```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Arrays;
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileNameExtensionFilter;

public class BibFormatter extends javax.swing.JFrame {
    private static String[] end;
    private static String[] total;
    private static String[] annotations;
    private static int count;
    private static int count2;

        public static String[] nonAnnotatedOrganize() {
            BufferedReader reader;
            BufferedReader reader2;

            JFileChooser chooser = new JFileChooser();
            FileNameExtensionFilter filter = new FileNameExtensionFilter("Text
File", "txt");
            chooser.setFileFilter(filter);
            int returnVal = chooser.showOpenDialog(null);

            try{
                reader = new BufferedReader(new
FileReader(chooser.getSelectedFile()));
                reader2 = new BufferedReader(new
FileReader(chooser.getSelectedFile()));

                String line = reader2.readLine();
                count = 0;
                while (line != null) { //count how many lines there are
                    count++;
                    // read next line
                    line = reader2.readLine();
                 }

                end = new String[count];
                for(int i = 0; i < count; i++){ //put each source into an array
                    end[i] = reader.readLine() + "\n";

                }
            }
            catch (IOException e)
            {
                e.printStackTrace();
            }

            return end;
```

```
        }
        public static String[] annotatedOrganize() {
            BufferedReader reader;
            BufferedReader reader2;

            JFileChooser chooser = new JFileChooser();
            FileNameExtensionFilter filter = new FileNameExtensionFilter("Text
File", "txt");
            chooser.setFileFilter(filter);
            int returnVal = chooser.showOpenDialog(null);

            try{
                reader = new BufferedReader(new
FileReader(chooser.getSelectedFile()));
                reader2 = new BufferedReader(new
FileReader(chooser.getSelectedFile()));

                String line = reader2.readLine();
                count = 0;
                while (line != null) { //count how many lines there are
                    count++;
                    // read next line
                    line = reader2.readLine();
                 }

                end = new String[count/2];
                for(int i = 0; i < count/2; i++){ //put each source into an array
                    end[i] = reader.readLine() + ">" + reader.readLine();

                }


            }
            catch (IOException e)
            {
                e.printStackTrace();
            }

            return end;

        }
        public static String[] alphabetize() {
                count2 = end.length;
                for(int i = 0; i < count2; i++){ //removes quotation marks from each
source (because they mess up alphabetizing)
                        if(end[i].charAt(0) == '\"')
                            end[i] = end[i].substring(1);
                    }

                Arrays.sort(end); //alpabetize
```

```
                for(int i = 0; i < count2; i++){ //goes through all sources and adds
back quotation marks
                    int quotes = 0;
                    for(int j = 0; j < end[i].length(); j++){ //counts how many
quotations marks there are
                            if(end[i].charAt(j) == '\"')
                                quotes++;
                        }

                    if(quotes % 2 != 0) //if the amount of quotation marks are odd,
then there has to be
                                        //a missing one (the one in front) so it
adds that
                        end[i] = "\"" + end[i];
                    }

                return end;
                }

        public static String[] hanging() {
            for(int a = 0; a < count2; a++){
                for(int b = 85; b < end[a].length(); b+=85){
                    int c = 0;
                    while(c == 0){
                        if(end[a].charAt(b) == ' '){
                            end[a] = end[a].substring(0, b) + "\n" + "\t" +
end[a].substring(b + 1);
                            c++;
                        }
                        else
                            b--;
                    }
                }

            }
            return end;
        }
        public static String annotated(){
            BibFormatter.annotatedOrganize();

            BibFormatter.alphabetize();//organize just sources

            total = new String[count];
            annotations = new String[count/2];
            for(int i = 0; i < count/2; i++){
                for(int j = 0; j < end[i].length(); j++){
                    if(end[i].charAt(j) == '>'){
                        annotations[i] = "\t" +
end[i].substring(j+1,end[i].length());
```

```
                    end[i] = end[i].substring(0,j);
                    }
            }
        }

        BibFormatter.hanging();

        int lastSou = 0;
        int lastAnn = 0;



        for(int a = 0; a < count/2; a++){
            for(int b = 85; b < annotations[a].length(); b+=85){
                int c = 0;
                while(c == 0){
                    if(annotations[a].charAt(b) == ' '){
                        annotations[a] = annotations[a].substring(0, b) + "\n" +
"\t" + annotations[a].substring(b + 1);
                        c++;
                    }
                    else
                        b--;
                }
            }
        }

        for(int h = 0; h < count; h++){ //put sources and annotations back
together
            if(h % 2 != 0){
                total[h] = annotations[lastAnn];
                lastAnn++;
            }
            else {
                total[h] = end[lastSou];
                lastSou++;
            }
        }

        for (String bib:total){
                 System.out.println(bib + "\n");
                }
        return "";
    }

    public static String NotAnnotated(){
        BibFormatter.nonAnnotatedOrganize();
        BibFormatter.alphabetize();
        BibFormatter.hanging();
        for (String bib:end){
```

```
                    System.out.println(bib + "\n");
                   }
            return "";
        }
```

```
    //GUI
    public BibFormatter() {
        initComponents();
    }
        @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">

    private void initComponents() {

        FormatMyBibliography = new java.awt.Button();
        Annotated = new javax.swing.JCheckBox();
        jPanel1 = new javax.swing.JPanel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        jPanel1.setBackground(new java.awt.Color(100, 240, 240));
        setLocation(new java.awt.Point(600, 300));
        setTitle("BibFormatter");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 400, Short.MAX_VALUE)
        );

        FormatMyBibliography.setActionCommand("Select File");
        FormatMyBibliography.setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        FormatMyBibliography.setLabel("Format My Bibliography");
        FormatMyBibliography.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                FormatMyBibliographyActionPerformed(evt);
            }
        });

        Annotated.setText("Annotated");
        Annotated.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                AnnotatedActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(21, 21, 21)
                .addComponent(FormatMyBibliography,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(28, 28, 28)
                .addComponent(Annotated)
                .addContainerGap(28, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(21, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(FormatMyBibliography,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(Annotated))
                .addGap(21, 21, 21))
        );

        pack();
    }// </editor-fold>

    private void FormatMyBibliographyActionPerformed(java.awt.event.ActionEvent evt)
{
        if(Annotated.isSelected())
            BibFormatter.annotated();
        else
            BibFormatter.NotAnnotated();
    }

    private void AnnotatedActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
    }

    /**
```

```
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(BibFormatter.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(BibFormatter.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(BibFormatter.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(BibFormatter.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new BibFormatter().setVisible(true);
            }
        });
    }
     // Variables declaration - do not modify
    private javax.swing.JCheckBox Annotated;
    private java.awt.Button FormatMyBibliography;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration
```

```
}
```