# CE203 Assignment 2

1603930

## The Task at Hand

With all the skills that the module has taught students so far, the assignment states to apply those skills and create any type of 2D game (excluding Tetris). This application must be able of holding a collection of different shapes, drawing these shapes to the screen and allowing them to be manipulated in position and size by the user. This program can be inspired by classic arcade games.

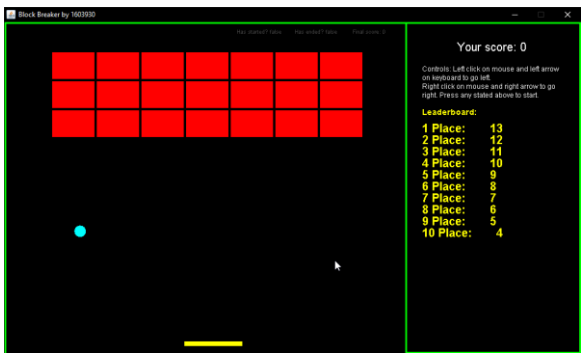
## Program Description

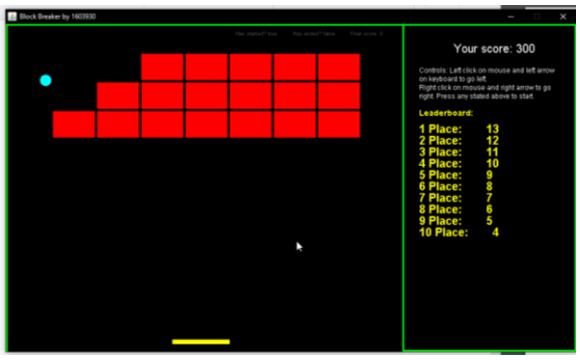
Block Breaker by 1603930

This game iterates with a Timer which allows for control of speed in movement of drawn shapes. The slider and the ball are generated through an abstract class, and then are used as objects in the Game.java file. The blocks that are implemented are written out in Blocks.java, this class utilises a two-dimensional array. The individual blocks hold either one of the two values, 1 or 0. 1 represents a present block that has not been hit and 0 representing a block that has been hit (intersected by the ball). When the Ball object intersects with a block, the value of that block hit in the nested for loop changes to 0 and is not drawn anymore, score also increases by 100. When the ball intersects with the borders, blocks or slider the direction of the ball changes (reverses). When left arrow/right arrow mouse 1/mouse 3 are pressed the position of the slider on X axis increments. The leader board functions through FileReader.java that reads a txt file and draws the top 10 scores sorted in descending order. The final score of each round is saved in the txt file through FileWriter.java, no matter how high it is because a top ten will be drawn always. All visuals are drawn on the JPanel, the leader board values are sorted before being drawn and with use of a for loop only 10 are displayed. The final score of the user will be displayed next round after the user presses the key ENTER.

## Testing to Meet Criteria

**a)** Ball shown changing position



**b)**

Main.java | Game.java | FileWriter.java | FileReader.java | leaderboard.txt | Blocks.java | Sha

```java
package assignment2;


import java.awt.*;

public abstract class Shape {

    int width, height, posX, posY;  // variables used to draw the shape

    Shape(int width, int height) {    // constructor
        this.width = width;
        this.height = height;
    }

    abstract void draw(Graphics graphics);  // abstract method with graphics in order to draw

}
```

```java
package assignment2;

import java.awt.*;

public class Slider extends Shape {

    Slider(int width, int height, int posX, int posY) {
        super(width, height);   // width and height stay consistent/define the shape and therefore are supers

        this.width = width;
        this.height = height;
        this.posX = posX;
        this.posY = posY;
    }

    void draw(Graphics graphics) {   // draws the shape
        graphics.setColor(Color.YELLOW);
        graphics.fillRect(posX, posY, width, height);
    }
}
```

```java
package assignment2;

import java.awt.*;

public class Ball extends Shape {

    Ball(int width, int height, int posX, int posY) {
        super(width, height);

        this.width = width;
        this.height = height;
        this.posX = posX;
        this.posY = posY;
    }

    void draw(Graphics graphics) {
        graphics.setColor(Color.CYAN);
        graphics.fillOval(posY, posX, width, height);
    }
}
```

Main.java | Game.java | FileWriter.java | FileReader.java | leaderboard.txt

abstract

```java
        graphics.fillRect( x: 691,  y: 0,  width: 3,  height: 592);

        // BORDERS OUTSIDE THE GAME
        graphics.fillRect( x: 991,  y: 0,  width: 3,  height: 592);
        graphics.fillRect( x: 691,  y: 568,  width: 300,  height: 3);

        // no bottom border added as once the ball passes the bottom the use

        // slider using abstract class
        Shape sliderShape;
        sliderShape = new Slider( width: 100,  height: 8, sliderX,  posY: 550);
        sliderShape.draw(graphics);

        // ball using abstract class
        Shape ballShape;
        ballShape = new Ball( width: 20,  height: 20, ballY, ballX);
        ballShape.draw(graphics);
```

**c)** Left: Keyboard, right: Mouse.

```
@Override
public void keyReleased(KeyEvent e) {
    if (e.getKeyCode() == KeyEvent.VK_ENTER) {    // restart time after ENTER is pressed
        // (timer is stopped to allow FileWriter WriteCurrentScore() to run only ONCE)
        requestFocus();
        time.restart();
    }
}

@Override
public void keyPressed(KeyEvent e) {


    if (e.getKeyCode() == KeyEvent.VK_RIGHT && !ended) {   // if right arrow key is pressed
        requestFocus();
        if (sliderX >= 600) { // checks that slider is not outside the border
            sliderX = 600;
        } else {
            goRight();
        }
    }
    if (e.getKeyCode() == KeyEvent.VK_LEFT && !ended) {   // if left arrow key is pressed
        requestFocus();
        if (sliderX < 10) {  // check for opposite side of border
            sliderX = 10;   // when at 10 stay at 10
        } else {
            goLeft();
        }
    }
    if (e.getKeyCode() == KeyEvent.VK_ENTER) {    // restart
        requestFocus();
        if (!start) {
            start = true;   // game restarts
            ended = false;
            ranOnce = false;

            finalScore = 0; // restart score relevant to leaderboard

            ballY = 350;    // resets position of objects and quantity of blocks
            ballX = 120;

            directionY = -2;
            directionX = -1;

            sliderX = 310;
            currentScore = 0;

            totalBlocks = 21;

            blocks = new Blocks( row: 3,  col: 7);

            repaint();
```

```
public void mousePressed(MouseEvent e) {
    requestFocus();
    if (e.getButton() == MouseEvent.BUTTON3 & !ended) {  // right
        requestFocus();
        if (sliderX >= 600) {
            sliderX = 600;
        } else {
            goRight();
        }
    }
    if (e.getButton() == MouseEvent.BUTTON1 & !ended) {  // left
        requestFocus();
        if (sliderX < 10) {
            sliderX = 10;
        } else {
            goLeft();
        }
    }
}

@Override
public void mouseClicked(MouseEvent e) {
}

@Override
public void mouseReleased(MouseEvent e) {
}

@Override
public void mouseEntered(MouseEvent e) {
}

@Override
public void mouseExited(MouseEvent e) {
}
```

**d)**

Left: FileReader class, Right: FileWriter class.

```
package assignment2;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;

public class FileReader {

    public static ArrayList<Integer> OpenAndRead() {
        ArrayList<Integer> leaderboardList = new ArrayList<>(); // array that holds the contents of the txt


        try {
            BufferedReader br = new BufferedReader(new java.io.FileReader( fileName: "src/assignment2/leaderboard.txt"));

            String line = null;

            while ((line = br.readLine()) != null) {    // stops when no characters are found
                Integer result = Integer.parseInt(line);
                leaderboardList.add(result);
            }

            br.close();

        } catch (FileNotFoundException e) {
            System.err.println("LEADERBOARD.TXT NOT FOUND");

        } catch (IOException e) {
            System.err.println("UNABLE TO READ FILE");
        }
        return leaderboardList;
    }
}
```

```
package assignment2;

import java.io.BufferedWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class FileWriter {
    public void WriteCurrentScore(int finalScore) {
        try {
            java.io.FileWriter fw = new java.io.FileWriter( fileName: "src/assignment2/leaderboard.txt",  append: true);
            BufferedWriter bw = new BufferedWriter(fw);
            PrintWriter out = new PrintWriter(bw)
        ) {

            String str = Integer.toString(finalScore);
            out.println(str);

        } catch (IOException e) {
            System.err.println("UNABLE TO READ FILE");
        }
    }
}
```

```
public void paint(Graphics graphics) {  // drawing the game and shapes involved
    FileReader returnList = new FileReader();    // creates object for reading txt list
    ArrayList<Integer> freshList = returnList.OpenAndRead();
    Collections.sort(freshList, Collections.reverseOrder());     // sorts descending
```

```
// displays leaderboard
graphics.setColor(Color.YELLOW);    // INSTRUCTIONS
graphics.setFont(new Font( name: "arial", Font.BOLD,  size: 14));
graphics.drawString( str: "Leaderboard:",  x: 720,  y: 160);
graphics.setFont(new Font( name: "arial", Font.BOLD,  size: 20));

int textSpaceY = 170;
for (int x = 0; x < 10; x++) {
    textSpaceY = textSpaceY + 20;   // spaces out the text
    graphics.drawString( str: (x + 1) + " Place:       " + freshList.get(x),  x: 720, textSpaceY);
}
```

## e)

Stated above.

## f)

Game.java is the biggest class, but in this project, I used a different class for the shapes and the file reader/writers.

I have written many comments throughout my code to explain how my program works.

## g)

-

## Known Bugs

Initially when I implemented the leader board I had a major issue that took me a few hours to fix. When it was time to write in the final score of the round, the leader board would fill up with that final score as the method to write the final score would be called numerous times due to the Timer. To fix this I implemented two Booleans that determine if the game was at its "Restart" screen and that determine if the method was ran once after the round. I also had to stop the time and restart it accordingly. After I fixed this, the program worked swiftly until it got to the end, every time I pressed any buttons excluding ENTER the final score would append to the text file rendering the leader board useless again, I fixed this by changing the if statements for the keys to run if the specific key was pressed and if the game not ended. I also implemented a timer restart in the enter key code which allowed the WriteCurrentScore() method to be only ran once per round.

## Possible Improvements

1. Add strings to the list using Map, to display users name. Names would be prompted on launch of program.

2. Add levels to the game, potentially increase ball speed, add another ball or change grid layout?

3. Add extra keyboard actions that add to the gameplay such as timing the ball hitting the slider correctly rewarding the user with two balls.

4. Using classes to manage the keyboard and mouse actions.

## Comments

I really enjoyed this assignment in comparison to the others, having the freedom to code whatever you like to fit specifications is something that I look forward to in the future. My skills were challenged a lot and If I had more time I would've happily continued to expand my program.

## Extra Credit

Implementation of Timer whilst managing to display changing leader boards all on one window.

## References

https://www.w3schools.com/js/js_timing.asp

https://www.w3schools.com/java/java_files.asp

https://www.w3schools.com/graphics/game_score.asp