```java
// Rachel Burke
// RobotAnimation.java: This program draws an animation of a robot grabbing a block on the floor.
// 495, 20, 0, 60

import java.awt.*;
import java.awt.event.*;
import java.awt.geom.AffineTransform;
import java.awt.geom.Arc2D;
import java.util.ArrayList;
import java.lang.Math;
import javax.swing.border.EmptyBorder;
import javax.swing.*;

public class RobotAnimation {
    // Robot Animation Input Variables
    int body, arm, elbow, hand; // Robot animation input - body location, arm, elbow, and hand
angles,

    // 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 JPanel controlPanel; // Contains the inputPanel and cvMap in the mainFrame
    private JPanel inputPanel; // Contains range map input in the control Panel
    private final TextField bodyText = new TextField(5);
    private final TextField armText = new TextField(5);
    private final TextField elbowText = new TextField(5);
    private final TextField handText = new TextField(5);
    private JButton enterButton; // Button that allows for the submission of user input
    private CvMap cvMap; // Range Map Canvas

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

    // Window Setup Functions

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

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

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

    /**
     * @Setup the instruction labels and add the instructions
     * @Initialize the Instruction JPanel Set
     * @Instruction Panel Properties Add Instruction labels
     */
    private void initializeInstructions() {
        instructionsLabel1 = new Label(
                "In each text field, input the information robot location, hand width, and angles
                of the arm. Continue to input information to move the robot across the screen to
                grab the block.");
        instructionsLabel1.setAlignment(Label.LEFT);
        instructionsLabel1.setForeground(Color.WHITE);

        instructionsLabel2 = new Label(
                "Default setting is a robot at \"0\" with arm angles \"50\" and \"60\" degrees
                with hand width \"5.\"");
        instructionsLabel2.setAlignment(Label.LEFT);
        instructionsLabel2.setForeground(Color.WHITE);

        instructionPanel = new JPanel();
        instructionPanel.setLayout(new FlowLayout());
        instructionPanel.setPreferredSize(new Dimension(1200, 40));
        instructionPanel.setBackground(Color.DARK_GRAY);
        instructionPanel.setBorder(new EmptyBorder(10, 5, 5, 5));

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

    /**
     * @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("Robot Animation");
        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 bodyLabel = new Label("Robot Position [0, 625]: ", Label.LEFT);
        Label handLabel = new Label("Hand Width [0, 20]: ", Label.LEFT);
        Label armLabel = new Label("Arm Angle [0, 90]: ", Label.LEFT);
        Label elbowLabel = new Label("Elbow Angle [0, 60]: ", Label.LEFT);
        resetAll();

        enterButton = new JButton("Enter");
        cvMap = new CvMap();
        cvMap.addPoints(body, arm, elbow, hand);
        btnEnter(enterButton, statusMsg);

        inputPanel.add(bodyLabel);
        inputPanel.add(bodyText);
        inputPanel.add(handLabel);
        inputPanel.add(handText);
        inputPanel.add(armLabel);
        inputPanel.add(armText);
        inputPanel.add(elbowLabel);
        inputPanel.add(elbowText);
        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(bodyText, statusMsg, cvMap) || isTxtEmpty(armText, statusMsg,
cvMap)
                        || isTxtEmpty(elbowText, statusMsg, cvMap) || isTxtEmpty(handText,
statusMsg, cvMap))) {
                    setBody();
                    setArm();
                    setElbow();
                    setHand();

                    boolean badValue = false;
                    if (body > 625) {
                        body = 625;
                        bodyText.setText("625");
                        bodyText.getEchoChar();
                        badValue = true;
                    }
                    if (body < 0) {
                        body = 0;
                        bodyText.setText("0");
                        bodyText.getEchoChar();
                        badValue = true;
                    }
                    if (arm > 90) {
                        arm = 90;
                        armText.setText("90");
                        armText.getEchoChar();
                        badValue = true;
                    }
                    if (arm < 0) {
                        arm = 0;
                        armText.setText("0");
                        armText.getEchoChar();
                        badValue = true;
                    }
                    if (elbow > 60) {
                        elbow = 60;
                        elbowText.setText("60");
                        elbowText.getEchoChar();
                        badValue = true;
                    }
                    if (elbow < 0) {
                        elbow = 0;
                        elbowText.setText("0");
                        elbowText.getEchoChar();
                        badValue = true;
                    }
                    if (hand > 20) {
                        hand = 20;
                        handText.setText("20");
                        handText.getEchoChar();
                        badValue = true;
                    }
                    if (hand < 0) {
                        hand = 0;
                        handText.setText("0");
                        handText.getEchoChar();
                        badValue = true;
                    }
                    if (badValue)
                        statusMsg.setText("Status Message: Bad information given. Closest valid
                        values used.");
                    else
                        statusMsg.setText("Status Message: Success!");
                }
                cvMap.addPoints(body, arm, elbow, hand);
                cvMap.repaint();
            }
        });
    }

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

    // Set input functions

    /**
     * @Set body to text field input
     */
    private void setBody() {
        body = Integer.parseInt(bodyText.getText());
    }

    /**
     * Set arm to text field input
     */
    private void setArm() {
        arm = Integer.parseInt(armText.getText());
    }

    /**
     * @Set elbow to text field input
     */
```

```java
    private void setElbow() {
        elbow = Integer.parseInt(elbowText.getText());
    }

    /**
     * @Set hand to text field input
     */
    private void setHand() {
        hand = Integer.parseInt(handText.getText());
    }

    // Reset Functions for range map inputs from control panel

    /**
     * @Resets all inputs
     */
    private void resetAll() {
        resetBody();
        resetArm();
        resetElbow();
        resetHand();
    }

    /**
     * @Reset robot body location
     */
    private void resetBody() {
        body = 0;
        bodyText.setText("0");
        bodyText.getEchoChar();
    }

    /**
     * @Reset robot arm angle
     */
    private void resetArm() {
        arm = 60;
        armText.setText("60");
        armText.getEchoChar();
    }

    /**
     * @Reset robot elbow angle
     */
    private void resetElbow() {
        elbow = 50;
        elbowText.setText("50");
        elbowText.getEchoChar();
    }

    /**
     * @Reset robot hand width
     */
    private void resetHand() {
        hand = 5;
        handText.setText("5");
        handText.getEchoChar();
    }

}

/**
 * Canvas for Drawing Robot Animation
 */
class CvMap extends Canvas {
    private static final long serialVersionUID = 1L;

    int maxX, maxY;
    int body, arm, elbow, hand;
    List bodyPosition, armPosition, elbowPosition, handPosition;
    /**
     * Initialize Canvas
     */
    public CvMap() {
        bodyPosition = new List();
        armPosition = new List();
        elbowPosition = new List();
        handPosition = new List();

        setBackground(Color.LIGHT_GRAY);
        setSize(851, 451);

        initializeGrid();
    }

    /**
     * Add points to the drawing
     */
    public void addPoints(int body, int arm, int elbow, int hand) {
        bodyPosition.add(Integer.toString(body));
        armPosition.add(Integer.toString(arm));
        elbowPosition.add(Integer.toString(elbow));
        handPosition.add(Integer.toString(hand));
    }

    /**
     * Clear the arrays
     */
    public void clearPoints() {
        bodyPosition.removeAll();
        armPosition.removeAll();
        elbowPosition.removeAll();
        handPosition.removeAll();
    }

    /**
     * Set body location
     */
    public void setBody(int body) {
        this.body = body;
    }

    /**
     * Set arm angle
     */
    public void setArm(int arm) {
        this.arm = arm;
    }

    /**
     * Set elbow angle
     */
    public void setElbow(int elbow) {
        this.elbow = elbow;
    }

    /**
     * Set hand width
     */
    public void setHand(int hand) {
        this.hand = hand;
    }

    /**
     * Find Grid Information
     */
    void initializeGrid() {
        Dimension d = getSize();
        maxX = d.width - 1;
        maxY = d.height - 1;
    }

    /**
     * Draw the Get Me Box
     */

    void drawBox(Graphics2D g) {
        g.drawRect(maxX - 50, maxY - 50, 50, 50);
        g.drawString("GET ME", maxX - 47, maxY - 20);
    }

    /**
     * Draw the Robot at its body location
     */
    void drawRobot(Graphics2D g) {
        // Base
        g.drawRect(body, maxY - 20, 175, 10);
        // Wheels
        g.drawOval(5 + body, maxY - 10, 10, 10);
        g.drawOval(160 + body, maxY - 10, 10, 10);
        // Body
        g.drawArc(30 + body, maxY - 200, 115, 100, 0, 180);
        g.drawLine(30 + body, maxY - 150, 30 + body, maxY - 20);
        g.drawLine(145 + body, maxY - 150, 145 + body, maxY - 20);
        // Pivot Angle
        g.drawOval(74 + body, maxY - 175, 24, 24);
        g.drawOval(79 + body, maxY - 171, 15, 15);
        g.fillOval(83 + body, maxY - 167, 7, 7);
        drawArm(g);
    }

    /**
     * Draw the Robot arm at a given angle
     */
    void drawArm(Graphics2D g) {
        g.rotate(Math.toRadians(90 - arm), 85 + body, maxY - 165);
        g.draw(new Arc2D.Double(75 + body, maxY - 145, 25, 25, 180, 180, Arc2D.OPEN));
        g.draw(new Arc2D.Double(75 + body, maxY - 345, 25, 25, 0, 180, Arc2D.OPEN));
        g.drawLine(75 + body, maxY - 130, 75 + body, maxY - 333);
        g.drawLine(100 + body, maxY - 130, 100 + body, maxY - 333);
        g.drawOval(80 + body, maxY - 340, 15, 15);
        g.fillOval(84 + body, maxY - 336, 7, 7);
        drawElbow(g);
    }

    /**
     * Draw the Robot elbow at a given angle
     */
    void drawElbow(Graphics2D g) {
        g.rotate(Math.toRadians(elbow), 85 + body, maxY - 335);
        g.draw(new Arc2D.Double(75 + body, maxY - 300, 25, 25, 180, 180, Arc2D.OPEN));
        g.drawLine(75 + body, maxY - 285, 75 + body, maxY - 425);
        g.drawLine(100 + body, maxY - 285, 100 + body, maxY - 425);
        g.drawLine(75 + body, maxY - 425, 100 + body, maxY - 425);
        drawHands(g);
    }

    /**
     * Draw the Robot hands at a given width
     */
    void drawHands(Graphics2D g) {
        // Hand Tips
        g.drawOval(65 + body - (int) (hand / 2), maxY - 480, 10, 10);
        g.drawOval(100 + body + (int) (hand / 2), maxY - 480, 10, 10);
        // Outer Bottom of Arm
        g.drawLine(65 + body - (int) (hand / 2), maxY - 425, 87 + body, maxY - 425);
        g.drawLine(88 + body, maxY - 425, 110 + body + (int) (hand / 2), maxY - 425);
        // Outer Outside of Arm
        g.drawLine(65 + body - (int) (hand / 2), maxY - 425, 65 + body - (int) (hand / 2), maxY -
        g.drawLine(65 + body - (int) (hand / 2), maxY - 425, 65 + body - (int) (hand / 2), maxY -
        475);
        g.drawLine(110 + body + (int) (hand / 2), maxY - 425, 110 + body + (int) (hand / 2), maxY
        - 475);
        // Outer Inside of Arm
        g.drawLine(87 + body - (int) (hand / 2), maxY - 425, 87 + body - (int) (hand / 2), maxY -
        435);
        g.drawLine(88 + body + (int) (hand / 2), maxY - 425, 88 + body + (int) (hand / 2), maxY -
        435);
        // Inner Outside of Arm
        g.drawLine(77 + body - (int) (hand / 2), maxY - 435, 87 + body - (int) (hand / 2), maxY -
        435);
        g.drawLine(88 + body + (int) (hand / 2), maxY - 435, 98 + body + (int) (hand / 2), maxY -
        435);
        // Inner Inside of Arm
        g.drawLine(77 + body - (int) (hand / 2), maxY - 435, 75 + body - (int) (hand / 2), maxY -
        475);
        g.drawLine(98 + body + (int) (hand / 2), maxY - 435, 100 + body + (int) (hand / 2), maxY -
        475);

    }

    /**
     * Paint the Canvas
     */
    @Override
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        AffineTransform oldTransform = g2.getTransform();

        drawBox(g2);

        for (int i = 0; i < bodyPosition.getItemCount(); i++) {
            setBody(Integer.valueOf(bodyPosition.getItem(i)));
            setArm(Integer.valueOf(armPosition.getItem(i)));
            setElbow(Integer.valueOf(elbowPosition.getItem(i)));
            setHand(Integer.valueOf(handPosition.getItem(i)));
            drawRobot(g2);
            g2.setTransform(oldTransform);
        }
    }
}
```