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

class MapGenerator {

    public int map[][];
    public int brickWidth;
    public int brickHeight;

    // this creates the brick of size 3x7
    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;
            }
        }

        brickWidth = 540 / col;
        brickHeight = 150 / row;
    }

    // this draws the bricks
    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(new Color(0XFF8787)); // brick color
                    g.fillRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight);

                    g.setStroke(new BasicStroke(4));
                    g.setColor(Color.BLACK);
                    g.drawRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight);
                }
            }

        }
    }

    // this sets the value of brick to 0 if it is hit by the ball
    public void setBrickValue(int value, int row, int col) {
        map[row][col] = value;
    }

}

class GamePlay extends JPanel implements KeyListener, ActionListener {
```

```
private boolean play = true;
private int score = 0;

private int totalBricks = 21;

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(3, 7);
    addKeyListener(this);
    setFocusable(true);
    setFocusTraversalKeysEnabled(false);
    timer = new Timer(delay, this);
    timer.start();
}

public void paint(Graphics g) {

    // background color
    g.setColor(Color.YELLOW);
    g.fillRect(1, 1, 692, 592);

    map.draw((Graphics2D) g);

    g.fillRect(0, 0, 3, 592);
    g.fillRect(0, 0, 692, 3);
    g.fillRect(691, 0, 3, 592);

    g.setColor(Color.blue);
    g.fillRect(playerX, 550, 100, 12);

    g.setColor(Color.RED); // ball color
    g.fillOval(ballposX, ballposY, 20, 20);

    g.setColor(Color.black);
    g.setFont(new Font("MV Boli", Font.BOLD, 25));
    g.drawString("Score: " + score, 520, 30);
```

```
        if (totalBricks <= 0) { // if all bricks are destroyed then you win
            play = false;
            ballXdir = 0;
            ballYdir = 0;
            g.setColor(new Color(0XFF6464));
            g.setFont(new Font("MV Boli", Font.BOLD, 30));
            g.drawString("You Won, Score: " + score, 190, 300);

            g.setFont(new Font("MV Boli", Font.BOLD, 20));
            g.drawString("Press Enter to Restart.", 230, 350);
        }

        if (ballposY > 570) { // if ball goes below the paddle then you lose
            play = false;
            ballXdir = 0;
            ballYdir = 0;
            g.setColor(Color.BLACK);
            g.setFont(new Font("MV Boli", Font.BOLD, 30));
            g.drawString("Game Over, Score: " + score, 190, 300);

            g.setFont(new Font("MV Boli", Font.BOLD, 20));
            g.drawString("Press Enter to Restart", 230, 350);

        }
        g.dispose();
    }

    @Override
    public void actionPerformed(ActionEvent arg0) {
        timer.start();
        if (play) {
            // Ball - Pedal interaction
            if (new Rectangle(ballposX, ballposY, 20, 20).intersects(new Rectangle(playerX, 550,
100, 8))) {
                ballYdir = -ballYdir;
            }
            for (int i = 0; i < map.map.length; i++) { // Ball - Brick interaction
                for (int j = 0; j < map.map[0].length; j++) { // map.map[0].length is the number of
columns
                    if (map.map[i][j] > 0) {
                        int brickX = j * map.brickWidth + 80;
                        int brickY = i * map.brickHeight + 50;
                        int brickWidth = map.brickWidth;
                        int brickHeight = map.brickHeight;

                        Rectangle rect = new Rectangle(brickX, brickY, brickWidth, brickHeight);
                        Rectangle ballRect = new Rectangle(ballposX, ballposY, 20, 20);
                        Rectangle brickRect = rect;
```

```
                    if (ballRect.intersects(brickRect)) {
                        map.setBrickValue(0, i, j);
                        totalBricks--;
                        score += 5;

                        if (ballposX + 19 <= brickRect.x || ballposX + 1 >= brickRect.x +
brickRect.width)
                            ballXdir = -ballXdir;
                        else {
                            ballYdir = -ballYdir;
                        }
                    }

                }

            }
        }

        ballposX += ballXdir;
        ballposY += ballYdir;
        if (ballposX < 0) { // if ball hits the left wall then it bounces back
            ballXdir = -ballXdir;
        }
        if (ballposY < 0) { // if ball hits the top wall then it bounces back
            ballYdir = -ballYdir;
        }
        if (ballposX > 670) { // if ball hits the right wall then it bounces back
            ballXdir = -ballXdir;

        }

    }

    repaint();

  }

  @Override
  public void keyTyped(KeyEvent arg0) {

  }

  @Override
  public void keyPressed(KeyEvent arg0) {
    if (arg0.getKeyCode() == KeyEvent.VK_RIGHT) { // if right arrow key is pressed then
paddle moves right
        if (playerX >= 600) {
```

```
                playerX = 600;
            } else {
                moveRight();

            }
        }
        if (arg0.getKeyCode() == KeyEvent.VK_LEFT) { // if left arrow key is pressed then
paddle moves left
            if (playerX < 10) {
                playerX = 10;
            } else {
                moveLeft();

            }
        }

        if (arg0.getKeyCode() == KeyEvent.VK_ENTER) { // if enter key is pressed then game
restarts
            if (!play) {
                play = true;
                ballposX = 120;
                ballposY = 350;
                ballXdir = -1;
                ballYdir = -2;
                score = 0;
                totalBricks = 21;
                map = new MapGenerator(3, 7);

                repaint();
            }
        }

    }

    public void moveRight() { // paddle moves right by 50 pixels
        play = true;
        playerX += 50;
    }

    public void moveLeft() { // paddle moves left by 50 pixels
        play = true;
        playerX -= 50;
    }

    @Override
    public void keyReleased(KeyEvent arg0) {

    }
```

```
}

class Main {

    public static void main(String[] args) {
        JFrame obj = new JFrame();
        GamePlay gamePlay = new GamePlay();
        obj.setBounds(10, 10, 700, 600);
        obj.setTitle("Brick Breaker");
        obj.setResizable(false);
        obj.setVisible(true);
        obj.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        obj.add(gamePlay);

    }

}
```