```
package main.CourseWork:
                                                                     }
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Point;
                                                                     public int getX(Link 1) {
                                                                         for (Link link : output) {
import java.awt.Rectangle;
                                                                              if (link == 1) {
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
                                                                                  return getX() + w;
import java.awt.event.MouseMotionListener;
import java.util.ArrayList;
import javax.swing.JComponent;
                                                                         return getX();
public class Component extends JComponent implements
                                                                     public int getY(Link 1) {
MouseMotionListener,
                                                                          for (Link link : output) {
        MouseListener {
                                                                              if (1 == link) {
                                                                                  return getY() + pinH;
    protected DrawPanel drawPanel;
    protected int numOfInputs;
    protected boolean not;
                                                                          for (int i = 0; i < numOfInputs; i++) {
    protected boolean or;
                                                                              if (input[i] == 1) {
    protected final int w = 75;
                                                                                  return getY() + pinH + i * pinH;
    protected int h;
    protected final int d = 3;
    protected final int dx = w / 4;
                                                                          return -1;
    protected final int pinH = 20;
                                                                     }
    protected ArrayList<Link> output = new
ArrayList();
                                                                     protected void removeLink(Link 1) {
                                                                         if (!output.remove(1)) {
    protected Link[] input;
    protected Point p;
                                                                              for (int i = 0; i < input.length; i++) {
    protected static Component component;
                                                                                  if (input[i] == 1) {
    protected int index = -2;
                                                                                      input[i] = null;
                                                                                      break;
    public Component(DrawPanel drawPanel, int
                                                                                  }
numOfInputs, boolean not, boolean or) {
                                                                              }
        this.drawPanel = drawPanel;
                                                                         }
        this.numOfInputs = numOfInputs;
        this.not = not;
        this.or = or;
                                                                     @Override
        input = new Link[numOfInputs];
                                                                     public void mouseDragged(MouseEvent e) {
                                                                          setLocation(getX() - (p.x -
        addMouseMotionListener(this);
        addMouseListener(this);
                                                                 e.getLocationOnScreen().x), getY() - (p.y -
        h = (numOfInputs + 1) * pinH;
                                                                 e.getLocationOnScreen().y));
        setBounds(0, h, w, h);
                                                                         p = e.getLocationOnScreen();
    }
                                                                          for (Link 1 : output) {
                                                                              1.repaint();
    @Override
                                                                          for (Link 1 : input) {
    public void paintComponent(Graphics g) {
        g.setColor(Color.WHITE);
                                                                              if (1 != null) {
        g.fillRect(dx, 0, w - dx * 2, h - d);
                                                                                  1.repaint();
        g.setColor(Color.BLACK);
        g.drawRect(dx, 0, w - dx * 2, h - d);
                                                                         }
        String text =
                                                                     }
        if (not) {
            text = "N";
                                                                     @Override
                                                                     public void mouseMoved(MouseEvent e) {
        if (or) {
            text += "OR";
                                                                     private int cont(Point p) {
        } else {
            text += "AND";
                                                                         if (new Rectangle(w - dx, pinH / 2, dx,
                                                                 pinH).contains(p)) {
        g.drawString(text, dx + d, pinH);
                                                                              return -1;
        if (index == -1) {
                                                                         int th = pinH / 2;
            g.setColor(Color.red);
                                                                         for (int i = 0; i < numOfInputs; i++) {
        g.drawLine(w - dx, pinH, w, pinH);
                                                                              if (new Rectangle(0, th, dx,
        if (index == -1) {
                                                                 pinH).contains(p)) {
            g.setColor(Color.black);
                                                                                  return i;
        if (not) {
                                                                              th += pinH;
            g.fillOval(w - dx - d * 2, pinH - d * 2,
d * 4, d * 4);
                                                                          if (component == this) {
                                                                              component = null;
        int th = pinH;
        for (int i = 0; i < numOfInputs; i++) {</pre>
                                                                          return -2;
            if (index == i) {
                g.setColor(Color.red);
                                                                     protected void setLink(Link 1) {
                                                                         if (index == -1) {
            g.drawLine(0, th, dx, th);
                                                                              output.add(1);
            if (index == -1) {
                 g.setColor(Color.black);
                                                                          } else if (index >= 0) {
                                                                              input[index] = 1;
            th += pinH;
```

```
index = -2:
                                                                             value = true:
        repaint();
    }
                                                                         for (Link 1 : input) {
                                                                             if (1.getFrom().getX() < getX()) {</pre>
    @Override
                                                                                 if (or) {
    public void mouseClicked(MouseEvent e) {
                                                                                     if (not) {
        if (e.getButton() == 3) {
                                                                                         value = value | !1.isValue();
            for (int i = 0; i < output.size(); i++) {
                                                                                     }else{
                output.get(i).removeLink();
                                                                                         value = value | l.isValue();
            for (int i = 0; i < numOfInputs; i++) {</pre>
                                                                                 } else {
                if (input[i] != null) {
                                                                                     if (not) {
                    input[i].removeLink();
                                                                                         value = value & !1.isValue();
                                                                                     }else{
                                                                                         value = value & 1.isValue();
            drawPanel.remove(this);
            drawPanel.repaint();
                                                                                 }
            return;
                                                                             }
                                                                         }
        index = cont(e.getPoint());
                                                                         return value;
        if (index != -2) {
            if (component == null) {
    if (index == -1) {
                                                                     public void setValue(boolean value) {
                     component = this;
                                                                         this.value = value;
                } else if (index >= 0) {
                    if (input[index] == null) {
                                                                }
                         component = this;
                                                                package main.CourseWork;
                    }
            } else if (component != this) {
                                                                import java.awt.Color;
                Link 1;
                                                                import java.awt.Graphics;
                if (index >= 0) {
                                                                import java.awt.Graphics2D;
                    1 = new Link(drawPanel,
                                                                import java.awt.Polygon;
component, this);
                                                                import java.awt.event.MouseEvent;
                } else {
                                                                import java.awt.event.MouseListener;
                    1 = new Link(drawPanel, this,
                                                                import java.awt.event.MouseMotionListener;
                                                                import java.util.PriorityQueue;
component):
                                                                import java.util.Queue;
                                                                import javax.swing.JComponent;
                component.setLink(1);
                setLink(1);
                drawPanel.add(1);
                                                                public class Link extends JComponent implements
                drawPanel.repaint();
                                                                MouseMotionListener,
                component = null;
                                                                         MouseListener {
                                                                    private static final Queue<Integer> ids = new
        repaint();
                                                                PriorityQueue<>();
    }
                                                                    private Component from, to;
                                                                    private DrawPanel drawPanel;
    @Override
    public void mousePressed(MouseEvent e) {
        p = e.getLocationOnScreen();
                                                                    private boolean test;
                                                                     private boolean value:
                                                                    private final int index;
    @Override
    public void mouseReleased(MouseEvent e) {
                                                                    private static int pollID() {
                                                                         if (ids.isEmpty()) {
                                                                             ids.offer(0);
    @Override
    public void mouseEntered(MouseEvent e) {
                                                                         int result = ids.poll();
                                                                         if (ids.isEmpty()) {
                                                                             ids.add(result + 1);
    @Override
    public void mouseExited(MouseEvent e) {
                                                                         return result;
                                                                    }
    public boolean isConnected() {
                                                                    public Link(DrawPanel drawPanel, Component from,
        if (output.isEmpty()) {
                                                                Component to) {
            return false;
                                                                         setLocation(0, 0);
                                                                         setSize(drawPanel.getSize());
        for (int i = 0; i < numOfInputs; i++) {</pre>
                                                                         addMouseListener(this);
            if (input[i] == null) {
                                                                         addMouseMotionListener(this);
                return false;
                                                                         this.from = from;
                                                                         this.to = to;
            }
                                                                         this.drawPanel = drawPanel;
                                                                         index = pollID();
        return true;
                                                                    }
    protected boolean value;
                                                                    @Override
                                                                    public void paintComponent(Graphics g) {
    public boolean isValue() {
        if (or) {
                                                                         Graphics2D g2 = (Graphics2D) g;
            value = false;
                                                                         if (!selected) {
                                                                             g2.setColor(Color.BLACK);
        } else {
```

```
if (new Polygon(xp, yp,
        } else {
            g2.setColor(Color.RED);
                                                               xp.length).contains(x, y)) {
                                                                                       return true;
        g2.drawLine(getX1Line(), getY1Line(),
}
getX2Line()) / 2, (getY1Line() + getY2Line()) / 2);
                                                                           return false;
                                                                       }
                                                                   }
    @Override
    public boolean contains(int x, int y) {
                                                                   @Override
        if (getX1Line() - DELTA < x && x <</pre>
                                                                   public void mouseClicked(MouseEvent me) {
                                                                       if (me.getButton() == 3) {
getX1Line() + DELTA && getY1Line() - DELTA < y && y <</pre>
getY1Line() + DELTA) {
                                                                           removeLink();
            return true;
        } else if (getX2Line() - DELTA < x && x <</pre>
                                                                   }
getX2Line() + DELTA && getY2Line() - DELTA < y && y <</pre>
getY2Line() + DELTA) {
                                                                   @Override
            return true;
                                                                   public void mousePressed(MouseEvent me) {
        } else {
            int minX = 0;
            int maxX = 0;
                                                                   @Override
            int minY = 0;
                                                                   public void mouseReleased(MouseEvent me) {
            int maxY = 0;
            if (getX1Line() > getX2Line()) {
                                                                   @Override
                minX = getX2Line();
                maxX = getX1Line();
                                                                   public void mouseEntered(MouseEvent me) {
                                                                       selected = true;
            } else {
                minX = getX1Line();
                                                                       repaint();
                maxX = getX2Line();
            if (getY1Line() > getY2Line()) {
                                                                   @Override
                minY = getY2Line();
                                                                   public void mouseExited(MouseEvent me) {
                maxY = getY1Line();
                                                                       selected = false;
            } else {
                                                                       repaint();
                minY = getY1Line();
                maxY = getY2Line();
                                                                   @Override
            if (getX1Line() < getX2Line()) {</pre>
                                                                   public void mouseDragged(MouseEvent me) {
                if (getY1Line() < getY2Line()) {</pre>
                    int[] xp = {minX - DELTA, minX -
DELTA, minX + DELTA, maxX + DELTA, maxX + DELTA, maxX
                                                                   @Override
- DELTA};
                                                                   public void mouseMoved(MouseEvent me) {
                    int[] yp = {minY + DELTA, minY -
DELTA, minY - DELTA, maxY - DELTA, maxY + DELTA, maxY
                                                                   private final int DELTA = 10;
                                                                   private boolean selected;
+ DELTA);
                    if (new Polygon(xp, yp,
xp.length).contains(x, y)) {
                                                                   private int getY1Line() {
                        return true;
                                                                       return from.getY(this);
                } else {
                    int[] xp = {minX + DELTA, minX -}
                                                                   private int getY2Line() {
DELTA, minX - DELTA, maxX - DELTA, maxX + DELTA, maxX
                                                                       return to.getY(this);
+ DELTA};
                    int[] yp = {maxY + DELTA, maxY +}
DELTA, maxY - DELTA, minY - DELTA, minY - DELTA, minY
                                                                   private int getX1Line() {
+ DELTA};
                                                                       return from.getX(this);
                    if (new Polygon(xp, yp,
xp.length).contains(x, y)) {
                        return true;
                                                                   private int getX2Line() {
                    }
                                                                       return to.getX(this);
                }
            } else {
                if (getY1Line() < getY2Line()) {</pre>
                                                                   public void removeLink() {
                    int[] xp = {minX + DELTA, minX -
                                                                       from.removeLink(this);
DELTA, minX - DELTA, maxX - DELTA, maxX + DELTA, maxX
                                                                       to.removeLink(this);
                                                                       ids.offer(index);
+ DELTA};
                    int[] yp = {maxY + DELTA, maxY +
                                                                       drawPanel.remove(this);
DELTA, maxY - DELTA, minY - DELTA, minY - DELTA, minY
                                                                       drawPanel.repaint();
+ DELTA};
                    if (new Polygon(xp, yp,
xp.length).contains(x, y)) {
                                                                   public Component getFrom() {
                        return true;
                                                                       return from;
                } else {
                    int[] xp = {minX - DELTA, minX -
                                                                   public Component getTo() {
DELTA, minX + DELTA, maxX + DELTA, maxX + DELTA, maxX
                                                                       return to;
- DELTA):
                    int[] yp = {minY + DELTA, minY -
DELTA, minY - DELTA, maxY - DELTA, maxY + DELTA, maxY
                                                                   public int getIndex() {
+ DELTA);
                                                                       return index;
```

```
if (e.getButton() == 3) {
                                                                            for(int i = 0; i < output.size(); i++){</pre>
    public void setValue(boolean value) {
        test = true;
                                                                                output.get(i).removeLink();
        this.value = value;
                                                                            for (int i = 0; i < numOfInputs; i++) {
                                                                                if (input[i] != null) {
    public boolean isValue() {
                                                                                    input[i].removeLink();
        if (test) {
            return value;
                                                                            drawPanel.remove(this);
        } else {
                                                                            drawPanel.repaint();
            return getFrom().isValue();
                                                                            return;
    public void reset(){
                                                                        index = cont(e.getPoint());
        test = false;
                                                                        if (index != -2) {
                                                                            if (component == null) {
                                                                                if (index == -1) {
    @Override
                                                                                    component = this;
    public String toString() {
        return "L" + index;
                                                                            } else if (component != this) {
                                                                                Link 1 = new Link(drawPanel, this,
                                                               component);
}
                                                                                component.setLink(1);
package main.CourseWork;
                                                                                setLink(1);
                                                                                drawPanel.add(1);
import java.awt.Color;
                                                                                drawPanel.repaint();
import java.awt.Graphics;
                                                                                component = null;
import java.awt.Point;
                                                                            }
import java.awt.Rectangle;
import java.awt.event.MouseEvent;
                                                                        repaint();
                                                                    }
public class InputPin extends Component {
                                                                    @Override
                                                                    public boolean isConnected() {
    public InputPin(DrawPanel drawPanel, String
caption) {
                                                                        if (output.isEmpty()) {
        super(drawPanel, 0, true, true);
                                                                            return false;
        this.caption = caption;
        addMouseMotionListener(this);
                                                                        return true;
        addMouseListener(this);
                                                                    }
        h = 2 * pinH;
        setBounds(0, h, w, h);
                                                                    @Override
                                                                    public boolean isValue() {
    }
                                                                        return value;
    @Override
    public void paintComponent(Graphics g) {
        g.setColor(Color.WHITE);
                                                                    @Override
        g.fillRect(0, 0, w - dx, h - d);
                                                                    public void setValue(boolean value) {
        g.setColor(Color.BLACK);
                                                                        this.value = value;
        g.drawRect(0, 0, w - dx, h - d);
                                                                    }
        g.drawString(caption, d, pinH);
        if (index == -1) {
                                                                    public String getCaption() {
            g.setColor(Color.red);
                                                                        return caption;
        g.drawLine(w - dx, pinH, w, pinH);
        if (index == -1) {
                                                               }
            g.setColor(Color.black);
                                                               package main.CourseWork;
    private String caption;
                                                                import java.awt.Color;
                                                                import java.awt.Graphics;
    private int cont(Point p) {
                                                                import java.awt.Point;
        if (new Rectangle(w - dx, pinH / 2, dx,
                                                                import java.awt.Rectangle;
pinH).contains(p)) {
                                                                import java.awt.event.MouseEvent;
            return -1;
                                                               public class OutputPin extends Component {
        if (component == this) {
            component = null;
                                                                    private String caption;
        return -2;
                                                                    public OutputPin(DrawPanel drawPanel, String
    }
                                                                caption) {
                                                                        super(drawPanel, 1, true, true);
    @Override
                                                                        this.drawPanel = drawPanel;
    protected void setLink(Link 1) {
                                                                        this.caption = caption;
        if (index == -1) {
                                                                        addMouseMotionListener(this);
            output.add(1);
                                                                        addMouseListener(this);
                                                                        h = 2 * pinH;
        index = -2;
                                                                        setBounds(0, h, w, h);
        repaint();
                                                                    }
    }
                                                                    @Override
    @Override
                                                                    public void paintComponent(Graphics g) {
```

public void mouseClicked(MouseEvent e) {

g.setColor(Color.WHITE);

```
g.fillRect(dx, 0, w - dx - d, h - d);
        g.setColor(Color.BLACK);
                                                                    @Override
        g.drawRect(dx, 0, w - dx - d, h - d);
                                                                    public void setValue(boolean value) {
        g.drawString(caption, dx + d, pinH);
                                                                        this.value = value;
        if (index == 0) {
            g.setColor(Color.red);
                                                                    public String getCaption() {
        g.drawLine(0, pinH, dx, pinH);
                                                                        return caption;
        if (index == 0) {
            g.setColor(Color.black);
    }
                                                                package main.CourseWork:
    private int cont(Point p) {
        if (new Rectangle(0, pinH / 2, dx,
                                                                import javax.swing.event.TableModelListener;
pinH).contains(p)) {
                                                                import javax.swing.table.TableModel;
            return 0;
                                                                public class TruthTableModel<T> implements TableModel
        if (component == this) {
            component = null;
                                                                    public TruthTableModel(T[][] matrix, String[]
        return -2;
                                                                names) {
    }
                                                                        this.matrix = matrix;
                                                                        this.names = names;
    @Override
                                                                    }
    protected void setLink(Link 1) {
        if (index == 0) {
                                                                    @Override
            input[index] = 1;
                                                                    public int getRowCount() {
                                                                        return matrix.length;
        index = -2;
        repaint();
    }
                                                                    @Override
                                                                    public int getColumnCount() {
    @Override
                                                                        return names.length;
    public void mouseClicked(MouseEvent e) {
        if (e.getButton() == 3) {
            for(int i = 0; i < output.size(); i++){</pre>
                                                                    @Override
                output.get(i).removeLink();
                                                                    public String getColumnName(int columnIndex) {
                                                                        return names[columnIndex];
            for (int i = 0; i < numOfInputs; i++) {</pre>
                if (input[i] != null) {
                    input[i].removeLink();
                                                                    @Override
                                                                    public Class<?> getColumnClass(int columnIndex) {
                                                                        if (matrix.length > 0) {
                                                                            if (matrix[0].length > 0) {
            drawPanel.remove(this);
            drawPanel.repaint();
                                                                                return matrix[0][0].getClass();
            return;
        index = cont(e.getPoint());
                                                                        return String.class;
        if (index != -2) {
            if (component == null) {
                if (index == 0) {
                                                                    @Override
                    if (input[index] == null) {
                                                                    public boolean isCellEditable(int rowIndex, int
                        component = this;
                                                                columnIndex) {
                                                                        return false;
            } else if (component != this) {
                Link 1 = new Link(drawPanel,
                                                                    @Override
component, this);
                                                                    public Object getValueAt(int rowIndex, int
                component.setLink(1);
                                                                columnIndex) {
                setLink(1);
                                                                        return matrix[rowIndex][columnIndex];
                drawPanel.add(1);
                drawPanel.repaint();
                                                                    @Override
                component = null;
                                                                    public void setValueAt(Object aValue, int
            }
                                                                rowIndex, int columnIndex) {
        }
        repaint();
                                                                    }
    @Override
                                                                    @Override
    public boolean isConnected() {
                                                                    public void
        for (int i = 0; i < numOfInputs; i++) {</pre>
                                                                addTableModelListener(TableModelListener 1) {
            if (input[i] == null) {
                                                                    }
                return false;
                                                                    @Override
                                                                    public void
                                                                removeTableModelListener(TableModelListener 1) {
        return true;
    }
                                                                    private T[][] matrix;
    @Override
                                                                    private String[] names;
    public boolean isValue() {
                                                                }
        return input[0].isValue();
                                                                package main.CourseWork;
```

```
return:
import java.util.ArrayList;
import java.util.Random;
                                                                            } else if (c instanceof Link) {
import javax.swing.JOptionPane;
                                                                                links.add((Link) c);
public class MainFrame extends javax.swing.JFrame {
                                                                        }
    public MainFrame() {
                                                                        Integer[][] matrix = new Integer[(int)
                                                                Math.pow(2, in.size())][in.size() + out.size()];
    private void
                                                                        String[] names = new String[in.size() +
addComponentButtonActionPerformed(java.awt.event.Acti
                                                                out.size()];
onEvent evt) {
                                                                        for (int i = 0; i < in.size(); i++) {
        int t = Math.abs(new
                                                                            names[i] = in.get(i).getCaption();
Integer(numOfInputsSpinner.getValue().toString()));
        if (t == 0) {
                                                                        for (int i = 0; i < out.size(); i++) {</pre>
            t = 1;
                                                                            names[i + in.size()] =
                                                                out.get(i).getCaption();
        drawPanel.add(new Component(drawPanel, t,
notCheckBox.isSelected();
                                                                        for (int i = 0; i < Math.pow(2, in.size());</pre>
orRadioButton.isSelected()));
                                                                i++) {
        drawPanel.repaint();
                                                                            String bin = Integer.toBinaryString(i);
    }
                                                                            boolean[] values = new
                                                                boolean[in.size()];
    private void
                                                                            int d = values.length - bin.length();
\verb|addInputButtonActionPerformed(java.awt.event.ActionEv|\\
                                                                            for (int j = bin.length() - 1; j >= 0; j-
ent evt) {
                                                                -) {
        if (!captionTextField.getText().isEmpty()) {
                                                                                if (bin.charAt(j) == '1') {
            drawPanel.add(new InputPin(drawPanel,
                                                                                     values[d + j] = true;
captionTextField.getText()));
            drawPanel.repaint();
                                                                            for (int j = 0; j < in.size(); j++) {
                                                                                in.get(j).setValue(values[j]);
    }
                                                                                if (values[j]) {
    private void
                                                                                     matrix[i][j] = 1;
addOutputButtonActionPerformed(java.awt.event.ActionE
                                                                                } else {
vent evt) {
                                                                                    matrix[i][j] = 0;
        if (!captionTextField.getText().isEmpty()) {
            drawPanel.add(new OutputPin(drawPanel,
                                                                            for (int j = 0; j < out.size(); j++) {
captionTextField.getText()));
                                                                                if (out.get(j).isValue()) {
            drawPanel.repaint();
        }
                                                                                    matrix[i][j + in.size()] = 1;
    }
                                                                                } else {
                                                                                    matrix[i][j + in.size()] = 0;
    private void
autoMenuItemActionPerformed(java.awt.event.ActionEven
                                                                            }
t evt) {
        ArrayList<Component> components = new
                                                                        truthTable.setModel(new
ArrayList<Component>();
                                                                TruthTableModel(matrix, names));
        ArrayList<InputPin> in = new
ArrayList<InputPin>();
        ArrayList<OutputPin> out = new
                                                                        Random r = new Random();
ArrayList<OutputPin>();
                                                                        int[][] faults = new
        ArrayList<Link> links = new
                                                                int[links.size()][links.size()];
ArrayList<Link>();
                                                                        for (int i = 0; i < links.size(); i++) {</pre>
        for (java.awt.Component c :
                                                                            for (int j = 0; j < Math.abs(r.nextInt())</pre>
                                                                % (links.size() / 2) + 1; j++) {
drawPanel.getComponents()) {
            if (c instanceof InputPin) {
                                                                                while (true) {
                in.add((InputPin) c);
                                                                                     int index = Math.abs(r.nextInt())
                if (!in.get(in.size() -
                                                                % links.size();
1).isConnected()) {
                                                                                     if (faults[i][index] == 0) {
                                                                                         if (Math.random() > 0.5) {
JOptionPane.showMessageDialog(this, "Error not
                                                                                             faults[i][index] = 1;
connected", "Error", JOptionPane.ERROR_MESSAGE);
                                                                                         } else {
                    return;
                                                                                             faults[i][index] = -1;
            } else if (c instanceof OutputPin) {
                                                                                         break;
                out.add((OutputPin) c);
                if (!out.get(out.size() -
                                                                                }
                                                                            }
1).isConnected()) {
JOptionPane.showMessageDialog(this, "Error not
                                                                        String[] faultsNames = {"N", "Faults"};
connected", "Error", JOptionPane.ERROR_MESSAGE);
                                                                        String[][] faultsMatrix = new
                    return;
                                                                String[links.size()][2];
                                                                        for (int i = 0; i < links.size(); i++) {</pre>
                                                                            faultsMatrix[i][0] = i + "";
            } else if (c instanceof Component) {
                components.add((Component) c);
                                                                            String value = "
                                                                            for (int j = 0; j < links.size(); j++) {
                if (!components.get(components.size()
                                                                                if (faults[i][j] > 0) {
- 1).isConnected()) {
                                                                                     value += links.get(j).toString()+
JOptionPane.showMessageDialog(this, "Error not
                                                                "/" + 1 +" ";
connected", "Error", JOptionPane.ERROR_MESSAGE);
                                                                                } else if (faults[i][j] < 0) {</pre>
```

```
for (int j = 0; j <
                     value += links.get(j).toString()+
"/" + 0 +" ";
                                                                 discrimination[0].length; j++) {
                                                                                      if (discrimination[i][j] > 0) {
                                                                                          count++;
            faultsMatrix[i][1] = value;
                                                                                          if (f.indexOf(j) != -1) {
                                                                                              need = true;
        faultTable.setModel(new
TruthTableModel(faultsMatrix, faultsNames));
                                                                                 if (count > max && need) {
        Integer[][] discrimination = new
                                                                                     max = count:
Integer[matrix.length][faults.length];
                                                                                      maxIndex = i;
        String[] discriminationNames = new
String[faults.length];
        for (int i = 0; i <
                                                                             if (m.indexOf(maxIndex) != -1 || max ==
discriminationNames.length; i++) {
                                                                0) {
            discriminationNames[i] = "" + i;
                                                                                 break:
        for (int i = 0; i < discrimination.length;</pre>
                                                                             m.add(maxIndex);
                                                                             result += maxIndex + " ";
i++) {
             for (int j = 0; j <
                                                                             for (Integer i = 0; i <
discrimination[0].length; j++) {
          boolean t = false;
                                                                discrimination[maxIndex].length; i++) {
                                                                                 if (discrimination[maxIndex][i] > 0)
                for (int k = 0; k < links.size();</pre>
k++) {
                                                                                      f.remove(i);
                     if (faults[j][k] > 0) {
                                                                                 }
                         links.get(k).setValue(true);
                                                                             }
                     } else if (faults[j][k] < 0) {</pre>
                                                                         JOptionPane.showMessageDialog(this, result,
                         links.get(k).setValue(false);
                                                                 "Result", JOptionPane.INFORMATION_MESSAGE);
                }
                                                                    }
                 for (int k = 0; k < in.size(); k++) {
                                                                     private void
                     if (matrix[i][k] > 0) {
                                                                manualMenuItemActionPerformed(java.awt.event.ActionEv
                         in.get(k).setValue(true);
                                                                 ent evt) {
                                                                         ArrayList<Component> components = new
                         in.get(k).setValue(false);
                                                                 ArrayList<Component>();
                                                                         ArrayList<InputPin> in = new
                                                                 ArrayList<InputPin>();
                 for (int k = 0; k < out.size(); k++)
                                                                         ArrayList<OutputPin> out = new
{
                                                                 ArrayList<OutputPin>();
                                                                         ArrayList<Link> links = new
                     if (out.get(k).isValue()) {
                                                                ArrayList<Link>();
                         if (matrix[i][k + in.size()]
== 0) {
                                                                         for (java.awt.Component c :
                                                                 drawPanel.getComponents()) {
                             t = true;
                                                                             if (c.getClass() == InputPin.class) {
                         }
                     } else {
                                                                                 in.add((InputPin) c);
                         if (matrix[i][k + in.size()]
                                                                                 if (!in.get(in.size() -
                                                                1).isConnected()) {
== 1) {
                             t = true;
                                                                 JOptionPane.showMessageDialog(this, "Error not
                                                                 connected", "Error", JOptionPane.ERROR_MESSAGE);
                     }
                                                                                      return;
                 for (int k = 0; k < links.size();</pre>
                                                                             } else if (c.getClass() ==
k++) {
                     links.get(k).reset();
                                                                 OutputPin.class) {
                                                                                 out.add((OutputPin) c);
                 if (t) {
                                                                                 if (!out.get(out.size() -
                     discrimination[i][j] = 1;
                                                                1).isConnected()) {
                 } else {
                                                                 JOptionPane.showMessageDialog(this, "Error not
                     discrimination[i][j] = 0;
                                                                 connected", "Error", JOptionPane.ERROR_MESSAGE);
                                                                                     return;
            }
        disctiminationTable.setModel(new
                                                                             } else if (c.getClass() ==
TruthTableModel(discrimination,
                                                                 Component.class) {
discriminationNames));
                                                                                 components.add((Component) c);
                                                                                 if (!components.get(components.size()
                                                                 - 1).isConnected()) {
        ArrayList<Integer> f = new ArrayList();
        for (int i = 0; i < faults.length; i++) {</pre>
                                                                 JOptionPane.showMessageDialog(this, "Error not
             f.add(i);
                                                                connected", "Error", JOptionPane.ERROR_MESSAGE);
                                                                                     return:
        String result = "";
        ArrayList<Integer> m = new ArrayList();
                                                                             } else if (c.getClass() == Link.class) {
        while (f.size() > 0) {
                                                                                 links.add((Link) c);
             int max = 0;
            int maxIndex = 0;
                                                                         }
            for (int i = 0; i <
discrimination.length; i++) {
                                                                         String[] fNames = new String[links.size()];
                int count = 0;
                                                                         for (int i = 0; i < fNames.length; <math>i++) {
                 boolean need = false;
                                                                             fNames[i] = links.get(i).toString();
```

```
for (int j = 0; j <
        MDialog md = new MDialog(this, true, fNames,
                                                                discrimination[0].length; j++) {
links.size());
                                                                                 boolean t = false;
                                                                                 for (int k = 0; k < links.size();</pre>
        md.setVisible(true);
        int[][] faults = md.getMatrix();
                                                                k++) {
        if (faults == null) {
                                                                                     if (faults[j][k] > 0) {
            return;
                                                                                         links.get(k).setValue(true);
                                                                                     } else if (faults[j][k] < 0) {</pre>
                                                                                         links.get(k).setValue(false);
        Integer[][] matrix = new Integer[(int)
                                                                                 }
Math.pow(2, in.size())][in.size() + out.size()];
        String[] names = new String[in.size() +
out.size()];
                                                                                 for (int k = 0; k < in.size(); k++) {
        for (int i = 0; i < in.size(); i++) {
                                                                                     if (matrix[i][k] > 0) {
            names[i] = in.get(i).getCaption();
                                                                                         in.get(k).setValue(true);
                                                                                     } else {
        for (int i = 0; i < out.size(); i++) {
                                                                                         in.get(k).setValue(false);
            names[i + in.size()] =
out.get(i).getCaption();
                                                                                 for (int k = 0; k < out.size(); k++)
        for (int i = 0; i < Math.pow(2, in.size());</pre>
i++) {
                                                                                     if (out.get(k).isValue()) {
            String bin = Integer.toBinaryString(i);
                                                                                         if (matrix[i][k + in.size()]
            boolean[] values = new
                                                                == 0) {
boolean[in.size()];
                                                                                             t = true;
                                                                                         }
            int d = values.length - bin.length();
            for (int j = bin.length() - 1; j >= 0; j-
                                                                                     } else {
-) {
                                                                                         if (matrix[i][k + in.size()]
                if (bin.charAt(j) == '1') {
                                                                == 1) {
                    values[d + j] = true;
                                                                                             t = true:
            for (int j = 0; j < in.size(); j++) {
                in.get(j).setValue(values[j]);
                                                                                 for (int k = 0; k < links.size();</pre>
                if (values[j]) {
                                                                k++) {
                    matrix[i][j] = 1;
                                                                                     links.get(k).reset();
                } else {
                                                                                 if (t) {
                    matrix[i][j] = 0;
                                                                                     discrimination[i][j] = 1;
                                                                                  else {
            }
            for (int j = 0; j < out.size(); j++) {</pre>
                                                                                     discrimination[i][j] = 0;
                if (out.get(j).isValue()) {
                    matrix[i][j + in.size()] = 1;
                                                                            }
                } else {
                                                                        disctiminationTable.setModel(new
                    matrix[i][j + in.size()] = 0;
                                                                TruthTableModel(discrimination,
                                                                discriminationNames));
                                                                        ArrayList<Integer> f = new ArrayList();
        truthTable.setModel(new
                                                                        for (int i = 0; i < faults.length; i++) {
TruthTableModel(matrix, names));
                                                                            f.add(i);
                                                                        String result = "";
        String[] faultsNames = {"N", "Faults"};
                                                                        ArrayList<Integer> m = new ArrayList();
        String[][] faultsMatrix = new
                                                                        while (f.size() > 0) {
String[links.size()][2];
                                                                            int max = 0;
        for (int i = 0; i < links.size(); i++) {</pre>
                                                                            int maxIndex = 0;
            faultsMatrix[i][0] = i + "";
                                                                            for (int i = 0; i <
                                                                discrimination.length; i++) {
            String value =
            for (int j = 0; j < links.size(); j++) {
                                                                                 int count = 0;
                if (faults[i][j] > 0) {
                                                                                 boolean need = false;
                                                                                 for (int j = 0; j <
                    value += links.get(j).toString()+
"/" + 1+" ";
                                                                discrimination[0].length; j++) {
                } else if (faults[i][j] < 0) {</pre>
                                                                                     if (discrimination[i][j] > 0) {
                                                                                         count++;
                    value += links.get(j).toString()+
"/" + 0+" ";
                                                                                         if (f.indexOf(j) != -1) {
                                                                                             need = true;
                }
            faultsMatrix[i][1] = value;
                                                                                 if (count > max && need) {
        faultTable.setModel(new
TruthTableModel(faultsMatrix, faultsNames));
                                                                                     max = count;
        Integer[][] discrimination = new
                                                                                     maxIndex = i;
Integer[matrix.length][faults.length];
        String[] discriminationNames = new
String[faults.length];
                                                                             if (m.indexOf(maxIndex) != -1 || max ==
        for (int i = 0; i <
                                                                0) {
discriminationNames.length; i++) {
                                                                                 break;
            discriminationNames[i] = "" + i;
                                                                            m.add(maxIndex);
                                                                            result += maxIndex + " ";
        for (int i = 0; i < discrimination.length;</pre>
i++) {
                                                                            for (Integer i = 0; i <
                                                                discrimination[maxIndex].length; i++) {
```

```
if (discrimination[maxIndex][i] > 0)
{
                    f.remove(i);
                }
            }
        JOptionPane.showMessageDialog(this, result,
"Result", JOptionPane.INFORMATION_MESSAGE);
    public static void main(String args[]) {
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new
Runnable() {
            public void run() {
                new MainFrame().setVisible(true);
        });
    // Variables declaration - do not modify
   private javax.swing.JButton addComponentButton;
   private javax.swing.JButton addInputButton;
   private javax.swing.JButton addOutputButton;
   private javax.swing.JRadioButton andRadioButton;
    private javax.swing.JMenuItem autoMenuItem;
    private javax.swing.JLabel captionLabel;
   private javax.swing.JTextField captionTextField;
   private javax.swing.ButtonGroup
componentTypeButtonGroup;
   private javax.swing.JScrollPane
discriminationTableScrollPane;
   private javax.swing.JTable disctiminationTable;
   private main.CourseWork.DrawPanel drawPanel;
   private javax.swing.JScrollPane drawScrollPane;
   private javax.swing.JTable faultTable;
   private javax.swing.JScrollPane
faultTableScrollPane;
   private javax.swing.JMenu generateTablesMenu;
   private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JToolBar.Separator
jSeparator3;
   private javax.swing.JToolBar.Separator
jSeparator4;
   private javax.swing.JToolBar.Separator
jSeparator5;
   private javax.swing.JTabbedPane jTabbedPane;
    private javax.swing.JMenuItem manualMenuItem;
   private javax.swing.JCheckBox notCheckBox;
   private javax.swing.JLabel numOfInputsLabel;
    private javax.swing.JSpinner numOfInputsSpinner;
   private javax.swing.JRadioButton orRadioButton;
    private javax.swing.JMenu testsMenu;
    private javax.swing.JToolBar toolBar;
   private javax.swing.JTable truthTable;
    private javax.swing.JScrollPane
truthTableScrollPane;
```