```java
// Rachel Burke and Lauren Marx
// rangeMap.java: This program draws the range map of a robot in an enclosed 2D space.

import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.lang.Math;

import javax.swing.border.EmptyBorder;
import javax.swing.*;

public class rangeMap {
    // Range Map Input Variables
    int v, n, rx, ry, rd; // Verticies, # of Sensors, Robot x-coord, Robot y-coord, Robot degrees
    (cc)
    ArrayList<Integer> x = new ArrayList<Integer>(); // List of x-coords of corners
    ArrayList<Integer> y = new ArrayList<Integer>(); // List of y-coords of corners

    // Window Variables
    private Frame mainFrame; // The Main Window
    private Label statusMsg; // Status label in mainFrame
    private JPanel instructionPanel; // Top panel of instructions in the mainFrame
    private Label instructionsLabel1; // Label 1 in instructionPanel
    private Label instructionsLabel2; // Label 2 in instructionPanel
    private Label instructionsLabel3; // Label 3 in instructionPanel
    private JPanel controlPanel; // Contains the inputPanel and cvMap in the mainFrame
    private JPanel inputPanel; // Contains range map input in the control Panel
    private final TextField vText = new TextField(5);
    private final TextField nText = new TextField(5);
    private final TextField rdText = new TextField(5);
    private final TextField coordText = new TextField(30);
    private final TextField robotText = new TextField(10);
    private JButton enterButton; // Button that allows for the submission of user input
    private CvMap cvMap; // Range Map Canvas

    /**
     * @The Main Function initializes a range map window and displays the content
     */
    public static void main(String[] args) {
        rangeMap map = new rangeMap();
        map.showContent();
    }

    // Window Setup Functions

    /**
     * @Prepare the GUI for the Range Map Project
     */
    public rangeMap() {
        prepareGUI();
    }

    /**
     * @Create the GUI
     */
    private void prepareGUI() {

        initializeInstructions();
        setupInputPanel();
        setupControlPanel();
        setupMainFrame();
    }

    /**
     * @Setup the instruciton labels and add the instructions
     * @Initialize the Instruction JPanel Set
     * @Instruction Panel Properties Add Instuction labels
     */
    private void initializeInstructions() {
```

```
        instructionsLabel1 = new Label(
                "Input the number of verticies in the 2D space. Then list in order the coordinates
                of each vertex given as an ordered pair (x,y) and separated by a \",\" comma.");
        instructionsLabel1.setAlignment(Label.LEFT);
        instructionsLabel1.setForeground(Color.WHITE);

        instructionsLabel2 = new Label(
                "Give the position (x,y) and the heading direction of the robot in the room as
                well as a number of sensors. Click \"Enter\" to see results.");
        instructionsLabel2.setAlignment(Label.LEFT);
        instructionsLabel2.setForeground(Color.WHITE);

        instructionsLabel3 = new Label(
                "Default setting is 4 vertices with locations (0,0), (0,100), (100,100), and
                (100,0). The robot is at (50,50) heading at 0 degrees with 20 sensors.");
        instructionsLabel3.setAlignment(Label.LEFT);
        instructionsLabel3.setForeground(Color.WHITE);

        instructionPanel = new JPanel();
        instructionPanel.setLayout(new FlowLayout());
        instructionPanel.setPreferredSize(new Dimension(1200, 60));
        instructionPanel.setBackground(Color.DARK_GRAY);
        instructionPanel.setBorder(new EmptyBorder(10, 5, 5, 5));

        instructionPanel.add(instructionsLabel1);
        instructionPanel.add(instructionsLabel2);
        instructionPanel.add(instructionsLabel3);
    }

    /**
     * @Initialize the input panel
     */
    private void setupInputPanel() {
        inputPanel = new JPanel();
        inputPanel.setLayout(new BoxLayout(inputPanel, BoxLayout.Y_AXIS));
    }

    /**
     * @Initialize the control panel
     * @Setup the control panel
     * @Add control panel content
     */
    private void setupControlPanel() {
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        controlPanel.setPreferredSize(new Dimension(1200, 450));
        controlPanel.setBorder(new EmptyBorder(10, 5, 0, 5));

        controlPanel.add(inputPanel);
    }

    /**
     * @Initialize the main frame
     * @Setup the main frame
     * @Add main frame content
     */
    private void setupMainFrame() {
        mainFrame = new Frame("Range Map");
        mainFrame.setSize(1200, 700);
        mainFrame.setResizable(false);
        mainFrame.setBackground(Color.DARK_GRAY);
        mainFrame.setForeground(Color.WHITE);
        mainFrame.setLayout(new BoxLayout(mainFrame, BoxLayout.Y_AXIS));
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });

        statusMsg = new Label("Status Message: ", Label.LEFT);
```

```
        mainFrame.add(instructionPanel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusMsg);
        mainFrame.setVisible(true);
    }

    /**
     * Show content in control panel
     */
    private void showContent() {
        Label vLabel = new Label("Number of Verticies: ", Label.LEFT);
        Label coordLabel = new Label("List of Coordinates (x,y): ", Label.LEFT);
        Label nLabel = new Label("Number of Robot Sensors: ", Label.LEFT);
        Label rdLabel = new Label("Directon of the Robot: ", Label.LEFT);
        Label robotLabel = new Label("Robot Position (x,y): ", Label.LEFT);
        resetAll();

        enterButton = new JButton("Enter");
        cvMap = new CvMap(v, x, y, n, rd, rx, ry);
        btnEnter(enterButton, statusMsg);

        inputPanel.add(vLabel);
        inputPanel.add(vText);
        inputPanel.add(coordLabel);
        inputPanel.add(coordText);
        inputPanel.add(nLabel);
        inputPanel.add(nText);
        inputPanel.add(rdLabel);
        inputPanel.add(rdText);
        inputPanel.add(robotLabel);
        inputPanel.add(robotText);
        inputPanel.add(enterButton);

        controlPanel.add(cvMap);

        mainFrame.setVisible(true);
    }

    /**
     * @Clicking Enter Button
     */
    private void btnEnter(JButton enterButton, Label statusMsg) {
        enterButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (!(isTxtEmpty(vText, statusMsg) || isTxtEmpty(coordText, statusMsg) ||
                isTxtEmpty(nText, statusMsg)
                        || isTxtEmpty(rdText, statusMsg) || isTxtEmpty(robotText, statusMsg))) {
                    setV();
                    setN();
                    setRD();
                    setCoords();
                    setRobot();

                    boolean badValue = false;

                    for (Integer xCoord : x)
                        if (xCoord > cvMap.maxX || xCoord < 0) {
                            resetAll();
                            badValue = true;
                        }
                    for (Integer yCoord : y)
                        if (yCoord > cvMap.maxY || yCoord < 0) {
                            resetAll();
                            badValue = true;
                        }
                    if (badValue)
                        statusMsg.setText("Status Message: Bad coordinates given. Bounds are 0-" +
                        cvMap.maxX
                                + " for x coordinates and 0-" + cvMap.maxY
```

```
                                + " for y coordinates. Default values used.");
                    else if (x.size() != y.size() || x.size() != v || x.size() < 3) {
                        resetAll();
                        statusMsg.setText("Status Message: Bad information given. Default values
                        used.");
                    } else
                        statusMsg.setText("Status Message: Success!");
                }
                cvMap.setv(v);
                cvMap.setx(x);
                cvMap.sety(y);
                cvMap.setn(n);
                cvMap.setrd(rd);
                cvMap.setrx(rx);
                cvMap.setry(ry);
                cvMap.repaint();
                controlPanel.add(cvMap);
            }
        });
    }

    /**
     * @Checks to see if text fields have input
     * @Sets to default if empty
     * @Returns true for defaul and false for detected input
     */
    private boolean isTxtEmpty(TextField txtField, Label statusMsg) {
        if (txtField.getText().isEmpty()) {
            resetAll();
            statusMsg.setText("Status Message: No information given. Default values used.");
            return true;
        }
        return false;
    }

    // Set input functions

    /**
     * @Set v to textfield input
     */
    private void setV() {
        v = Integer.parseInt(vText.getText());
    }

    /**
     * Set x and y array list coordinates
     */
    private void setCoords() {
        String coords = coordText.getText().replaceAll("[()]", "");
        coords = coords.replaceAll("\\s+", "");
        String[] xy = coords.split(",");
        x.clear();
        y.clear();
        for (String xypair : xy) {
            if (!(xypair.isEmpty() || xypair.equals(" "))) {
                if (x.size() < y.size())
                    x.add(Integer.parseInt(xypair));
                else
                    y.add(Integer.parseInt(xypair));
            }
        }
    }

    /**
     * @Set n to textfield input
     */
    private void setN() {
        n = Integer.parseInt(nText.getText());
    }
```

```
    /**
     * @Set d to textfield input
     */
    private void setRD() {
        rd = Integer.parseInt(rdText.getText());
    }

    /**
     * @Set robot position
     */
    private void setRobot() {
        String position = robotText.getText().replaceAll("[()]", "");
        position = position.replaceAll("\\s+", "");
        String[] coords = position.split(",");
        boolean robotXFound = false;
        boolean robotYFound = false;
        for (String coord : coords) {
            if (!(coord.isEmpty() || coord.equals(" "))) {
                if (!robotXFound) {
                    rx = Integer.parseInt(coord);
                    robotXFound = true;
                } else if (!robotYFound) {
                    ry = Integer.parseInt(coord);
                    robotYFound = true;
                }
            }
        }
    }

    // Reset Functions for range map inputs from control panel

    /**
     * @Resets all inputs
     */
    private void resetAll() {
        resetV();
        resetCoord();
        resetN();
        resetRD();
        resetRobot();
    }

    /**
     * @Reset corners/verticies
     */
    private void resetV() {
        v = 4;
        vText.setText("4");
        vText.getEchoChar();
    }

    /**
     * @Reset coordiniate arrays
     */
    private void resetCoord() {
        x.removeAll(x);
        y.removeAll(y);
        x.add(0);
        y.add(0);
        x.add(0);
        y.add(100);
        x.add(100);
        y.add(100);
        x.add(100);
        y.add(0);
        coordText.setText("(0,0), (0,100), (100,100), (100,0)");
        coordText.getEchoChar();
    }

    /**
```

```
     * @Reset sensor number
     */
    private void resetN() {
        n = 20;
        nText.setText("20");
        nText.getEchoChar();
    }

    /**
     * @Reset robot direction
     */
    private void resetRD() {
        rd = 0;
        rdText.setText("0");
        rdText.getEchoChar();
    }

    /**
     * @Reset robot position
     */
    private void resetRobot() {
        rx = 50;
        ry = 50;
        robotText.setText("(50, 50)");
        robotText.getEchoChar();
    }

}

/**
 * Canvas for Drawing Polygon and Robot with Sensors
 */
class CvMap extends Canvas {
    private static final long serialVersionUID = 1L;

    Polygon p;
    int maxX, maxY;
    double diagonal;
    int v, n, rd, rx, ry;
    int minXCoord, maxXCoord, minYCoord, maxYCoord;
    ArrayList<Integer> x = new ArrayList<Integer>();
    ArrayList<Integer> y = new ArrayList<Integer>();

    /**
     * Initialize Canvas
     */
    public CvMap(int v, ArrayList<Integer> x, ArrayList<Integer> y, int n, int rd, int rx, int ry)
    {
        setBackground(Color.LIGHT_GRAY);
        setSize(451, 451);

        setv(v);
        setx(x);
        sety(y);
        setn(n);
        setrd(rd);
        setrx(rx);
        setry(ry);
    }

    /**
     * Set number of vertices
     */
    public void setv(int v) {
        this.v = v;
    }

    /**
     * Set number of sensors
     */
```

```
    public void setn(int n) {
        this.n = n;
    }

    /**
     * Set direction of robot
     */
    public void setrd(int rd) {
        this.rd = rd;
    }

    /**
     * Set robot x coordinate
     */
    public void setrx(int rx) {
        this.rx = rx;
    }

    /**
     * Set robot y coordinate
     */
    public void setry(int ry) {
        this.ry = ry;
    }

    /**
     * Set polygon x coordinates
     */
    public void setx(ArrayList<Integer> x) {
        this.x.clear();
        for (Integer xcoord : x)
            this.x.add(xcoord);
    }

    /**
     * Set polygon y coordinates
     */
    public void sety(ArrayList<Integer> y) {
        this.y.clear();
        for (Integer ycoord : y)
            this.y.add(ycoord);
    }

    /**
     * Find Grid Information
     */
    void initgr() {
        Dimension d = getSize();
        maxX = d.width - 1;
        maxY = d.height - 1;
        diagonal = Math.sqrt(Math.pow(maxX, 2) + Math.pow(maxY, 2));
    }

    /**
     * Draw Polygon Edges
     */
    void drawEdges(Graphics2D g2) {
        for (int i = 1; i < v; i++)
            g2.drawLine(x.get(i - 1), y.get(i - 1), x.get(i), y.get(i));
        g2.drawLine(x.get(v - 1), y.get(v - 1), x.get(0), y.get(0));
    }

    /**
     * Draw Robot and Robot Direction
     */
    void drawRobot(Graphics2D g2) {

        g2.drawOval(rx - 10, ry - 10, 20, 20);

        double r2x = 10 * Math.cos(Math.toRadians(-rd));
```

```
        double r2y = 10 * Math.sin(Math.toRadians(-rd));
        g2.drawLine(rx, ry, (int) r2x + rx, (int) r2y + ry);
    }

    /**
     * Find and Draw Robot Sensors
     */
    void drawSensors(Graphics2D g2) {
        for (int i = 0; i < n; i++) {
            double x1 = 10 * Math.cos((((-2 * Math.PI * i)) / n) + Math.toRadians(-rd));
            double y1 = 10 * Math.sin((((-2 * Math.PI * i)) / n) + Math.toRadians(-rd));
            double x2 = diagonal * Math.cos((((-2 * Math.PI * i)) / n) + Math.toRadians(-rd));
            double y2 = diagonal * Math.sin((((-2 * Math.PI * i)) / n) + Math.toRadians(-rd));
            double startX = x1 + rx;
            double startY = y1 + ry;
            double endX = x2 + rx;
            double endY = y2 + ry;

            clipAndDraw(startX, startY, endX, endY, g2);
        }
    }

    /**
     * Determine if intersection point is in the polygon
     */
    void inPolygon(double x1, double x2, double startX, double endX, double iX, double y1, double
    y2, double startY,
            double endY, double iY, Graphics2D g2) {
        // Determine if intersection is in 2D plane
        if ((x1 <= iX && x2 >= iX) || (x1 >= iX && x2 <= iX)) {
            if ((startX <= iX && endX >= iX) || (startX >= iX && endX <= iX))
                if ((y1 <= iY && y2 >= iY) || (y1 >= iY && y2 <= iY))
                    if ((startY <= iY && endY >= iY) || (startY >= iY && endY <= iY))
                        g2.drawLine((int) startX, (int) startY, (int) iX, (int) iY);
        }
    }

    /**
     * Helper Function to Clip and Draw the Sensor lines
     */
    void clipAndDraw(double startX, double startY, double endX, double endY, Graphics2D g2) {
        for (int i = 0; i < x.size(); i++) {
            // Get an Edge
            double x1 = x.get(i).doubleValue();
            double y1 = y.get(i).doubleValue();
            double x2 = x.get((i + 1) % x.size()).doubleValue();
            double y2 = y.get((i + 1) % y.size()).doubleValue();

            // Find Slope
            double sSlope = (endY - startY) / (endX - startX);
            double eSlope = (y2 - y1) / (x2 - x1);

            double iX = 0, iY = 0;

            // Find intersection
            if (x1 != x2 && y1 != y2 && startX != endX && startY != endY) {
                iX = ((eSlope * x1) - (sSlope * startX) + startY - y1) / (eSlope - sSlope);
                iY = eSlope * (iX - x1) + y1;
            } else if (x1 == x2) {
                iX = x1;
                iY = (sSlope * (iX - startX)) + startY;
            } else if (y1 == y2) {
                iY = y1;
                iX = ((iY - startY) / sSlope) + startX;

            } else if (startX == endX) {
                iX = startX;
                iY = eSlope * (iX - x1) + y1;
            } else if (startY == endY) {
                iY = startY;
```

```
                iX = ((iY - y1) / eSlope) + x1;
            }
            inPolygon(x1, x2, startX, endX, iX, y1, y2, startY, endY, iY, g2);
        }
    }

    /**
     * Paint the Canvas
     */
    public void paint(Graphics g) {
        initgr();

        Graphics2D g2 = (Graphics2D) g;

        drawEdges(g2);
        drawRobot(g2);
        drawSensors(g2);
    }
}
```