**Name: MUHAMMAD AIMAN AQIL BIN MD SAFFIE**

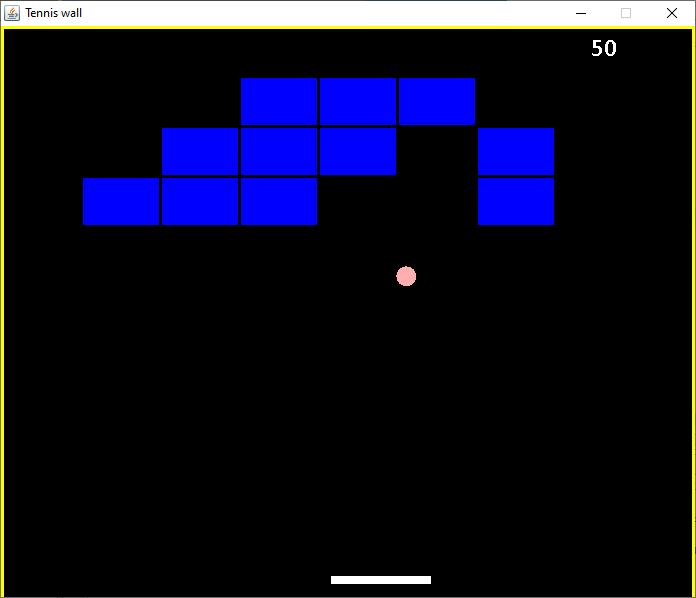
**Matric No: BI19110084**

**Object Oriented Concept Implementation**

My java project is about a simple game based on the old brick breaker game but I rename it to wall breaker. User can play the game by using right and left arrow on the keyboard to move the paddle. The objective is to see if user can break the wall and without letting the ball fall in the endless hole. My project applied the five concepts of object-oriented programming (OOP), the first OOP concept would be abstraction since all the processes are hidden from the user and only certain and relevant information are visible. The purpose of abstraction is to hides the underlying complexity of data, presents only the signature of internal functionality and gives flexibility to programmers to change the implementation of the abstract behaviour. The second OOP concept is encapsulation since most of the variables and methods are private so it would not confuse the programmer when calling a variable or method inside a class. Encapsulation is to restricts direct access to data members (fields) of a class, each field has a getter and setter method and setter methods let us change the value of the field. The third one is inheritance, this concept is applied and implements the ActionListener interface. Inheritance is a class (child class) that can extend another class (parent class) by inheriting its features and it improves code reusability. The fourth OOP concept that was applied into the project is polymorphism since each class contains multiple functions because in java polymorphism is refers to the ability to perform a certain action in different ways. It used the same method name several times and different methods of the same name can be called from the object. The fifth OOP concept is association, where all classes are associated with one another whether just one or multiple classes to run the whole programme. Two separate classes are associated through their objects and the two classes are unrelated, each can exist without the other one.

**Read and Write Implementation**

When the user run the programme, the programme will show:



As the user interact, the programme will read the inputs that were given by the user and the programme will produce certain action based on what information it received.

Manual

1. Run the Wall Breaker programme
2. Press the left or right arrow on the keyboard to start the game and try to break all the walls
3. If you dropped the ball to the endless hole. The “Well played, your score is:” will be displayed
4. You can play again by pressing “Enter” button.

Code

import java.awt.Color;

import javax.swing.JFrame;

public class Main {

public static void main(String[] args) {

JFrame obj=new JFrame();

Gameplay gamePlay = new Gameplay();

obj.setBounds(10, 10, 700, 600);

obj.setTitle("Tennis wall");

obj.setResizable(false);

obj.setVisible(true);

obj.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

obj.add(gamePlay);

obj.setVisible(true);

}

}

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Graphics2D;

public class MapGenerator

{

public int map[][];

public int blockwidth;

public int blockheight;

public MapGenerator (int row, int col)

{

map = new int[row][col];

for(int i = 0; i<map.length; i++)

{

for(int j =0; j<map[0].length; j++)

{

map[i][j] = 1;

}

}

blockwidth = 555/col;

blockheight = 150/row;

}

public void draw(Graphics2D g)

{

for(int i = 0; i<map.length; i++)

{

for(int j =0; j<map[0].length; j++)

{

if(map[i][j] > 0)

{

g.setColor(Color.blue);

g.fillRect(j \* blockwidth + 80, i \* blockheight + 50, blockwidth, blockheight);

g.setStroke(new BasicStroke(3));

g.setColor(Color.black);

g.drawRect(j \* blockwidth + 80, i \* blockheight + 50, blockwidth, blockheight);

}

}

}

}

public void BrickValue(int value, int row, int col)

{

map[row][col] = value;

}

}

import java.util.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.awt.\*;

import javax.swing.Timer;

public class Gameplay extends JPanel implements KeyListener, ActionListener

{

private boolean play = false;

private int score = 0;

private int Bricks = 48;

private Timer timer;

private int delay=8;

private int playerX = 310;

private int ballposX = 120;

private int ballposY = 350;

private int ballXdir = -1;

private int ballYdir = -2;

private MapGenerator map;

public Gameplay()

{

map = new MapGenerator(4, 12);

addKeyListener(this);

setFocusable(true);

setFocusTraversalKeysEnabled(false);

timer=new Timer(delay,this);

timer.start();

}

public void paint(Graphics g)

{

g.setColor(Color.black);

g.fillRect(1, 1, 692, 592);

map.draw((Graphics2D) g);

g.setColor(Color.yellow);

g.fillRect(0, 0, 3, 592);

g.fillRect(0, 0, 692, 3);

g.fillRect(691, 0, 3, 592);

g.setColor(Color.white);

g.setFont(new Font("calibri",Font.BOLD, 25));

g.drawString(""+score, 590,30);

g.setColor(Color.white);

g.fillRect(playerX, 550, 100, 8);

g.setColor(Color.pink);

g.fillOval(ballposX, ballposY, 20, 20);

if(ballposY > 570)

{

play = false;

ballXdir = 0;

ballYdir = 0;

g.setColor(Color.YELLOW);

g.setFont(new Font("calibri",Font.BOLD, 30));

g.drawString("Well Played, Scores: "+score, 190,300);

g.setColor(Color.RED);

g.setFont(new Font("calibri",Font.BOLD, 20));

g.drawString("Press (Enter) to Restart", 230,350);

}

}

public void keyPressed(KeyEvent e)

{

if (e.getKeyCode() == KeyEvent.VK\_RIGHT)

{

if(playerX >= 600)

{

playerX = 600;

}

else

{

moveRight();

}

}

if (e.getKeyCode() == KeyEvent.VK\_LEFT)

{

if(playerX < 10)

{

playerX = 10;

}

else

{

moveLeft();

}

}

if (e.getKeyCode() == KeyEvent.VK\_ENTER)

{

if(!play)

{

play = true;

ballposX = 120;

ballposY = 350;

ballXdir = -1;

ballYdir = -2;

playerX = 310;

score = 0;

Bricks = 48;

map = new MapGenerator(3, 7);

repaint();

}

}

}

public void keyReleased(KeyEvent e) {}

public void keyTyped(KeyEvent e) {}

public void moveRight()

{

play = true;

playerX+=20;

}

public void moveLeft()

{

play = true;

playerX-=20;

}

public void actionPerformed(ActionEvent e)

{

timer.start();

if(play)

{

if(new Rectangle(ballposX, ballposY, 20, 20).intersects(new Rectangle(playerX, 550, 30, 8)))

{

ballYdir = -ballYdir;

ballXdir = -2;

}

else if(new Rectangle(ballposX, ballposY, 20, 20).intersects(new Rectangle(playerX + 70, 550, 30, 8)))

{

ballYdir = -ballYdir;

ballXdir = ballXdir + 1;

}

else if(new Rectangle(ballposX, ballposY, 20, 20).intersects(new Rectangle(playerX + 30, 550, 40, 8)))

{

ballYdir = -ballYdir;

}

A: for(int i = 0; i<map.map.length; i++)

{

for(int j =0; j<map.map[0].length; j++)

{

if(map.map[i][j] > 0)

{

//scores++;

int brickX = j \* map.blockwidth + 80;

int brickY = i \* map.blockheight + 50;

int blockwidth = map.blockwidth;

int blockheight = map.blockheight;

Rectangle rect = new Rectangle(brickX, brickY, blockwidth, blockheight);

Rectangle ballRect = new Rectangle(ballposX, ballposY, 20, 20);

Rectangle brickRect = rect;

if(ballRect.intersects(brickRect))

{

map.BrickValue(0, i, j);

score+=5;

Bricks--;

if(ballposX + 19 <= brickRect.x || ballposX + 1 >= brickRect.x + brickRect.width)

{

ballXdir = -ballXdir;

}

else

{

ballYdir = -ballYdir;

}

break A;

}

}

}

}

ballposX += ballXdir;

ballposY += ballYdir;

if(ballposX < 0)

{

ballXdir = -ballXdir;

}

if(ballposY < 0)

{

ballYdir = -ballYdir;

}

if(ballposX > 670)

{

ballXdir = -ballXdir;

}

repaint();

}

}

}