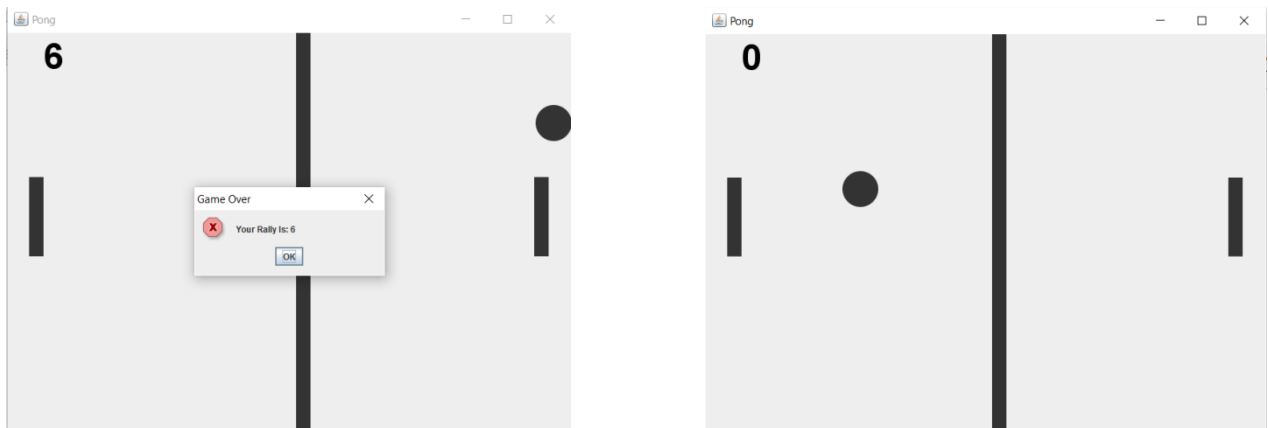


## Programming Testing – Pong Game



After the Pong Java program was created, I began testing the application to see how robust the code is. I am also testing to find any improvements that can be made in the program.

This game is a co-operative game in which the two players are trying to get the biggest rally they can while the ball increases in speed every collision with the player.

I will be using White Box Testing (Clear Box Testing) to test the software's internal structure, will also grade the programming skills produced and compare it from the design plan before the program was created.

## White Box Testing

No.	Purpose	Expected Response	Actual Response	Conclusion
1	Stopping the ball from leaving the frame.	Make the ball collide with the top and bottom walls of the frame and reflect from them.	Ball collides with both sides of the wall, the ball doesn't speed up and the game doesn't end.	Program working efficiently. Not problems occurring.
2	Giving the user the ability to play the game properly	Player movement throughout the game	The user was able to move the keyboard effectively.	Keyboard Input was correctly used in the program
3	Player to not be allowed to leave the frame or move closer to the ball.	Restricting movement in the x direction.	The X coordinates for each player is stationary. User not able to move in the x-direction.	User restriction enabled correctly
4	Window to appear to show that the game is over and to reveal your rally score.	Expecting a JOptionPane to appear once the game is over.	JOptionPane working effectively. Game closes once the player fails.	JOptionPane works but closing the game stops the user from playing again.
5	Give the user keyboard inputs, using their keys to move their characters.	Player to use the keys to move their rectangle.	Users able to use the four keys (W, S, UP and DOWN) to move their characters. Restricting other keys from working.	Only the four keys affect the player's movement. Works well, however sometimes movement restriction happens when both players move.
6	Stops the user from getting points when they collide with the ball once.	Rally score to increase by one every time the player hits the ball.	The user gets one point, however when the user hits the ball at the corner, gets given more points than they should	The points system works well on the top left of the game. JOptionPane tells the users their score before the program terminates.
7	Making sure the program doesn't stay on forever and run properly	Program to start and finish without any problems	Pong game runs and terminates properly.	
8	Testing whether maximising and minimising affects the user experience.	Maximising the screen to not interfere with player experience.	Maximising the screen affects the game.	Program doesn't adjust when the user maximises the frame. Errors occur.
9	To ensure that there are no errors when both players press their keys	Expecting the two keys to work and move each paddle at the same time.	90% of the time, it works as required, however it sometimes stops one of the paddles to move.	Small error sometimes when both players are interacting with the keys.

There is a recurring error where the keyboard will only listen to one of the keys that both players pressed; one way of fixing this problem is to give the users KeyBindings.

Using this will ensure that more than one key can be pressed at the same time without either of them being interfered.

There is also an error that happens now and again where the score increases significantly once the edge of the paddle has interacted with the ball, this occurs because the ball has collided with the edge of the paddle more than once, thus increasing the speed significantly and giving the users a higher score than they deserved.

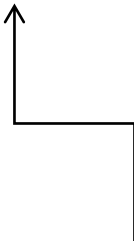
One way to improve this will be to removing the ability for the ball to interact with the side of the paddle and only allowing it to rebound off the front of the paddle.

```
import java.awt.*;
import java.awt.event.*;

public class Player1 {
    private Game game;
    private static final int Y = 30;
    private static final int WIDTH = 20;
    private static final int HEIGHT = 110;
    int x = 200;
    int xx = 0;
    public Player1(Game game) {
        this.game = game;
    }

    public void move() {
        if (x + xx > 0 && x + xx < game.getHeight() - HEIGHT)
            x = x + xx;
    }

    public void paint(Graphics2D graph2D) {
        graph2D.fillRect(Y, x, WIDTH, HEIGHT);
    }
}
```



Testing to make sure that the paddle is working effectively.

When I presses the key, the keyPressed method of "Racquet" is called and this will set "xx" to 1, if the key pressed is the upwards direction (KeyEvent.VK\_UP), the paddle will move to the right the next time the move method due to the move() method (x = x + xx;).

In the same way if we press the key KeyEvent.VK\_DOWN it will move downwards just as predicted.

```
void move() {
    if (x + xx < 0)
        game.endGame1();
    if (x + xx > game.getWidth() - DIAMETER)
        game.endGame1();
    if (y + yy < 0)
        yy = game.speed1;
    if (y + yy > game.getHeight() - DIAMETER)
        yy = -game.speed1;
    if (collision()) {
        xx = game.speed1;
        game.speed1++;
    }
    if (collision2()) {
        xx = -game.speed1;
        game.speed1++;
    }
    x = x + xx;
    y = y + yy;
}
```



Testing the move() method in the Ball class to see how well it runs

The move() method has if functions for every scenario that the ball can have in the frame, hitting each side of the wall and colliding with each of the paddles. Change of direction will always occur when interacting with the paddles and the top and bottom walls.

The game.speed1++ increases the speed of the ball every time there is a collision with the paddles. No errors are occurring apart from when the ball collides with the sides of the paddle, causing multiple collisions to happen.