```java
package brickbreaker;

/**
 * Required imports
 */
import java.awt.Color;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.Timer;

/**
 * BrickBreakerForm.java - This form is the main game form it contains mostly
 * everything for the game, It contains ball, brick and paddle also the score
 * and the game should end when the score is 225.
 *
 * @author Abhishek Shah
 * @since Jan. 5, 2020
 */
public class BrickBreakerForm extends javax.swing.JFrame {

    /**
     * Creates new form BrickBreakerForm
     */
    public BrickBreakerForm() {
        initComponents();
        start();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void formKeyPressed(java.awt.event.KeyEvent evt) {
        pressKey(evt);
    }

    private void formKeyReleased(java.awt.event.KeyEvent evt) {
        releasedKey(evt);
    }


    // Variables declaration - do not modify
```

```
    private javax.swing.JLabel bottomWall;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel18;
    private javax.swing.JLabel jLabel19;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel20;
    private javax.swing.JLabel jLabel21;
    private javax.swing.JLabel jLabel22;
    private javax.swing.JLabel jLabel23;
    private javax.swing.JLabel jLabel24;
    private javax.swing.JLabel jLabel25;
    private javax.swing.JLabel jLabel26;
    private javax.swing.JLabel jLabel27;
    private javax.swing.JLabel jLabel28;
    private javax.swing.JLabel jLabel29;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel30;
    private javax.swing.JLabel jLabel31;
    private javax.swing.JLabel jLabel32;
    private javax.swing.JLabel jLabel33;
    private javax.swing.JLabel jLabel34;
    private javax.swing.JLabel jLabel35;
    private javax.swing.JLabel jLabel36;
    private javax.swing.JLabel jLabel37;
    private javax.swing.JLabel jLabel38;
    private javax.swing.JLabel jLabel39;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel40;
    private javax.swing.JLabel jLabel41;
    private javax.swing.JLabel jLabel42;
    private javax.swing.JLabel jLabel43;
    private javax.swing.JLabel jLabel44;
    private javax.swing.JLabel jLabel45;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JLabel lblBall;
    private javax.swing.JLabel lblEnter;
    private javax.swing.JLabel lblGameOver;
    private javax.swing.JLabel lblPaddle;
```

```
    private javax.swing.JLabel lblScore;
    private javax.swing.JLabel leftCurb;
    private javax.swing.JLabel rightCurb;
    private javax.swing.JLabel topWall;
    // End of variables declaration

    // Here are some global variables
    private Timer ballTimer;
    private Timer paddleTimer;

    private int ballDirection;
    private int ballX = 120;
    private int ballY = 350;
    private final int ballW = 20;
    private final int ballH = 20;

    private int paddleDirection;
    private final int LEFT = 0;
    private final int RIGHT = 1;
    private final int UP_LEFT = 1;
    private final int UP_RIGHT = 2;
    private final int DOWN_LEFT = 3;
    private final int DOWN_RIGHT = 4;

    private int score = 0;

    private final int TOTAL_BRICKS = 45;
    private final JLabel[] brickLabels = new JLabel[TOTAL_BRICKS];

    Rectangle ballHitbox;
    Rectangle paddleHitbox;
    Rectangle leftWallHitbox;
    Rectangle rightWallHitbox;
    Rectangle topWallHitbox;
    Rectangle bottomWallHitbox;

    /**
     * This method makes the form settings and make the bricks appear also
     * randomizes the ball direction
     */
    private void start() {

        // Ball starts with a random direction
        ballDirection = (int) ((4 - 1 + 1) * Math.random() + 1);
        // Making arrays of bricks at the start for loops
        brickLabels[0] = jLabel1;
        brickLabels[1] = jLabel2;
        brickLabels[2] = jLabel3;
        brickLabels[3] = jLabel4;
        brickLabels[4] = jLabel5;
```

```
        brickLabels[5] = jLabel6;
        brickLabels[6] = jLabel7;
        brickLabels[7] = jLabel8;
        brickLabels[8] = jLabel9;
        brickLabels[9] = jLabel10;
        brickLabels[10] = jLabel11;
        brickLabels[11] = jLabel12;
        brickLabels[12] = jLabel13;
        brickLabels[13] = jLabel14;
        brickLabels[14] = jLabel15;
        brickLabels[15] = jLabel16;
        brickLabels[16] = jLabel17;
        brickLabels[17] = jLabel18;
        brickLabels[18] = jLabel19;
        brickLabels[19] = jLabel20;
        brickLabels[20] = jLabel21;
        brickLabels[21] = jLabel22;
        brickLabels[22] = jLabel23;
        brickLabels[23] = jLabel24;
        brickLabels[24] = jLabel25;
        brickLabels[25] = jLabel26;
        brickLabels[26] = jLabel27;
        brickLabels[27] = jLabel28;
        brickLabels[28] = jLabel29;
        brickLabels[29] = jLabel30;
        brickLabels[30] = jLabel31;
        brickLabels[31] = jLabel32;
        brickLabels[32] = jLabel33;
        brickLabels[33] = jLabel34;
        brickLabels[34] = jLabel35;
        brickLabels[35] = jLabel36;
        brickLabels[36] = jLabel37;
        brickLabels[37] = jLabel38;
        brickLabels[38] = jLabel39;
        brickLabels[39] = jLabel40;
        brickLabels[40] = jLabel41;
        brickLabels[41] = jLabel42;
        brickLabels[42] = jLabel43;
        brickLabels[43] = jLabel44;
        brickLabels[44] = jLabel45;

        //Some settings for the form
        this.setTitle(Globals.APPLICATION_TITLE);
        this.setResizable(false);
        this.setSize(632, 581);
        this.setLocationRelativeTo(null);
        setTimers();
        this.setBackground(Color.black);
        this.getContentPane().setBackground(Color.black);
        this.setVisible(true);
```

```
    }

    /**
     * When the user presses a key on the keyboard
     *
     * @param eventthe keyboard event for the specific key
     */
    private void pressKey(KeyEvent event) {
        // Check which key was preseed
        if (paddleTimer.isRunning() == false) {
            paddleTimer.start();
        }
        if (event.getKeyCode() == KeyEvent.VK_LEFT) {
            // Set to move left
            paddleDirection = LEFT;
        } else if (event.getKeyCode() == KeyEvent.VK_RIGHT) {
            // Set to move right
            paddleDirection = RIGHT;
        } else if (event.getKeyCode() == KeyEvent.VK_ENTER) {
            // If user presses an enter key the game restarts
            Restart();
        }
    }

    /**
     * This method gets the coordinates of the paddle also runs a check
     * collisions method in it with the ball and the side walls
     */
    private void paddleTick() {
        int x = lblPaddle.getX();
        int y = lblPaddle.getY();
        int w = lblPaddle.getWidth();
        int h = lblPaddle.getHeight();

        if (paddleDirection == LEFT) {
            x = x - Globals.PADDLE_AMOUNT;
        } else if (paddleDirection == RIGHT) {
            x = x + Globals.PADDLE_AMOUNT;
        }
        checkCollisions();
        lblPaddle.setBounds(x, y, w, h);
    }

    /**
     * This method randomizes the ball direction and also runs check collisions
     * method for the every brick, walls, and paddle
     */
    private void ballTick() {
        ballX = lblBall.getX();
        ballY = lblBall.getY();
```

```
        if (ballDirection == UP_LEFT) {
            ballX = ballX - Globals.BALL_AMOUNT;
            ballY = ballY - Globals.BALL_AMOUNT;
        } else if (ballDirection == UP_RIGHT) {
            ballX = ballX + Globals.BALL_AMOUNT;
            ballY = ballY - Globals.BALL_AMOUNT;
        } else if (ballDirection == DOWN_LEFT) {
            ballX = ballX - Globals.BALL_AMOUNT;
            ballY = ballY + Globals.BALL_AMOUNT;
        } else if (ballDirection == DOWN_RIGHT) {
            ballX = ballX + Globals.BALL_AMOUNT;
            ballY = ballY + Globals.BALL_AMOUNT;
        }
        checkCollisions();
        lblBall.setBounds(ballX, ballY, ballW, ballH);
    }

    /**
     * This method run different methods of collisions with objects
     */
    private void checkCollisions() {
        // get all hitboxes
        ballHitbox = getHitbox(lblBall);
        paddleHitbox = getHitbox(lblPaddle);
        leftWallHitbox = getHitbox(leftCurb);
        rightWallHitbox = getHitbox(rightCurb);
        topWallHitbox = getHitbox(topWall);
        bottomWallHitbox = getHitbox(bottomWall);
        checkPaddleWallsCollison();
        checkBallWallsCollisions();
        checkPaddleBallCollison();
        checkBallBricksCollsion();
    }

    /**
     * This method is for ball to brick collisions
     */
    private void checkBallBricksCollsion() {
        // Now check for any ball to brick collisions
        // loop through all bricks in the array
        for (int i = 0; i < brickLabels.length; i++) {
            // pull a brick label out of the array of brick labels
            JLabel brickLabel = brickLabels[i];
            // only worry about hits label if it is visible
            if (brickLabel.isVisible()) {
                // make a hitbox for that one brick label
                Rectangle brickHitbox = getHitbox(brickLabel);
                // check ball to brick collision
                if (ballHitbox.intersects(brickHitbox)) {
```

```
                    // make brick disappear
                    brickLabel.setVisible(false);
                    score = score + 5;
                    lblScore.setText("Score: " + score);
                    // now leave the loop and leave the method
                    return;
                }
            }
        }
    }

    /**
     * This method is for paddle to ball collisions
     */
    private void checkPaddleBallCollison() {
        // check if ball hits paddle
        if (ballHitbox.intersects(paddleHitbox)) {
            if (ballDirection == DOWN_LEFT) {
                ballDirection = UP_LEFT;
            } else if (ballDirection == DOWN_RIGHT) {
                ballDirection = UP_RIGHT;
            }
        }
    }

    /**
     * This method is for paddle to wall collisions
     */
    private void checkPaddleWallsCollison() {
        // check if paddle is hitting a wall
        if (paddleHitbox.intersects(leftWallHitbox)) {
            paddleDirection = RIGHT;
        } else if (paddleHitbox.intersects(rightWallHitbox)) {
            paddleDirection = LEFT;
        }
    }

    /**
     * This method is for ball to wall collisions
     */
    private void checkBallWallsCollisions() {
        // Now check for ball to any wall collisions
        if (ballHitbox.intersects(leftWallHitbox)) {
            if (ballDirection == UP_LEFT) {
                ballDirection = UP_RIGHT;
            } else if (ballDirection == DOWN_LEFT) {
                ballDirection = DOWN_RIGHT;
            }
        } else if (ballHitbox.intersects(rightWallHitbox)) {
            if (ballDirection == UP_RIGHT) {
```

```
            ballDirection = UP_LEFT;
        } else if (ballDirection == DOWN_RIGHT) {
            ballDirection = DOWN_LEFT;
        }
    } else if (ballHitbox.intersects(topWallHitbox)) {
        if (ballDirection == UP_LEFT) {
            ballDirection = DOWN_LEFT;
        } else if (ballDirection == UP_RIGHT) {
            ballDirection = DOWN_RIGHT;
        }
    } else if (ballHitbox.intersects(bottomWallHitbox)) {
        lblGameOver.setVisible(true);
        lblGameOver.setHorizontalAlignment((int) CENTER_ALIGNMENT);
        lblGameOver.setText("GAME OVER!,Your score was : " + score);

        lblEnter.setVisible(true);
        lblEnter.setHorizontalAlignment((int) CENTER_ALIGNMENT);
        lblEnter.setText(" Press (ENTER) to restart ");
    } else if (score == 225) {
        lblGameOver.setVisible(true);
        lblGameOver.setHorizontalAlignment((int) CENTER_ALIGNMENT);
        lblGameOver.setText("YOU WON!");

        paddleTimer.stop();
        ballTimer.stop();

        lblEnter.setVisible(true);
        lblEnter.setHorizontalAlignment((int) CENTER_ALIGNMENT);
        lblEnter.setText(" Press (ENTER) to restart ");
    }
}

/**
 * Gets a rectangle object from the passed label object to be used for
 * collision detection for any object
 *
 * @param labelthe label object to use
 * @return a rectangle object
 */
private Rectangle getHitbox(JLabel label) {
    int x = label.getX();                    // Using built-in label methods
    int y = label.getY();
    int w = label.getWidth();
    int h = label.getHeight();
    Rectangle hitbox = new Rectangle(x, y, w, h); // creates rectangle
    return hitbox;
}

/**
 * This method is for the timers to start
```

```
     */
    private void setTimers() {
        //Some timers start
        paddleTimer = new Timer(Globals.PADDLE_SPEED, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                paddleTick();
            }
        });

        ballTimer = new Timer(Globals.BALL_SPEED, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                ballTick();
            }
        });
        ballTimer.start();
    }

    /**
     * This method restarts the game
     */
    private void Restart() {
        BrickBreakerForm screen = new BrickBreakerForm();
    }

    /**
     * This method is for when the key is pressed then only it moves
     *
     * @param evt it controls how it moves when the keys are pressed
     */
    private void releasedKey(KeyEvent evt) {
        paddleTimer.stop();
    }
}
```

```java
package brickbreaker;

/**
 * Required imports
 */
import java.awt.Color;

/**
 * BrickBreakerStartForm.java - This form asks for user to press the start
 * button to play brick breaker game.
 *
 * @author Abhishek Shah
 * @since Jan. 5, 2020
 */
public class BrickBreakerStartForm extends javax.swing.JFrame {

    /**
     * Creates new form BrickBreakerStartForm
     */
    public BrickBreakerStartForm() {
        initComponents();
        start();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void start() {
        // Some settings for the form
        this.setTitle(Globals.APPLICATION_TITLE);
        this.setResizable(false);
        this.setSize(440, 345);
        this.setLocationRelativeTo(null);
        this.setBackground(Color.black);
        this.getContentPane().setBackground(Color.black);
        this.setVisible(true);
    }

    /**
     * When the mouse is clicked it should switch the form
     *
     * @param evt opens the new game form
     */
    private void btnStartMouseClicked(java.awt.event.MouseEvent evt) {
        BrickBreakerForm screen = new BrickBreakerForm();
```

```
    }


    // Variables declaration - do not modify
    private javax.swing.JButton btnStart;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration
}
```

```java
package brickbreaker;

/**
 * BrickBreaker.java - This is main class to generate the new form of
 * the brick breaker start class.
 *
 * @author Abhishek Shah
 * @since Jan. 5, 2020
 */
public class Brickbreaker {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // This runs the start class to appear the start form
        BrickBreakerStartForm screen = new BrickBreakerStartForm();
    }
}
```

```java
package brickbreaker;

/**
 * Required imports
 */
import java.awt.Color;

/**
 * Globals.java - classes can contain "properties" which are essentially "global
 * variables" to that class - but can also be thought of as "things about that
 * class" (or "descriptors" or even "adjectives"). This class is using
 * properties which are "global" to all the other classes in the project.
 *
 * @author Abhishek Shah
 * @since Jan. 5, 2020
 */
public class Globals {
    //Some global variables  to transfer them class by class easily.

    public static final String APPLICATION_TITLE = "Brick Breaker";
    public static final Color FORM_COLOR = Color.black;

    public static final Color PADDLE_COLOR = Color.blue;
    public static final int PADDLE_SPEED = 14;
    public static final int PADDLE_AMOUNT = 12;

    public static final Color BALL_COLOR = Color.yellow;
    public static final int BALL_DELAY = 10;
    public static final int BALL_SPEED = 18;
    public static final int BALL_AMOUNT = 4;

    public static final int BRICK_AMOUNT = 10;
    public static final int BRICK_WIDTH = 50;
    public static final int BRICK_HEIGHT = 30;
}
```