RayCastJava.java (main)

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

public class RayMaze extends JFrame {
    private JLabel titleLabel;
    private JLabel gridSizeLabel;
    private JTextField gridSizeTextField;
    private JLabel wallInstanceLabel;
    private JTextField wallInstanceTextField;
    private JButton startButton;
    private JButton exitButton;
    public RayMaze() {

        setUndecorated(true);
        setTitle("Ray Maze");
        setBackground( new Color(0,0,0,85) );
        setSize(400, 325);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridLayout(7, 8));

        exitButton = new JButton("<html><font color='white'>Exit</font></html>");
        exitButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                dispose();
            }
        });
        exitButton.setBackground( new Color(0,0,0,85) );
        add(exitButton);
        titleLabel = new JLabel("<html><font color='white'>Ray
Maze</font></html>");
        titleLabel.setHorizontalAlignment(SwingConstants.CENTER);
        add(titleLabel);

        gridSizeLabel = new JLabel("<html><font color='white'>Grid
Size</font></html>", SwingConstants.CENTER);
        add(gridSizeLabel);

        gridSizeTextField = new JTextField("50");
        gridSizeTextField.setHorizontalAlignment(JTextField.CENTER);
        gridSizeTextField.setForeground(Color.BLACK);
        add(gridSizeTextField);

        wallInstanceLabel = new JLabel("<html><font color='white'>Wall Instance
Amount</font></html>", SwingConstants.CENTER);
```

```
        add(wallInstanceLabel);

        wallInstanceTextField = new JTextField("50");
        wallInstanceTextField.setHorizontalAlignment(JTextField.CENTER);

        wallInstanceTextField.setForeground(Color.BLACK);
        add(wallInstanceTextField);

        startButton = new JButton("<html><font color='white'>Start
Game</font></html>");
        startButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int gridSize = Integer.parseInt(gridSizeTextField.getText());
                int wallInstanceAmount =
Integer.parseInt(wallInstanceTextField.getText());
                DrawFrame frame = new DrawFrame(gridSize, wallInstanceAmount);
                dispose();
            }
        });
        startButton.setBackground( new Color(0,0,0,85) );
        add(startButton);

        setVisible(true);
    }
}
```

DrawFrame.java

```
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import javax.swing.JFrame;
import javax.swing.border.Border;
import java.util.Random;
import static java.lang.System.out;
import javax.swing.JPanel;
import java.awt.Color;
import java.awt.Font;
import java.net.CookieHandler;
public class DrawFrame {
    int pixel_size = 12;
    int pixel_size_x = pixel_size;
    int pixel_size_y = pixel_size;
    int pixel_amount_x = 69;
    int pixel_amount_y = 60;
    int currentMove = 0;
    JPanel[][] pixelList;
    public DrawFrame(int gridSize, int wallSpawnProbability) {
```

```java
        JFrame frame = new JFrame();
        DrawMap map = new DrawMap(gridSize, wallSpawnProbability);
        pixelList = new JPanel[pixel_amount_x][pixel_amount_y];
        frame.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                int keyCode = e.getKeyCode();
                currentMove += 1;
                if (keyCode == KeyEvent.VK_W) {
                    map.move(false);
                }
                else if (keyCode == KeyEvent.VK_A) {
                    map.faceDirection("left");
                }
                else if (keyCode == KeyEvent.VK_D) {
                    map.faceDirection("right");
                } else if (keyCode == KeyEvent.VK_S) {
                    map.move(true);
                }
                map.printMap();
                System.out.println(map.getFacing());
                refresh(map.getMap(), map.getFacing(), pixelList);
            }
        });
        Font f1 = new Font(Font.SANS_SERIF, Font.PLAIN, 20);
        frame.setLayout(null);
        frame.setTitle("Ray Maze - Active Game");
        frame.setVisible(false);
        frame.getContentPane().setBackground(Color.BLACK);
        frame.setSize(pixel_size_x*pixel_amount_x,pixel_size_y*pixel_amount_y);
        frame.setResizable(false);
        int count = 0;
        for (int x = 0; x < pixel_amount_x; x++) {
            for (int y = 0; y < pixel_amount_y; y++) {
                JPanel pixel = new JPanel();
                if (y == pixel_amount_y/2 || y == (pixel_amount_y/2)+0.5) {
                    count++;
                    pixel.setBackground(Color.WHITE);
                } else {
                    pixel.setBackground(Color.BLACK);
                }

pixel.setBounds(x*pixel_size_x,y*pixel_size_y,pixel_size_x,pixel_size_y);
                pixelList[x][y] = pixel;
                frame.add(pixel);
            }
        }
        frame.setVisible(true);

    }


    public void updatePixel(int row, int column, String clr, JPanel[][]
pixelList_Parameter) {

        if (row < 0) {
            row = 0;
        }

        if (column < 0) {
            column = 0;
        }

        if (row >= pixel_amount_x) {
            row -= (row-pixel_amount_x) + 1;
        }

        if (column >= pixel_amount_y) {
            column -= (column-pixel_amount_y) + 1;
        }

        if (pixelList_Parameter != null && pixelList_Parameter[row] != null ||
pixelList_Parameter[row][column] != null) {
            if (clr == "white" &&
!pixelList_Parameter[row][column].getBackground().equals(Color.WHITE)) {
                pixelList_Parameter[row][column].setBackground(Color.WHITE);
                pixelList_Parameter[row][column].repaint();
            } else if (clr == "black" &&
!pixelList_Parameter[row][column].getBackground().equals(Color.BLACK)) {
                pixelList_Parameter[row][column].setBackground(Color.BLACK);
                pixelList_Parameter[row][column].repaint();
            } else if (clr == "gray" &&
!pixelList_Parameter[row][column].getBackground().equals(Color.LIGHT_GRAY)) {
                pixelList_Parameter[row][column].setBackground(Color.LIGHT_GRAY);
                pixelList_Parameter[row][column].repaint();
            } else if (clr == "green" &&
!pixelList_Parameter[row][column].getBackground().equals(Color.GREEN)) {
                pixelList_Parameter[row][column].setBackground(Color.GREEN);
                pixelList_Parameter[row][column].repaint();
            }
        }
    }
    public void updatePixelColumn(int column, int size, JPanel[][]
pixelList_Parameter) {
        int dimension = pixelList[0].length;
        double midPoint;
        if (dimension % 2 == 0) {
            midPoint = (dimension/2)+1;
        } else {
            midPoint = (dimension/2) + 0.5;
        }

        int middle = (int) midPoint;

        for (int iterate = 0; iterate < dimension; iterate++) {
            updatePixel(column,iterate,"black",pixelList_Parameter);
        }
        updatePixel(column,middle,"gray",pixelList_Parameter);
        for (int iterate = 0; iterate < size; iterate++) {
            if (size < 2) {
                updatePixel(column,middle-iterate,"gray",pixelList_Parameter);
                updatePixel(column,middle+iterate,"gray",pixelList_Parameter);
            } else {
                updatePixel(column,middle-iterate,"white",pixelList_Parameter);
                updatePixel(column,middle+iterate,"white",pixelList_Parameter);
            }
        }
    }

    public int distanceFromFacing(String map, String facing, int x, int y) {
        int distance = 0;
        boolean startCounting = false;
        if (facing == "North") {
            for (int iterate = y; iterate > 0; iterate--) {
                if ((map.split("\n")[iterate].charAt(x)+"").equals("#")) {
                    break;
                } else {
                    distance += 1;
                }
            }
        } else if (facing == "East") {
            String line = map.split("\n")[y];
            for (int iterating = 0; iterating < line.length(); iterating++) {
                String currentLetter = line.charAt(iterating) + "";
                if (startCounting) {
                    distance += 1;
                }
                if (currentLetter.equals("*")) {
```

```java
                    startCounting = true;
                } else if (currentLetter.equals("#") && startCounting) {
                    break;
                }
            }
        } else if (facing == "South") {
            for (int iterate = y; iterate < map.split("\n").length; iterate++) {
                if ((map.split("\n")[iterate].charAt(x)+"").equals("#")) {
                    break;
                } else {
                    distance += 1;
                }
            }
        } else if (facing == "West") {
            String line = map.split("\n")[y];
            for (int iterating = line.length()-1; iterating >= 0; iterating--) {
                String currentLetter = line.charAt(iterating) + "";
                if (startCounting) {
                    distance += 1;
                }
                if (currentLetter.equals("*")) {
                    startCounting = true;
                } else if (currentLetter.equals("#") && startCounting) {
                    break;
                }
            }
        }
        return distance;
    }

    public int[] locateCharFromString(String map) {
        int x_local = 0;
        int y_local = 0;

        for (int character = 0; character < map.length(); character++) {
            if (String.valueOf(map.charAt(character)).equals("*")) {
                break;
            }
            if (String.valueOf(map.charAt(character)).equals("\n")) {
                y_local += 1;
                x_local = 0;
            } else {
                x_local += 1;
            }
        }
        int[] result = {x_local,y_local};
        return result;
    }

    public int sendRayForDistance(String map, String facing, double tilt) {
        int[] xy = locateCharFromString(map);
        int x = xy[0];
        int y = xy[1];
        double new_x = x;
        double new_y = y;
        int distance;
        int increment = 0;
        while (true) {
            if (facing.equals("North")) {
                new_y -= 1;
                new_x += tilt;
            } else if (facing.equals("East")) {
                new_x += 1;
                new_y += tilt;
            } else if (facing.equals("South")) {
                new_y += 1;
                new_x -= tilt;
            } else if (facing.equals("West")) {
                new_x -= 1;
                new_y -= tilt;
            }
            int temp_x = (int) Math.round(new_x);
            int temp_y = (int) Math.round(new_y);

            if (temp_y >= 0 && temp_y < map.split("\n").length && temp_x >= 0 &&
temp_x < map.split("\n")[temp_y].length()) {
                if (map.split("\n")[temp_y].charAt(temp_x) == '#') {
                    break;
                }
            } else {
                break;
            }
            increment++;
        }
        return increment+1;
    }




    public void refresh(String map, String facing, JPanel[][] pixelList_Parameter)
{
        int[] xy = locateCharFromString(map);
        int x = xy[0];
        int y = xy[1];

        double midPoint;
        int dimension = pixelList_Parameter.length;
        if (dimension % 2 == 0) {
            midPoint = (dimension/2)+1;
        } else {
            midPoint = (dimension/2) + 0.5;
        }

        int middle = (int) midPoint;

        int[] charPos = locateCharFromString(map);
        int distance_straight_ahead =
distanceFromFacing(map,facing,charPos[0],charPos[1]);
        int orig = distance_straight_ahead;
        distance_straight_ahead = (middle-distance_straight_ahead)+1;
        if (distance_straight_ahead == -1) {
            distance_straight_ahead = (middle-orig);
        }
        updatePixelColumn(middle,(middle-distance_straight_ahead)+1,
pixelList_Parameter);
        if (facing == "South" || facing == "West") {
            for (double current = dimension-1; current > -1; current--) {
                double tilt = -0.5 + current/(dimension-1);
                int distance = sendRayForDistance(map,facing,tilt);
                updatePixelColumn((int) current,middle-distance+1,
pixelList_Parameter);
            }
        } else {
            for (double current = 0; current < dimension; current++) {
                double tilt = -0.5 + current/(dimension-1);
                int distance = sendRayForDistance(map,facing,tilt);
                updatePixelColumn((int) current,middle-distance+1,
pixelList_Parameter);
            }
        }
    }
}
```

DrawMap.java

```
import java.util.Random;

public class DrawMap {
    private String MOVE_KEY = "W";

    private int sizeGrid;
    private int plrSpawnX = 6;
    private int plrSpawnY = 6;
    private String grid = "";
    private boolean horizontal;
    private boolean placedFinishLine;
    private boolean placedPlayer;
    private int dimension;
    // 0 = West
    // 1 = North
    // 2 = East
    // 3 = South
    private String[] facingWays = {"West", "North", "East", "South"};
    private int facing = 2;
    public DrawMap(int gridSize, int wallSpawnProbability) {
        sizeGrid = gridSize;
        if (sizeGrid % 2 != 0) {
            sizeGrid -= 1;
        }
        this.dimension = sizeGrid;
        int offset_x = new Random().nextInt(sizeGrid/2);
        int offset_y = new Random().nextInt(sizeGrid/2);

        placedFinishLine = false;
        placedPlayer = false;
        int finish_x = ((int) new Random().nextInt(50)) + 1;
        int iterate = 0;

        for (int x = 0; x < sizeGrid; x++) {
            for (int y = 0; y < sizeGrid; y++) {
                iterate += 1;
                if (iterate == finish_x && !placedFinishLine) {
                    this.grid += "F";
                    placedFinishLine = true;
                } else if (!placedPlayer && x == plrSpawnX && y == plrSpawnY) {
                    placedPlayer = true;
                    this.grid += "*";
                } else if (x == 0 || x == sizeGrid - 1 || y == 0 || y == sizeGrid -
1) {
                    this.grid += "#";
                } else if (new Random().nextInt(wallSpawnProbability) == 1) {
                    this.grid += "#";
                } else {
                    this.grid += " ";
                }
```

```java
            }
            this.grid += "\n";
        }
    }

    public int getDimension() {
        return this.dimension;
    }

    public String getMap() {
        return this.grid;
    }

    public String locateChar() {
        int x_local = 0;
        int y_local = 0;

        for (int character = 0; character < getMap().length(); character++) {
            if (String.valueOf(getMap().charAt(character)).equals("*")) {
                break;
            }
            if (String.valueOf(getMap().charAt(character)).equals("\n")) {
                y_local += 1;
                x_local = 0;
            } else {
                x_local += 1;
            }
        }

        return Integer.toString(x_local) + "," + Integer.toString(y_local);
    }


    public void printMap() {
        System.out.println(this.grid);
    }

    public void faceDirection(String direction) {
        if (direction == "left") {
            this.facing -= 1;
        } else {
            this.facing += 1;
        }

        if (this.facing == -1) {
            this.facing = 3;
        } else if (facing == 4) {
            this.facing = 0;
        }
    }

    public String getFacing() {
        return facingWays[this.facing];
    }


    // x and y start at 0
    public String indexGrid(String grid, int x,int y) {
        return grid.split("\n")[y].substring(x,x+1);
    }

    public String changeGridAtIndex(String grid, int x, int y, String changeTo) {
        String[] output = grid.split("\n");
        if (y >= 0 && y < output.length && x >= 0 && x < output[y].length()) {
            output[y] = output[y].substring(0, x) + changeTo +
output[y].substring(x + 1);
        } else {
            System.out.println("Error: Coordinates are out of bounds.");
        }

            return String.join("\n", output);
    }

    public void move(boolean reverse) {
        int moveAmount;
        if (reverse) {
            moveAmount = -1;
        } else {
            moveAmount = 1;
        }
        String charPosString = locateChar();
        boolean blockage = false;
        int[] charPos = {Integer.parseInt(charPosString.split(",")[0]),
Integer.parseInt(charPosString.split(",")[1])};
        System.out.println(charPosString);
        String direction = getFacing();

        if (direction.equals("North")) {
            if (indexGrid(this.grid, charPos[0], charPos[1] -
moveAmount).equals("#")) {
                blockage = true;
            }
        } else if (direction.equals("South")) {
            if (indexGrid(this.grid, charPos[0], charPos[1] +
moveAmount).equals("#")) {
                blockage = true;
            }
        } else if (direction.equals("East")) {
            if (indexGrid(this.grid, charPos[0] + moveAmount,
charPos[1]).equals("#")) {
                blockage = true;
            }
        } else if (direction.equals("West")) {
            if (indexGrid(this.grid, charPos[0] - moveAmount,
charPos[1]).equals("#")) {
                blockage = true;
            }
        }
        if (!blockage) {
            this.grid = this.grid.replace("*", " ");

            if (direction.equals("North")) {
                this.grid = changeGridAtIndex(this.grid, charPos[0], charPos[1] -
moveAmount, "*");
            } else if (direction.equals("South")) {
                this.grid = changeGridAtIndex(this.grid, charPos[0], charPos[1] +
moveAmount, "*");
            } else if (direction.equals("East")) {
                this.grid = changeGridAtIndex(this.grid, charPos[0] + moveAmount,
charPos[1], "*");
            } else if (direction.equals("West")) {
                this.grid = changeGridAtIndex(this.grid, charPos[0] - moveAmount,
charPos[1], "*");
            }
        }
    }

}
```