Appendix

Program Code:

```
import java.awt.Color;
import java.awt.Graphics2D;
public abstract class AbstractFunctions {
	protected final double INCREMENT = 0.1;
	private final int TWO_FIFTY_SIX = 256;
	protected final double DELTA_H = 0.001;
	protected final double ACCEPTABLE_ERROR_LIMIT = 0.01;
	protected final double DX = 0.001;
	protected final double ACCEPTABLE_ERROR = 0.0001;
	public static int numOfFunctions=0;
	protected String expression;
	protected boolean visible = true;
	protected Color color;
	protected int num =0;
	public String readExp = "";
	public boolean show = true;
	public AbstractFunctions(String expr) {
		this.expression = expr;
		color = randomColor();
	}
	public Color randomColor() {
		int R = (int)(Math.random()*TWO_FIFTY_SIX);
		int G = (int)(Math.random()*TWO_FIFTY_SIX);
		int B= (int)(Math.random()*TWO_FIFTY_SIX);
		Color color = new Color(R, G, B);
		return color;
	}
	public abstract Color getColor();
	public abstract void setColor(Color c);
	public abstract String getExpr();
	public abstract void setExpr(String s);
	public abstract double eval(double x);//consider divide by 0 situation
	public abstract String derivative(double x);
	public abstract double integral(double upper, double lower);
	public abstract String equal(Function f, double leftX, double rightX);
	public abstract String equal(double x, double leftX, double rightX);
	public abstract String toString();
	public abstract void drawFunction(Graphics2D g2d, int minX, int maxX, double f);
}
```

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

```
public class AddFunctionPanel extends JPanel implements ActionListener {
        private final String FONT = "Verdana";
        private final int FUNCTION_EQUALS_FONT_SIZE = 30;
        private final int DISPLAY_LABEL_FONT_SIZE = 30;
        private final int CALCULATOR_BUTTONS_COLUMN = 8;
        private final int CALCULATOR_BUTTONS_ROW = 4;

        private final int DISPLAY_LABEL_X = 240;
        private final int DISPLAY_LABEL_Y = 150;
        private final int DISPLAY_LABEL_WIDTH = 685;
        private final int DISPLAY_LABEL_HEIGHT = 130;

        private final int CALCULATOR_BUTTONS_TOP_LEFT_X = 150;
        private final int CALCULATOR_BUTTONS_TOP_LEFT_Y = 350;
        private final int CALCULATOR_BUTTONS_WIDTH = 80;
        private final int CALCULATOR_BUTTONS_HEIGHT = 50;

        private final int BACKGROUND_RECTANGLE_X = 115;
        private final int BACKGROUND_RECTANGLE_Y = 150;
        private final int BACKGROUND_RECTANGLE_WIDTH = 715;
        private final int BACKGROUND_RECTANGLE_HEIGHT = 130;

        private final int BACK_BUTTONS_TOP_LEFT_X = 135;
        private final int BACK_BUTTONS_TOP_LEFT_Y = 25;
        private final int BACK_BUTTONS_WIDTH = 180;
        private final int BACK_BUTTONS_HEIGHT = 50;

        private final int FUNCTION_EQUALS_X = 130;
        private final int FUNCTION_EQUALS_Y = 190;
        private final int FUNCTION_EQUALS_WIDTH = 150;
        private final int FUNCTION_EQUALS_HEIGHT = 50;

        private final double X_BOUND_FOR_CHECKING = 10000;
        private final double X_INCREMENT_FOR_CHECKING = 100;

        public boolean show = true;
        private boolean lastButtonWasEqual = false;
        private JLabel functionEquals = new JLabel("f(x) = ");
        private Image calculatorButtonImages;
        private JButton[][] calculatorButtons = new JButton[CALCULATOR_BUTTONS_COLUMN][CALCULATOR_BUTTONS_ROW];
        public JButton backButton = new JButton();
        private String processText = "";
        private String displayText = "";
        private StringStack input = new StringStack();
        private StringStack processInput = new StringStack();
        private JLabel displayLabel = new JLabel(displayText);
        private String[][] addToDisplayText = { { "(", ")", "^(", "7", "8", "9", "del", "ac" },
                        { "arcsin(", "sin(", "(x)", "4", "5", "6", "*", "/" }, { "arcos(", "cos(", "ln(", "1", "2", "3", "+", "-" },
                        { "arctan(", "tan(", "abs(", "0", ".", "e", "pi", "=" } };
        private String[][] addToProcessText = { { "(", ")", "^(", "7", "8", "9", "b", "c" },
                        { "d(", "e(", "(x)", "4", "5", "6", "*", "/" }, { "h(", "i(", "g(", "1", "2", "3", "+", "@" },
                        { "l(", "m(", "j(", "0", ".", "o", "n", "q" },
	};

	private Image backImage = new ImageIcon("zImages\\AddFunction\\back.png").getImage();

	public AddFunctionPanel(int w, int h) {
		this.setLayout(null);
		this.setSize(w, h);
		setUpButtons();
		backButton.setBounds(BACK_BUTTONS_TOP_LEFT_X, BACK_BUTTONS_TOP_LEFT_Y, BACK_BUTTONS_WIDTH,
BACK_BUTTONS_HEIGHT);
		backButton.setIcon(new ImageIcon(
				backImage.getScaledInstance(BACK_BUTTONS_WIDTH, BACK_BUTTONS_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		backButton.addActionListener(this);
		this.add(backButton);

		functionEquals.setBackground(Color.white);
		functionEquals.setForeground(Color.black);
		functionEquals.setFont(new Font(FONT, Font.PLAIN, FUNCTION_EQUALS_FONT_SIZE));
		functionEquals.setBounds(FUNCTION_EQUALS_X, FUNCTION_EQUALS_Y, FUNCTION_EQUALS_WIDTH,
FUNCTION_EQUALS_HEIGHT);
		this.add(functionEquals);

		displayLabel.setBackground(Color.white);
		displayLabel.setForeground(Color.black);
		displayLabel.setFont(new Font(FONT, Font.PLAIN, DISPLAY_LABEL_FONT_SIZE));
		displayLabel.setBounds(DISPLAY_LABEL_X, DISPLAY_LABEL_Y, DISPLAY_LABEL_WIDTH, DISPLAY_LABEL_HEIGHT);
		this.add(displayLabel);
	}

	public void setUpButtons() {
		for (int i = 0; i < CALCULATOR_BUTTONS_COLUMN; i++) {
			for (int j = 0; j < CALCULATOR_BUTTONS_ROW; j++) {
				calculatorButtons[i][j] = new JButton();
				calculatorButtons[i][j].setBounds(CALCULATOR_BUTTONS_TOP_LEFT_X + i *
CALCULATOR_BUTTONS_WIDTH,
						CALCULATOR_BUTTONS_TOP_LEFT_Y + j * CALCULATOR_BUTTONS_HEIGHT,
CALCULATOR_BUTTONS_WIDTH,
						CALCULATOR_BUTTONS_HEIGHT);
				calculatorButtons[i][j].addActionListener(this);
				calculatorButtonImages = new ImageIcon("src\\Images\\AddFunction\\" + i + "_" + j +
".png").getImage();
				calculatorButtons[i][j]
						.setIcon(new
ImageIcon(calculatorButtonImages.getScaledInstance(CALCULATOR_BUTTONS_WIDTH,
								CALCULATOR_BUTTONS_HEIGHT, java.awt.Image.SCALE_SMOOTH)));
				this.add(calculatorButtons[i][j]);
			}
		}
	}

	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		Graphics2D g2d = (Graphics2D) g;
		((Graphics2D) g2d).setStroke(new BasicStroke(3));
		g.setColor(Color.black);
		g.drawRect(BACKGROUND_RECTANGLE_X, BACKGROUND_RECTANGLE_Y, BACKGROUND_RECTANGLE_WIDTH,
				BACKGROUND_RECTANGLE_HEIGHT);
		displayLabel.setText(displayText);
		repaint();
	}

	public void actionPerformed(ActionEvent e) {
		try {
			if (lastButtonWasEqual) {
				displayText = "";
				processText = "";
				input.clear();
				processInput.clear();
				lastButtonWasEqual = false;
			}
			if (e.getSource() == calculatorButtons[CALCULATOR_BUTTONS_COLUMN - 1][CALCULATOR_BUTTONS_ROW -
1]) {
				if (!processText.isEmpty()) {
					boolean isValid = false;
					String s;
					for (double z = -X_BOUND_FOR_CHECKING; z < 0; z+=X_INCREMENT_FOR_CHECKING) {
						s = processText.replaceAll("x", "0@"+z);
						try {
							eval(s);
							isValid = true;
							break;
						} catch (Exception ex) {

						}
					}
					if (isValid) {
						Function f = new Function(processText, displayText);
						GraphingPanel.functions.add(f); backButton.doClick();
					}
					else {
						for (double z = 0; z < X_BOUND_FOR_CHECKING; z+=X_INCREMENT_FOR_CHECKING)
{
							s = processText.replaceAll("x", z+"");
							try {
								eval(s);
								isValid = true;
								break;
							} catch (Exception ex) {

							}
						}
						if (isValid) {
							Function f = new Function(processText,displayText);
							GraphingPanel.functions.add(f); backButton.doClick();
						}
```

```
                        else {
                            displayText = "Invalid Input";
                        }
                    }

                } else {
                    displayText = "Invalid Input";
                }
                lastButtonWasEqual = true;
            } else if (e.getSource() == calculatorButtons[7][0]) {
                displayText = "";
                input.clear();
                processText = "";
                processInput.clear();
            }
            // If the user clicks del button
            else if (e.getSource() == calculatorButtons[6][0]) {
                if (!displayText.isEmpty()) {
                    int lastItemLength = input.pop().length();
                    displayText = displayText.substring(0, displayText.length() - lastItemLength);
                    lastItemLength = processInput.pop().length();
                    processText = processText.substring(0, processText.length() - lastItemLength);
                }
            }

            else {
                for (int i = 0; i < CALCULATOR_BUTTONS_COLUMN; i++) {
                    for (int j = 0; j < CALCULATOR_BUTTONS_ROW; j++) {
                        if (e.getSource() == calculatorButtons[i][j]) {
                            displayText += addToDisplayText[j][i];
                            input.add(addToDisplayText[j][i]);
                            processText += addToProcessText[j][i];
                            processInput.add(addToProcessText[j][i]);
                        }
                    }
                }
            }
        } catch (

        Exception exception) {
            displayText = "Invalid Input";
        }
    }

    public double eval(String s) {
        s += "*1";
        String num = "1234567890.no";
        String operations = "+@,*/%,abcdefghijklmpq^(),";
        StringStack operator = new StringStack();
        DoubleStack term = new DoubleStack();
        for (int i = 0; i < s.length(); i++) {
            if (num.contains(s.charAt(i) + "")) {
                if (s.charAt(i) == 'n') {
```

```
        term.add(Math.PI);
    } else if (s.charAt(i) == 'o') {
        term.add(Math.E);
    } else {
        for (int j = i; j < s.length(); j++) {
            if (!num.contains(s.charAt(j) + "")) {
                double d = Double.parseDouble(s.substring(i, j));
                term.add(d);
                i = j;
                break;
            } else if (j == s.length() - 1) {
                double d = Double.parseDouble(s.substring(i));
                term.add(d);
                i = j;
                break;
            }
        }
    }
}
if (operations.contains(s.charAt(i) + "")) {
    String op = "";
    if (s.charAt(i) == '(') {
        int endIndex = i;
        int counter = 0;
        for (int z = i + 1; z < s.length(); z++) {
            if (s.charAt(z) == ')') {
                if (counter == 0) {
                    endIndex = z;
                    break;
                } else {
                    counter--;
                }

            } else if (s.charAt(z) == '(')
                counter++;
        }
        String p = eval(s.substring(i + 1, endIndex)) + "";
        term.add(Double.parseDouble(p));
        i = endIndex;
    } else if (s.charAt(i) == 'i') {
        int endIndex = i;
        int counter = 0;
        for (int z = i + 2; z < s.length(); z++) {
            if (s.charAt(z) == ')') {
                if (counter == 0) {
                    endIndex = z;
                    break;
                } else {
                    counter--;
                }

            } else if (s.charAt(z) == '(')
                counter++;
				}
				String p = Math.cos(eval(s.substring(i + 1, endIndex + 1))) + "";
				term.add(Double.parseDouble(p));
				i = endIndex;
			} else if (s.charAt(i) == 'e') {
				int endIndex = i;
				int counter = 0;
				for (int z = i + 2; z < s.length(); z++) {
					if (s.charAt(z) == ')') {
						if (counter == 0) {
							endIndex = z;
							break;
						} else {
							counter--;
						}

					} else if (s.charAt(z) == '(')
						counter++;
				}
				String p = Math.sin(eval(s.substring(i + 1, endIndex + 1))) + "";
				term.add(Double.parseDouble(p));
				i = endIndex;
			} else if (s.charAt(i) == 'm') {
				int endIndex = i;
				int counter = 0;
				for (int z = i + 2; z < s.length(); z++) {
					if (s.charAt(z) == ')') {
						if (counter == 0) {
							endIndex = z;
							break;
						} else {
							counter--;
						}

					} else if (s.charAt(z) == '(')
						counter++;
				}
				String p = Math.tan(eval(s.substring(i + 1, endIndex + 1))) + "";
				term.add(Double.parseDouble(p));
				i = endIndex;
			} else if (s.charAt(i) == 'j') {
				int endIndex = i;
				int counter = 0;
				for (int z = i + 2; z < s.length(); z++) {
					if (s.charAt(z) == ')') {
						if (counter == 0) {
							endIndex = z;
							break;
						} else {
							counter--;
						}

					} else if (s.charAt(z) == '(')
                    counter++;
            }
            String p = Math.abs(eval(s.substring(i + 1, endIndex + 1))) + "";
            term.add(Double.parseDouble(p));
            i = endIndex;

        } else if (s.charAt(i) == 'f') {
            int endIndex = i;
            int counter = 0;
            for (int z = i + 2; z < s.length(); z++) {
                if (s.charAt(z) == ')') {
                    if (counter == 0) {
                        endIndex = z;
                        break;
                    } else {
                        counter--;
                    }

                } else if (s.charAt(z) == '(')
                    counter++;
            }
            String p = Math.log10(eval(s.substring(i + 1, endIndex + 1))) + "";
            term.add(Double.parseDouble(p));
            i = endIndex;
        } else if (s.charAt(i) == 'd') {
            int endIndex = i;
            int counter = 0;
            for (int z = i + 2; z < s.length(); z++) {
                if (s.charAt(z) == ')') {
                    if (counter == 0) {
                        endIndex = z;
                        break;
                    } else {
                        counter--;
                    }

                } else if (s.charAt(z) == '(')
                    counter++;
            }
            String p = Math.asin(eval(s.substring(i + 1, endIndex + 1))) + "";
            term.add(Double.parseDouble(p));
            i = endIndex;
        } else if (s.charAt(i) == 'h') {
            int endIndex = i;
            int counter = 0;
            for (int z = i + 2; z < s.length(); z++) {
                if (s.charAt(z) == ')') {
                    if (counter == 0) {
                        endIndex = z;
                        break;
                    } else {
                        counter--;
                    }
        } else if (s.charAt(z) == '(')
            counter++;
    }
    String p = Math.acos(eval(s.substring(i + 1, endIndex + 1))) + "";
    term.add(Double.parseDouble(p));
    i = endIndex;
} else if (s.charAt(i) == 'l') {
    int endIndex = i;
    int counter = 0;
    for (int z = i + 2; z < s.length(); z++) {
        if (s.charAt(z) == ')') {
            if (counter == 0) {
                endIndex = z;
                break;
            } else {
                counter--;
            }

        } else if (s.charAt(z) == '(')
            counter++;
    }
    String p = Math.atan(eval(s.substring(i + 1, endIndex + 1))) + "";
    term.add(Double.parseDouble(p));
    i = endIndex;
} else if (s.charAt(i) == 'a') {
    int endIndex = i;
    int counter = 0;
    for (int z = i + 2; z < s.length(); z++) {
        if (s.charAt(z) == ')') {
            if (counter == 0) {
                endIndex = z;
                break;
            } else {
                counter--;
            }

        } else if (s.charAt(z) == '(')
            counter++;
    }
    String p = Math.exp(eval(s.substring(i + 1, endIndex + 1))) + "";
    term.add(Double.parseDouble(p));
    i = endIndex;
} else if (s.charAt(i) == 'g') {
    int endIndex = i;
    int counter = 0;
    for (int z = i + 2; z < s.length(); z++) {
        if (s.charAt(z) == ')') {
            if (counter == 0) {
                endIndex = z;
                break;
            } else {
                counter--;
```

```
                            }

                        } else if (s.charAt(z) == '(')
                            counter++;
                    }
                    String p = Math.log(eval(s.substring(i + 1, endIndex + 1))) + "";
                    term.add(Double.parseDouble(p));
                    i = endIndex;
                } else {
                    op = s.charAt(i) + "";
                    if (operator.isEmpty()) {
                        operator.add(op);
                    } else {
                        if (operations.indexOf(",", operations.indexOf(operator.peek())) >=
operations.indexOf(",",
                                operations.indexOf(op))) {
                            if (operator.peek().equals("(")) {

                            } else if (operator.peek().equals("*")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(term1 * term2);
                                operator.pop();
                                i--;
                            } else if (operator.peek().equals("/")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(term2 / term1);
                                operator.pop();
                                i--;
                            } else if (operator.peek().equals("%")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(term2 % term1);
                                operator.pop();
                                i--;
                            } else if (operator.peek().equals("+")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(term2 + term1);
                                operator.pop();
                                i--;
                            } else if (operator.peek().equals("@")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(term2 - term1);
                                operator.pop();
                                i--;
                            } else if (operator.peek().equals("^")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(Math.pow(term2, term1));
                                operator.pop();
                                i--;
                            }

                        } else {
                            operator.add(op);
                        }
                    }
                }
            }

        }
        term.pop();
        operator.pop();
        if (operator.isEmpty()) {
            if (term.size() == 1)
                return term.pop();
            else {
                return (Double) null;
            }
        }

        else {
            if (operator.peek().equals("+")) {
                return term.pop() + term.pop();
            } else {
                double term1 = term.pop();
                double term2 = term.pop();
                return term2 - term1;
            }
        }

    }
}
```

```
public class DoubleStack {
    public int maxCapacity=100;
    public double arr[]= new double[maxCapacity];
    public int top = -1;
    public boolean isFull() {
        return top ==maxCapacity-1;
    }
    public boolean isEmpty() {
        return top ==-1;
    }
    public double peek() {
```

```
		if (!isEmpty()) {
			return arr[top];
		}
		return (Double) null;

	}
	public double pop() {
		if (!isEmpty()) {
			double x = arr[top];
			top--;
			return x;
		}
		return (Double) null;

	}
	public void add(double x) {
		if (!isFull()) {
			top++;
			arr[top]= x;
		}

	}
	public int size() {
		return top+1;
	}
}
```

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
public class EditFunctionPanel extends JPanel implements ActionListener {
	private final String FONT = "Veranda";
	private final int LABEL_FONT_SIZE = 20;
	private final int TEXT_FIELD_FONT_SIZE = 15;
	private final int ANSWER_TEXT_FIELD_FONT_SIZE = 15;

	private final String LEFT_BUTTON_FILE = "src\\Images\\EditFunctionPanel\\previous.png";
	private final int LEFT_BUTTON_X = 50;
	private final int LEFT_BUTTON_Y = 50;
	private final int LEFT_BUTTON_WIDTH = 150;
	private final int LEFT_BUTTON_HEIGHT = 50;

	private final String RIGHT_BUTTON_FILE = "src\\Images\\EditFunctionPanel\\next.png";
	private final int RIGHT_BUTTON_X = 800;
private final int RIGHT_BUTTON_Y = 50;
private final int RIGHT_BUTTON_WIDTH = 150;
private final int RIGHT_BUTTON_HEIGHT = 50;

private final String BACK_BUTTON_FILE = "src\\Images\\EditFunctionPanel\\back.png";
private final int BACK_BUTTON_X = 50;
private final int BACK_BUTTON_Y = 800;
private final int BACK_BUTTON_WIDTH = 170;
private final int BACK_BUTTON_HEIGHT = 50;

private final String REMOVE_BUTTON_FILE = "src\\Images\\EditFunctionPanel\\remove.png";
private final int REMOVE_BUTTON_X = 50;
private final int REMOVE_BUTTON_Y = 700;
private final int REMOVE_BUTTON_WIDTH = 170;
private final int REMOVE_BUTTON_HEIGHT = 50;

private final String SHOW_BUTTON_FILE = "src\\Images\\EditFunctionPanel\\show.png";
private final String HIDE_BUTTON_FILE = "src\\Images\\EditFunctionPanel\\hide.png";
private final int SHOW_BUTTON_X = 300;
private final int SHOW_BUTTON_Y = 700;
private final int SHOW_BUTTON_WIDTH = 150;
private final int SHOW_BUTTON_HEIGHT = 50;

private final String COLOR_BUTTON_FILE = "src\\Images\\EditFunctionPanel\\color.png";
private final int COLOR_BUTTON_X = 500;
private final int COLOR_BUTTON_Y = 700;
private final int COLOR_BUTTON_WIDTH = 120;
private final int COLOR_BUTTON_HEIGHT = 50;

private final int LABEL_DY = 100;
private final int LABEL_X = 50;
private final int LABEL_Y = 100;
private final int LABEL_WIDTH = 1000;
private final int LABEL_HEIGHT = 75;

private final int TEXT_FIELD_0_X = 147;
private final int TEXT_FIELD_0_Y = 225;
private final int TEXT_FIELD_0_WIDTH = 120;
private final int TEXT_FIELD_0_HEIGHT = 25;

private final int TEXT_FIELD_1_X = 147;
private final int TEXT_FIELD_1_Y = 327;
private final int TEXT_FIELD_1_WIDTH = 120;
private final int TEXT_FIELD_1_HEIGHT = 25;

private final int TEXT_FIELD_2_X = 170;
private final int TEXT_FIELD_2_Y = 427;
private final int TEXT_FIELD_2_WIDTH = 120;
private final int TEXT_FIELD_2_HEIGHT = 25;

private final int TEXT_FIELD_3_X = 350;
private final int TEXT_FIELD_3_Y = 427;
private final int TEXT_FIELD_3_WIDTH = 120;
private final int TEXT_FIELD_3_HEIGHT = 25;

private final int TEXT_FIELD_4_X = 293;
private final int TEXT_FIELD_4_Y = 527;
private final int TEXT_FIELD_4_WIDTH = 60;
private final int TEXT_FIELD_4_HEIGHT = 25;

private final int TEXT_FIELD_5_X = 525;
private final int TEXT_FIELD_5_Y = 527;
private final int TEXT_FIELD_5_WIDTH = 60;
private final int TEXT_FIELD_5_HEIGHT = 25;

private final int TEXT_FIELD_6_X = 650;
private final int TEXT_FIELD_6_Y = 527;
private final int TEXT_FIELD_6_WIDTH = 60;
private final int TEXT_FIELD_6_HEIGHT = 25;

private final int ANSWER_TEXT_FIELD_1_X = 300;
private final int ANSWER_TEXT_FIELD_1_Y = 225;
private final int ANSWER_TEXT_FIELD_1_WIDTH = 120;
private final int ANSWER_TEXT_FIELD_1_HEIGHT = 25;

private final int ANSWER_TEXT_FIELD_2_X = 300;
private final int ANSWER_TEXT_FIELD_2_Y = 327;
private final int ANSWER_TEXT_FIELD_2_WIDTH = 120;
private final int ANSWER_TEXT_FIELD_2_HEIGHT = 25;

private final int ANSWER_TEXT_FIELD_3_X = 507;
private final int ANSWER_TEXT_FIELD_3_Y = 427;
private final int ANSWER_TEXT_FIELD_3_WIDTH = 120;
private final int ANSWER_TEXT_FIELD_3_HEIGHT = 25;

private final int ANSWER_TEXT_FIELD_4_X = 800;
private final int ANSWER_TEXT_FIELD_4_Y = 527;
private final int ANSWER_TEXT_FIELD_4_WIDTH = 120;
private final int ANSWER_TEXT_FIELD_4_HEIGHT = 25;
private final String space = "                    ";

private boolean perform = false;
private boolean valid1 = false;
private boolean valid2 = false;
private boolean valid3 = false;
private int currFunctionNumber = 0;
private JLabel labels[] = new JLabel[5];
private JTextField[] textField = new JTextField[7];
private Image image = new ImageIcon(LEFT_BUTTON_FILE).getImage();
private boolean showLeftButton = false;
private boolean showRightButton = true;
private JButton leftButton = new JButton();
private JButton rightButton = new JButton();
public JButton backButton = new JButton();
private JButton showButton = new JButton();
private JButton removeButton = new JButton();
```

```
	private JButton colorButton = new JButton();
	private JTextField answerTextField1 = new JTextField();
	private JTextField answerTextField2 = new JTextField();
	private JTextField answerTextField3 = new JTextField();
	private JTextField answerTextField4 = new JTextField();

	public EditFunctionPanel(int width, int height) {
		this.setLayout(null);
		this.setSize(width, height);
		leftButton.setBounds(LEFT_BUTTON_X, LEFT_BUTTON_Y, LEFT_BUTTON_WIDTH, LEFT_BUTTON_HEIGHT);
		leftButton.addActionListener(this);
		leftButton.setVisible(showLeftButton);
		leftButton.setIcon(new ImageIcon(
				image.getScaledInstance(LEFT_BUTTON_WIDTH, LEFT_BUTTON_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		this.add(leftButton);

		rightButton.setBounds(RIGHT_BUTTON_X, RIGHT_BUTTON_Y, RIGHT_BUTTON_WIDTH, RIGHT_BUTTON_HEIGHT);
		rightButton.addActionListener(this);
		rightButton.setVisible(showRightButton);
		image = new ImageIcon(RIGHT_BUTTON_FILE).getImage();
		rightButton.setIcon(new ImageIcon(
				image.getScaledInstance(RIGHT_BUTTON_WIDTH, RIGHT_BUTTON_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		this.add(rightButton);

		backButton.setBounds(BACK_BUTTON_X, BACK_BUTTON_Y, BACK_BUTTON_WIDTH, BACK_BUTTON_HEIGHT);
		backButton.addActionListener(this);
		image = new ImageIcon(BACK_BUTTON_FILE).getImage();
		backButton.setIcon(new ImageIcon(
				image.getScaledInstance(BACK_BUTTON_WIDTH, BACK_BUTTON_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		this.add(backButton);

		removeButton.setBounds(REMOVE_BUTTON_X, REMOVE_BUTTON_Y, REMOVE_BUTTON_WIDTH,
REMOVE_BUTTON_HEIGHT);
		removeButton.addActionListener(this);
		image = new ImageIcon(REMOVE_BUTTON_FILE).getImage();
		removeButton.setIcon(new ImageIcon(
				image.getScaledInstance(REMOVE_BUTTON_WIDTH, REMOVE_BUTTON_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		this.add(removeButton);

		showButton.setBounds(SHOW_BUTTON_X, SHOW_BUTTON_Y, SHOW_BUTTON_WIDTH, SHOW_BUTTON_HEIGHT);
		showButton.addActionListener(this);
		image = new ImageIcon(HIDE_BUTTON_FILE).getImage();
		showButton.setIcon(new ImageIcon(
				image.getScaledInstance(SHOW_BUTTON_WIDTH, SHOW_BUTTON_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		this.add(showButton);



		colorButton.setBounds(COLOR_BUTTON_X, COLOR_BUTTON_Y, COLOR_BUTTON_WIDTH, COLOR_BUTTON_HEIGHT);
		colorButton.addActionListener(this);
		image = new ImageIcon(COLOR_BUTTON_FILE).getImage();
		colorButton.setIcon(new ImageIcon(
				image.getScaledInstance(COLOR_BUTTON_WIDTH, COLOR_BUTTON_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		this.add(colorButton);
		setUpLabels();
	}
	public void setUpLabels() {
		labels[0] = new JLabel();
		labels[1] = new JLabel("f(x) at x =  " + space + " is:");
		labels[2] = new JLabel("f'(x) at x =  " + space + " is:");
		labels[3] = new JLabel("F(x) from x = " + space + " to x =" + space + "  is:");
		labels[4] = new JLabel("f(x) equals function number          in the interval x =             to x =           at x = ");
		for (int i = 0; i < labels.length; i++) {
			labels[i].setBounds(LABEL_X, LABEL_Y + LABEL_DY * i, LABEL_WIDTH, LABEL_HEIGHT);
			labels[i].setBackground(Color.white);
			labels[i].setForeground(Color.black);
			labels[i].setFont(new Font(FONT, Font.PLAIN, LABEL_FONT_SIZE));
			this.add(labels[i]);
			if (i < textField.length) {
				textField[i] = new JTextField();
				textField[i].setFont(new Font(FONT, Font.PLAIN, TEXT_FIELD_FONT_SIZE));
				textField[i].setForeground(Color.black);
				textField[i].addActionListener(this);
				this.add(textField[i]);
			}

		}
		textField[5] = new JTextField();
		textField[5].setFont(new Font(FONT, Font.PLAIN, TEXT_FIELD_FONT_SIZE));
		textField[5].setForeground(Color.black);
		textField[5].addActionListener(this);
		this.add(textField[5]);
		textField[6] = new JTextField();
		textField[6].setFont(new Font(FONT, Font.PLAIN, TEXT_FIELD_FONT_SIZE));
		textField[6].setForeground(Color.black);
		textField[6].addActionListener(this);
		this.add(textField[6]);
		textField[0].setBounds(TEXT_FIELD_0_X, TEXT_FIELD_0_Y, TEXT_FIELD_0_WIDTH, TEXT_FIELD_0_HEIGHT);
		textField[1].setBounds(TEXT_FIELD_1_X, TEXT_FIELD_1_Y, TEXT_FIELD_1_WIDTH, TEXT_FIELD_1_HEIGHT);
		textField[2].setBounds(TEXT_FIELD_2_X, TEXT_FIELD_2_Y, TEXT_FIELD_2_WIDTH, TEXT_FIELD_2_HEIGHT);
		textField[3].setBounds(TEXT_FIELD_3_X, TEXT_FIELD_3_Y, TEXT_FIELD_3_WIDTH, TEXT_FIELD_3_HEIGHT);
		textField[4].setBounds(TEXT_FIELD_4_X, TEXT_FIELD_4_Y, TEXT_FIELD_4_WIDTH, TEXT_FIELD_4_HEIGHT);
		textField[5].setBounds(TEXT_FIELD_5_X, TEXT_FIELD_5_Y, TEXT_FIELD_5_WIDTH, TEXT_FIELD_5_HEIGHT);
		textField[6].setBounds(TEXT_FIELD_6_X, TEXT_FIELD_6_Y, TEXT_FIELD_6_WIDTH, TEXT_FIELD_6_HEIGHT);


		answerTextField1.setBounds(ANSWER_TEXT_FIELD_1_X, ANSWER_TEXT_FIELD_1_Y, ANSWER_TEXT_FIELD_1_WIDTH,
				ANSWER_TEXT_FIELD_1_HEIGHT);
		answerTextField2.setBounds(ANSWER_TEXT_FIELD_2_X, ANSWER_TEXT_FIELD_2_Y, ANSWER_TEXT_FIELD_2_WIDTH,
				ANSWER_TEXT_FIELD_2_HEIGHT);
		answerTextField3.setBounds(ANSWER_TEXT_FIELD_3_X, ANSWER_TEXT_FIELD_3_Y, ANSWER_TEXT_FIELD_3_WIDTH,
				ANSWER_TEXT_FIELD_3_HEIGHT);
		answerTextField4.setBounds(ANSWER_TEXT_FIELD_4_X, ANSWER_TEXT_FIELD_4_Y, ANSWER_TEXT_FIELD_4_WIDTH,
				ANSWER_TEXT_FIELD_4_HEIGHT);
		answerTextField1.setFont(new Font(FONT, Font.PLAIN, ANSWER_TEXT_FIELD_FONT_SIZE));
		answerTextField2.setFont(new Font(FONT, Font.PLAIN, ANSWER_TEXT_FIELD_FONT_SIZE));
		answerTextField3.setFont(new Font(FONT, Font.PLAIN, ANSWER_TEXT_FIELD_FONT_SIZE));
		answerTextField4.setFont(new Font(FONT, Font.PLAIN, ANSWER_TEXT_FIELD_FONT_SIZE));
		answerTextField1.setForeground(Color.black);
		answerTextField2.setForeground(Color.black);
		answerTextField3.setForeground(Color.black);
		answerTextField4.setForeground(Color.black);
		this.add(answerTextField1);
		this.add(answerTextField2);
		this.add(answerTextField3);
		this.add(answerTextField4);
	}

	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		if (Function.numOfFunctions == 1) {
			showLeftButton = false;
			showRightButton = false;
		} else {
			if (currFunctionNumber == 0) {
				showLeftButton = false;
				showRightButton = true;
			} else if (currFunctionNumber == Function.numOfFunctions - 1) {
				showLeftButton = true;
				showRightButton = false;
			} else {
				showLeftButton = true;
				showRightButton = true;
			}
		}
		leftButton.setVisible(showLeftButton);
		rightButton.setVisible(showRightButton);
		if (GraphingPanel.functions.get(currFunctionNumber).show) {
			image = new ImageIcon(HIDE_BUTTON_FILE).getImage();
			showButton.setIcon(new ImageIcon(
					image.getScaledInstance(SHOW_BUTTON_WIDTH, SHOW_BUTTON_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		} else {
			image = new ImageIcon(SHOW_BUTTON_FILE).getImage();
			showButton.setIcon(new ImageIcon(
					image.getScaledInstance(SHOW_BUTTON_WIDTH, SHOW_BUTTON_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		}
		labels[0].setText(GraphingPanel.functions.get(currFunctionNumber).toString());
		if (valid1 && valid2&&valid3&& perform) {
			try {
				int n = Integer.parseInt(textField[4].getText());
				double x1= Double.parseDouble(textField[5].getText());
				double x2= Double.parseDouble(textField[6].getText());
```

```
                                        String s =
GraphingPanel.functions.get(currFunctionNumber).equal(GraphingPanel.functions.get(n-1),x1,x2);
                                        answerTextField4.setText(s);
                              }
                              catch(Exception ex) {
                                        answerTextField4.setText("Not Continuous");
                              }
                              perform = false;
                    }
                    repaint();
          }

          public double round(double d) {
                    return Math.round(d * Math.pow(10, 5)) / Math.pow(10, 5);
          }

          public void actionPerformed(ActionEvent e) {
                    if (e.getSource() == leftButton) {
                              currFunctionNumber--;
                    }
                    if (e.getSource() == rightButton) {
                              currFunctionNumber++;
                    }
                    if (e.getSource() == textField[0]) {
                              boolean isDouble = false;
                              String s = textField[0].getText();
                              try {
                                        double d = Double.parseDouble(s);
                                        isDouble = true;
                                        d = GraphingPanel.functions.get(currFunctionNumber).eval(d);
                                        if (Double.isNaN(d)) {
                                                  answerTextField1.setText("undefined");
                                        } else
                                                  answerTextField1.setText(round(d) + "");
                              } catch (Exception ex) {
                                        if (!isDouble)
                                                  JOptionPane.showMessageDialog(null, "The x value is not a real number.");
                              }
                    }
                    if (e.getSource() == textField[1]) {
                              boolean isDouble = false;
                              String s = textField[1].getText();

                              try {
                                        double d = Double.parseDouble(s);
                                        isDouble = true;
                                        s = GraphingPanel.functions.get(currFunctionNumber).derivative(d);
                                        if (s.equals("NaN")) {
                                                  answerTextField2.setText("DNE");
                                        } else
                                                  answerTextField2.setText(s);
                              } catch (Exception ex) {
                                        if (!isDouble)
```

```
					JOptionPane.showMessageDialog(null, "The x value is not a real number.");
			}

		}
		if (e.getSource()==textField[2]) {
			if (!textField[3].getText().isEmpty()) {
				if (textField[2].getText().length() == 0) {
					JOptionPane.showMessageDialog(null, "The first x bound is empty.");
				} else {
					boolean error = false;
					double x1=0;
					double x2=0;
					try {
						x1 = Double.parseDouble(textField[2].getText());


					} catch (Exception ex) {
						error = true;
						JOptionPane.showMessageDialog(null, "The first x bound is not a real number.");
					}
					if (!error) {
						if (textField[2].getText().length() == 0) {
							JOptionPane.showMessageDialog(null, "The second x bound is empty.");
						} else {
							try {
								x2 = Double.parseDouble(textField[3].getText());
								try{
									Double d =
GraphingPanel.functions.get(currFunctionNumber).integral(x1,x2);
									answerTextField3.setText(d+"");
								}
								catch(Exception ex) {
									JOptionPane.showMessageDialog(null, "The function is
undefined within the x bounds from x = "+x1+" to x = "+x2+".");
								}
							} catch (Exception ex) {
								JOptionPane.showMessageDialog(null, "The second x bound is not a
real number.");
							}
						}
					}
				}
			}
		}
		if (e.getSource() == textField[3]) {
			if (textField[2].getText().length() == 0) {
				JOptionPane.showMessageDialog(null, "The first x bound is empty.");
			} else {
				boolean error = false;
				double x1=0;
				double x2=0;
				try {
					x1 = Double.parseDouble(textField[2].getText());
```

```
                                        } catch (Exception ex) {
                                                  error = true;
                                                  JOptionPane.showMessageDialog(null, "The first x bound is not a real number.");
                                        }
                                        if (!error) {
                                                  if (textField[2].getText().length() == 0) {
                                                            JOptionPane.showMessageDialog(null, "The second x bound is empty.");
                                                  } else {
                                                            try {
                                                                      x2 = Double.parseDouble(textField[3].getText());
                                                                      try{
                                                                                Double d = 
GraphingPanel.functions.get(currFunctionNumber).integral(x1,x2);
                                                                                answerTextField3.setText(d+"");
                                                                      }
                                                                      catch(Exception ex) {
                                                                                JOptionPane.showMessageDialog(null, "The function is undefined 
within the x bounds from x = "+x1+" to x = "+x2+".");
                                                                      }
                                                            } catch (Exception ex) {
                                                                      JOptionPane.showMessageDialog(null, "The second x bound is not a real 
number.");
                                                            }
                                                  }
                                        }
                              }
                    }
                    if (e.getSource() == removeButton) {
                              GraphingPanel.functions.remove(currFunctionNumber);

                              for (int i =currFunctionNumber; i<Function.numOfFunctions-1; i++ ) {
                                        GraphingPanel.functions.get(i).num--;
                              }

                              Function.numOfFunctions--;
                              backButton.doClick();
                    }
                    if (e.getSource() == showButton) {
                              GraphingPanel.functions
                                                  .get(currFunctionNumber).show = !GraphingPanel.functions.get(currFunctionNumber).show;
                    }
                    if (e.getSource() == colorButton) {
                              
GraphingPanel.functions.get(currFunctionNumber).setColor(GraphingPanel.functions.get(currFunctionNumber).randomColor());
                    }
                    if (e.getSource() == textField[4]) {
                              JOptionPane.showMessageDialog(null, "Note: This feature only works if the two functions are continuous.");
                              try {
                                        int n = Integer.parseInt(textField[4].getText());
                                        if (n<1) {
                                                  valid1 = false;
```

```
					JOptionPane.showMessageDialog(null, "The function number of the other function cannot be 
less than 1.");
				}
				else if (n>Function.numOfFunctions) {
					valid1 = false;
					JOptionPane.showMessageDialog(null, "The function number of the other function cannot be 
greater than the current total number of functions of "+Function.numOfFunctions+".");
				}
				else {
					valid1 = true;
					perform = true;
				}
			}
			catch(Exception ex) {
				valid1 = false;
				JOptionPane.showMessageDialog(null, "The function number of the other function is not an 
integer.");
			}

		}
		if (e.getSource()==textField[5]) {
			String s = textField[5].getText();
			try {
				double d = Double.parseDouble(s);
				valid2 = true;
				perform = true;
			}
			catch (Exception ex) {
				valid2 = false;
				JOptionPane.showMessageDialog(null, "The x value is not a real number.");
			}
		}
		if (e.getSource()==textField[6]) {
			String s = textField[6].getText();
			try {
				double d = Double.parseDouble(s);
				valid3 = true;
				perform = true;
			}
			catch (Exception ex) {
				valid3 = false;
				JOptionPane.showMessageDialog(null, "The x value is not a real number.");
			}
		}
	}
}
```

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
public class Frame extends JFrame implements ActionListener {
	private final static String FRAME_NAME = "IB Math Calculator";
	private final int FRAME_WIDTH = 1000;
	private final int FRAME_HEIGHT = 1000;
	private final int SCIENTIFIC_PANEL_HEIGHT = 500;
	private final int STATS_PANEL_HEIGHT = 840;
	private final int ADD_FUNCTIONS_PANEL_HEIGHT = 700;
	private final int MENU_ITEM_SIZE = 5;

	private Container c = getContentPane();
	private JMenuBar menuBar;
	private JMenu menu;
	private JMenuItem menuItems[];
	private ScientificPanel scientificPanel = new ScientificPanel(FRAME_WIDTH, SCIENTIFIC_PANEL_HEIGHT);
	private StatisticsPanel statisticsPanel = new StatisticsPanel(FRAME_WIDTH, STATS_PANEL_HEIGHT);
	private TrigPanel trigPanel = new TrigPanel(FRAME_WIDTH, FRAME_HEIGHT);
	private GraphingPanel graphingPanel = new GraphingPanel(FRAME_WIDTH, FRAME_HEIGHT);
	private AddFunctionPanel addFunctionPanel = new AddFunctionPanel(FRAME_WIDTH, ADD_FUNCTIONS_PANEL_HEIGHT);
	private EditFunctionPanel editFunctionPanel = new EditFunctionPanel(FRAME_WIDTH, FRAME_HEIGHT);

	public Frame() {
		super(FRAME_NAME);
		c.add(scientificPanel);
		graphingPanel.addFunction.addActionListener(this);
		graphingPanel.editFunctions.addActionListener(this);
		addFunctionPanel.backButton.addActionListener(this);
		editFunctionPanel.backButton.addActionListener(this);
		setUpFrame();
		this.setSize(FRAME_WIDTH,SCIENTIFIC_PANEL_HEIGHT);
		if (!scientificPanel.notifcationShown) {

			JOptionPane.showMessageDialog(null,
					"Please enter negative numbers with a leading negative as 0 minus their magnitude. Ex: -6-2
can be expressed as 0-6-2");
			scientificPanel.notifcationShown = true;
		}
	}

	public void setUpFrame() {
		this.setJMenuBar(null);
		menuBar = new JMenuBar();
		menu = new JMenu("Menu");
		menuItems = new JMenuItem[MENU_ITEM_SIZE];
		menuItems[0] = new JMenuItem("Scientific Calculator");
		menuItems[1] = new JMenuItem("Statistics");
		menuItems[2] = new JMenuItem("Graphing");
		menuItems[3] = new JMenuItem("Trigonometry");
```

```
        menuItems[4] = new JMenuItem("Exit");
        for (int i = 0; i < MENU_ITEM_SIZE; i++) {
            menuItems[i].addActionListener(this);
            menu.add(menuItems[i]);
        }
        menuBar.add(menu);
        this.setJMenuBar(menuBar);
        this.setSize(FRAME_WIDTH, FRAME_HEIGHT);
        this.setResizable(false);
        this.setVisible(true);
        this.setLayout(null);
        this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == menuItems[0]) {
            c.removeAll();
            c.add(scientificPanel);
            setUpFrame();
            this.setSize(FRAME_WIDTH, SCIENTIFIC_PANEL_HEIGHT);
        }
        if (e.getSource() == menuItems[1]) {
            c.removeAll();
            c.add(statisticsPanel);
            setUpFrame();
            this.setSize(FRAME_WIDTH, STATS_PANEL_HEIGHT);
        }
        if (e.getSource() == menuItems[2]) {
            c.removeAll();
            c.add(graphingPanel);
            setUpFrame();
            graphingPanel.setFocusable(true);
            graphingPanel.requestFocusInWindow();
        }
        if (e.getSource() == menuItems[3]) {
            c.removeAll();
            c.add(trigPanel);
            setUpFrame();
        }
        if (e.getSource() == menuItems[4]) {
            System.exit(1);
        }
        if (e.getSource() == graphingPanel.addFunction) {
            c.removeAll();
            addFunctionPanel = new AddFunctionPanel(FRAME_WIDTH, ADD_FUNCTIONS_PANEL_HEIGHT);
            addFunctionPanel.backButton.addActionListener(this);
            c.add(addFunctionPanel);
            setUpFrame();
            this.setSize(FRAME_WIDTH, ADD_FUNCTIONS_PANEL_HEIGHT);
        }
        if (e.getSource() == addFunctionPanel.backButton) {
            c.removeAll();
```

```
			c.add(graphingPanel);
			setUpFrame();
			graphingPanel.setFocusable(true);
			graphingPanel.requestFocusInWindow();
		}
		if (e.getSource() == graphingPanel.editFunctions) {
			if (Function.numOfFunctions != 0) {
				c.removeAll();
				editFunctionPanel = new EditFunctionPanel(FRAME_WIDTH, FRAME_HEIGHT);
				editFunctionPanel.backButton.addActionListener(this);
				c.add(editFunctionPanel);
				setUpFrame();
			}
			else {
				JOptionPane.showMessageDialog(null, "There are no functions to edit.");
			}
		}
		if (e.getSource() == editFunctionPanel.backButton) {
			c.removeAll();
			c.add(graphingPanel);
			setUpFrame();
			graphingPanel.setFocusable(true);
			graphingPanel.requestFocusInWindow();
		}
	}
}
```

```
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;
public class Function extends AbstractFunctions {
	public Function(String s, String d) {
		super(s);
		readExp = d;
		numOfFunctions++;
		num = numOfFunctions;
	}

	public Color getColor() {
		return color;
	}
```

```
public void setColor(Color c) {
        color = c;
}

public String getExpr() {
        return expression;
}

public void setExpr(String s) {
        expression = s;
}

public double eval(double value) {
        String input = value + "";
        String s = expression;
        if (value < 0) {
                input = "0@" + input.substring(1);
        }
        s = s.replaceAll("x", input);
        return eval(s);
}
public double round(double d) {
        return Math.round(d * Math.pow(10, 5)) / Math.pow(10, 5);
}
public String derivative(double x) {
        try {
                double leftLimit;
                leftLimit = (eval(x + DELTA_H) - eval(x)) / DELTA_H;
                double rightLimit;
                rightLimit = (eval(x - DELTA_H) - eval(x)) / (-1 * DELTA_H);
                if (Math.abs(leftLimit - rightLimit) > ACCEPTABLE_ERROR_LIMIT) {
                        return "DNE";
                } else
                        return round(leftLimit) + "";
        } catch (Exception e) {
                return "DNE";
        }
}

public double integral(double upper, double lower) {
        double integral = 0;
        double max = Math.max(upper, lower);
        double min = Math.min(upper, lower);
        for (double x = min; x < max; x += DX) {
                integral += (DX * (eval(x) + eval(x + DX)) / 2);
        }
        if (upper < lower)
                return round(integral);
        else
                return round(-integral);
}

public String equal(Function f, double leftX, double rightX) {
		Function g = new Function("(" + expression + ")@(" + f.expression + ")", "");
		numOfFunctions--;
		double maxX = Math.max(leftX, rightX);
		double minX = Math.min(rightX, leftX);
		double midX = (maxX + minX) / 2;
		double counter = 0;
		while (true) {
			if (Math.abs(g.eval(midX)) < ACCEPTABLE_ERROR) {
				return round(midX) + "";
			} else if (g.eval(midX) < 0) {
				minX = midX;
				midX = (minX + maxX) / 2;
			} else if (g.eval(midX) > 0) {
				maxX = midX;
				midX = (minX + maxX) / 2;
			}
			counter++;
			if (counter > Math.log(Math.abs(rightX-leftX)*(1/(Math.pow(10, -5))))/Math.log(2)) {
				break;
			}
		}
		maxX = Math.max(leftX, rightX);
		minX = Math.min(rightX, leftX);
		midX = (maxX + minX) / 2;
		counter = 0;
		while (true) {
			if (Math.abs(g.eval(midX)) < ACCEPTABLE_ERROR) {
				return round(midX) + "";
			} else if (g.eval(midX) > 0) {
				minX = midX;
				midX = (minX + maxX) / 2;
			} else if (g.eval(midX) < 0) {
				maxX = midX;
				midX = (minX + maxX) / 2;
			}
			counter++;
			if (counter > Math.log(Math.abs(rightX-leftX)*(1/(Math.pow(10, -5))))/Math.log(2)) {
				break;
			}
		}
		return "No Intersection";
	}

	public String equal(double x, double leftX, double rightX) {
		Function f = new Function(expression + "@" + x, "");
		numOfFunctions--;
		return equal(f, leftX, rightX);
	}

	public String toString() {
		if (numOfFunctions == 1)
			return "The function f(x) = " + readExp + " is function number " + num + " out of " + numOfFunctions
					+ " function.";
```

```
        else
            return "The function f(x) = " + readExp + " is function number " + num + " out of " + numOfFunctions
                    + " functions.";
    }

    public void drawFunction(Graphics2D g2d, int minX, int maxX, double f) {
        g2d.setColor(color);
        ((Graphics2D) g2d).setStroke(new BasicStroke(3));
        for (double x = minX; x < maxX; x += INCREMENT) {
            double n = x / f;
            try {
                if (!Double.isNaN(f * eval(n))) {
                    int d = (int) (f * eval(n));
                    g2d.drawLine((int) x, d, (int) (x + 1), (d));
                }
            } catch (Exception ex) {

            }
        }
    }

    public double eval(String s) {
        s += "*1";
        String num = "1234567890.no";
        String operations = "+@,*/%,abcdefghijklmpq^(),";
        StringStack operator = new StringStack();
        DoubleStack term = new DoubleStack();
        for (int i = 0; i < s.length(); i++) {
            if (num.contains(s.charAt(i) + "")) {
                if (s.charAt(i) == 'n') {
                    term.add(Math.PI);
                } else if (s.charAt(i) == 'o') {
                    term.add(Math.E);
                } else {
                    for (int j = i; j < s.length(); j++) {
                        if (!num.contains(s.charAt(j) + "")) {
                            double d = Double.parseDouble(s.substring(i, j));
                            term.add(d);
                            i = j;
                            break;
                        } else if (j == s.length() - 1) {
                            double d = Double.parseDouble(s.substring(i));
                            term.add(d);
                            i = j;
                            break;
                        }
                    }
                }
            }
            if (operations.contains(s.charAt(i) + "")) {
                String op = "";
                if (s.charAt(i) == '(') {
                    int endIndex = i;
                    int counter = 0;
                    for (int z = i + 1; z < s.length(); z++) {
                        if (s.charAt(z) == ')') {
                            if (counter == 0) {
                                endIndex = z;
                                break;
                            } else {
                                counter--;
                            }

                        } else if (s.charAt(z) == '(')
                            counter++;
                    }
                    String p = eval(s.substring(i + 1, endIndex)) + "";
                    term.add(Double.parseDouble(p));
                    i = endIndex;
                } else if (s.charAt(i) == 'i') {
                    int endIndex = i;
                    int counter = 0;
                    for (int z = i + 2; z < s.length(); z++) {
                        if (s.charAt(z) == ')') {
                            if (counter == 0) {
                                endIndex = z;
                                break;
                            } else {
                                counter--;
                            }

                        } else if (s.charAt(z) == '(')
                            counter++;
                    }
                    String p = Math.cos(eval(s.substring(i + 1, endIndex + 1))) + "";
                    term.add(Double.parseDouble(p));
                    i = endIndex;
                } else if (s.charAt(i) == 'e') {
                    int endIndex = i;
                    int counter = 0;
                    for (int z = i + 2; z < s.length(); z++) {
                        if (s.charAt(z) == ')') {
                            if (counter == 0) {
                                endIndex = z;
                                break;
                            } else {
                                counter--;
                            }

                        } else if (s.charAt(z) == '(')
                            counter++;
                    }
                    String p = Math.sin(eval(s.substring(i + 1, endIndex + 1))) + "";
                    term.add(Double.parseDouble(p));
                    i = endIndex;
                } else if (s.charAt(i) == 'm') {
```

```
            int endIndex = i;
            int counter = 0;
            for (int z = i + 2; z < s.length(); z++) {
                if (s.charAt(z) == ')') {
                    if (counter == 0) {
                        endIndex = z;
                        break;
                    } else {
                        counter--;
                    }

                } else if (s.charAt(z) == '(')
                    counter++;
            }
            String p = Math.tan(eval(s.substring(i + 1, endIndex + 1))) + "";
            term.add(Double.parseDouble(p));
            i = endIndex;
        } else if (s.charAt(i) == 'j') {
            int endIndex = i;
            int counter = 0;
            for (int z = i + 2; z < s.length(); z++) {
                if (s.charAt(z) == ')') {
                    if (counter == 0) {
                        endIndex = z;
                        break;
                    } else {
                        counter--;
                    }

                } else if (s.charAt(z) == '(')
                    counter++;
            }
            String p = Math.abs(eval(s.substring(i + 1, endIndex + 1))) + "";
            term.add(Double.parseDouble(p));
            i = endIndex;

        } else if (s.charAt(i) == 'f') {
            int endIndex = i;
            int counter = 0;
            for (int z = i + 2; z < s.length(); z++) {
                if (s.charAt(z) == ')') {
                    if (counter == 0) {
                        endIndex = z;
                        break;
                    } else {
                        counter--;
                    }

                } else if (s.charAt(z) == '(')
                    counter++;
            }
            String p = Math.log10(eval(s.substring(i + 1, endIndex + 1))) + "";
            term.add(Double.parseDouble(p));
```

```
		i = endIndex;
	} else if (s.charAt(i) == 'd') {
		int endIndex = i;
		int counter = 0;
		for (int z = i + 2; z < s.length(); z++) {
			if (s.charAt(z) == ')') {
				if (counter == 0) {
					endIndex = z;
					break;
				} else {
					counter--;
				}

			} else if (s.charAt(z) == '(')
				counter++;
		}
		String p = Math.asin(eval(s.substring(i + 1, endIndex + 1))) + "";
		term.add(Double.parseDouble(p));
		i = endIndex;
	} else if (s.charAt(i) == 'h') {
		int endIndex = i;
		int counter = 0;
		for (int z = i + 2; z < s.length(); z++) {
			if (s.charAt(z) == ')') {
				if (counter == 0) {
					endIndex = z;
					break;
				} else {
					counter--;
				}

			} else if (s.charAt(z) == '(')
				counter++;
		}
		String p = Math.acos(eval(s.substring(i + 1, endIndex + 1))) + "";
		term.add(Double.parseDouble(p));
		i = endIndex;
	} else if (s.charAt(i) == 'l') {
		int endIndex = i;
		int counter = 0;
		for (int z = i + 2; z < s.length(); z++) {
			if (s.charAt(z) == ')') {
				if (counter == 0) {
					endIndex = z;
					break;
				} else {
					counter--;
				}

			} else if (s.charAt(z) == '(')
				counter++;
		}
		String p = Math.atan(eval(s.substring(i + 1, endIndex + 1))) + "";
```

```
                    term.add(Double.parseDouble(p));
                    i = endIndex;
                } else if (s.charAt(i) == 'a') {
                    int endIndex = i;
                    int counter = 0;
                    for (int z = i + 2; z < s.length(); z++) {
                        if (s.charAt(z) == ')') {
                            if (counter == 0) {
                                endIndex = z;
                                break;
                            } else {
                                counter--;
                            }

                        } else if (s.charAt(z) == '(')
                            counter++;
                    }
                    String p = Math.exp(eval(s.substring(i + 1, endIndex + 1))) + "";
                    term.add(Double.parseDouble(p));
                    i = endIndex;
                } else if (s.charAt(i) == 'g') {
                    int endIndex = i;
                    int counter = 0;
                    for (int z = i + 2; z < s.length(); z++) {
                        if (s.charAt(z) == ')') {
                            if (counter == 0) {
                                endIndex = z;
                                break;
                            } else {
                                counter--;
                            }

                        } else if (s.charAt(z) == '(')
                            counter++;
                    }
                    String p = Math.log(eval(s.substring(i + 1, endIndex + 1))) + "";
                    term.add(Double.parseDouble(p));
                    i = endIndex;
                } else {
                    op = s.charAt(i) + "";
                    if (operator.isEmpty()) {
                        operator.add(op);
                    } else {
                        if (operations.indexOf(",", operations.indexOf(operator.peek())) >=
operations.indexOf(",",
                                operations.indexOf(op))) {
                            if (operator.peek().equals("(")) {

                            } else if (operator.peek().equals("*")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(term1 * term2);
                                operator.pop();
                        i--;
                    } else if (operator.peek().equals("/")) {
                        double term1 = term.pop();
                        double term2 = term.pop();
                        term.add(term2 / term1);
                        operator.pop();
                        i--;
                    } else if (operator.peek().equals("%")) {
                        double term1 = term.pop();
                        double term2 = term.pop();
                        term.add(term2 % term1);
                        operator.pop();
                        i--;
                    } else if (operator.peek().equals("+")) {
                        double term1 = term.pop();
                        double term2 = term.pop();
                        term.add(term2 + term1);
                        operator.pop();
                        i--;
                    } else if (operator.peek().equals("@")) {
                        double term1 = term.pop();
                        double term2 = term.pop();
                        term.add(term2 - term1);
                        operator.pop();
                        i--;
                    } else if (operator.peek().equals("^")) {
                        double term1 = term.pop();
                        double term2 = term.pop();
                        term.add(Math.pow(term2, term1));
                        operator.pop();
                        i--;
                    }

                } else {
                    operator.add(op);
                }
            }
        }
    }
}
term.pop();
operator.pop();
if (operator.isEmpty()) {
    if (term.size() == 1)
        return term.pop();
    else {
        return (Double) null;
    }
}

else {
    if (operator.peek().equals("+")) {
        return term.pop() + term.pop();
```

```
                        } else {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                return term2 - term1;
                        }
                }

        }
}
```

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.*;
public class GraphingPanel extends JPanel implements KeyListener, ActionListener {
        private final int DEFAULT_DX = 0;
        private final int DEFAULT_DY = 0;
        private final int ORIGIN_X = 50;
        private final int ORIGIN_Y = 50;
        private final int ORIGIN_WIDTH = 200;
        private final int ORIGIN_HEIGHT = 50;

        private final int ZOOM_IN_X = 50;
        private final int ZOOM_IN_Y = 125;
        private final int ZOOM_IN_WIDTH = 200;
        private final int ZOOM_IN_HEIGHT = 50;

        private final int ZOOM_OUT_X = 50;
        private final int ZOOM_OUT_Y = 200;
        private final int ZOOM_OUT_WIDTH = 200;
        private final int ZOOM_OUT_HEIGHT = 50;

        private final int EDIT_FUNCTIONS_X = 50;
        private final int EDIT_FUNCTIONS_Y = 275;
        private final int EDIT_FUNCTIONS_WIDTH = 200;
        private final int EDIT_FUNCTIONS_HEIGHT = 50;

        private final int ADD_FUNCTIONS_X = 50;
        private final int ADD_FUNCTIONS_Y = 350;
        private final int ADD_FUNCTIONS_WIDTH = 200;
        private final int ADD_FUNCTIONS_HEIGHT = 50;

        private final int SPEED = 150;
        private final int TWO_FIFTY_SIX = 256;
        private final double F_CHANGE = 1.3;
        private final double MIN_F = 3;
        private final double DEFAULT_F = 50;
```

```
	private double f = DEFAULT_F;
	private int width;
	private int height;
	private int dx = 0;
	private int dy = 0;
	private boolean up = false;
	private boolean down = false;
	private boolean right = false;
	private boolean left = false;
	private JButton zoomIn = new JButton();
	private JButton zoomOut = new JButton();
	public JButton editFunctions = new JButton();
	private JButton origin = new JButton();
	public JButton addFunction = new JButton();

	public static SinglyLinkedList functions = new SinglyLinkedList();
	private Image originImage = new ImageIcon("src\\Images\\Calculus\\origin.png").getImage();
	private Image zoomInImage = new ImageIcon("src\\Images\\Calculus\\zoom in.png").getImage();
	private Image zoomOutImage = new ImageIcon("src\\Images\\Calculus\\zoom out.png").getImage();
	private Image editImage = new ImageIcon("src\\Images\\Calculus\\edit.png").getImage();
	private Image addFunctionImage = new ImageIcon("src\\Images\\Calculus\\add function.png").getImage();

	public GraphingPanel(int w, int h) {
		width = w;
		height = h;
		this.setSize(w, h);
		this.setLayout(null);
		this.addKeyListener(this);
		this.setFocusable(true);
		this.requestFocusInWindow();

		zoomIn.setBounds(ZOOM_IN_X, ZOOM_IN_Y, ZOOM_IN_WIDTH, ZOOM_IN_HEIGHT);
		zoomIn.addActionListener(this);
		zoomIn.setIcon(new ImageIcon(zoomInImage.getScaledInstance(ZOOM_IN_WIDTH, ZOOM_IN_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		this.add(zoomIn);


		zoomOut.setBounds(ZOOM_OUT_X, ZOOM_OUT_Y, ZOOM_OUT_WIDTH, ZOOM_OUT_HEIGHT);
		zoomOut.addActionListener(this);
		zoomOut.setIcon(new ImageIcon(zoomOutImage.getScaledInstance(ZOOM_OUT_WIDTH, ZOOM_OUT_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		this.add(zoomOut);


		editFunctions.setBounds(EDIT_FUNCTIONS_X, EDIT_FUNCTIONS_Y, EDIT_FUNCTIONS_WIDTH,
EDIT_FUNCTIONS_HEIGHT);
		editFunctions.addActionListener(this);
		editFunctions.setIcon(new ImageIcon(editImage.getScaledInstance(EDIT_FUNCTIONS_WIDTH,
EDIT_FUNCTIONS_HEIGHT, java.awt.Image.SCALE_SMOOTH)));
		this.add(editFunctions);
```

```
			origin.setBounds(ORIGIN_X, ORIGIN_Y, ORIGIN_WIDTH, ORIGIN_HEIGHT);
			origin.setIcon(new ImageIcon(originImage.getScaledInstance(ORIGIN_WIDTH, ORIGIN_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
			origin.addActionListener(this);
			this.add(origin);


addFunction.setBounds(ADD_FUNCTIONS_X,ADD_FUNCTIONS_Y,ADD_FUNCTIONS_WIDTH,ADD_FUNCTIONS_HEIGHT);
			addFunction.addActionListener(this);
			addFunction.setIcon(new ImageIcon(addFunctionImage.getScaledInstance(ADD_FUNCTIONS_WIDTH,
ADD_FUNCTIONS_HEIGHT, java.awt.Image.SCALE_SMOOTH)));
			this.add(addFunction);
		}

		public void paint(Graphics g) {
			super.paint(g);
			Graphics2D g2d = (Graphics2D) g;
			g2d.translate(width / 2 - dx, height / 2 + dy);
			g2d.scale(1, -1);
			setUpGrid(g2d);
			for (int i = 0; i < functions.size(); i++) {
				if (functions.get(i).show)
					functions.get(i).drawFunction(g2d, -width / 2 + dx, width / 2 + dx, f);
			}
			if (left) {
				dx -= SPEED;
			}
			if (right) {
				dx += SPEED;
			}
			if (up) {
				dy += SPEED;
			}
			if (down) {
				dy -= SPEED;
			}
			repaint();
		}
		public Color randomColor() {
			int R = (int) (Math.random() * TWO_FIFTY_SIX);
			int G = (int) (Math.random() * TWO_FIFTY_SIX);
			int B = (int) (Math.random() * TWO_FIFTY_SIX);
			Color color = new Color(R, G, B);
			return color;
		}

		public void setUpGrid(Graphics2D g2d) {
			g2d.setColor(Color.gray);
			g2d.setStroke(new BasicStroke(1));
			for (int i = 0; i < width / 2 + dx; i += f) {
				g2d.drawLine(i, height / 2 + dy, i, -height / 2 + dy);
			}
		for (int i = 0; i > -width / 2 + dx; i -= f) {
			g2d.drawLine(i, height / 2 + dy, i, -height / 2 + dy);
		}
		for (int i = 0; i < height / 2 + dy; i += f) {
			g2d.drawLine(width / 2 + dx, i, -width / 2 + dx, i);
		}
		for (int i = 0; i > -height / 2 + dy; i -= f) {
			g2d.drawLine(width / 2 + dx, i, -width / 2 + dx, i);
		}
		g2d.setStroke(new BasicStroke(3));
		g2d.setColor(Color.black);
		g2d.drawLine(0, height / 2 + dy, 0, -height / 2 + dy);
		g2d.drawLine(width / 2 + dx, 0, -width / 2 + dx, 0);
	}

	public void keyTyped(KeyEvent e) {

	}

	public void keyPressed(KeyEvent e) {
		int key = e.getKeyCode();
		switch (key) {
		case KeyEvent.VK_LEFT: {
			left = true;
			break;
		}
		case KeyEvent.VK_RIGHT: {
			right = true;
			break;
		}
		case KeyEvent.VK_DOWN: {
			down = true;
			break;
		}
		case KeyEvent.VK_UP: {
			up = true;
			break;
		}

		}
	}

	public void keyReleased(KeyEvent e) {
		int key = e.getKeyCode();
		switch (key) {
		case KeyEvent.VK_LEFT: {
			left = false;
			break;
		}
		case KeyEvent.VK_RIGHT: {
			right = false;
			break;
		}
```

```
                    case KeyEvent.VK_DOWN: {
                              down = false;
                              break;
                    }
                    case KeyEvent.VK_UP: {
                              up = false;
                              break;
                    }
                    }
          }

          public void actionPerformed(ActionEvent e) {
                    if (e.getSource() == zoomIn) {
                              f = f * F_CHANGE;
                    }
                    if (e.getSource() == zoomOut) {
                              f = f / F_CHANGE;
                              if (f <= MIN_F)
                                        f = MIN_F;
                    }
                    if (e.getSource()==origin) {
                              dx = DEFAULT_DX;
                              dy = DEFAULT_DY;
                              f = DEFAULT_F;
                    }
                    this.addKeyListener(this);
                    this.setFocusable(true);
                    this.requestFocusInWindow();

          }

}
```

```
public class Main {
          public static void main(String[]args) {
                    Frame f = new Frame();
          }
}
```

```
public class Node {
	Function data;
	Node next = null;

	public Node(Function data) {
		this.data = data;
	}
	public boolean hasNext() {
		if (next==null)
			return false;
		return true;
	}
}
```

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
public class ScientificPanel extends JPanel implements ActionListener {
	private final String DISPLAY_LABEL_FONT = "Verdana";
	private final int DISPLAY_LABEL_FONT_SIZE = 20;

	private final int DISPLAY_LABEL_X = 155;
	private final int DISPLAY_LABEL_Y = 30;
	private final int DISPLAY_LABEL_WIDTH = 685;
	private final int DISPLAY_LABEL_HEIGHT = 130;

	private final int BACKGROUND_RECTANGLE_X = 135;
	private final int BACKGROUND_RECTANGLE_Y = 30;
	private final int BACKGROUND_RECTANGLE_WIDTH = 715;
	private final int BACKGROUND_RECTANGLE_HEIGHT = 130;

	private final int CALCULATOR_BUTTONS_TOP_LEFT_X = 135;
	private final int CALCULATOR_BUTTONS_TOP_LEFT_Y = 200;
	private final int CALCULATOR_BUTTONS_WIDTH = 80;
	private final int CALCULATOR_BUTTONS_HEIGHT = 50;
	private final int CALCULATOR_BUTTONS_COLUMN = 9;
	private final int CALCULATOR_BUTTONS_ROW = 4;

	public boolean notifcationShown = false;
	private int panelWidth;
	private int panelHeight;
	private boolean lastButtonWasEqual = false;
	private String processText = "";
	private String displayText = "";
	private StringStack input = new StringStack();
	private StringStack processInput = new StringStack();
	private JLabel displayLabel = new JLabel(displayText);
	private JButton[][] calculatorButtons = new JButton[CALCULATOR_BUTTONS_COLUMN][CALCULATOR_BUTTONS_ROW];
	private String[][] addToDisplayText = { { "(", ")", "^(", "e^(", "7", "8", "9", "del", "ac" },
			{ "arcsin(", "sin(", "log(", "ln(", "4", "5", "6", "*", "/" },
			{ "arcos(", "cos(", "abs(", "factorial(", "1", "2", "3", "+", "-" },
			{ "arctan(", "tan(", "pi", "e", "0", ".", "factor(", "%", "=", } };
	private String[][] addToProcessText = { { "(", ")", "^(", "a(", "7", "8", "9", "b", "c" },
			{ "d(", "e(", "f(", "g(", "4", "5", "6", "*", "/" }, { "h(", "i(", "j(", "k(", "1", "2", "3", "+", "@" },
			{ "l(", "m(", "n", "o", "0", ".", "p(", "%", "q" }, };

	private Image calculatorButtonImages;
	private Image backgroundImage;

	public ScientificPanel(int panelWidth, int panelHeight) {
		this.setLayout(null);
		this.setSize(panelWidth, panelHeight);
		setUpButtons();
		this.panelWidth = panelWidth;
		this.panelHeight = panelHeight;
		displayLabel.setForeground(Color.white);
		displayLabel.setBackground(Color.black);
		displayLabel.setFont(new Font(DISPLAY_LABEL_FONT, Font.PLAIN, DISPLAY_LABEL_FONT_SIZE));
		displayLabel.setBounds(DISPLAY_LABEL_X, DISPLAY_LABEL_Y, DISPLAY_LABEL_WIDTH, DISPLAY_LABEL_HEIGHT);
		this.add(displayLabel);

	}

	public void setUpButtons() {
		for (int i = 0; i < CALCULATOR_BUTTONS_COLUMN; i++) {
			for (int j = 0; j < CALCULATOR_BUTTONS_ROW; j++) {
				calculatorButtons[i][j] = new JButton();
				calculatorButtons[i][j].setBounds(CALCULATOR_BUTTONS_TOP_LEFT_X + i *
CALCULATOR_BUTTONS_WIDTH,
						CALCULATOR_BUTTONS_TOP_LEFT_Y + j * CALCULATOR_BUTTONS_HEIGHT,
CALCULATOR_BUTTONS_WIDTH,
						CALCULATOR_BUTTONS_HEIGHT);
				calculatorButtons[i][j].addActionListener(this);
				calculatorButtonImages = new ImageIcon("src\\Images\\ScientificPanel\\" + i + "_" + j + ".png")
						.getImage();
				calculatorButtons[i][j]
						.setIcon(new
ImageIcon(calculatorButtonImages.getScaledInstance(CALCULATOR_BUTTONS_WIDTH,
								CALCULATOR_BUTTONS_HEIGHT, java.awt.Image.SCALE_SMOOTH)));
				this.add(calculatorButtons[i][j]);
			}
		}
	}

	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		backgroundImage = new ImageIcon("src\\Images\\ScientificPanel\\background.png").getImage();
		g.drawImage(backgroundImage, 0, 0, panelWidth, panelHeight, this);
		Graphics2D g2d = (Graphics2D) g;
		((Graphics2D) g2d).setStroke(new BasicStroke(3));
		g.setColor(Color.black);
		g.fillRect(BACKGROUND_RECTANGLE_X, BACKGROUND_RECTANGLE_Y, BACKGROUND_RECTANGLE_WIDTH,
				BACKGROUND_RECTANGLE_HEIGHT);
		displayLabel.setText(displayText);
		repaint();

	}

	public String factor(long n) {
		if (n == 0)
			return "0=0";
		if (n == 1)
			return "1=1";
		String s = n + "=";
		long count = 2;
		while (n != 1) {
			if (n % count == 0) {
				n = n / count;
				s += count + "*";
				count--;
			}
			count++;
		}
		s = s.substring(0, s.length() - 1);
		return s;
	}

	public double factorial(long n) {
		if (n == 0)
			return 1;
		else
			return n * factorial(n - 1);
	}

	public void actionPerformed(ActionEvent e) {
		try {
			if (lastButtonWasEqual) {
				displayText = "";
				processText = "";
				input.clear();
				processInput.clear();
				lastButtonWasEqual = false;
			}
			if (e.getSource() == calculatorButtons[CALCULATOR_BUTTONS_COLUMN - 1][CALCULATOR_BUTTONS_ROW -
1]) {
				try {
					if (processText.charAt(0) == 'p') {
						String s = (eval(processText)) + "";
						int indexOfDecimal = s.indexOf(".");
```

```java
                    boolean isInteger = true;
                    for (int i = indexOfDecimal + 1; i < s.length(); i++) {
                        if (s.charAt(i) != '0') {
                            isInteger = false;
                            break;
                        }
                    }
                    if (isInteger)
                        s = (int) (eval(processText)) + "";
                    int n = Integer.parseInt(s);
                    displayText = factor(n);
                } else {
                    displayText = eval(processText) + "";
                    int indexOfDecimal = displayText.indexOf(".");
                    boolean isInteger = true;
                    for (int i = indexOfDecimal + 1; i < displayText.length(); i++) {
                        if (displayText.charAt(i) != '0') {
                            isInteger = false;
                            break;
                        }
                    }
                    if (isInteger)
                        displayText = (int) (eval(processText)) + "";
                }
            } catch (Exception ex) {
                displayText = "Invalid Input";
            }
            lastButtonWasEqual = true;
        } else if (e.getSource() == calculatorButtons[8][0]) {
            displayText = "";
            input.clear();
            processText = "";
            processInput.clear();
        } else if (e.getSource() == calculatorButtons[7][0]) {
            if (!displayText.isEmpty()) {
                int lastItemLength = input.pop().length();
                displayText = displayText.substring(0, displayText.length() - lastItemLength);
                lastItemLength = processInput.pop().length();
                processText = processText.substring(0, processText.length() - lastItemLength);
            }
        } else {
            for (int i = 0; i < CALCULATOR_BUTTONS_COLUMN; i++) {
                for (int j = 0; j < CALCULATOR_BUTTONS_ROW; j++) {
                    if (e.getSource() == calculatorButtons[i][j]) {
                        displayText += addToDisplayText[j][i];
                        input.add(addToDisplayText[j][i]);
                        processText += addToProcessText[j][i];
                        processInput.add(addToProcessText[j][i]);
                    }
                }
            }
        }
    } catch (Exception exception) {
```

```
                    displayText = "Invalid Input";
            }
    }

    public double eval(String s) {
            s += "*1";
            String num = "1234567890.no";
            String operations = "+@,*/%,abcdefghijklmpq^(),";
            StringStack operator = new StringStack();
            DoubleStack term = new DoubleStack();
            for (int i = 0; i < s.length(); i++) {
                    if (num.contains(s.charAt(i) + "")) {
                            if (s.charAt(i) == 'n') {
                                    term.add(Math.PI);
                            } else if (s.charAt(i) == 'o') {
                                    term.add(Math.E);
                            } else {
                                    for (int j = i; j < s.length(); j++) {
                                            if (!num.contains(s.charAt(j) + "")) {
                                                    double d = Double.parseDouble(s.substring(i, j));
                                                    term.add(d);
                                                    i = j;
                                                    break;
                                            } else if (j == s.length() - 1) {
                                                    double d = Double.parseDouble(s.substring(i));
                                                    term.add(d);
                                                    i = j;
                                                    break;
                                            }
                                    }
                            }
                    }
                    if (operations.contains(s.charAt(i) + "")) {
                            String op = "";
                            if (s.charAt(i) == '(') {
                                    int endIndex = i;
                                    int counter = 0;
                                    for (int z = i + 1; z < s.length(); z++) {
                                            if (s.charAt(z) == ')') {
                                                    if (counter == 0) {
                                                            endIndex = z;
                                                            break;
                                                    } else {
                                                            counter--;
                                                    }

                                            } else if (s.charAt(z) == '(')
                                                    counter++;
                                    }
                                    String p = eval(s.substring(i + 1, endIndex)) + "";
                                    term.add(Double.parseDouble(p));
                                    i = endIndex;
                            } else if (s.charAt(i) == 'i') {
```

```
                int endIndex = i;
                int counter = 0;
                for (int z = i + 2; z < s.length(); z++) {
                        if (s.charAt(z) == ')') {
                                if (counter == 0) {
                                        endIndex = z;
                                        break;
                                } else {
                                        counter--;
                                }

                        } else if (s.charAt(z) == '(')
                                counter++;
                }
                String p = Math.cos(eval(s.substring(i + 1, endIndex + 1))) + "";
                term.add(Double.parseDouble(p));
                i = endIndex;
        } else if (s.charAt(i) == 'e') {
                int endIndex = i;
                int counter = 0;
                for (int z = i + 2; z < s.length(); z++) {
                        if (s.charAt(z) == ')') {
                                if (counter == 0) {
                                        endIndex = z;
                                        break;
                                } else {
                                        counter--;
                                }

                        } else if (s.charAt(z) == '(')
                                counter++;
                }
                String p = Math.sin(eval(s.substring(i + 1, endIndex + 1))) + "";
                term.add(Double.parseDouble(p));
                i = endIndex;
        } else if (s.charAt(i) == 'm') {
                int endIndex = i;
                int counter = 0;
                for (int z = i + 2; z < s.length(); z++) {
                        if (s.charAt(z) == ')') {
                                if (counter == 0) {
                                        endIndex = z;
                                        break;
                                } else {
                                        counter--;
                                }

                        } else if (s.charAt(z) == '(')
                                counter++;
                }
                String p = Math.tan(eval(s.substring(i + 1, endIndex + 1))) + "";
                term.add(Double.parseDouble(p));
                i = endIndex;
```

```
} else if (s.charAt(i) == 'j') {
        int endIndex = i;
        int counter = 0;
        for (int z = i + 2; z < s.length(); z++) {
                if (s.charAt(z) == ')') {
                        if (counter == 0) {
                                endIndex = z;
                                break;
                        } else {
                                counter--;
                        }

                } else if (s.charAt(z) == '(')
                        counter++;
        }
        String p = Math.abs(eval(s.substring(i + 1, endIndex + 1))) + "";
        term.add(Double.parseDouble(p));
        i = endIndex;
} else if (s.charAt(i) == 'k') {
        int endIndex = i;
        int counter = 0;
        for (int z = i + 2; z < s.length(); z++) {
                if (s.charAt(z) == ')') {
                        if (counter == 0) {
                                endIndex = z;
                                break;
                        } else {
                                counter--;
                        }

                } else if (s.charAt(z) == '(')
                        counter++;
        }
        try {
                double value = eval(s.substring(i + 1, endIndex + 1));
                if (value == Math.floor(value) && !Double.isInfinite(value)) {
                        String p = factorial((long) value) + "";
                        term.add(Double.parseDouble(p));
                        i = endIndex;
                } else {
                        return (Double) (null);
                }
        } catch (Exception ex) {
                return (Double) null;
        }
} else if (s.charAt(i) == 'f') {
        int endIndex = i;
        int counter = 0;
        for (int z = i + 2; z < s.length(); z++) {
                if (s.charAt(z) == ')') {
                        if (counter == 0) {
                                endIndex = z;
                                break;
                        } else {
                            counter--;
                        }

                    } else if (s.charAt(z) == '(')
                        counter++;
                }
                String p = Math.log10(eval(s.substring(i + 1, endIndex + 1))) + "";
                term.add(Double.parseDouble(p));
                i = endIndex;
            } else if (s.charAt(i) == 'd') {
                int endIndex = i;
                int counter = 0;
                for (int z = i + 2; z < s.length(); z++) {
                    if (s.charAt(z) == ')') {
                        if (counter == 0) {
                            endIndex = z;
                            break;
                        } else {
                            counter--;
                        }

                    } else if (s.charAt(z) == '(')
                        counter++;
                }
                String p = Math.asin(eval(s.substring(i + 1, endIndex + 1))) + "";
                term.add(Double.parseDouble(p));
                i = endIndex;
            } else if (s.charAt(i) == 'h') {
                int endIndex = i;
                int counter = 0;
                for (int z = i + 2; z < s.length(); z++) {
                    if (s.charAt(z) == ')') {
                        if (counter == 0) {
                            endIndex = z;
                            break;
                        } else {
                            counter--;
                        }

                    } else if (s.charAt(z) == '(')
                        counter++;
                }
                String p = Math.acos(eval(s.substring(i + 1, endIndex + 1))) + "";
                term.add(Double.parseDouble(p));
                i = endIndex;
            } else if (s.charAt(i) == 'l') {
                int endIndex = i;
                int counter = 0;
                for (int z = i + 2; z < s.length(); z++) {
                    if (s.charAt(z) == ')') {
                        if (counter == 0) {
                            endIndex = z;
                break;
            } else {
                counter--;
            }

        } else if (s.charAt(z) == '(')
            counter++;
    }
    String p = Math.atan(eval(s.substring(i + 1, endIndex + 1))) + "";
    term.add(Double.parseDouble(p));
    i = endIndex;
} else if (s.charAt(i) == 'a') {
    int endIndex = i;
    int counter = 0;
    for (int z = i + 2; z < s.length(); z++) {
        if (s.charAt(z) == ')') {
            if (counter == 0) {
                endIndex = z;
                break;
            } else {
                counter--;
            }

        } else if (s.charAt(z) == '(')
            counter++;
    }
    String p = Math.exp(eval(s.substring(i + 1, endIndex + 1))) + "";
    term.add(Double.parseDouble(p));
    i = endIndex;
} else if (s.charAt(i) == 'g') {
    int endIndex = i;
    int counter = 0;
    for (int z = i + 2; z < s.length(); z++) {
        if (s.charAt(z) == ')') {
            if (counter == 0) {
                endIndex = z;
                break;
            } else {
                counter--;
            }

        } else if (s.charAt(z) == '(')
            counter++;
    }
    String p = Math.log(eval(s.substring(i + 1, endIndex + 1))) + "";
    term.add(Double.parseDouble(p));
    i = endIndex;
} else {
    op = s.charAt(i) + "";
    if (operator.isEmpty()) {
        operator.add(op);
    } else {
```

```
                        if (operations.indexOf(",", operations.indexOf(operator.peek())) >=
operations.indexOf(",",
                                operations.indexOf(op))) {
                            if (operator.peek().equals("(")) {

                            } else if (operator.peek().equals("*")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(term1 * term2);
                                operator.pop();
                                i--;
                            } else if (operator.peek().equals("/")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(term2 / term1);
                                operator.pop();
                                i--;
                            } else if (operator.peek().equals("%")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(term2 % term1);
                                operator.pop();
                                i--;
                            } else if (operator.peek().equals("+")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(term2 + term1);
                                operator.pop();
                                i--;
                            } else if (operator.peek().equals("@")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(term2 - term1);
                                operator.pop();
                                i--;
                            } else if (operator.peek().equals("^")) {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                term.add(Math.pow(term2, term1));
                                operator.pop();
                                i--;
                            }

                        } else {
                            operator.add(op);
                        }
                    }
                }
            }
        }
        term.pop();
        operator.pop();
        if (operator.isEmpty()) {
```

```
                        if (term.size() == 1)
                                return term.pop();
                        else {
                                return (Double) null;
                        }
                }

                else {
                        if (operator.peek().equals("+")) {
                                return term.pop() + term.pop();
                        } else {
                                double term1 = term.pop();
                                double term2 = term.pop();
                                return term2 - term1;
                        }
                }

        }
}
```

```
public class SinglyLinkedList {
        private Node head = null;
        private Node curr = null;
        private int size = 0;

        public void add(Function f) {
                Node n = new Node(f);
                if (head == null) {
                        head = n;
                        curr = n;
                } else {
                        curr.next = n;
                        curr = n;

                }
                size++;
        }

        public int size() {
                return size;
        }

        public Function get(int index) {
            Node n = head;
            int i=0;
            while(i<index && n!=null) {
                n = n.next;
                i++;
            }
            return n.data;

        }

        public void remove(int index) {
                if(index==0) {
                        if (size!=1)
                                head = head.next;
                        else {
                                head = null;
                                curr = null;
                        }
          }
                else if (index==size-1) {
                        Node n = head;
                        for (int i =0; i<index-1; i++) {
                                n = n.next;
                        }
                        curr = n;
                        curr.next=null;
                }
                else {
            Node n = head;
            for (int i =0; i<index-1; i++) {
                n= n.next;
            }
            n.next= n.next.next;
                }
                size--;
        }

        public void display() {
                Node current = head;
                while (current != null) {
                        System.out.println(current.data);
                        current = current.next;
                }
                System.out.println();
        }
}
```

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
import java.util.*;
import javax.swing.*;
public class StatisticsPanel extends JPanel implements ActionListener {
	private final String FONT = "Verdana";
	private final int INPUT_LABEL_FONT_SIZE = 15;
	private final int TEXT_FIELD_FONT_SIZE = 15;
	private final int STAT_LABEL_FONT_SIZE = 20;
	private final int TEXT_AREA_FONT_SIZE = 15;

	private final int BINOM_DIS_SIZE = 18;
	private final int BINOM_DIS_TEXT_SIZE = 13;

	private final int INPUT_LABEL_X = 50;
	private final int INPUT_LABEL_Y = 25;
	private final int INPUT_LABEL_WIDTH = 300;
	private final int INPUT_LABEL_HEIGHT = 20;

	private final int TEXT_FIELD_X = 50;
	private final int TEXT_FIELD_Y = 60;
	private final int TEXT_FIELD_WIDTH = 300;
	private final int TEXT_FIELD_HEIGHT = 20;

	private final int SCROLL_PANE_X = 50;
	private final int SCROLL_PANE_Y = 100;
	private final int SCROLL_PANE_WIDTH = 300;
	private final int SCROLL_PANE_HEIGHT = 200;

	private final int TEXT_AREA_ROWS = 150;
	private final int TEXT_AREA_COLUMNS = 300;

	private final int SORT_BUTTON_X = 100;
	private final int SORT_BUTTON_Y = 310;
	private final int SORT_BUTTON_WIDTH = 200;
	private final int SORT_BUTTON_HEIGHT = 50;

	private final int REMOVE_BUTTON_X = 100;
	private final int REMOVE_BUTTON_Y = 370;
	private final int REMOVE_BUTTON_WIDTH = 200;
	private final int REMOVE_BUTTON_HEIGHT = 50;

	private final int REMOVE_ALL_BUTTON_X = 100;
	private final int REMOVE_ALL_BUTTON_Y = 430;
	private final int REMOVE_ALL_BUTTON_WIDTH = 200;
	private final int REMOVE_ALL_BUTTON_HEIGHT = 50;

	private final int STAT_LABEL_X = 50;
	private final int STAT_LABEL_Y = 500;
```

```java
private final int STAT_LABEL_WIDTH = 600;
private final int STAT_LABEL_HEIGHT = 50;

private final int BINOM_DIS_X = 450;
private final int BINOM_DIS_Y = 75;
private final int BINOM_DIS_WIDTH = 500;
private final int BINOM_DIS_HEIGHT = 20;
private final int BINOM_DIS_CHANGE_IN_HEIGHT = 50;

private final int BINOM_DIS_TEXT_X = 650;
private final int BINOM_DIS_TEXT_Y = 125;
private final int BINOM_DIS_TEXT_WIDTH = 100;
private final int BINOM_DIS_TEXT_HEIGHT = 20;
private final int BINOM_DIS_TEXT_CHANGE_IN_HEIGHT = 50;

private final String SPACE = "		";
private final int DX = 5;

private final int NUM_OF_DECIMAL_ACCURACY = 9;
private final double FOUR_DECIMAL_ACCURACY = 10000.0;
private final double TEN = 10.0;
private final int SHIFT_BINOM_LEFT=80;
private final int SHIFT_BINOM_UP=-40;

private final String FILE = "src\\StatsText.txt";

private ArrayList<Double> dataValues = new ArrayList<Double>();
private JLabel inputLabel = new JLabel("Enter To Add A Data Value:");
private JTextField textField = new JTextField();
private String textAreaString = "";
private JTextArea textArea = new JTextArea(textAreaString, TEXT_AREA_ROWS, TEXT_AREA_COLUMNS);
private JScrollPane scrollPane;
private JButton sortButton = new JButton();
private JButton removeButton = new JButton();
private JButton removeAll = new JButton();
private JLabel statLabels[] = new JLabel[5];
private JLabel binomDis[] = new JLabel[6];
private JTextField binomDisText[] = new JTextField[5];
private boolean validNumberOfTrials = false;
private boolean validProbabilityOfSuccess = false;
private boolean validRandomVariable = false;
private long numberOfTrials = 0;
private double probabilityOfSuccess = 0;
private long randomVariable = 0;
private int panelWidth;
private int panelHeight;
private Image backgroundImage = new ImageIcon("src\\Images\\StatisticsPanel\\background.png").getImage();
private Image sortButtonImage = new ImageIcon("src\\Images\\StatisticsPanel\\sortButton.png").getImage();
private Image removeButtonImage = new ImageIcon("src\\Images\\StatisticsPanel\\removeItem.png").getImage();
private Image removeAllButtonImage = new ImageIcon("src\\Images\\StatisticsPanel\\removeAll.png").getImage();

public StatisticsPanel(int w, int h) {
	panelWidth = w;
```

```
		panelHeight = h;
		this.setBackground(Color.gray);
		this.setLayout(null);
		this.setSize(w, h);

		inputLabel.setBounds(INPUT_LABEL_X, INPUT_LABEL_Y, INPUT_LABEL_WIDTH, INPUT_LABEL_HEIGHT);
		inputLabel.setBackground(Color.white);
		inputLabel.setForeground(Color.black);
		inputLabel.setFont(new Font(FONT, Font.PLAIN, INPUT_LABEL_FONT_SIZE));
		this.add(inputLabel);

		textField.setBounds(TEXT_FIELD_X, TEXT_FIELD_Y, TEXT_FIELD_WIDTH, TEXT_FIELD_HEIGHT);
		textField.setFont(new Font(FONT, Font.PLAIN, TEXT_FIELD_FONT_SIZE));
		textField.addActionListener(this);
		this.add(textField);

		textArea.setLineWrap(true);
		textAreaString = "";
		setArrayFromText();
		for (int i = 0; i < dataValues.size(); i++) {
			textAreaString += ("Data item " + (i + 1) + " is: " + dataValues.get(i) + "\n\n");
		}
		textArea.setText(textAreaString);
		textArea.setFont(new Font(FONT, Font.PLAIN, TEXT_AREA_FONT_SIZE));
		writeArrayInText();
		this.add(textArea);

		scrollPane = new JScrollPane(textArea);
		scrollPane.setBounds(SCROLL_PANE_X, SCROLL_PANE_Y, SCROLL_PANE_WIDTH, SCROLL_PANE_HEIGHT);
		scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
		scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
		this.add(scrollPane);

		sortButton.setBounds(SORT_BUTTON_X, SORT_BUTTON_Y, SORT_BUTTON_WIDTH, SORT_BUTTON_HEIGHT);
		sortButton.setIcon(new ImageIcon(sortButtonImage.getScaledInstance(SORT_BUTTON_WIDTH,
SORT_BUTTON_HEIGHT, java.awt.Image.SCALE_SMOOTH)));
		sortButton.addActionListener(this);
		this.add(sortButton);

		removeButton.setBounds(REMOVE_BUTTON_X, REMOVE_BUTTON_Y, REMOVE_BUTTON_WIDTH,
REMOVE_BUTTON_HEIGHT);
		removeButton.setIcon(new ImageIcon(removeButtonImage.getScaledInstance(REMOVE_BUTTON_WIDTH,
REMOVE_BUTTON_HEIGHT, java.awt.Image.SCALE_SMOOTH)));
		removeButton.addActionListener(this);
		this.add(removeButton);

		removeAll.setBounds(REMOVE_ALL_BUTTON_X, REMOVE_ALL_BUTTON_Y, REMOVE_ALL_BUTTON_WIDTH,
REMOVE_ALL_BUTTON_HEIGHT);
		removeAll.setIcon(new ImageIcon(removeAllButtonImage.getScaledInstance(REMOVE_ALL_BUTTON_WIDTH,
REMOVE_ALL_BUTTON_HEIGHT, java.awt.Image.SCALE_SMOOTH)));
		removeAll.addActionListener(this);
		this.add(removeAll);
		statLabels[0] = new JLabel("Mean: ");
		statLabels[1] = new JLabel("Range: ");
		statLabels[2] = new JLabel("Standard Deviation: ");
		statLabels[3] = new JLabel("Variance: ");
		statLabels[4] = new JLabel("Median: ");
		for (int i = 0; i < statLabels.length; i++) {
			statLabels[i].setBounds(STAT_LABEL_X, STAT_LABEL_Y + STAT_LABEL_HEIGHT * i,
STAT_LABEL_WIDTH,STAT_LABEL_HEIGHT);
			statLabels[i].setBackground(Color.white);
			statLabels[i].setForeground(Color.black);
			statLabels[i].setFont(new Font(FONT, Font.PLAIN, STAT_LABEL_FONT_SIZE));
			this.add(statLabels[i]);
		}

		binomDis[0] = new JLabel("Binomial Distribution Functions");
		binomDis[1] = new JLabel("Number Of Trials:            " + SPACE + isValid(validNumberOfTrials));
		binomDis[2] = new JLabel("Probability of Sucess:       " + SPACE + isValid(validProbabilityOfSuccess));
		binomDis[3] = new JLabel("Random Variable:            " + SPACE + isValid(validRandomVariable));
		binomDis[4] = new JLabel("PDF: ");
		binomDis[5] = new JLabel("CDF: ");
		for (int i = 0; i < binomDis.length; i++) {
			binomDis[i].setBounds(BINOM_DIS_X, BINOM_DIS_Y + BINOM_DIS_CHANGE_IN_HEIGHT * i,
BINOM_DIS_WIDTH,
					BINOM_DIS_HEIGHT);
			binomDis[i].setBackground(Color.white);
			binomDis[i].setForeground(Color.black);

			binomDis[i].setFont(new Font(FONT, Font.PLAIN, BINOM_DIS_SIZE));
			this.add(binomDis[i]);
		}
		binomDis[0].setBounds(BINOM_DIS_X+SHIFT_BINOM_LEFT, BINOM_DIS_Y+SHIFT_BINOM_UP,
BINOM_DIS_WIDTH,BINOM_DIS_HEIGHT);

		for (int i = 0; i < binomDisText.length; i++) {
			binomDisText[i] = new JTextField();
			binomDisText[i].setBounds(BINOM_DIS_TEXT_X, BINOM_DIS_TEXT_Y +
BINOM_DIS_TEXT_CHANGE_IN_HEIGHT * i,
					BINOM_DIS_TEXT_WIDTH, BINOM_DIS_TEXT_HEIGHT);
			binomDisText[i].setFont(new Font(FONT, Font.PLAIN, BINOM_DIS_TEXT_SIZE));
			binomDisText[i].addActionListener(this);
			this.add(binomDisText[i]);
		}

		setUpStats();
	}

	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		g.drawImage(backgroundImage, 0, 0, panelWidth, panelHeight, this);
		g.setColor(Color.black);
		g.fillRect(SCROLL_PANE_X-DX, SCROLL_PANE_Y-DX, SCROLL_PANE_WIDTH+2*DX, SCROLL_PANE_HEIGHT+2*DX);
		g.fillRect(TEXT_FIELD_X-DX, TEXT_FIELD_Y-DX, TEXT_FIELD_WIDTH+2*DX, TEXT_FIELD_HEIGHT+2*DX);
		for (int i = 0; i < binomDisText.length; i++) {
				g.fillRect(BINOM_DIS_TEXT_X-DX, BINOM_DIS_TEXT_Y + BINOM_DIS_TEXT_CHANGE_IN_HEIGHT * 
i-DX,BINOM_DIS_TEXT_WIDTH+2*DX, BINOM_DIS_TEXT_HEIGHT+2*DX);
			}
			repaint();
	}

	public void trialsSetValidLabel(boolean b) {
		binomDis[1].setText("Number Of Trials:		" + SPACE + isValid(validNumberOfTrials));

	}

	public void pSetValidLabel(boolean b) {
		binomDis[2].setText("Probability of Sucess:		" + SPACE+isValid(validProbabilityOfSuccess));
	}

	public void randomVariableSetValidLabel(boolean b) {
		binomDis[3].setText("Random Variable:		" + SPACE + isValid(validRandomVariable));
	}

	public String isValid(boolean b) {
		if (b)
			return "Input Is Entered";
		else
			return "Input Is Not Entered";
	}

	public void updateBinom() {
		double pdf = pdf(numberOfTrials, probabilityOfSuccess, randomVariable);
		binomDisText[3].setText(round(pdf, NUM_OF_DECIMAL_ACCURACY) + "");
		double cdf = 0;
		for (int i = 0; i <= randomVariable; i++) {
			cdf += pdf(numberOfTrials, probabilityOfSuccess, i);
		}
		binomDisText[4].setText(round(cdf, NUM_OF_DECIMAL_ACCURACY) + "");
	}

	public double pdf(long trials, double p, long randomVariable) {
		double pdf = (factorial(trials) / ((factorial(randomVariable) * factorial(trials - randomVariable))))
					* Math.pow(p, randomVariable) * Math.pow(1 - p, trials - randomVariable);
		return pdf;
	}

	public double round(double x) {
		return Math.round(x * FOUR_DECIMAL_ACCURACY) / FOUR_DECIMAL_ACCURACY;
	}

	public double round(double x, int n) {
		return Math.round(x * Math.pow(TEN, n)) / Math.pow(TEN, n);
	}

	public void setUpStats() {
		if (dataValues.isEmpty()) {
			statLabels[0].setText("Mean: ");
```

```
            statLabels[1].setText("Range: ");
            statLabels[2].setText("Standard Deviation: ");
            statLabels[3].setText("Variance: ");
            statLabels[4].setText("Median: ");
        } else {
            double maxData = dataValues.get(0);
            double minData = dataValues.get(0);
            double mean = 0;
            for (int i = 0; i < dataValues.size(); i++) {
                mean += dataValues.get(i);
                maxData = Math.max(maxData, dataValues.get(i));
                minData = Math.min(minData, dataValues.get(i));
            }
            mean = mean / dataValues.size();
            statLabels[0].setText("Mean: " + round(mean));
            statLabels[1].setText("Range: " + (maxData - minData));
            double standardDeviation = 0;
            for (int i = 0; i < dataValues.size(); i++) {
                standardDeviation += Math.pow(dataValues.get(i) - mean, 2);
            }
            standardDeviation = Math.sqrt(standardDeviation / dataValues.size());
            statLabels[2].setText("Standard Deviation: " + round(standardDeviation));
            statLabels[3].setText("Variance: " + round(Math.pow(standardDeviation, 2)));

            ArrayList<Double> a = new ArrayList<Double>();
            for (int i = 0; i < dataValues.size(); i++) {
                a.add(dataValues.get(i));
            }
            bubbleSort(a);
            double median = 0;
            if (a.size() == 1)
                median = a.get(0);
            else if (a.size() % 2 == 0) {
                median = (a.get(a.size() / 2) + a.get(a.size() / 2 - 1)) / 2;
            } else if (a.size() % 2 == 1) {
                median = a.get(a.size() / 2);
            }
            statLabels[4].setText("Median: " + median);
        }
    }

    public void bubbleSort(ArrayList<Double> data) {
        boolean swapped = true;
        while (swapped) {
            swapped = false;
            for (int i = 1; i < data.size(); i++) {
                if (data.get(i - 1) > data.get(i)) {
                    double temp = data.get(i);
                    data.set(i, data.get(i - 1));
                    data.set(i - 1, temp);
                    swapped = true;
                }
            }
        }
    }

    public double factorial(long x) {
        if (x == 0)
            return 1;
        else
            return x * factorial(x - 1);
    }

    public void writeArrayInText() {
        try {
            BufferedWriter bw = new BufferedWriter(new FileWriter(FILE));
            for (int i = 0; i < dataValues.size(); i++) {
                bw.write(("Data Item " + (i + 1) + " is: " + dataValues.get(i) + "\n"));
            }
            bw.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    public void setArrayFromText() {
        try {
            BufferedReader br = new BufferedReader(new FileReader(FILE));
            String s;
            dataValues.clear();
            while ((s = br.readLine()) != null) {
                int index = s.indexOf(':');
                s = s.substring(index + 1).trim();
                dataValues.add(Double.parseDouble(s));
            }
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (Exception e) {

        }
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == textField) {
            try {
                dataValues.add(Double.parseDouble(textField.getText()));
                textAreaString = "";
                for (int i = 0; i < dataValues.size(); i++) {
                    textAreaString += ("Data Item " + (i + 1) + " is: " + dataValues.get(i) + "\n\n");
                }
                textArea.setText(textAreaString);
                writeArrayInText();
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(null, "The input is not a real number.");
            }
				textField.setText("");
				setUpStats();
			}
			if (e.getSource() == removeButton) {
				String input = JOptionPane.showInputDialog("Remove The Data Item With Index Number Of: ");
				try {
					int t = Integer.parseInt(input);
					if (t < 0)
						JOptionPane.showMessageDialog(null, "The input is not a positive integer.");
					else if (t > dataValues.size())
						JOptionPane.showMessageDialog(null, "There is no term number with an index as large as " +
t + ".");
					else {
						dataValues.remove(t - 1);
						setUpStats();
						textAreaString = "";
						for (int i = 0; i < dataValues.size(); i++) {
							textAreaString += ("Data Item " + (i + 1) + " is: " + dataValues.get(i) + "\n\n");
						}
						textArea.setText(textAreaString);
					}
				} catch (Exception ex) {
				}
				writeArrayInText();
			}
			if (e.getSource() == removeAll) {
				dataValues.clear();
				setUpStats();
				textAreaString = "";
				textArea.setText(textAreaString);
				writeArrayInText();
			}
			if (e.getSource() == sortButton) {
				bubbleSort(dataValues);
				textAreaString = "";
				for (int i = 0; i < dataValues.size(); i++) {
					textAreaString += ("Data item " + (i + 1) + " is: " + dataValues.get(i) + "\n\n");
				}
				textArea.setText(textAreaString);
				writeArrayInText();
			}
			if (e.getSource() == binomDisText[0]) {
				try {
					int t = Integer.parseInt(binomDisText[0].getText());
					if (t < 0) {
						JOptionPane.showMessageDialog(null, "The input is not a positive integer.");
						validNumberOfTrials = false;
						trialsSetValidLabel(validNumberOfTrials);
					} else {
						numberOfTrials = t;
						validNumberOfTrials = true;
						trialsSetValidLabel(validNumberOfTrials);
						if (validNumberOfTrials && validProbabilityOfSuccess && validRandomVariable) {
```

```
                                                updateBinom();
                                        }
                                }

                        } catch (Exception ex) {
                                JOptionPane.showMessageDialog(null, "The input is not a positive integer.");
                                validNumberOfTrials = false;
                                trialsSetValidLabel(validNumberOfTrials);
                        }
                }
                if (e.getSource() == binomDisText[1]) {
                        try {
                                double prob = Double.parseDouble(binomDisText[1].getText());
                                if (prob < 0 || prob > 1) {
                                        JOptionPane.showMessageDialog(null, "The input is within the range from 0 to 1 inclusive.");
                                        validProbabilityOfSuccess = false;
                                        pSetValidLabel(validProbabilityOfSuccess);
                                } else {
                                        probabilityOfSuccess = prob;
                                        validProbabilityOfSuccess = true;
                                        pSetValidLabel(validProbabilityOfSuccess);
                                        if (validNumberOfTrials && validProbabilityOfSuccess && validRandomVariable) {
                                                updateBinom();
                                        }
                                }

                        } catch (Exception ex) {
                                JOptionPane.showMessageDialog(null, "The input is not a real number.");
                                validProbabilityOfSuccess = false;
                                pSetValidLabel(validProbabilityOfSuccess);
                        }
                }
                if (e.getSource() == binomDisText[2]) {
                        try {
                                int t = Integer.parseInt(binomDisText[2].getText());
                                if (!validNumberOfTrials) {
                                        JOptionPane.showMessageDialog(null, "Input the number of trials first.");
                                        validRandomVariable = false;
                                        randomVariableSetValidLabel(validRandomVariable);
                                } else {
                                        if (t < 0) {
                                                JOptionPane.showMessageDialog(null, "The input is not a positive integer.");
                                                validRandomVariable = false;
                                                randomVariableSetValidLabel(validRandomVariable);
                                        } else if (t > numberOfTrials) {
                                                JOptionPane.showMessageDialog(null, "The input cannot be greater than the
number of trials.");
                                                validRandomVariable = false;
                                                randomVariableSetValidLabel(validRandomVariable);
                                        } else {
                                                randomVariable = t;
                                                validRandomVariable = true;
                                                randomVariableSetValidLabel(validRandomVariable);
                                                if (validNumberOfTrials && validProbabilityOfSuccess && validRandomVariable) {
                                                        updateBinom();
                                                }
                                        }
                                }

                        } catch (Exception ex) {
                                JOptionPane.showMessageDialog(null, "The input is not a positive integer.");
                                validRandomVariable = false;
                                randomVariableSetValidLabel(validRandomVariable);
                        }
                }
        }
}
```

```
public class StringStack {
        public int maxCapacity=100;
        public String arr[]= new String[maxCapacity];
        public int top = -1;
        public boolean isFull() {
                return top ==maxCapacity-1;
        }
        public boolean isEmpty() {
                return top ==-1;
        }
        public String peek() {
                if (!isEmpty()) {
                        return arr[top];
                }
                return null;

        }
        public String pop() {
                if (!isEmpty()) {
                        String str = arr[top];
                        top--;
                        return str;
                }
                return null;

        }
        public void add(String s) {
                if (!isFull()) {
                        top++;
                        arr[top]= s;
                }

        }
        public void print() {
                for (int i =0; i<=top; i++) {
```

```
                System.out.print(arr[i]+" ");
            }
            System.out.println();
        }

        public void clear() {
            arr = new String[maxCapacity];
            top = -1;
        }
}
```

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
public class
```

TrigPanel

```
extends JPanel implements ActionListener {
	private final String FONT = "Verdana";
	private final int TRIG_VALUE_FONT_SIZE = 15;

	private final int SPIN_BUTTON_X = 700;
	private final int SPIN_BUTTON_Y = 620;
	private final int SPIN_BUTTON_WIDTH = 150;
	private final int SPIN_BUTTON_HEIGHT = 70;

	private final int SPEED_UP_BUTTON_X = 100;
	private final int SPEED_UP_BUTTON_Y = 620;
	private final int SPEED_UP_BUTTON_WIDTH = 250;
	private final int SPEED_UP_BUTTON_HEIGHT = 70;

	private final int SLOW_DOWN_BUTTON_X = 400;
	private final int SLOW_DOWN_BUTTON_Y = 620;
	private final int SLOW_DOWN_BUTTON_WIDTH = 250;
	private final int SLOW_DOWN_BUTTON_HEIGHT = 70;

	private final int RADIANS_BUTTON_X = 500;
	private final int RADIANS_BUTTON_Y = 850;
	private final int RADIANS_BUTTON_WIDTH = 300;
	private final int RADIANS_BUTTON_HEIGHT = 70;

	private final int DEGREES_BUTTON_X = 200;
	private final int DEGREES_BUTTON_Y = 850;
	private final int DEGREES_BUTTON_WIDTH = 300;
	private final int DEGREES_BUTTON_HEIGHT = 70;

	private final int TRIG_VALUES_X = 25;
	private final int TRIG_VALUES_Y = 740;
	private final int TRIG_VALUES_WIDTH = 230;
	private final int TRIG_VALUES_HEIGHT = 30;
	private final int TRIG_VALUES_CHANGE_IN_WIDTH = 240;
	private final int TRIG_VALUES_CHANGE_IN_HEIGHT = 20;

	private final int DEGREES_IN_PI_RADIANS = 180;
	private final int RADIUS = 200;
	private final double FOUR_DECIMAL_PLACES = 10000;
	private final int ARROW_LENGTH = 50;
	private final int ARROW_WIDTH = 20;

	private boolean spin = true;
	private double dSpin = 0.001;
	private int panelWidth;
	private int panelHeight;
	private double angle = 0;
	private JButton spinButton = new JButton();
	private JButton speedUp = new JButton();
	private JButton slowDown = new JButton();
	private JButton inputAngleRad = new JButton();
	private JButton inputAngleDeg = new JButton();
	private JLabel[][] trigValues = new JLabel[2][4];
	private Image backgroundImage = new ImageIcon("src\\Images\\TrigPanel\\background.png").getImage();
	private Image boxImage = new ImageIcon("src\\Images\\TrigPanel\\box.png").getImage();
	private Image spinImage = new ImageIcon("src\\Images\\TrigPanel\\spin.png").getImage();
	private Image slowDownImage = new ImageIcon("src\\Images\\TrigPanel\\slow down.png").getImage();
	private Image speedUpImage = new ImageIcon("src\\Images\\TrigPanel\\speed up.png").getImage();
	private Image radiansImage = new ImageIcon("src\\Images\\TrigPanel\\radians.png").getImage();
	private Image degreesImage = new ImageIcon("src\\Images\\TrigPanel\\degrees.png").getImage();

	public TrigPanel(int w, int h) {
		panelWidth = w;
		panelHeight = h;
		this.setBackground(Color.gray);
		this.setLayout(null);
		this.setSize(w, h);
		spinButton.setBounds(SPIN_BUTTON_X, SPIN_BUTTON_Y, SPIN_BUTTON_WIDTH, SPIN_BUTTON_HEIGHT);
		spinButton.setIcon(new ImageIcon(spinImage.getScaledInstance(SPIN_BUTTON_WIDTH, SPIN_BUTTON_HEIGHT,
java.awt.Image.SCALE_SMOOTH)));
		spinButton.addActionListener(this);
		this.add(spinButton);

		speedUp.setIcon(new ImageIcon(speedUpImage.getScaledInstance(SPEED_UP_BUTTON_WIDTH,
SPEED_UP_BUTTON_HEIGHT, java.awt.Image.SCALE_SMOOTH)));
		speedUp.setBounds(SPEED_UP_BUTTON_X, SPEED_UP_BUTTON_Y, SPEED_UP_BUTTON_WIDTH,
SPEED_UP_BUTTON_HEIGHT);
		speedUp.addActionListener(this);
		this.add(speedUp);

		slowDown.setBounds(SLOW_DOWN_BUTTON_X, SLOW_DOWN_BUTTON_Y, SLOW_DOWN_BUTTON_WIDTH,
SLOW_DOWN_BUTTON_HEIGHT);
		slowDown.setIcon(new ImageIcon(slowDownImage.getScaledInstance(SLOW_DOWN_BUTTON_WIDTH,
SLOW_DOWN_BUTTON_HEIGHT,		java.awt.Image.SCALE_SMOOTH)));
		slowDown.addActionListener(this);
		this.add(slowDown);

		inputAngleRad.setBounds(RADIANS_BUTTON_X, RADIANS_BUTTON_Y, RADIANS_BUTTON_WIDTH,
RADIANS_BUTTON_HEIGHT);
		inputAngleRad.setIcon(new ImageIcon(radiansImage.getScaledInstance(RADIANS_BUTTON_WIDTH,
RADIANS_BUTTON_HEIGHT, java.awt.Image.SCALE_SMOOTH)));
		inputAngleRad.addActionListener(this);
		this.add(inputAngleRad);

		inputAngleDeg.setBounds(DEGREES_BUTTON_X, DEGREES_BUTTON_Y, DEGREES_BUTTON_WIDTH,
DEGREES_BUTTON_HEIGHT);
		inputAngleDeg.setIcon(new ImageIcon(degreesImage.getScaledInstance(DEGREES_BUTTON_WIDTH,
DEGREES_BUTTON_HEIGHT,java.awt.Image.SCALE_SMOOTH)));
		inputAngleDeg.addActionListener(this);
		this.add(inputAngleDeg);

		setUpLabels();
	}

	public void setUpLabels() {
		for (int i = 0; i < trigValues.length; i++) {
				for (int z = 0; z < trigValues[i].length; z++) {
					trigValues[i][z] = new JLabel();
					trigValues[i][z].setBounds(TRIG_VALUES_X + TRIG_VALUES_CHANGE_IN_WIDTH * z, TRIG_VALUES_Y
+ TRIG_VALUES_CHANGE_IN_HEIGHT * i, TRIG_VALUES_WIDTH, TRIG_VALUES_HEIGHT);
					trigValues[i][z].setBackground(Color.white);
					trigValues[i][z].setForeground(Color.black);
					trigValues[i][z].setFont(new Font(FONT, Font.PLAIN, TRIG_VALUE_FONT_SIZE));
					this.add(trigValues[i][z]);
				}
			}
		}

		public void paint(Graphics g) {
			super.paint(g);
			Graphics2D g2d = (Graphics2D) g;
			g2d.translate(panelWidth / 2, panelHeight / 3);
			if (spin)
				angle += dSpin;
			if (angle > 2 * Math.PI) {
				angle -= 2 * Math.PI;
			}
			g2d.setStroke(new BasicStroke(3));
			g2d.setColor(Color.black);
			g.drawImage(boxImage, -RADIUS - 2 * ARROW_LENGTH, -RADIUS - 2 * ARROW_LENGTH + 20, 2 * RADIUS + 4 *
ARROW_LENGTH, 2 * RADIUS + 4 * ARROW_LENGTH - 40, this);
			g2d.drawOval(-RADIUS, -RADIUS, RADIUS * 2, RADIUS * 2);
			g2d.drawLine(-RADIUS - ARROW_LENGTH, 0, RADIUS + ARROW_LENGTH, 0);
			g2d.drawLine(0, -RADIUS - ARROW_LENGTH, 0, RADIUS + ARROW_LENGTH);
			g2d.setColor(Color.red);
			g2d.drawLine(0, 0, (int) (RADIUS * Math.cos(-angle)), (int) (RADIUS * Math.sin(-angle)));
			g2d.drawLine((int) (RADIUS * Math.cos(-angle)), 0, (int) (RADIUS * Math.cos(-angle)),(int) (RADIUS *
Math.sin(-angle)));
			g2d.drawArc(-RADIUS / 4, -RADIUS / 4, RADIUS / 2, RADIUS / 2, 0, (int) radToDeg(angle));
			g2d.setColor(Color.black);
			g2d.drawLine(-RADIUS - ARROW_LENGTH, 0, -RADIUS - ARROW_LENGTH / 2, ARROW_WIDTH);
			g2d.drawLine(-RADIUS - ARROW_LENGTH, 0, -RADIUS - ARROW_LENGTH / 2, -ARROW_WIDTH);
			g2d.drawLine(RADIUS + ARROW_LENGTH, 0, RADIUS + ARROW_LENGTH / 2, ARROW_WIDTH);
			g2d.drawLine(RADIUS + ARROW_LENGTH, 0, RADIUS + ARROW_LENGTH / 2, -ARROW_WIDTH);
			g2d.drawLine(0, RADIUS + ARROW_LENGTH, ARROW_WIDTH, RADIUS + ARROW_LENGTH / 2);
			g2d.drawLine(0, RADIUS + ARROW_LENGTH, -ARROW_WIDTH, RADIUS + ARROW_LENGTH / 2);
			g2d.drawLine(0, -RADIUS - ARROW_LENGTH, ARROW_WIDTH, -RADIUS - ARROW_LENGTH / 2);
			g2d.drawLine(0, -RADIUS - ARROW_LENGTH, -ARROW_WIDTH, -RADIUS - ARROW_LENGTH / 2);

			trigValues[0][0].setText("Angle = " + round(angle) + " radians");
			trigValues[1][0].setText("Angle = " + round(radToDeg(angle)) + " degrees");
			trigValues[0][1].setText("sin(" + round(angle) + ") = " + round(Math.sin(angle)));
			trigValues[0][2].setText("cos(" + round(angle) + ") = " + round(Math.cos(angle)));
			trigValues[0][3].setText("tan(" + round(angle) + ") = " + round(Math.tan(angle)));
			trigValues[1][1].setText("csc(" + round(angle) + ") = " + round(1 / Math.sin(angle)));
			trigValues[1][2].setText("sec(" + round(angle) + ") = " + round(1 / Math.cos(angle)));
			trigValues[1][3].setText("cot(" + round(angle) + ") = " + round(1 / Math.tan(angle)));

			repaint();
```

```
}

public void paintComponent(Graphics g) {
		super.paintComponent(g);
		g.drawImage(backgroundImage, 0, 0, panelWidth, panelHeight, this);
}

public double radToDeg(double x) {
		return DEGREES_IN_PI_RADIANS * x / Math.PI;
}

public double degToRad(double x) {
		return Math.PI * x / DEGREES_IN_PI_RADIANS;
}

public double round(double x) {
		return Math.round(x * FOUR_DECIMAL_PLACES) / FOUR_DECIMAL_PLACES;
}

public void actionPerformed(ActionEvent e) {
		if (e.getSource() == spinButton) {
				spin = !spin;
		}

		if (e.getSource() == speedUp) {
				dSpin = dSpin * 2;
				spin = true;
		}

		if (e.getSource() == slowDown) {
				dSpin = dSpin / 2;
				spin = true;
		}

		if (e.getSource() == inputAngleRad) {
				String input = JOptionPane.showInputDialog("Please input the value of the angle in radians.");
				try {
						angle = Double.parseDouble(input);
						spin = false;

				} catch (Exception ex) {
						JOptionPane.showMessageDialog(null, "The input is not an real number.");
				}
		}

		if (e.getSource() == inputAngleDeg) {
				String input = JOptionPane.showInputDialog("Please input the value of the angle in degrees.");
				try {
						angle = degToRad(Double.parseDouble(input));
						spin = false;

				} catch (Exception ex) {
```

```
                                        JOptionPane.showMessageDialog(null, "The input is not an real number.");
                              }
                    }
          }
}
```