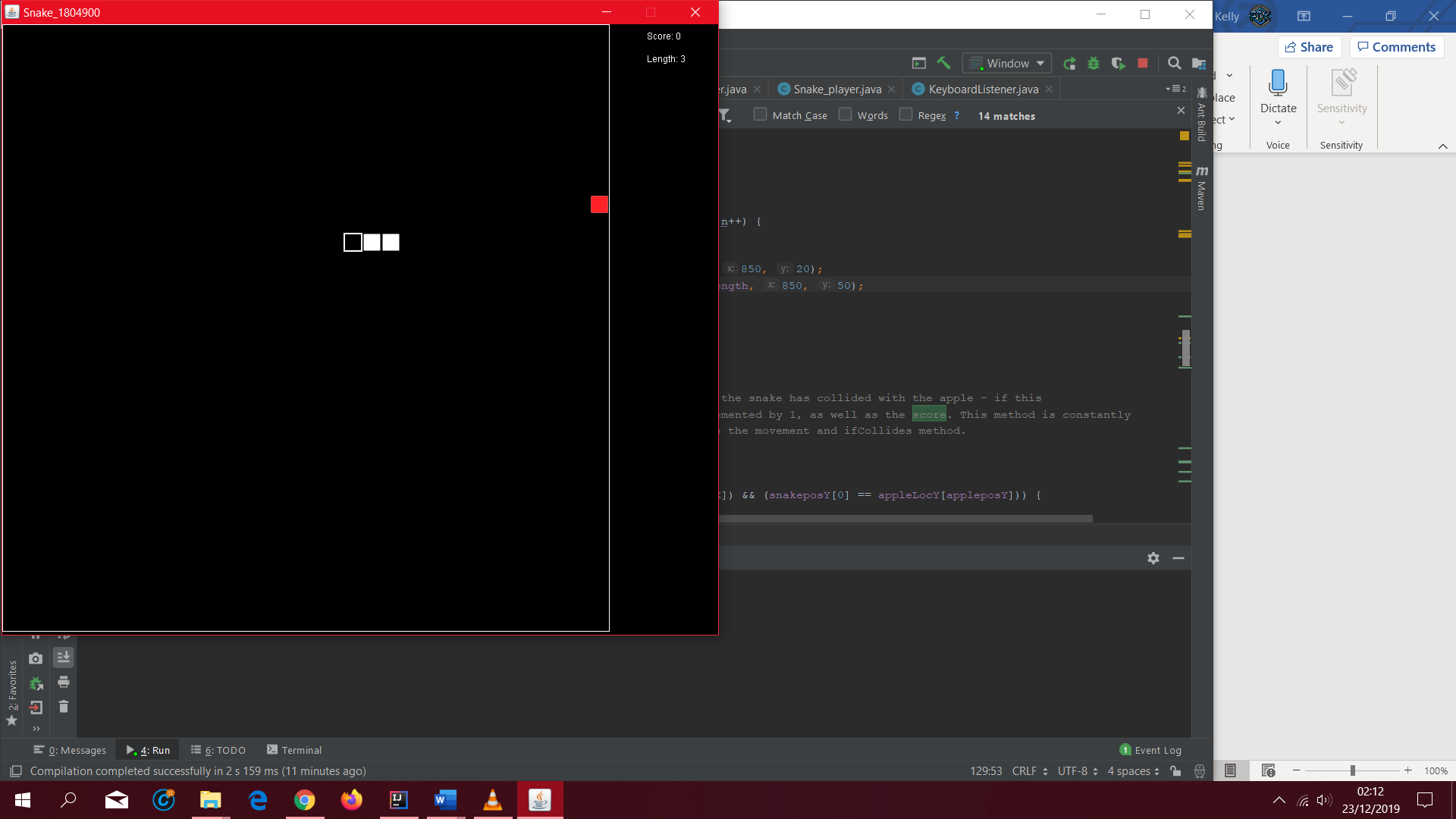
Screenshot 5.

Snake moving in upwards direction (user presses UP arrow key)



Screenshot 6.

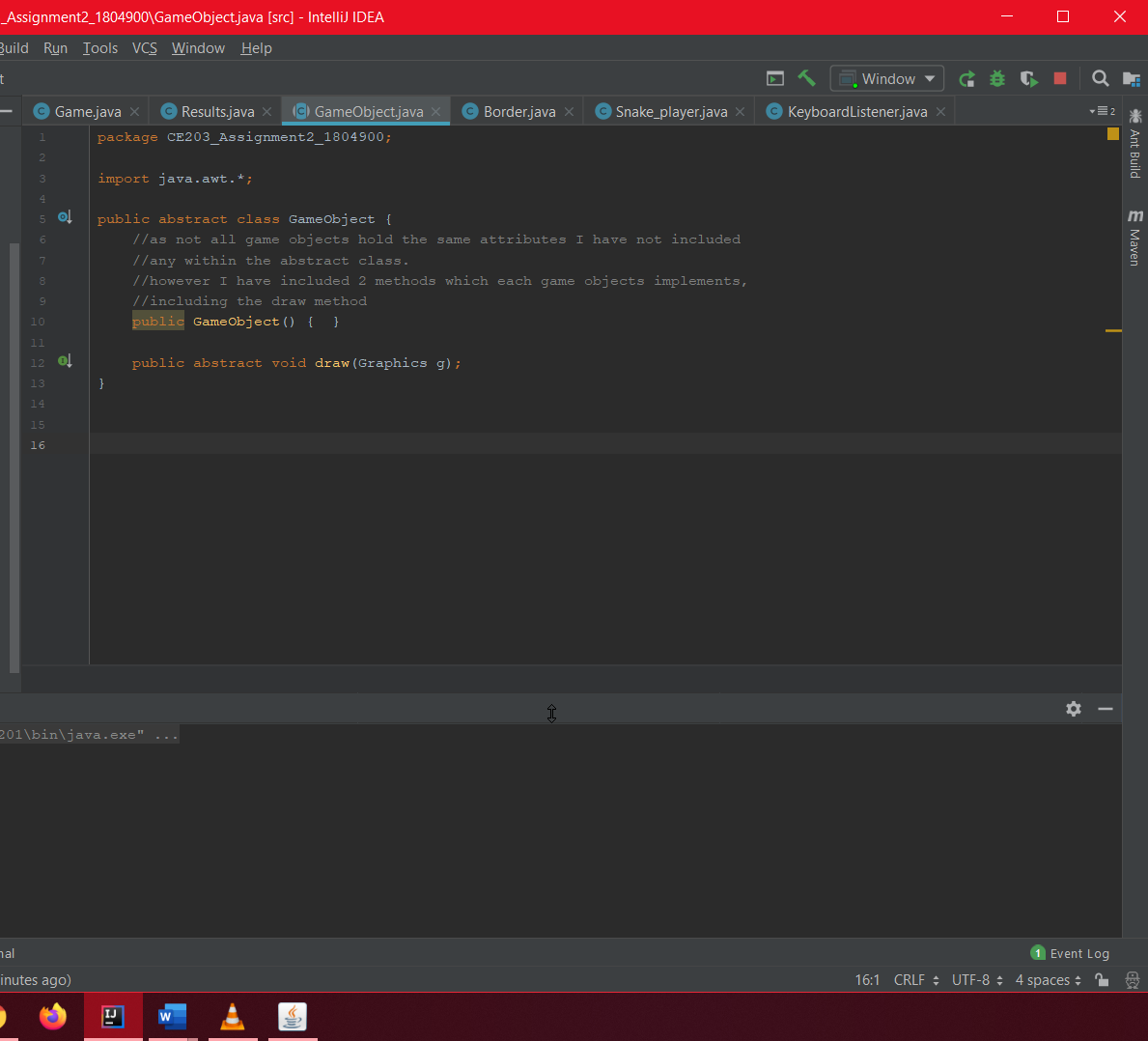
Snake moving Left (user presses LEFT arrow key)



### Task B

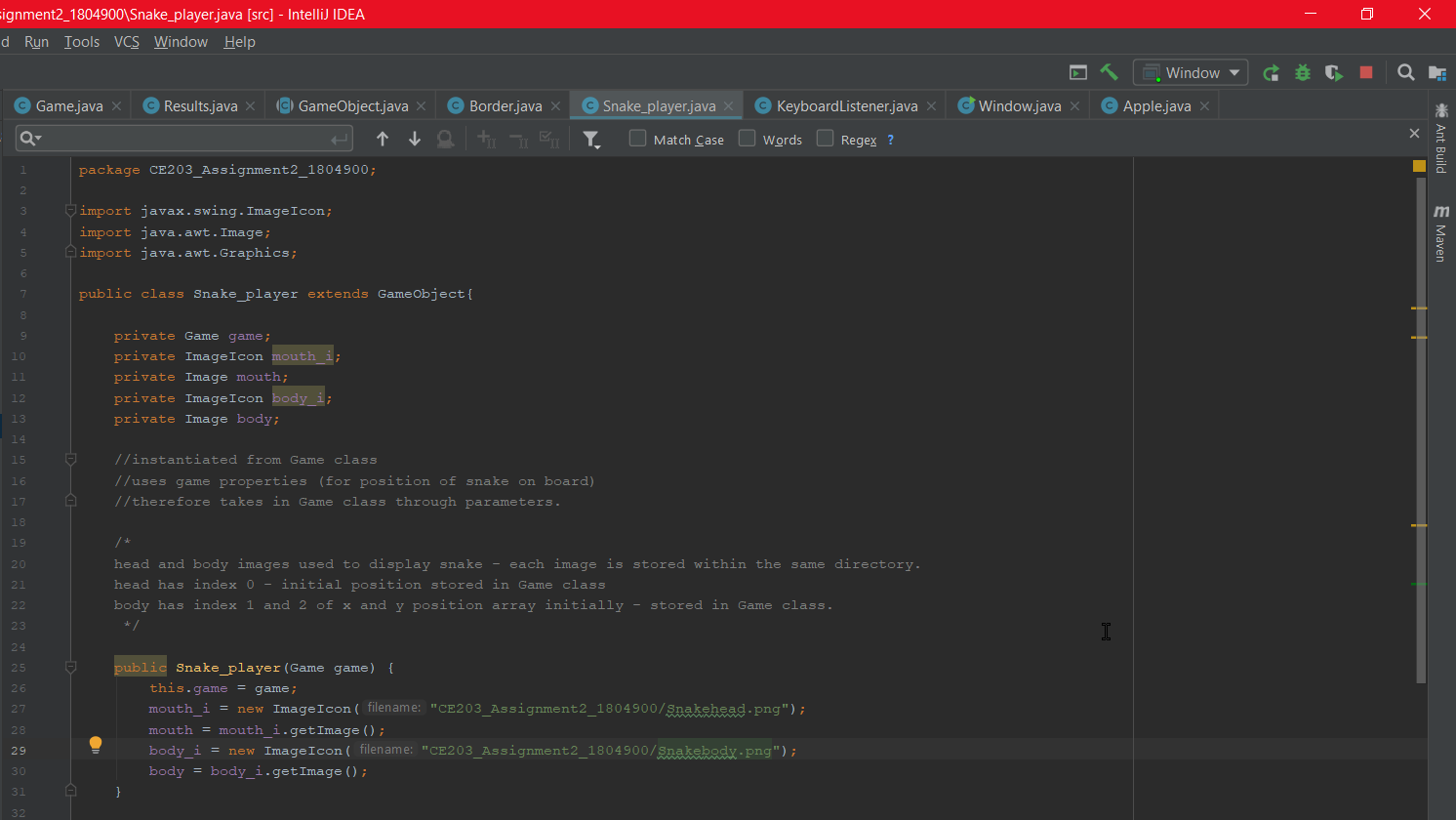
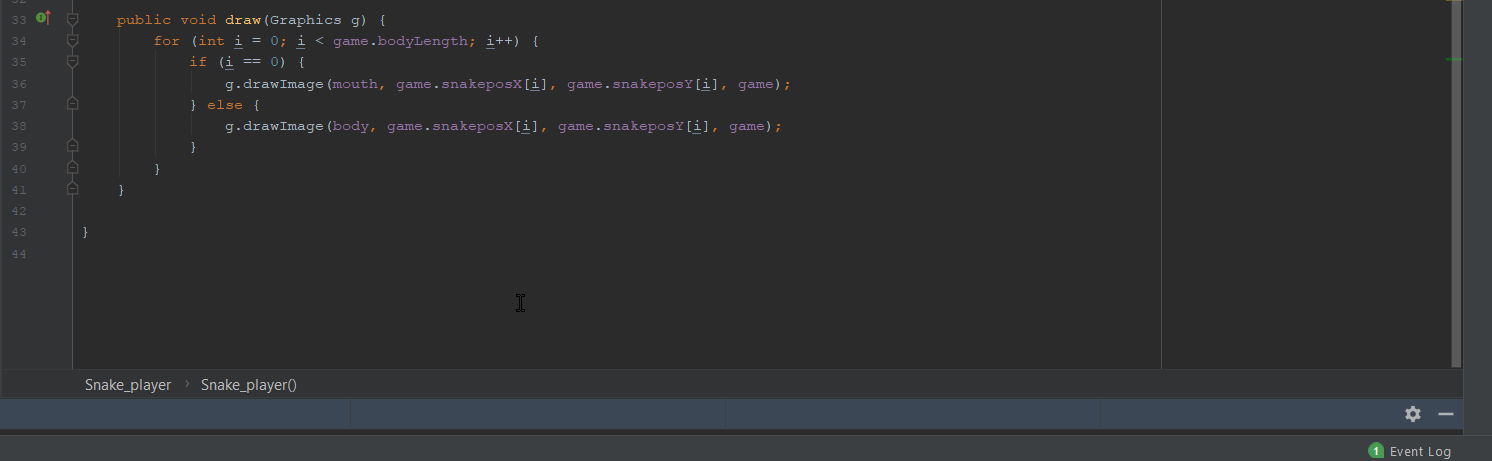
Screenshot 1.

GameObject Abstract class.



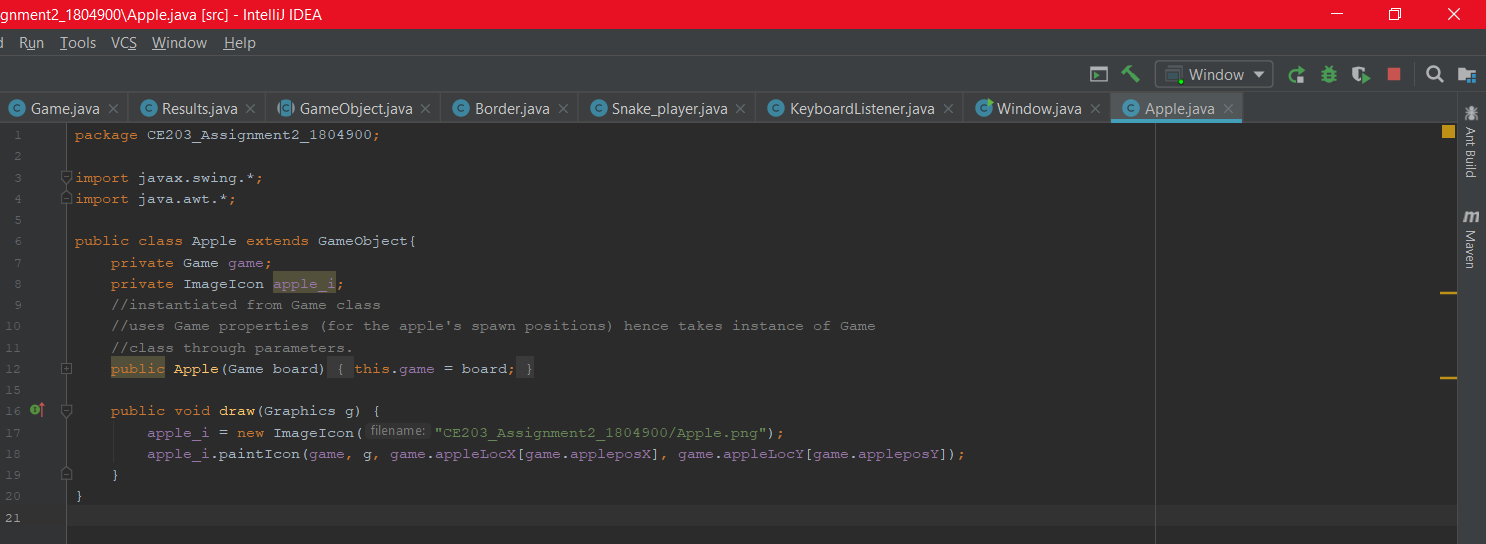
Screenshot2 and 3.

Snake\_player is its own class and encapsulates its own methods. Extends from GameObject abstract class.

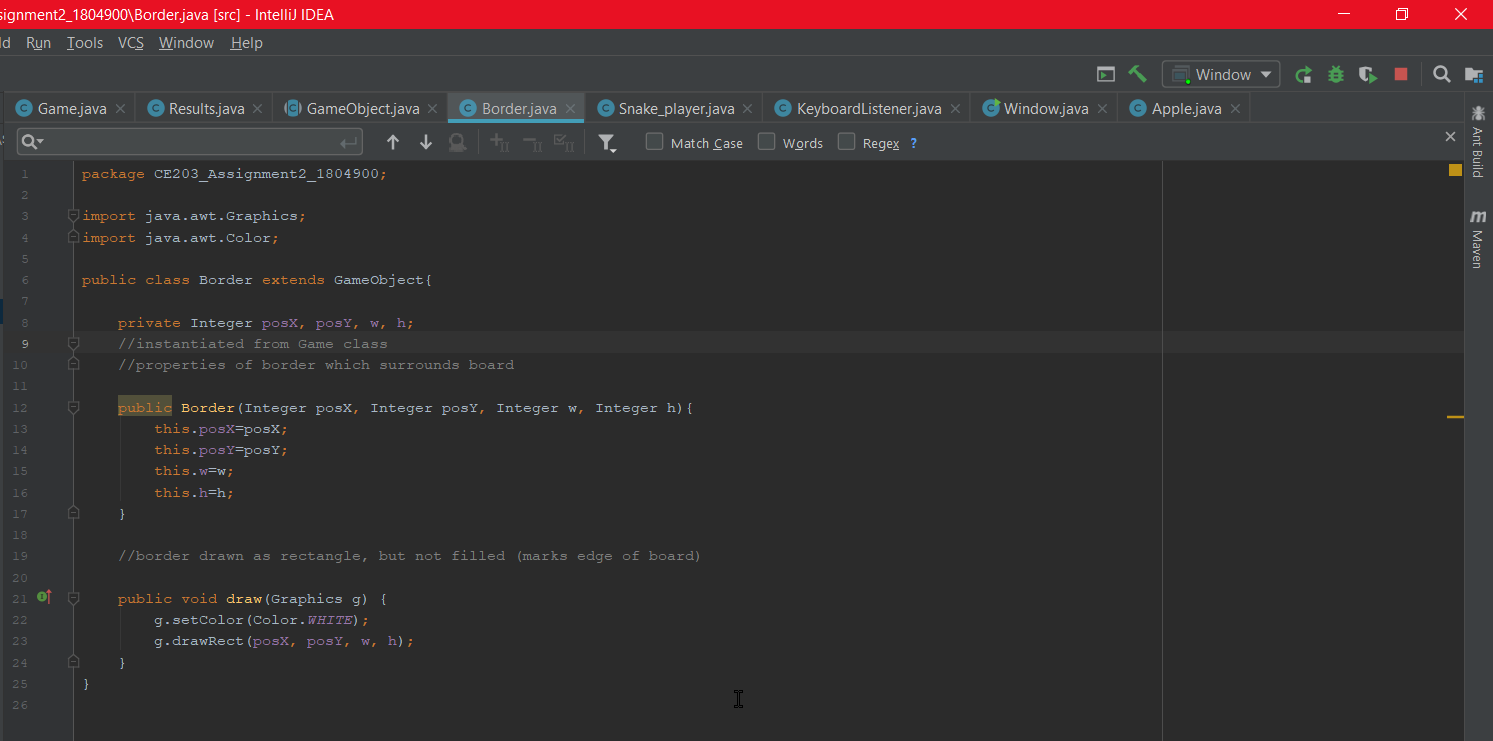
Screenshot4.

Apple within its own class, encapsulates methods. Extends from GameObject abstract class.



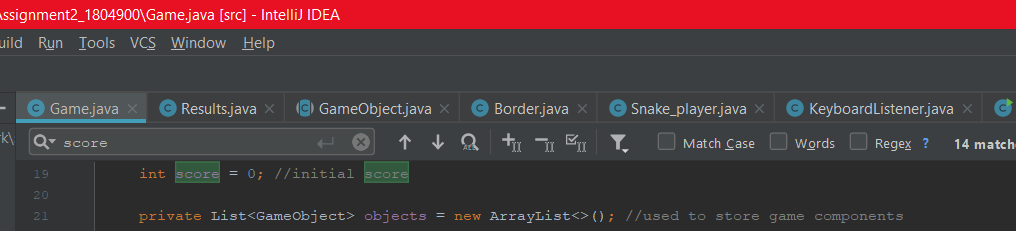
Screenshot5.

Border within own class, encapsulates methods. Extends from GameObject abstract class.



Screenshot6 and 7.

All game objects stored within a collection (List<GameObject> object), in Game class.

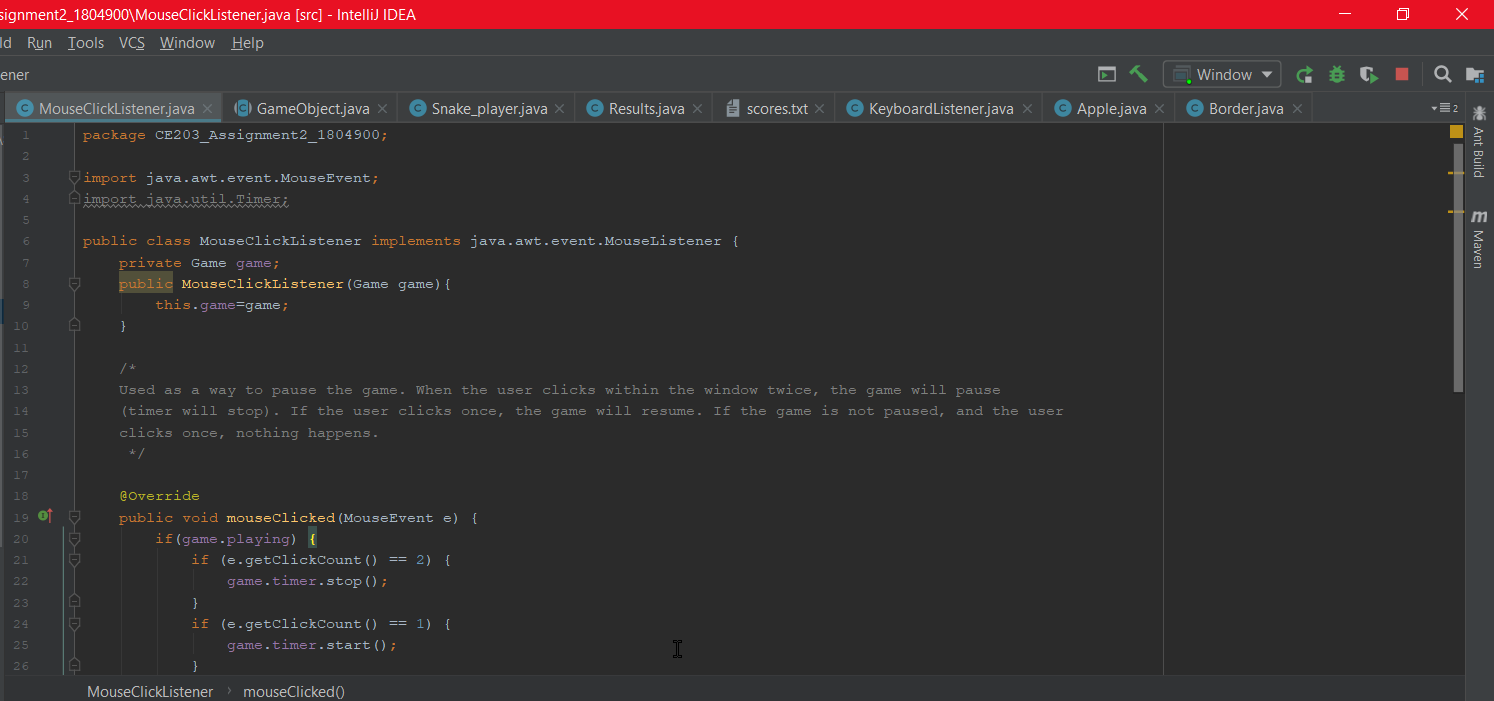




### Task C

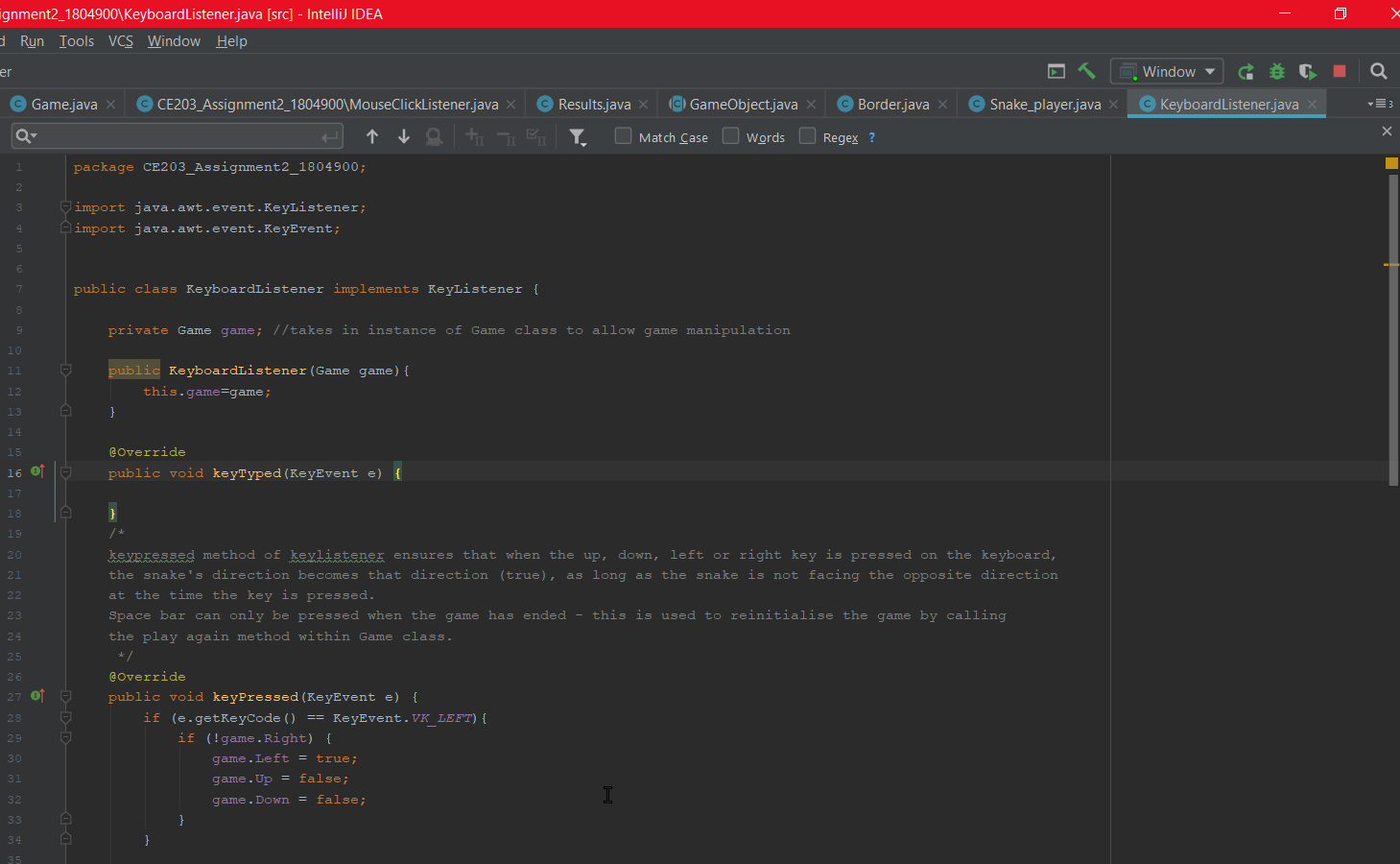
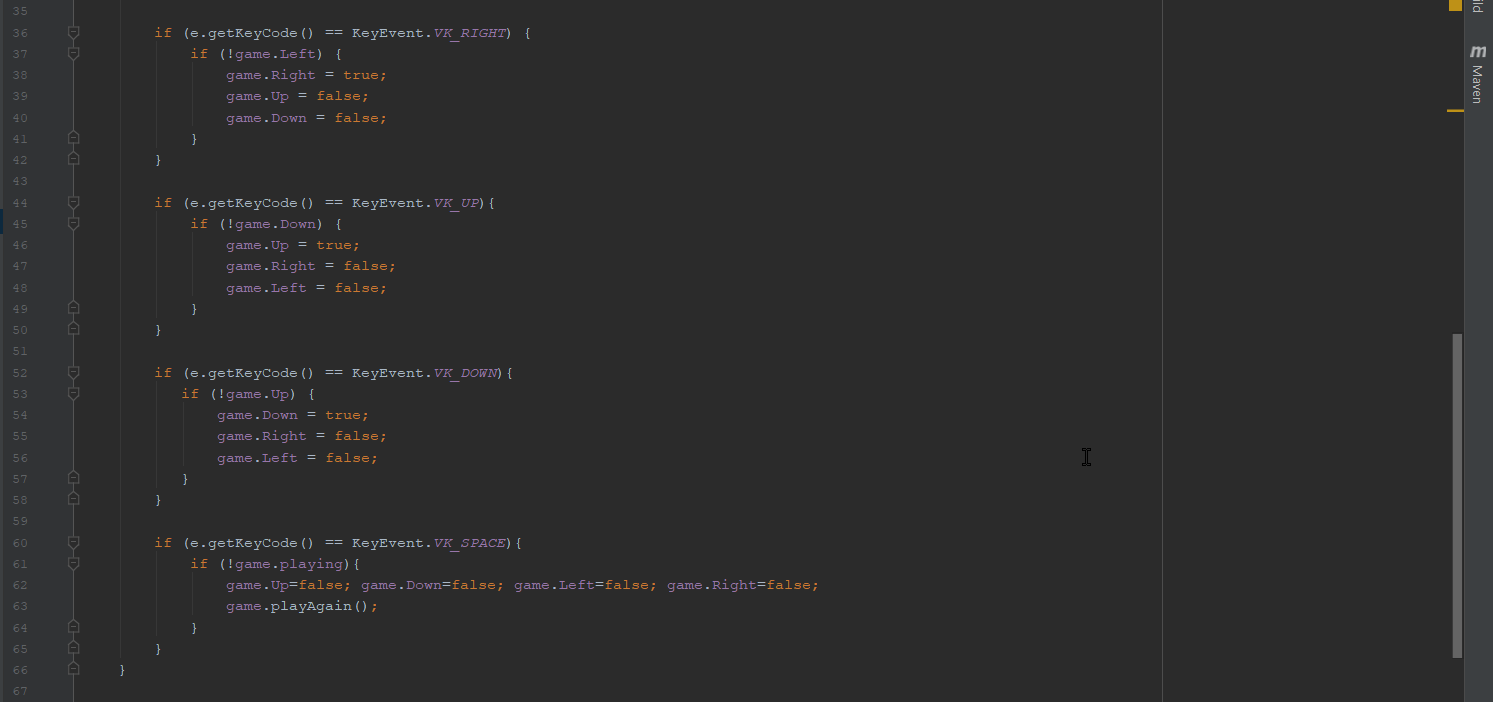
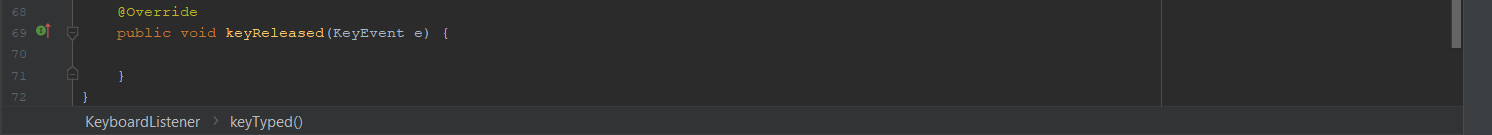
Screenshot1.

Shows MouseListener class, MouseClickListener



Screenshot2 and 3 and 4.

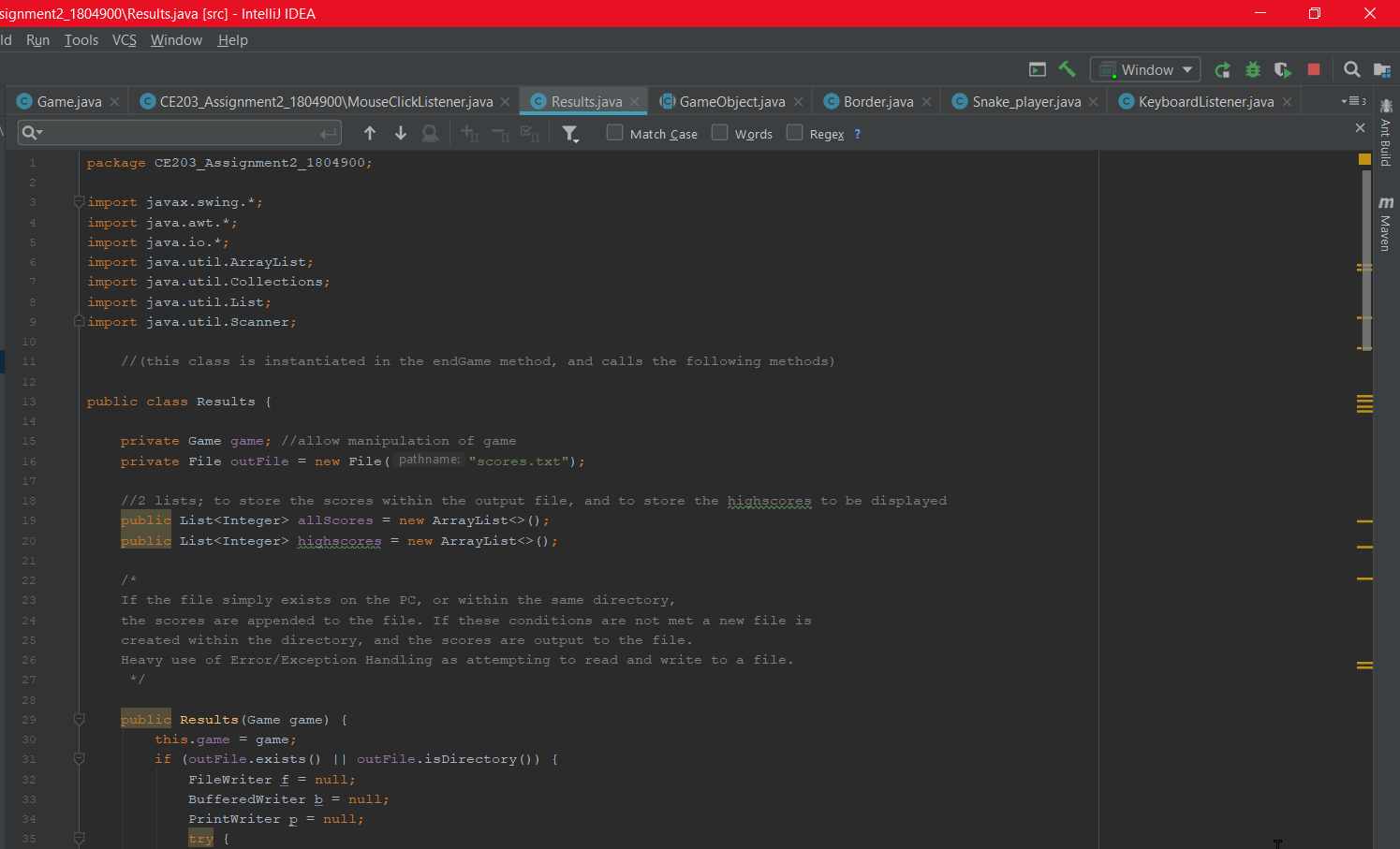
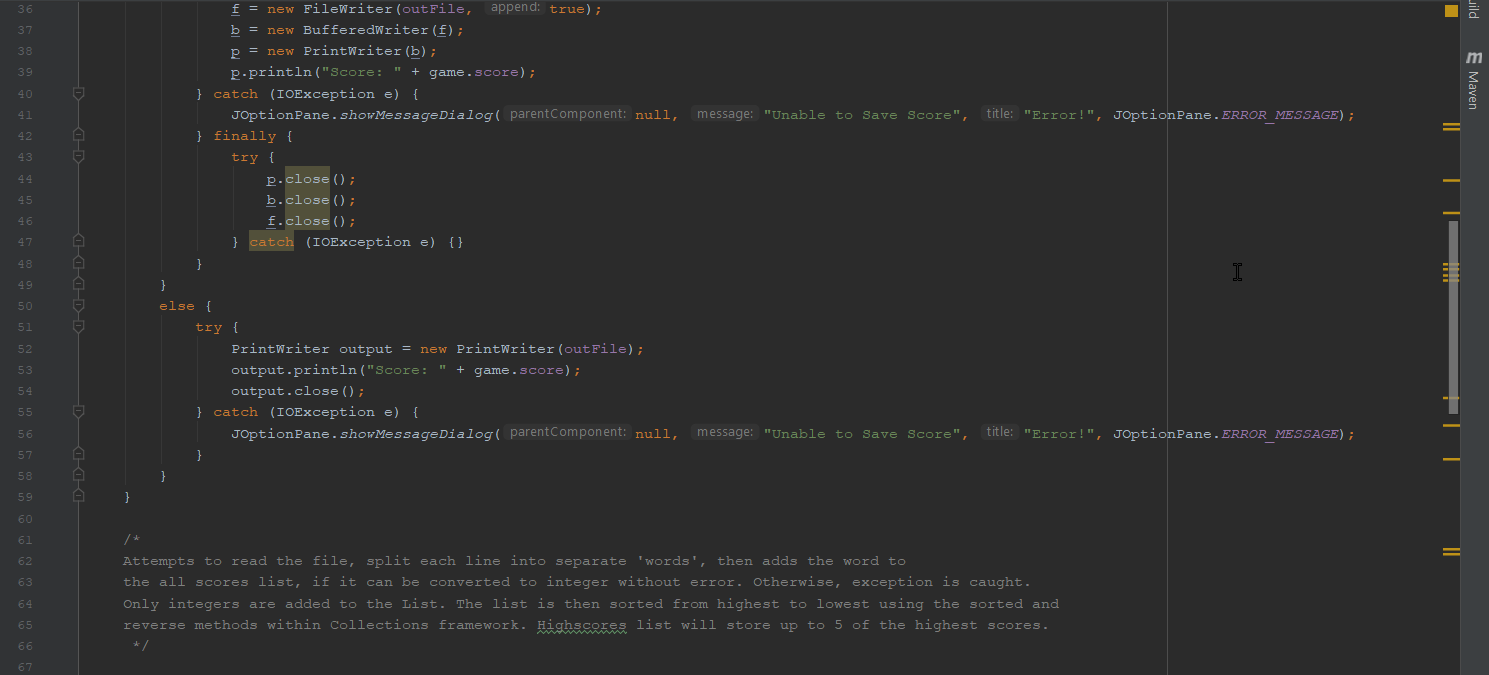
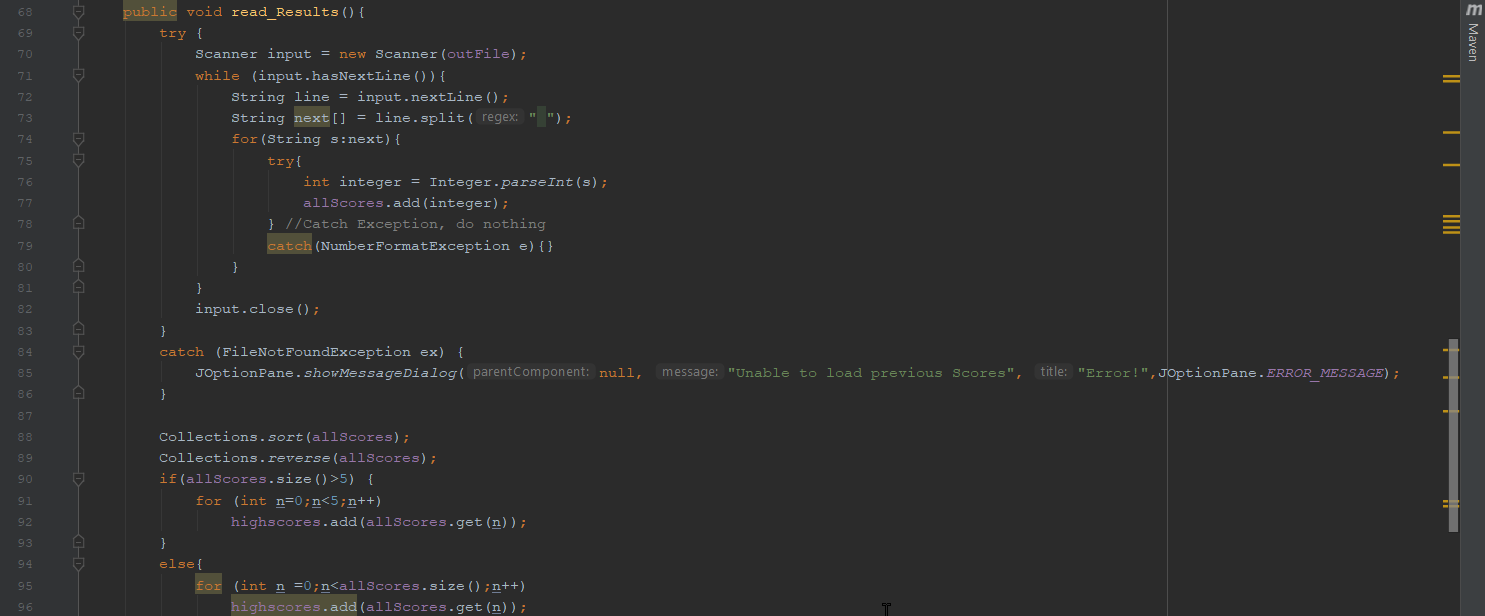
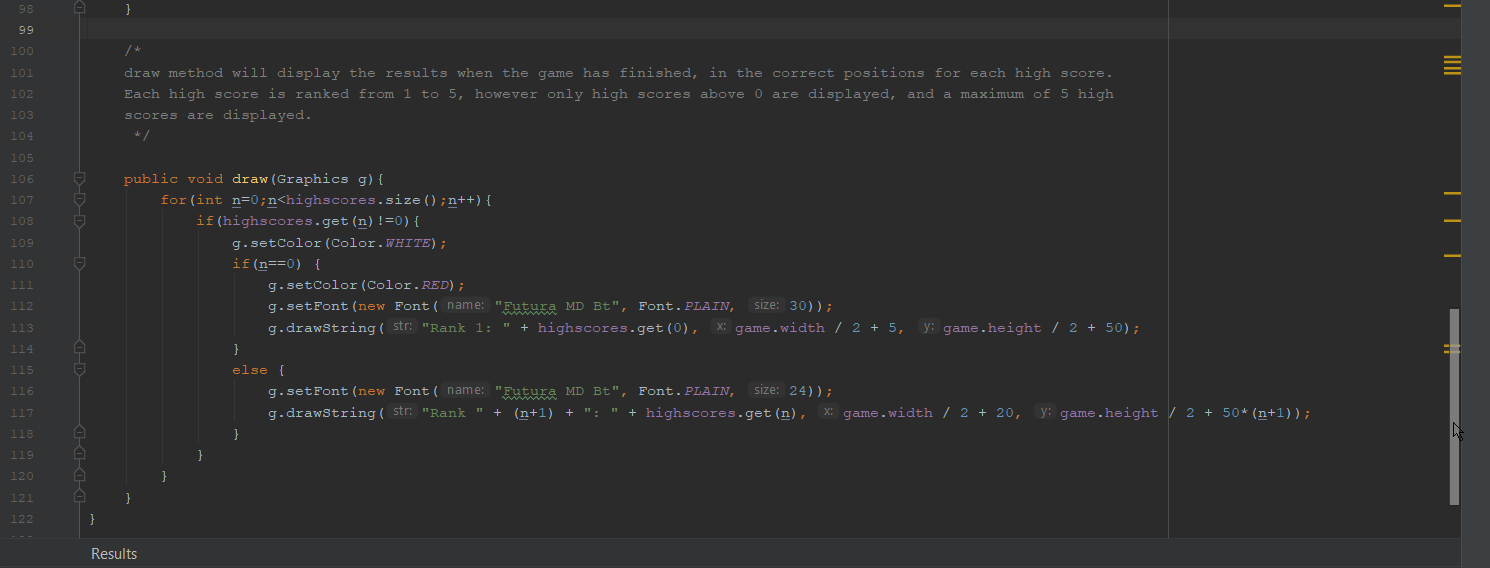
Shows KeyListener class, KeyboardListener

### Task D

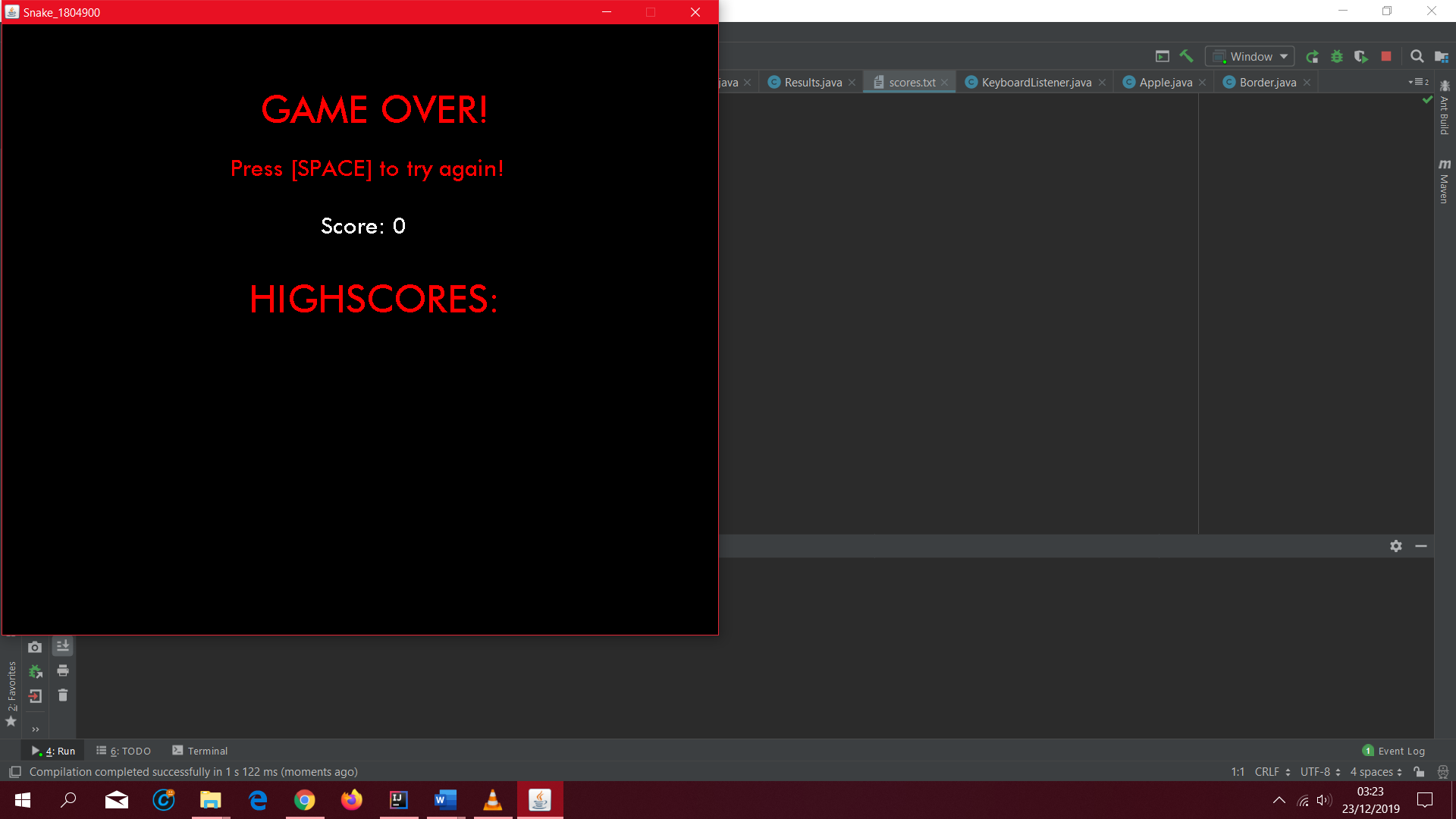
Screenshot1 and 2 and 3 and 4.

Shows Results class.

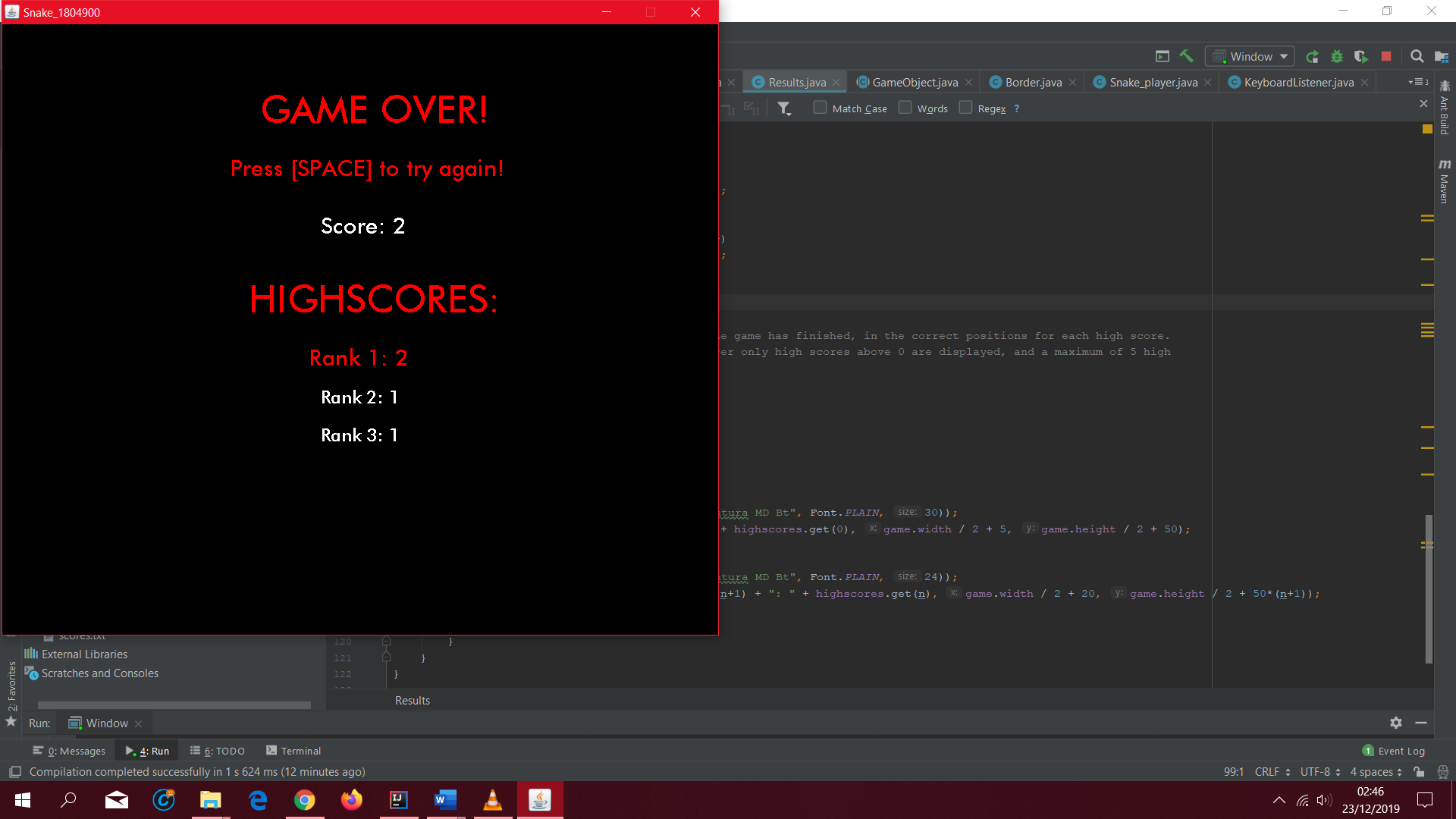
Screenshot5.

1st game, score of 0. There are no High scores yet.



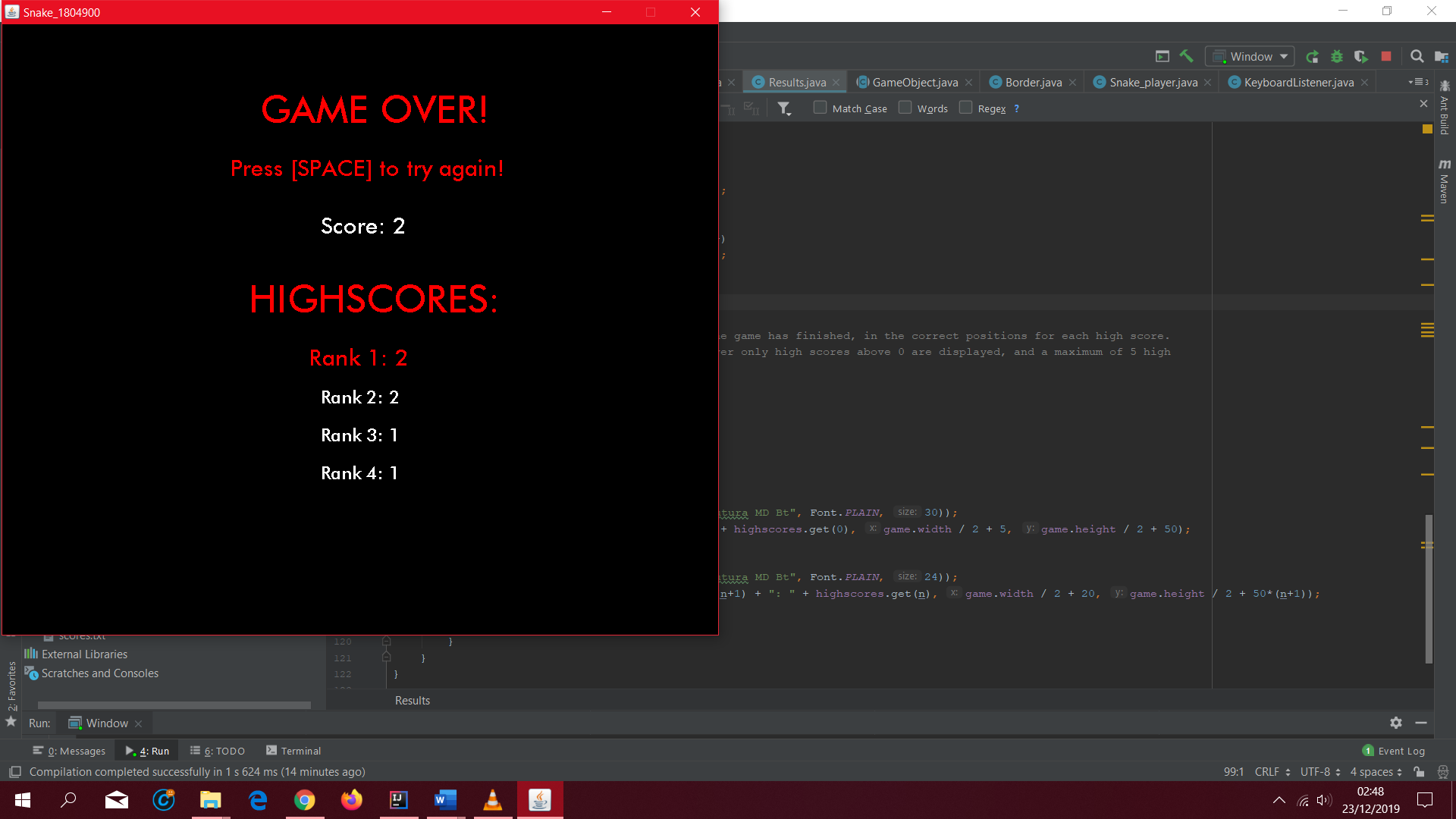
Screenshot6.

After playing 3 games, in which the score achieved was above 0.



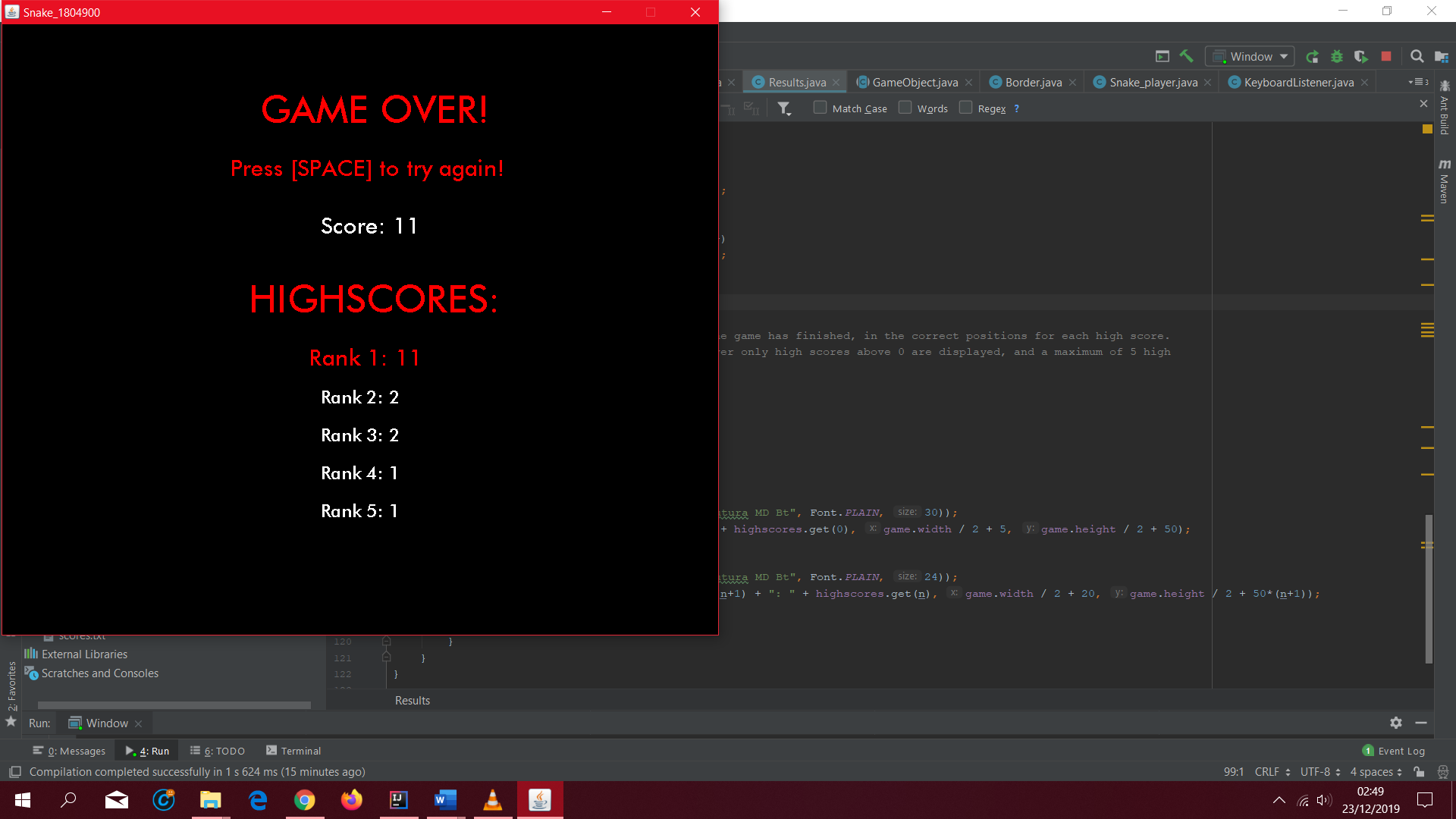
Screenshot7.

After 4 games



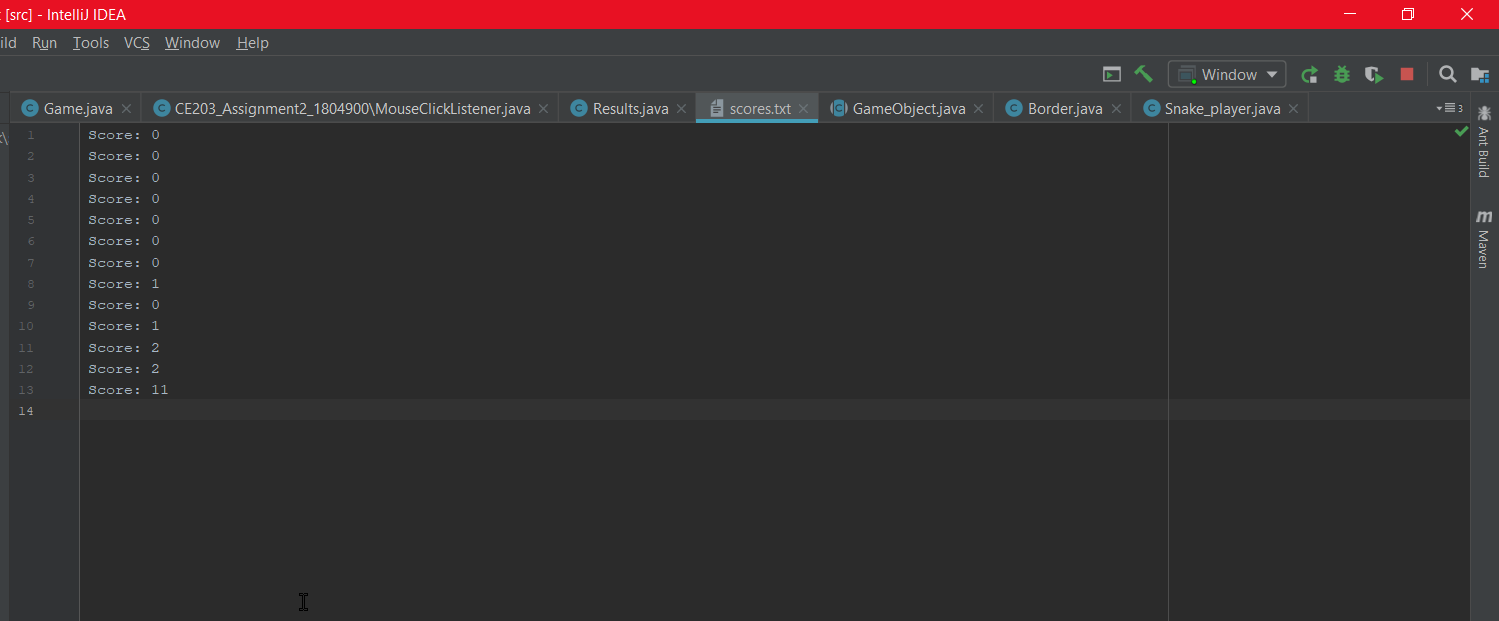
Screenshot8.

After 5 games



Screenshot9.

Showing scores.txt updated



### Task E

Shown within Code

### Task F

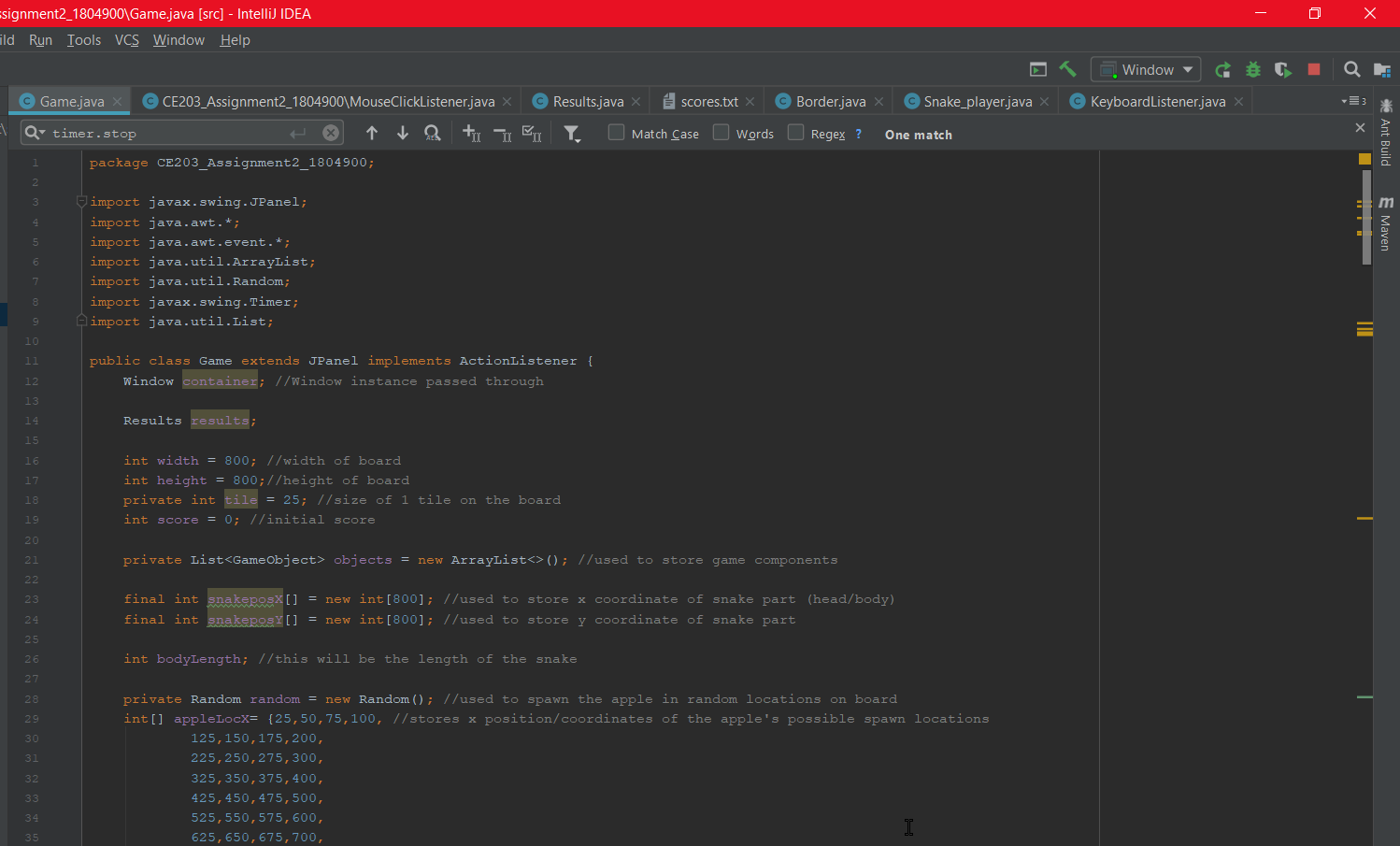
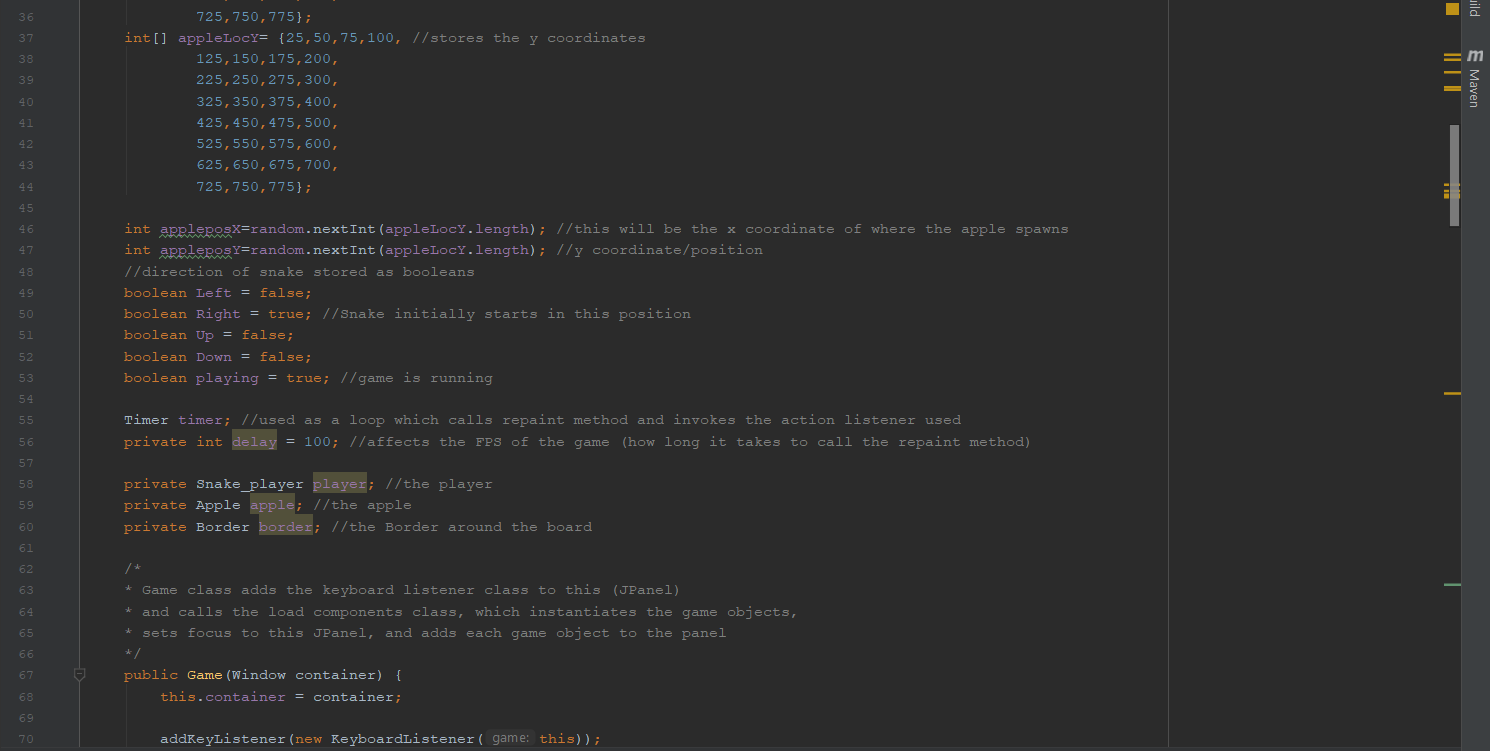
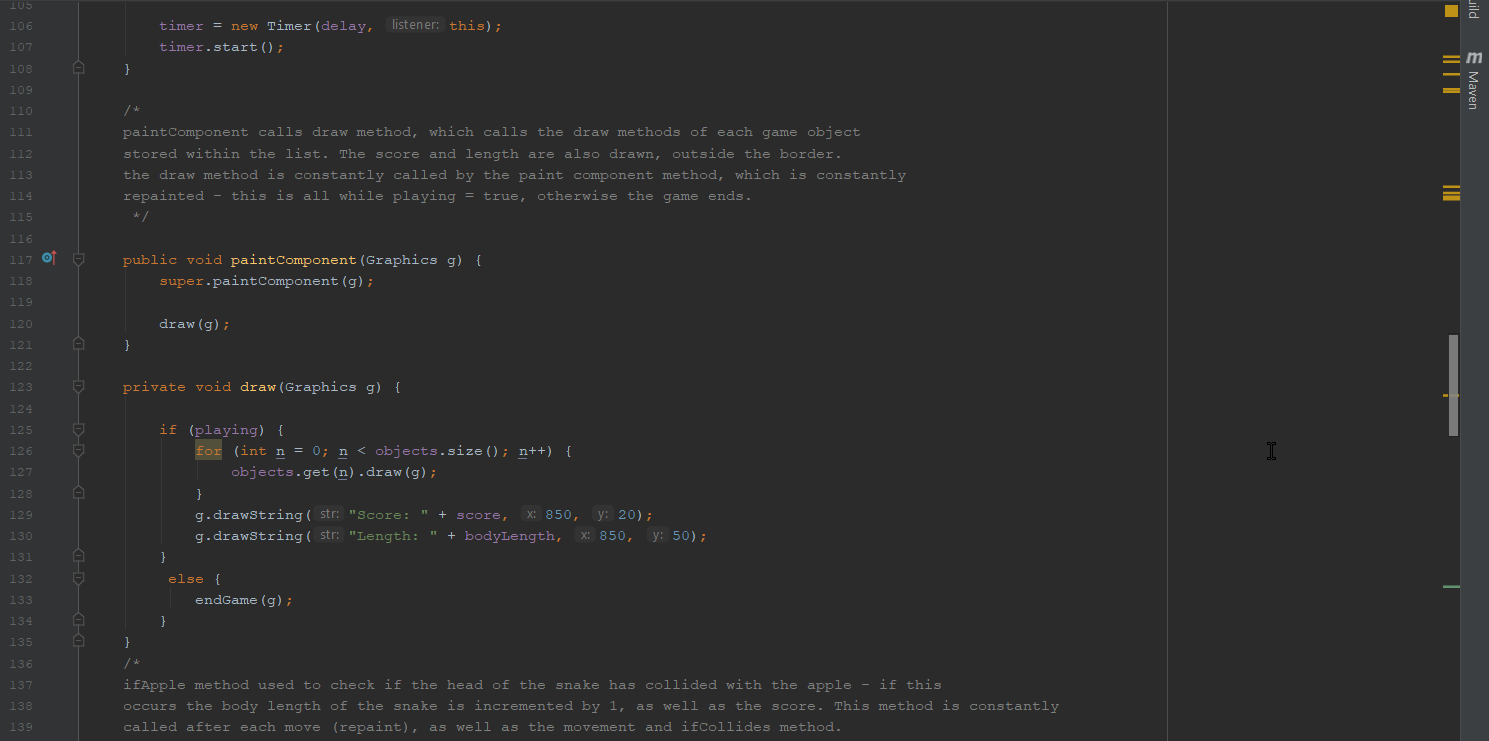
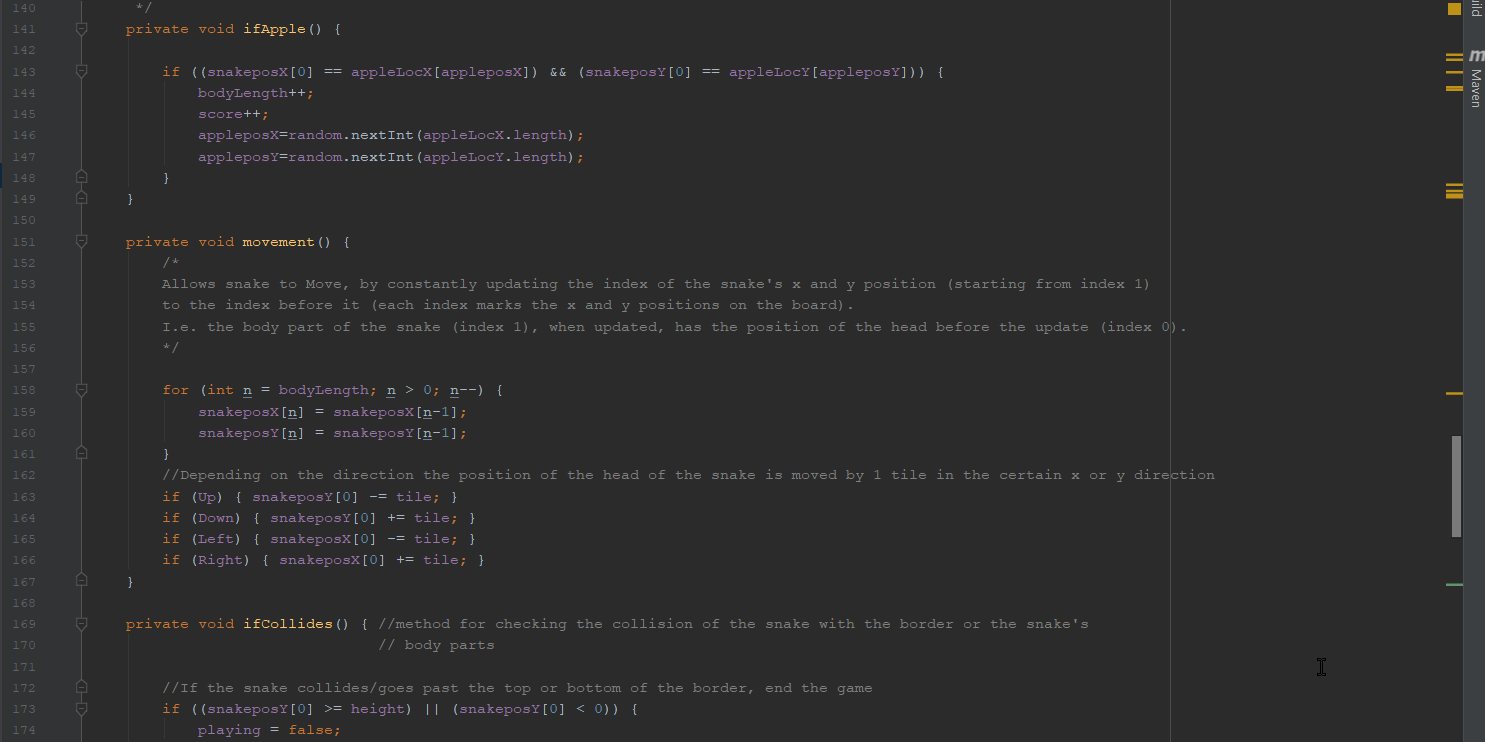
Screenshot1.

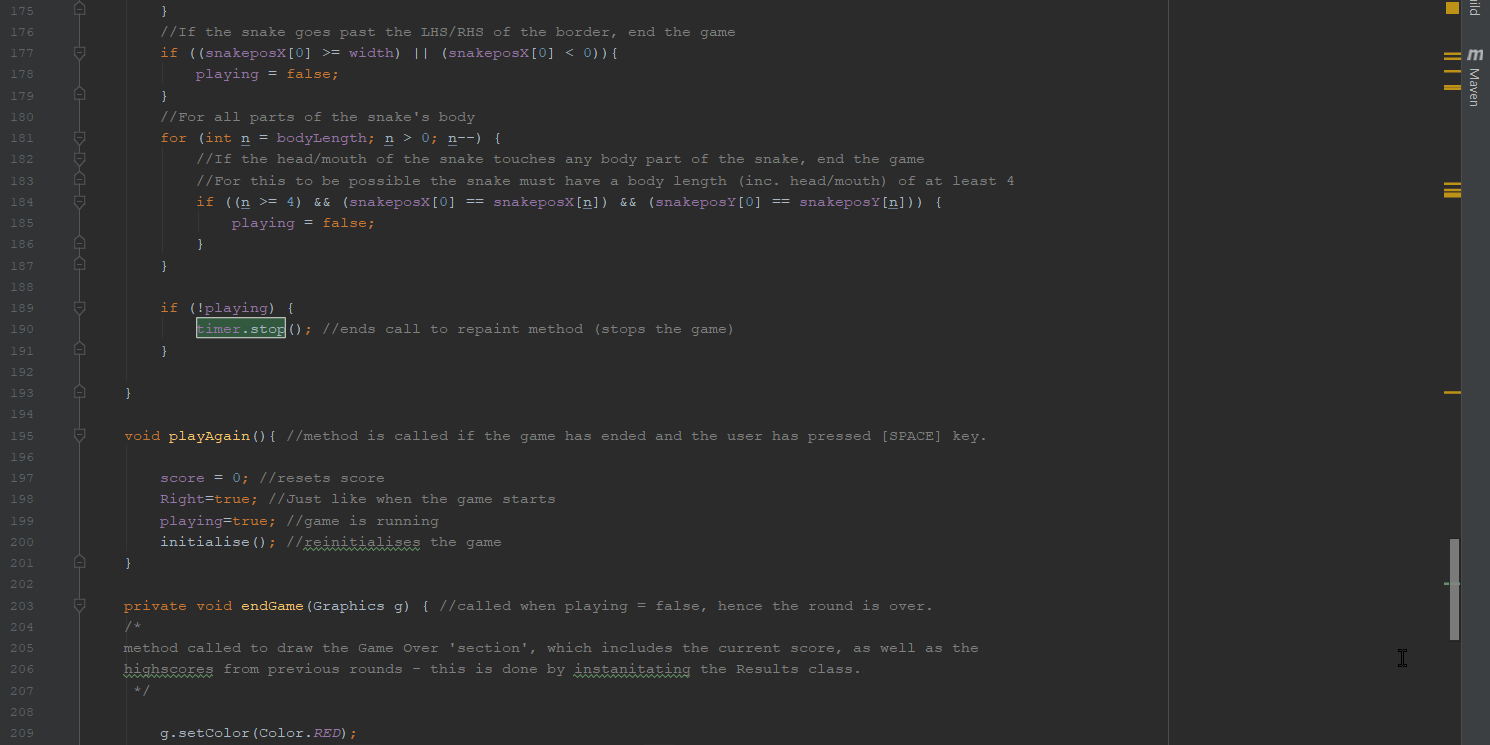
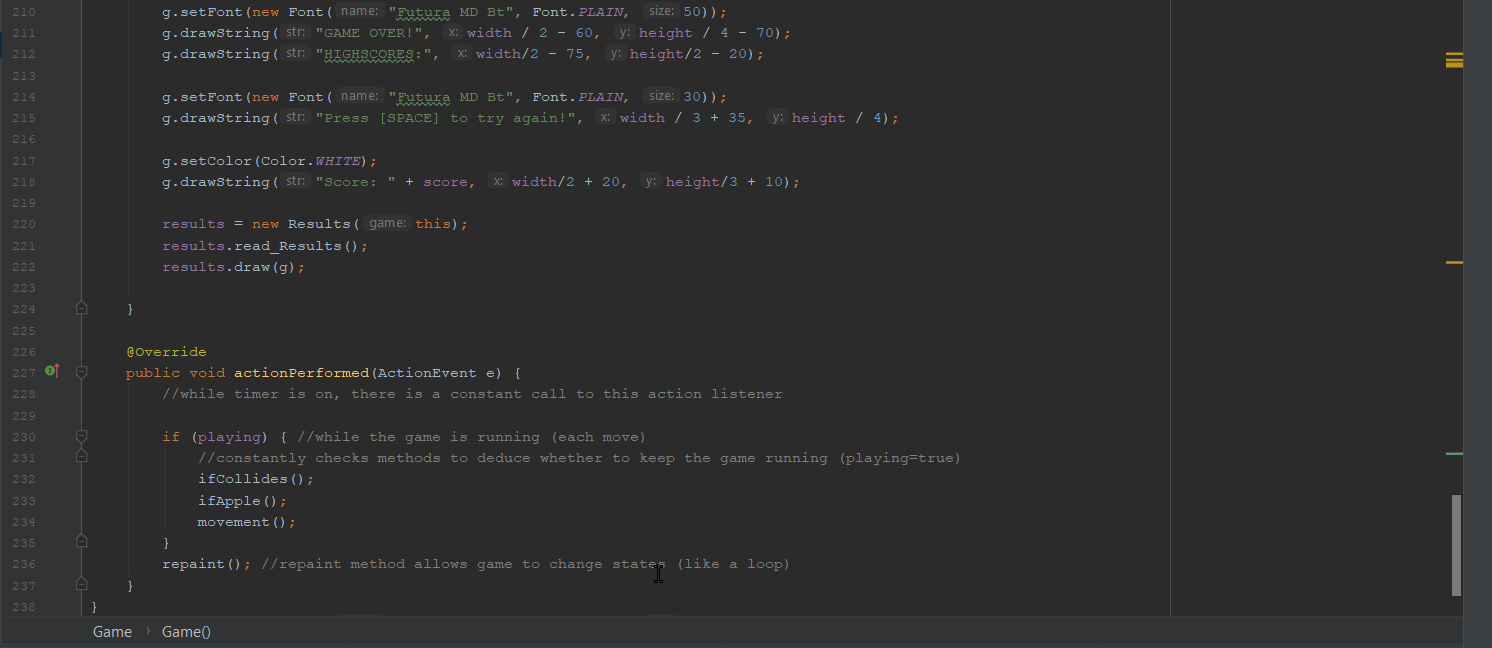
Evidence of Window class and comments.



Screenshot2, 3, 4, 5, 6, 7, 8.

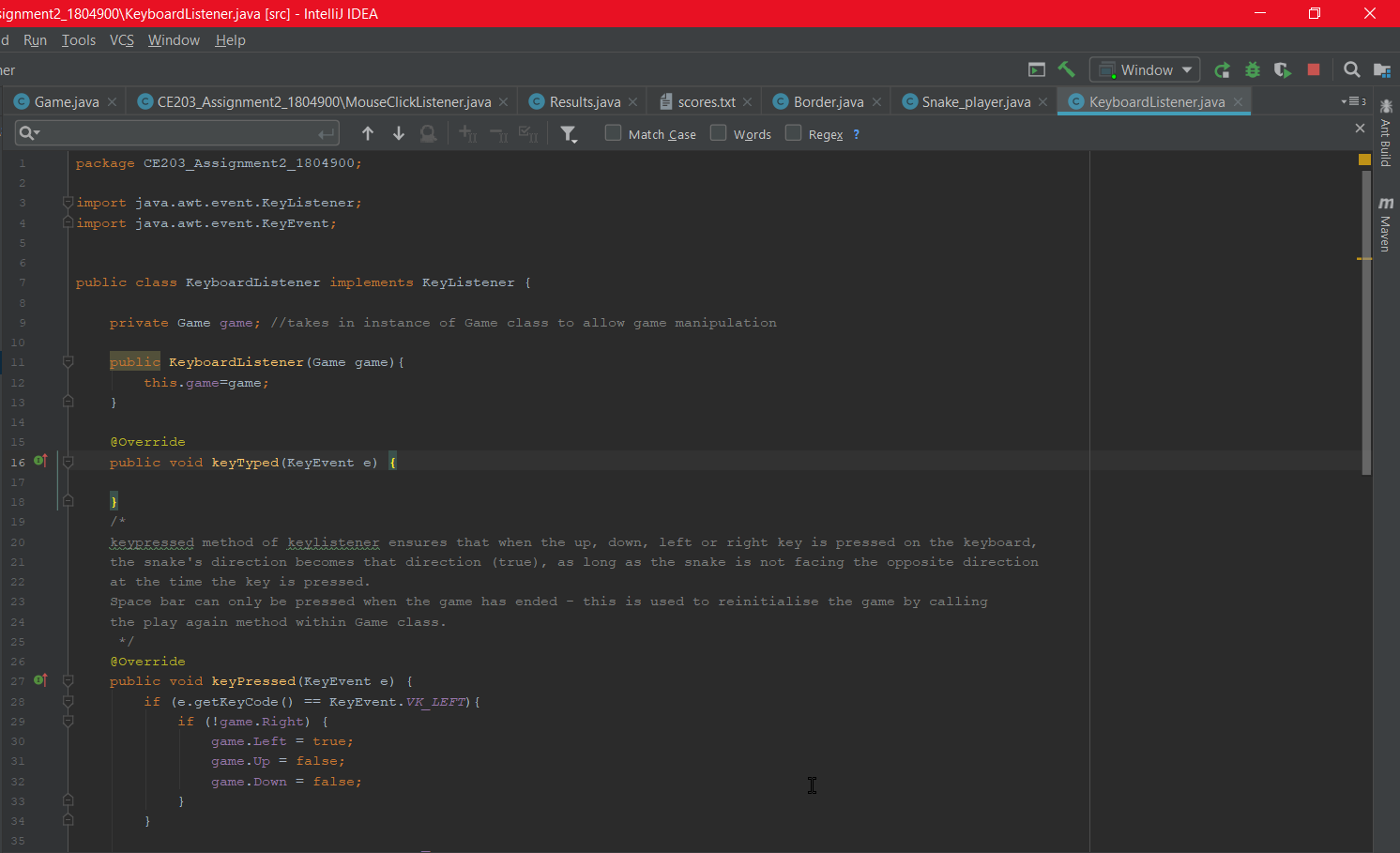
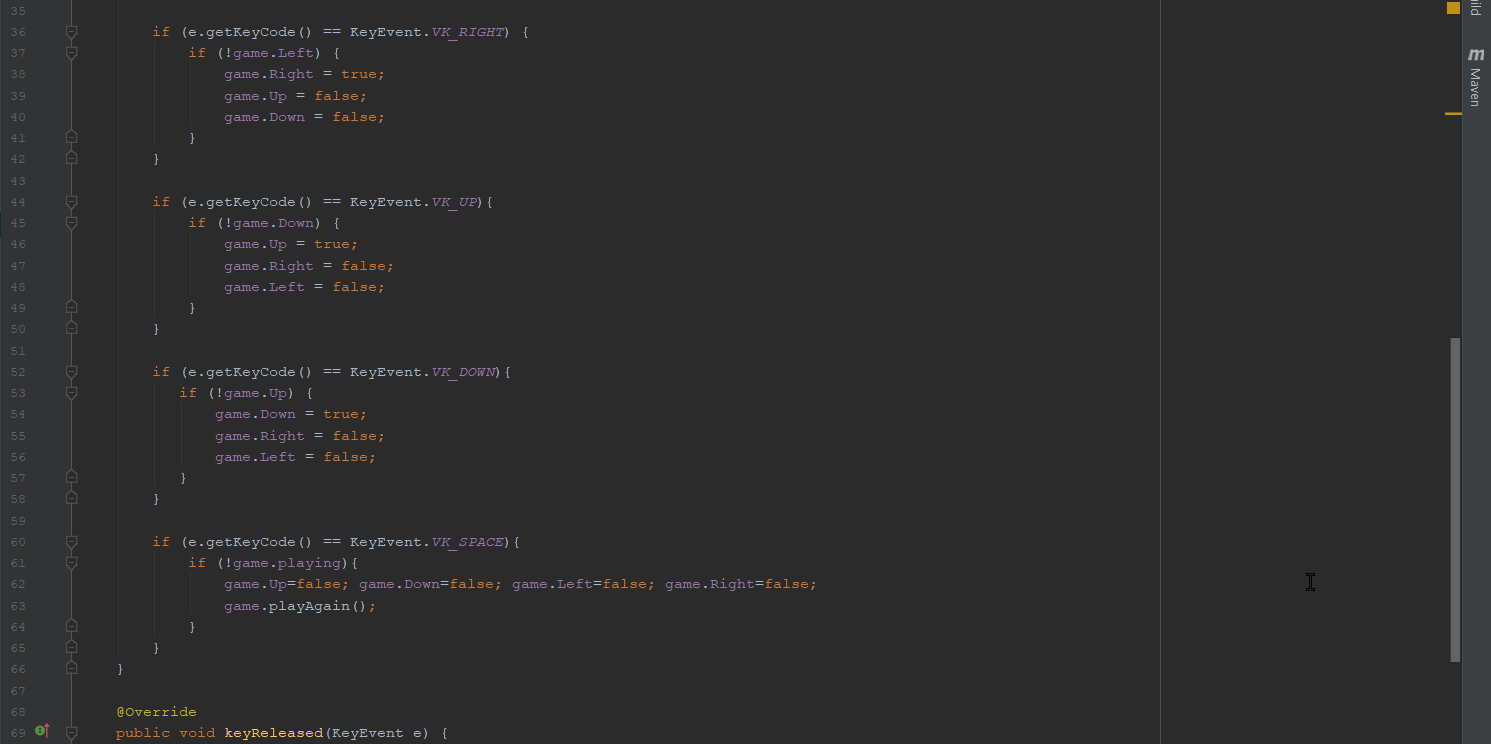
Proof of Game class and commenting.

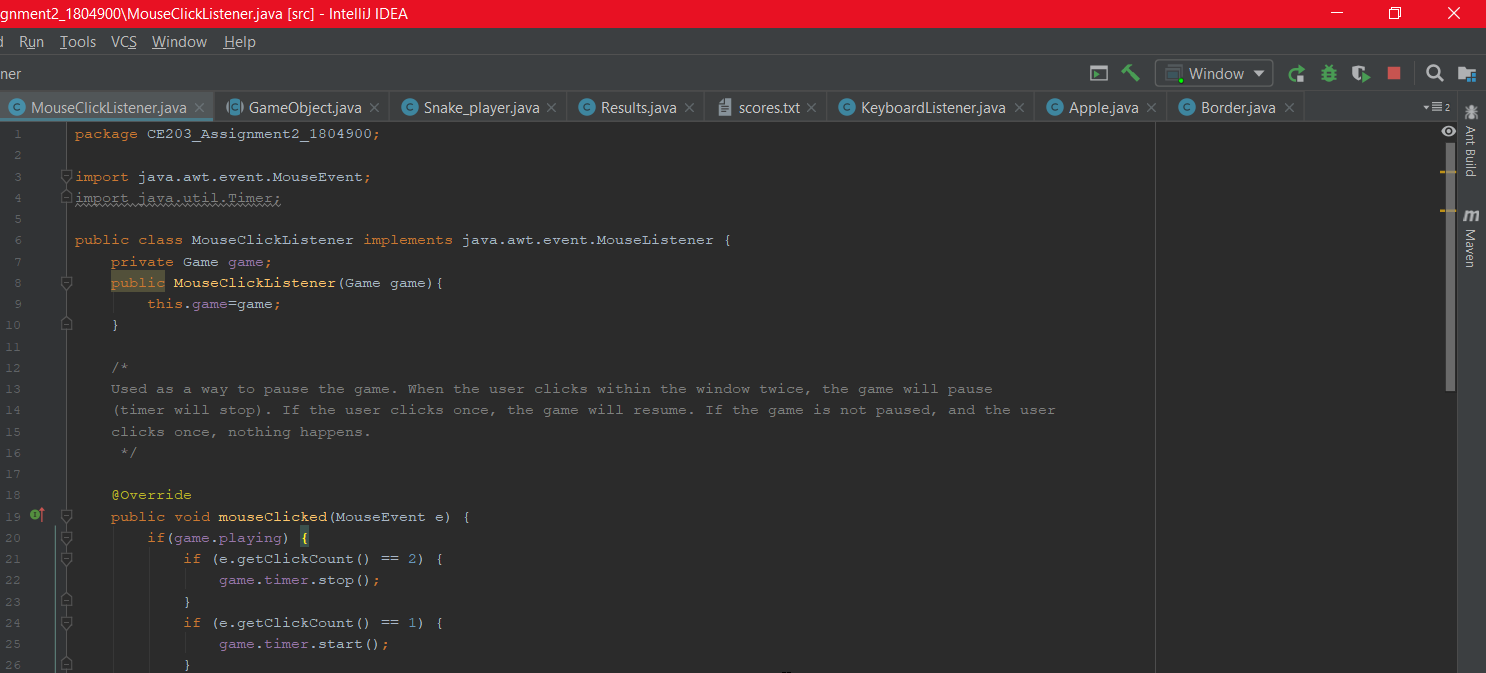
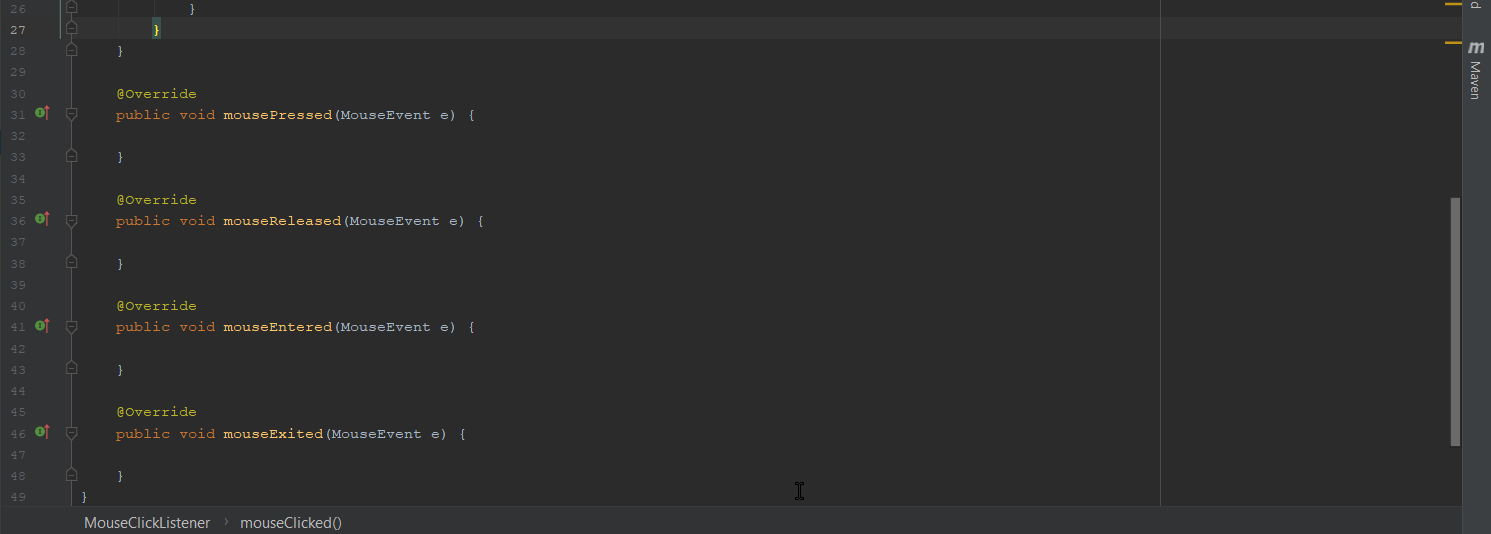
Screenshot 9 and 10 and 11.

Proof of KeyboardListener class with commenting

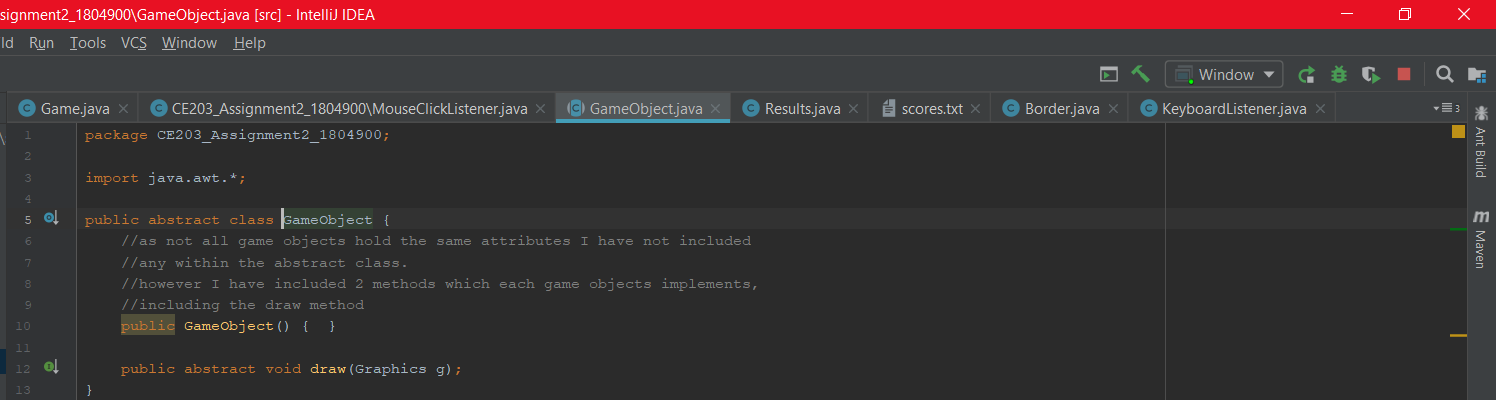
Screenshot 12 and 13.

Proof of MouseClickListener class with commenting

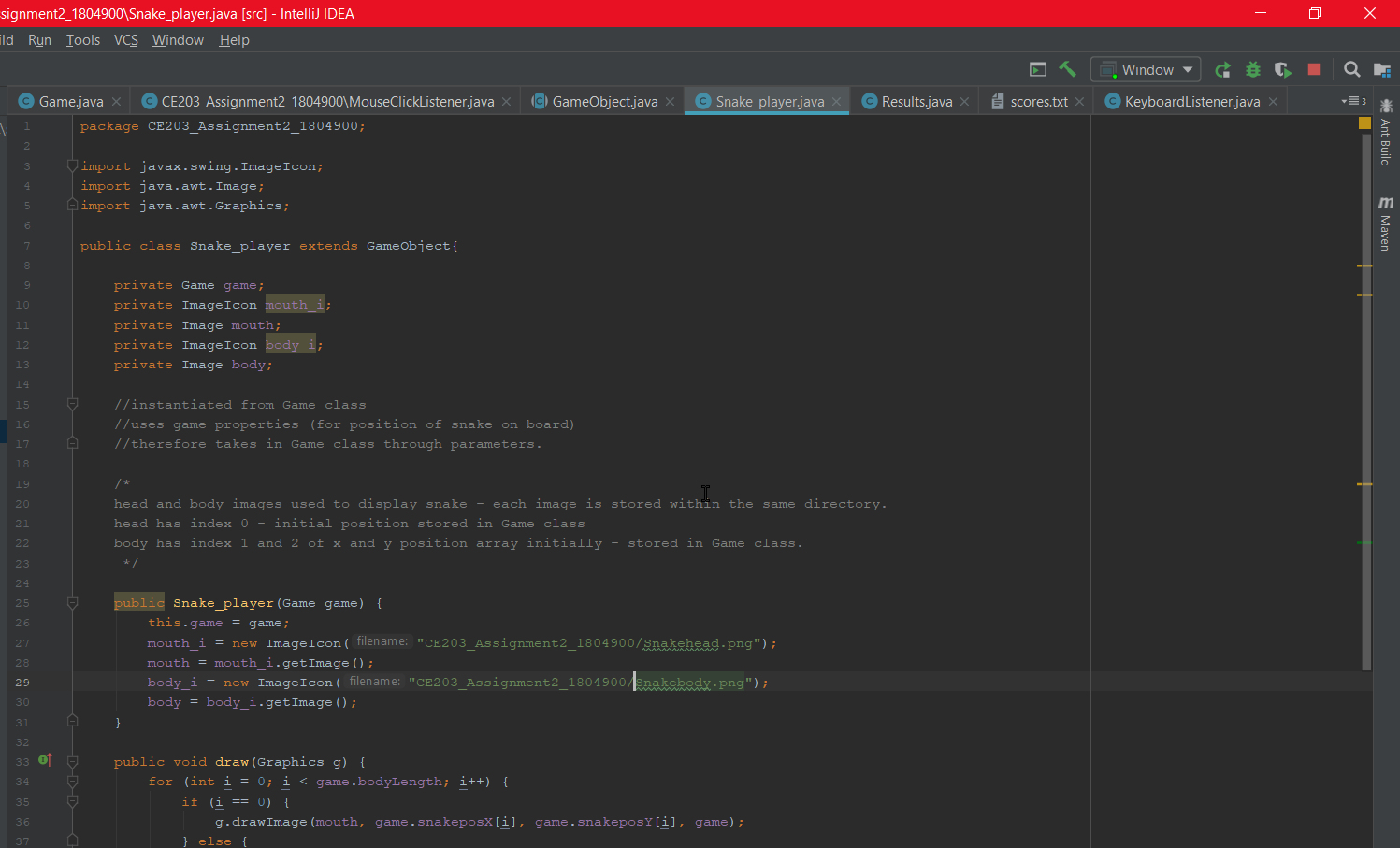
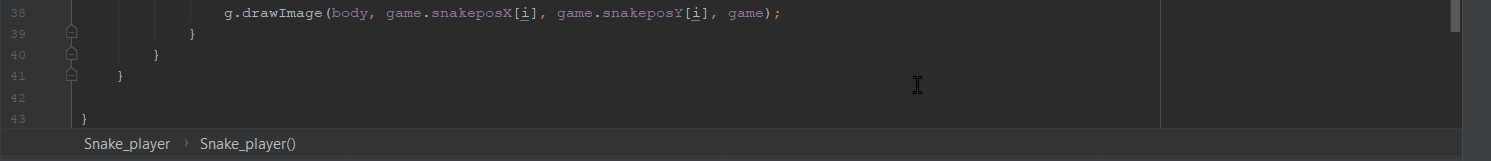
Screenshot 14.

Proof of GameObject class with commenting



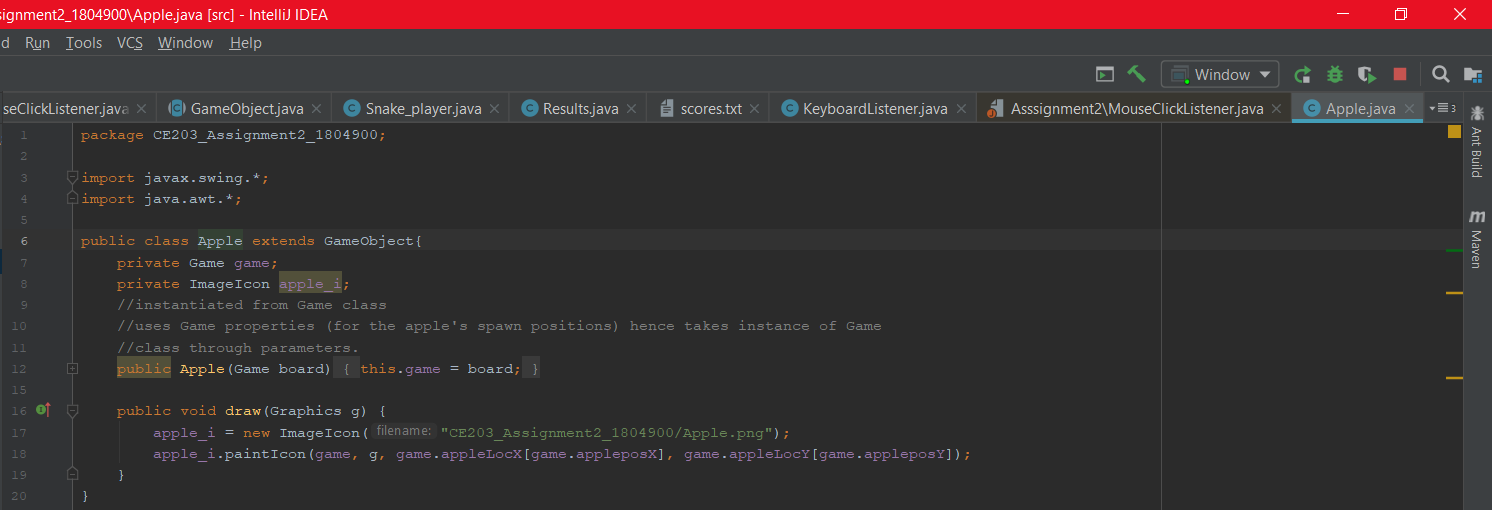
Screenshot 15 and 16.

Proof of Snake\_player class with commenting

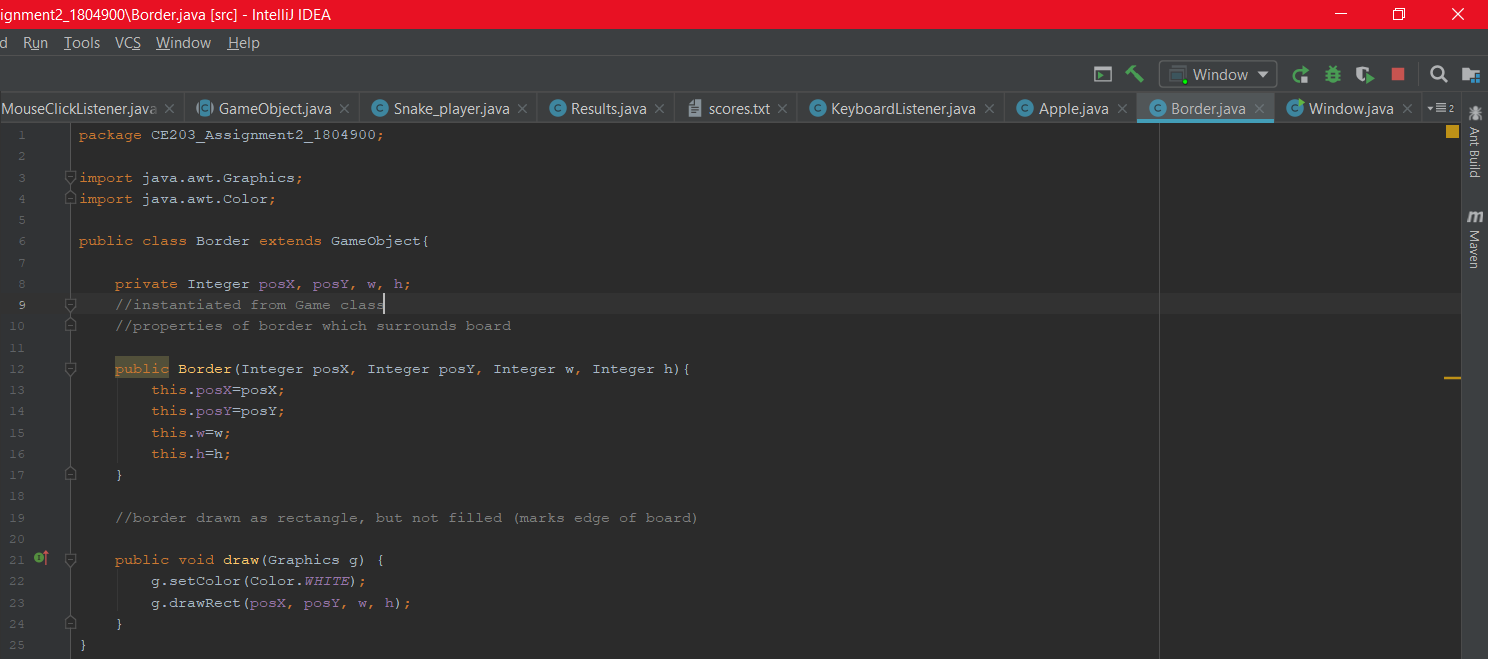
Screenshot 17.

Proof of Apple class with commenting



Screenshot18.

Proof of Border class with commenting

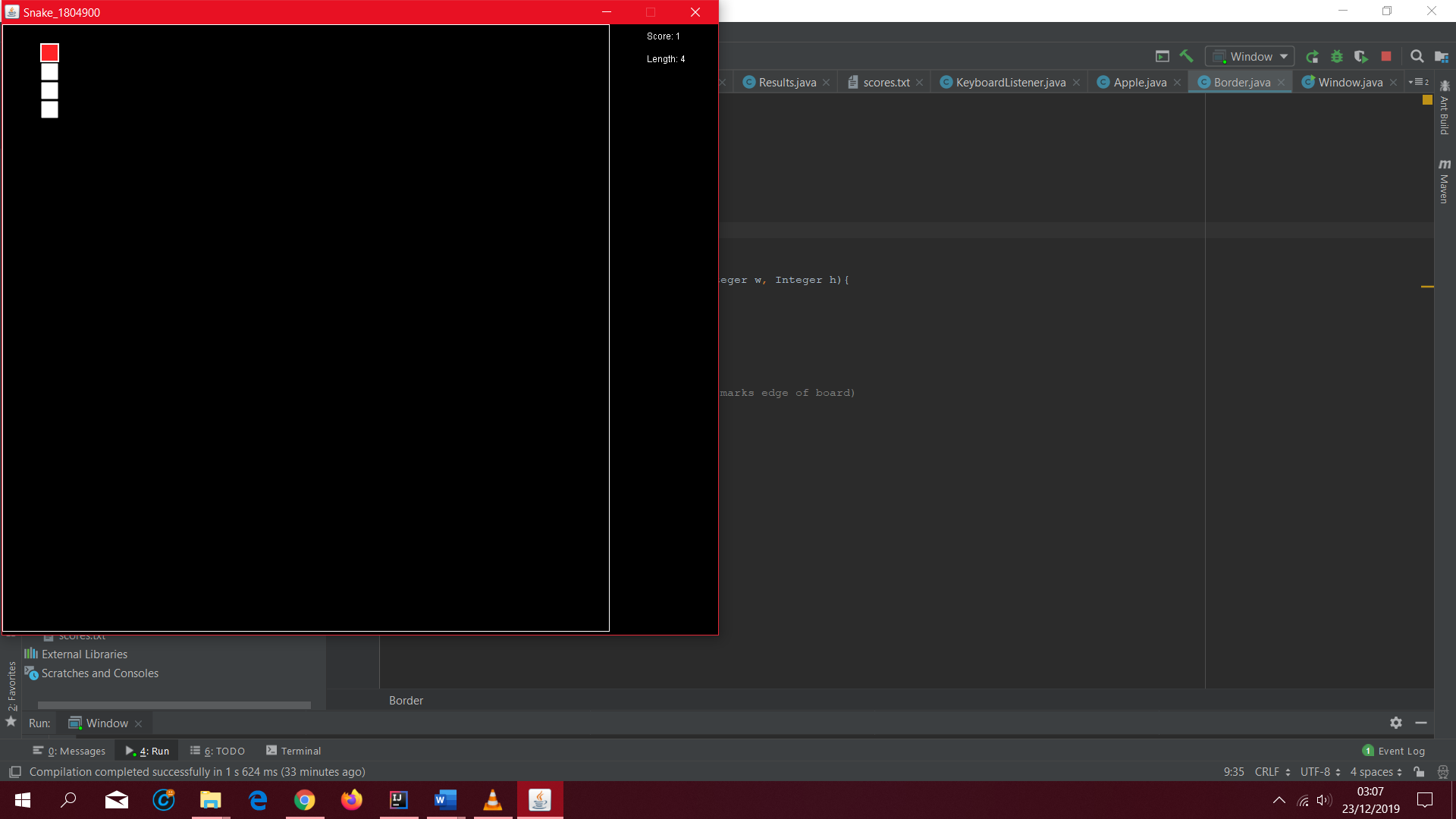


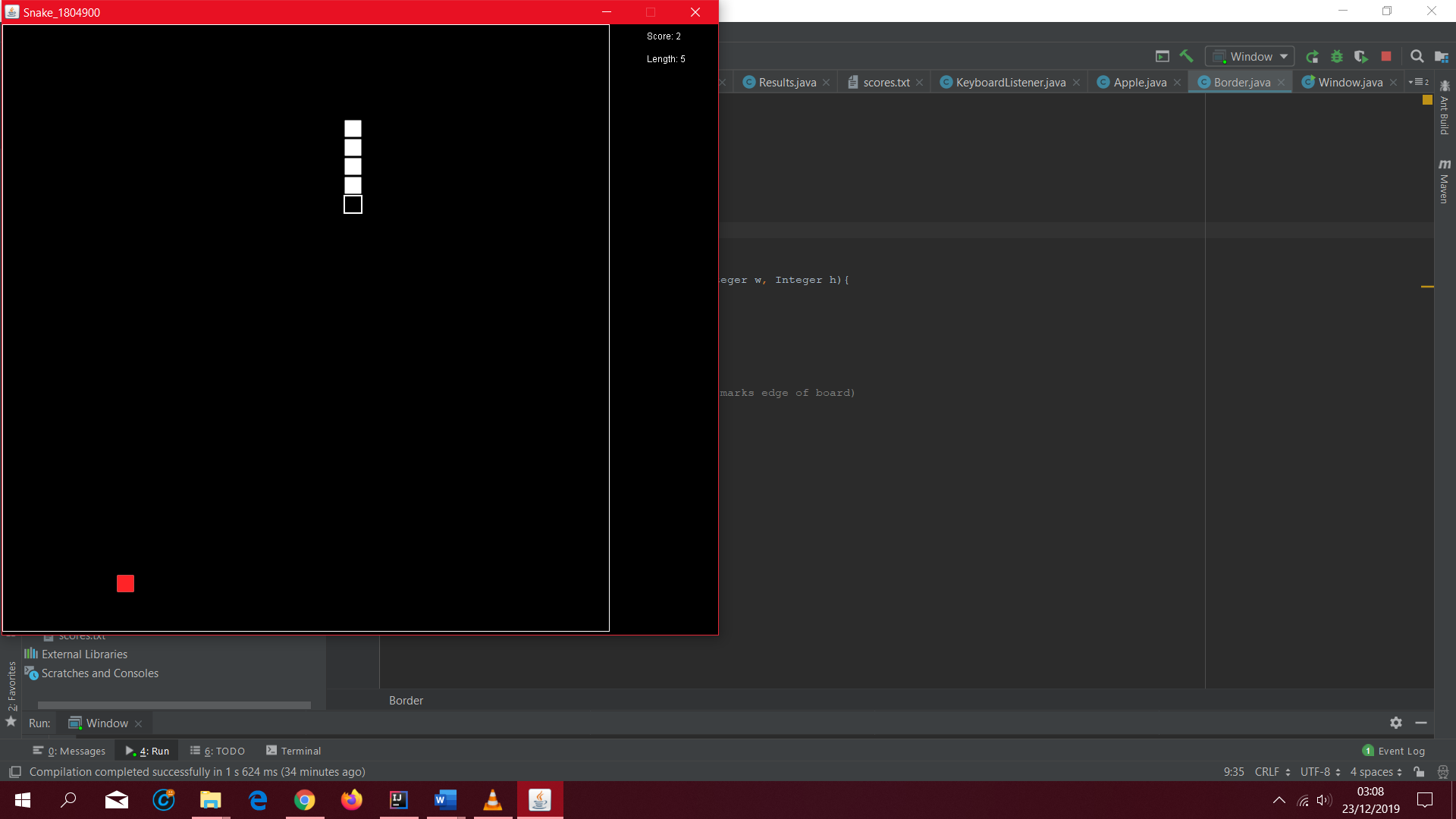
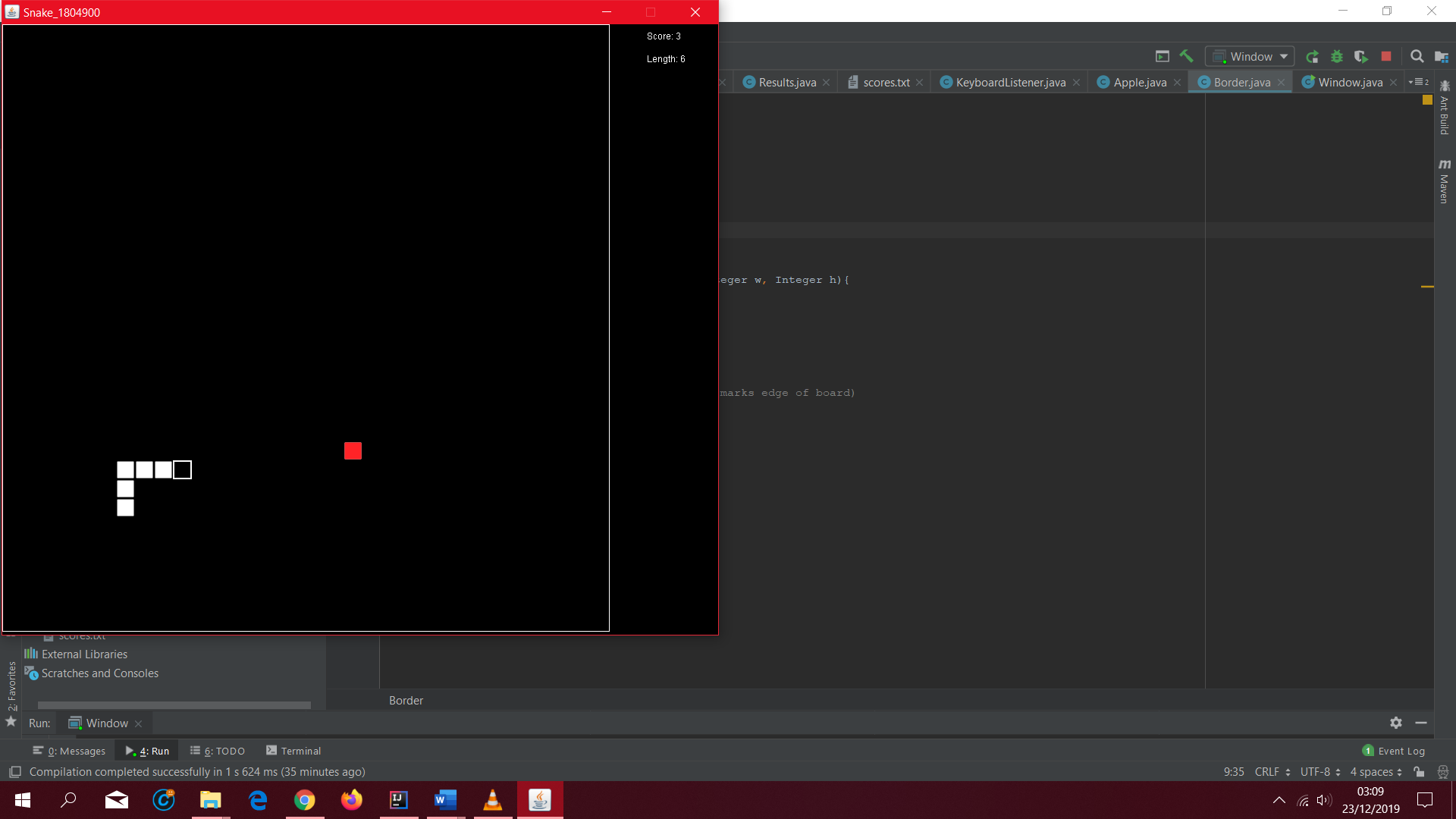
### Task G

Screenshot1 and 2 and 3.

Showing snake eat apple – score and length incremented.

Apple spawns in new location on board.

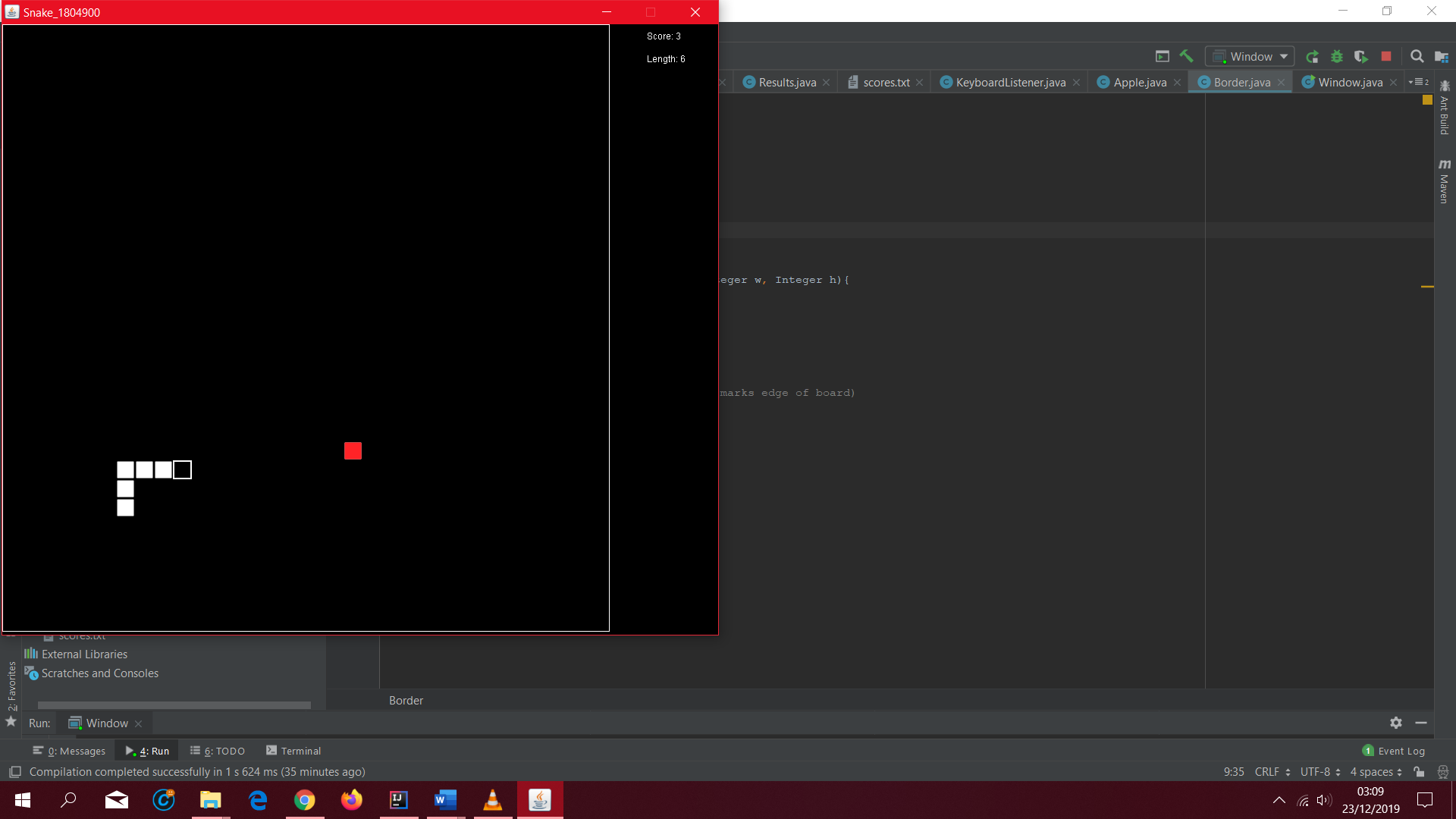
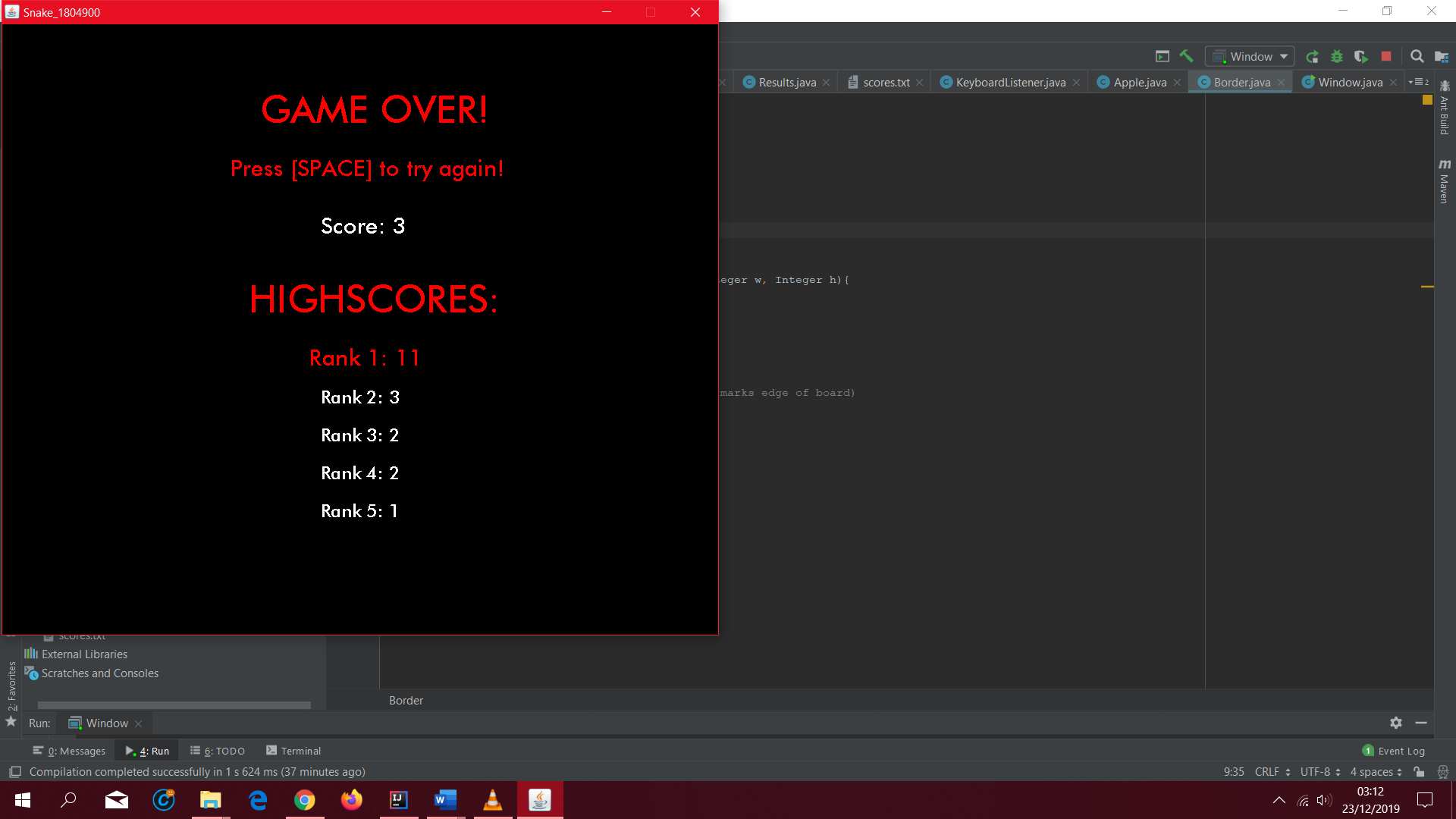


Screenshot4, 5.

Snake goes past border, hence game ends.

Scores match up with current score and high score.

Screenshot6 and 7.

New high score matches current score.

