Mini Project 2 Critique Document

CI101 – Programming

James Shaw – 17848280

Contents

[What Grade I am claiming 2](#_Toc512514703)

[What I’ve added to the game 3](#_Toc512514704)

[Model.java 3](#_Toc512514705)

[Adding bricks to the game 3](#_Toc512514706)

[Stopping the bat moving off the screen 3](#_Toc512514707)

[Checking if a brick has been hit 4](#_Toc512514708)

[Making the ball a bit more random 5](#_Toc512514709)

[Lives and Levels 5](#_Toc512514710)

[View.java 6](#_Toc512514711)

[Displaying the bricks 6](#_Toc512514712)

[Changing the update method 6](#_Toc512514713)

[Changing the shape of the ball 6](#_Toc512514714)

[Creating a menu 7](#_Toc512514715)

[How could the game be improved? 8](#_Toc512514716)

[What I struggled with 9](#_Toc512514717)

# What Grade I am claiming

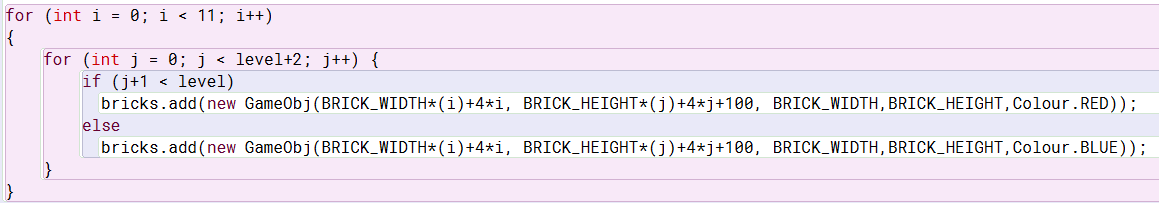
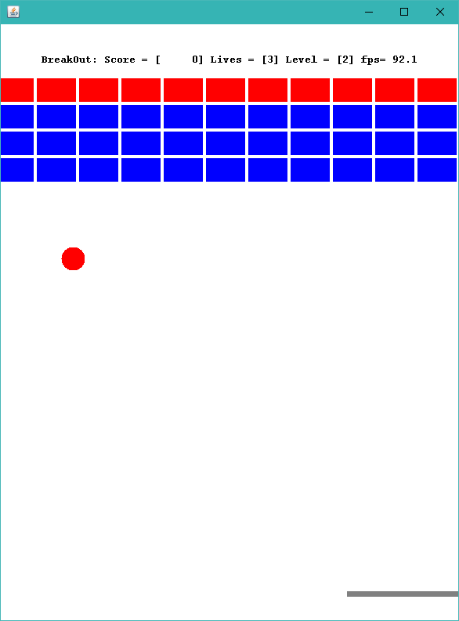
For this project, I am claiming an A-A\* grade. This is because I have fulfilled all the criteria within each grade band up to and including the A band. I have also added a menu to my game which means the player isn’t thrown straight into the game. On top of this, I have also created a life system so that the player only has 3 lives. These are reset at the beginning of a new level.

# What I’ve added to the game

## Model.java

### Adding bricks to the game

Within the Model.java file, I have created two for loops that deal with adding bricks to the game.



This adds bricks to the list ‘bricks’, creating a grid of bricks 11 across and level+2 down. I decided on 11 across as this is what fitted the screen best. Any more than 11 and the bricks were off the edge of the playing area, and less than 11 and the bricks did not fill the full width of the playing area. I then used level+2 because this meant there was always 3 lines of one hit blocks on each level, and a line of 2 hit blocks were added each level.

I have also added an if statement that creates either red or blue bricks. The red bricks need two hits to destroy them. I have made it so that as the player goes up by a level, a line changes from blue bricks that take 1 hit to red bricks that take two hits. The first level has all blue bricks. The screenshot shown here is from the second level of the game, where there is 1 layer of red bricks. In the next level, the second layer of bricks will turn red, and the fourth level will have all red bricks.

Within the new GameObj function, I have set the first variable (x) to BRICK\_WIDTH\*(i)+4\*i and the second variable (y) to BRICK\_HEIGHT\*(j)+4\*j+100. These set the starting position of each brick in the array. The BRICK\_WIDTH\*i and BRICK\_HEIGHT\*j parts ensure that each brick is not covering another brick. In one row of bricks, each x position is the bricks width multiplied by its number in the row. This puts all the bricks in 1 row. The same is done for columns, using height and the bricks row number.   
The +4\*j and +4\*i give each brick a bit of padding. This was decided by me playing around with which number to multiply i and j by until I found a value I was happy with.  
I have also added +100 to the y variable of the brick to ensure the bricks did not cover the text at the top of the game.

I have then set the height and width of the bricks to the given variables BRICK\_HEIGHT and BRICK\_WIDTH, and set the colour accordingly.

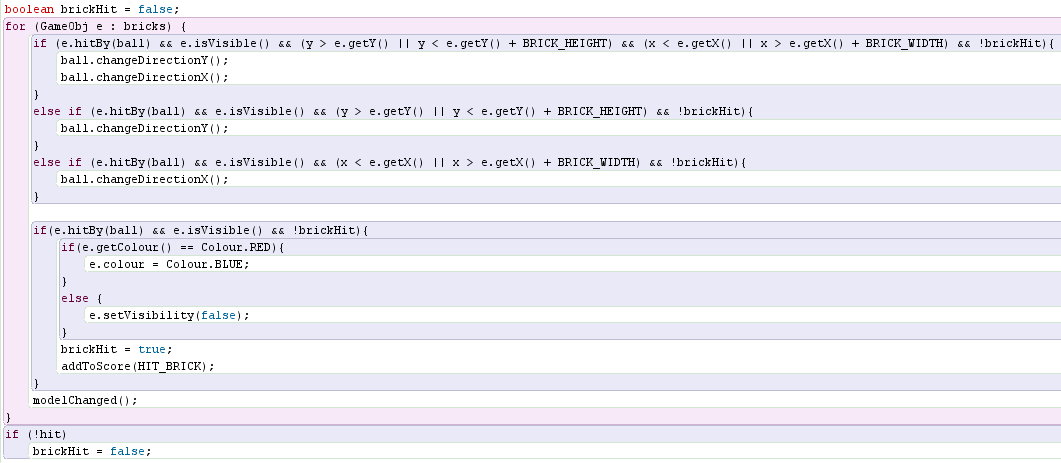
### Stopping the bat moving off the screen



To start, I created a variable x which holds the x position of the bat using the getX() method in the gameObj class. I then used that to see if the bat was at the edge of the screen with this if statement. The first part of this if statement stops the bat going off the right side of the game area, and says if x is greater of equal to the width of the game area minus the border offset minus the width of the bat, and the bat’s direction = 1, so it’s moving to the right, then set the direction to 0, so it’s not moving.

I then added an OR to stop the bat going off the left side of the game area. This part of the if statement only needs to check if x is less than or equal to the border offset, and if the direction is -1, so it’s moving left. This then sets the direction to 0.

### Checking if a brick has been hit

  
Firstly, I created a for loop that checked through the list of bricks. In this for loop, each individual brick within the bricks array is represented by the variable e.  
The first if else statement deals with changing the direction of the ball when it hits the brick, and the second if else statement deals with changing the bricks to either invisible or change the colour. This also deals with scoring.

The first part of the first if else statement says if the brick is hit by the ball, and e is set to visible, and the ball is hitting one of the corners of the brick, and the brickHit variable is set to false, then change both the X and Y direction.   
Else, if the brick is hit by the ball, and e is set to visible, and the ball hit the brick at the top or bottom, and the brickHit variable is set to false, then change the Y direction.  
Else, if the brick is hit by the ball, and e is set to visible, and the ball hit the brick on one of the sides, and the brickHit variable is set to false, then change the X direction.

The second if statement checks if the brick is hit by the ball and is visible and if the brickHit variable is set to false. This then runs another if else that says if the colour of the brick is set to red, then change the colour to blue. Otherwise, set the visibility to false.  
This then sets the variable brickHit to true, and adds the points for hitting the brick.

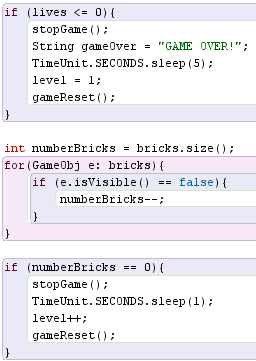
Finally, I added an if statement to say if the ball isn’t touching anything, then set the brickHit variable to false. I have added this variable as I encountered a bug where the ball would remove multiple bricks in one hit, or would destroy a red brick in one hit as it didn’t move away from the brick fast enough.

### Making the ball a bit more random

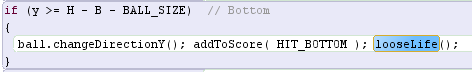
To start, the ball would always bounce off the bat in the same X direction as it hit the bat. This is because the code to change the direction when the ball hit the bat only changed the Y direction. This meant the ball followed the same path every time the game was played, and it became very predictable. To make the game more interesting, I added some code to randomise the X direction of the ball when it bounced off the bat.

This code works as Math.random() produces a random number between 0.0 and 1.0. Therefore, saying Math.random() < 0.5 is saying that half of these numbers will cause the if statement to run, and half won’t, and therefore it is a 50-50 chance of the ball also changing X direction when the bat is hit. This makes the game a bit less predictable and more fun to play.

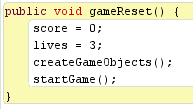
### Lives and Levels



I started by creating two new variables called lives and level, and setting each of these to the values I wanted to start the game with. I then created a method called looseLife which decreased the lives variable by 1.   
I then called the looseLife method within the if statement that checks if the ball has hit the bottom of the playing area.

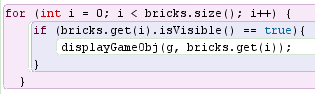


Once I had done this, I added the code on the right within the runAsSeparateThread method.   
The first if statement checks if the lives variable has reached 0. If it has, then stop the game, pause for 5 seconds, reset the level to 1 and run the gameReset method which I created within the Model class.

I then wrote a for loop to calculate how many bricks are still visible, and an if statement to say if all the bricks have been removed from the game, then pause for a second, increase the level and run the gameReset method.

## View.java

### Displaying the bricks

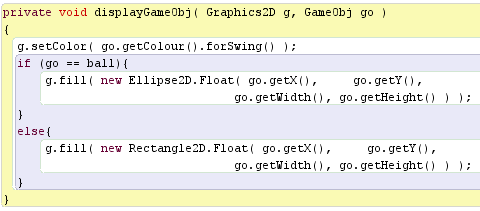
To display the bricks that I had previously stored in the list ‘bricks’, I created a for loop to check each brick, and then check if the brick was visible. If it was, then I used the displayGameObj method to show the brick.

### Changing the update method

I had to add a couple of lines of code to the update method to ensure that the lives and levels that I created earlier were included in this.

### Changing the shape of the ball

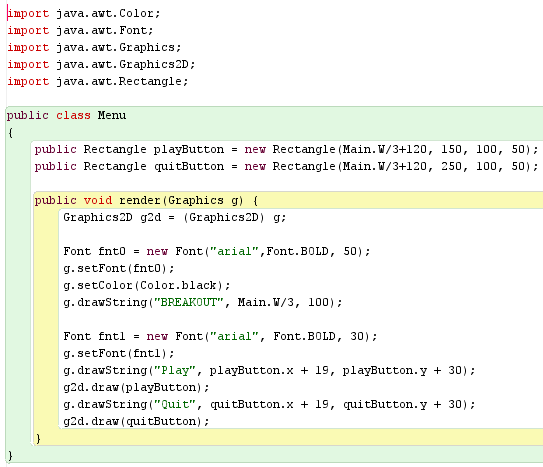
I wanted the ball to be round instead of rectangular because I feel this makes more sense – balls are round. To do this, I used an if statement to use Ellipse2D to create the ball, and left the rest as Rectangle 2D. I had to import java.awt.geom.Ellipse2D into the View class.



## Creating a menu

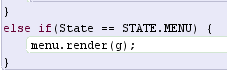
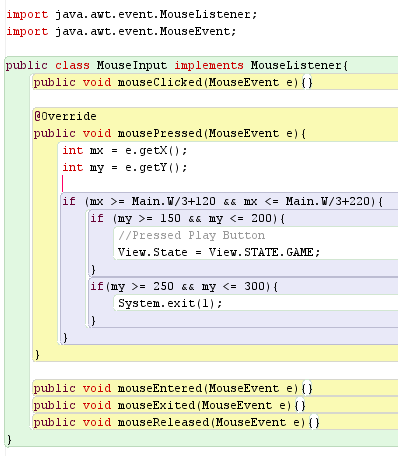
To create the menu in my game, I first created some states within the View class. I then set the starting state to menu to ensure the game started on the menu screen.

I then added a new class called menu to create the menu screen

I used this menu class to create all of the menu items. I could have done this within the view class, but I felt this way was neater.

I then had to add an iteration of menu within the view class  
  
and within the drawActualPicture method, I created my new menu, added a mouse listener, and wrote an if statement to check which state was active.

This if statement then says: if the state is set to game, then run all of the code that was originally within the drawActualPicture method. Otherwise, render the menu.



Then, I added one more class called MouseInput. This checked for any mouse clicks and checked for where the input was. I’ve used if statements to detect if the click was on one of the menu buttons. If the button clicked was the play button, then change the state to game. If the button clicked was the exit button, then close the program.

Finally, I added another if statement to the runAsSeparateThread method, which checks if the state is menu. If it is, then the game is reset. This is because I found that the game would run in the background while I was in the menu screen.

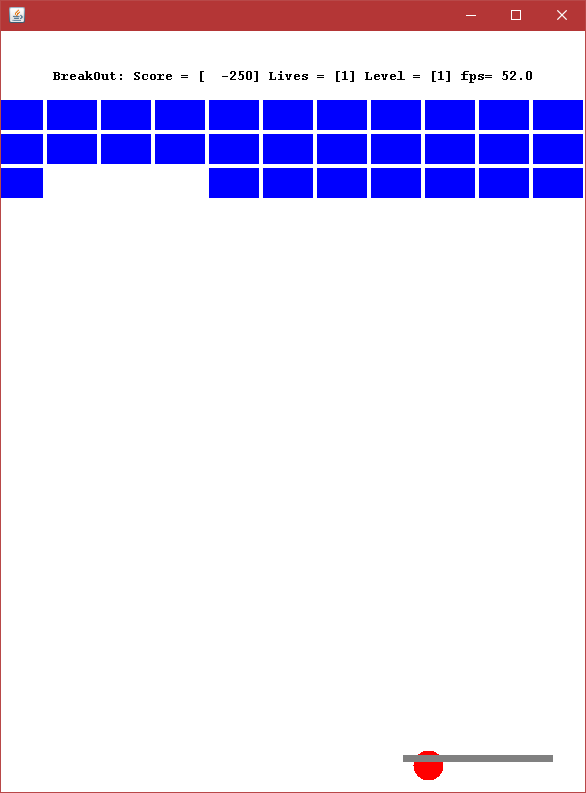
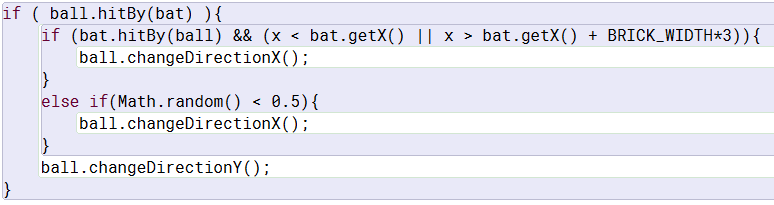


# How could the game be improved?

There is a slight issue on the university computers when moving from the menu to the game. When the play button is pressed in the menu, there is a 3-5 second delay where the screen changes, but the game doesn’t start. I thought this might be because I’ve used an if statement to check the game state. I tried changing this to a while statement but this seemed to crash the game so I changed it back. I think it may be where the game is processing a lot as it has to reset variables and change from the menu state to the game state, as it doesn’t happen on my home computer which I imagine has more processing power.

Also, I would like to have the ball bounce off the bat at a slightly different direction depending on where the bat is hit. I had a bit of a play with this, but didn’t find a solution. I tried doing a calculation to work out how far along the bat the ball had hit, and then convert that to a number between -1 and 1 to give the ball a direction. This seemed to mess the ball up as when it hit the bat, it just started moving vertically. I would like to find a way to do this, but I haven’t managed to so far. Instead, I implemented a feature where the ball would sometimes change X direction when it hit the bat, and sometimes didn’t.

The maths isn’t quite right with the ball hitting the bricks. Because of this, the ball will occasionally bounce off the brick at an unexpected angle. This would be solved by spending some time refining the math that controls this function. The game still runs fine, but it’s just not quite there.

Finally, there is a bug with when the ball hits the side of the bat. It seems to get stuck in the bat, trying to change Y direction constantly and then changing back. I tried to fix it with this code but that didn’t seem to work. With a bit more time I might have been able to find a solution to this

# What I struggled with

Getting the math to work correctly for the brick collisions took a while to calculate. This was mostly just trial and error and it still doesn’t work quite right, but it’s almost there. Also, the menu was a bit of a struggle to implement as I wasn’t familiar enough with java to know how to do this. After a bit of research, I figured it out.