Table of Contents

MainFrame.java 1
Tagger.java 19
Tuple.java 23

MainFrame.java

```
/*
 * MainFrame.java
 *
 * Created on 15-Apr-2010, 8:25:18 PM
 */

package CL_Book;

import com.google.api.translate.Language;
import com.google.api.translate.Translate;
import edu.stanford.nlp.util.Pair;
import java.awt.Color;
import java.awt.Component;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.Arrays;
import java.util.Iterator;
import java.util.List;
import java.util.TreeSet;
import java.util.Vector;
import javax.swing.ButtonGroup;
import javax.swing.ButtonModel;
import javax.swing.JComponent;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPopupMenu;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.JTextPane;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.text.BadLocationException;
import javax.swing.text.DefaultHighlighter;
import javax.swing.text.Highlighter;
import net.java.balloontip.BalloonTip;
import net.java.balloontip.CustomBalloonTip;
import net.java.balloontip.styles.RoundedBalloonStyle;

/**
 *
 * @author Dmitri Shuralyov and Ameeta Agrawal
 */
public class MainFrame extends JFrame
{
        private class MyPopupMenu extends JPopupMenu implements ActionListener
        {
                private JMenuItem translatePopupMenuItem;
                private JTextPane textPane;
                private String textToTranslate;

                // A custom popup menu class, used to display the right-click context menu
                MyPopupMenu()
                {
                        super();

                        translatePopupMenuItem = new JMenuItem("Translate");
                        translatePopupMenuItem.setMnemonic('T');
                        translatePopupMenuItem.addActionListener(this);
                }

                // Called to show the context menu
                @Override
                public void show(Component invoker, int x, int y) {
                        JPopupMenu popupMenu = new JPopupMenu();

                        textPane = (JTextPane)invoker;
                        int caretPosition = textPane.viewToModel(new Point(x, y));

                        if ((textPane.getSelectionStart() == textPane.getSelectionEnd())
                                        || !(caretPosition >= textPane.getSelectionStart() &&
caretPosition <= textPane.getSelectionEnd()))
                        {
                                textPane.setCaretPosition(caretPosition);
                        }

                        Tuple<Integer> range = expandRange(textPane.getText(),
textPane.getSelectionStart(), textPane.getSelectionEnd());
                        textPane.setSelectionStart(range.getFirst());
                        textPane.setSelectionEnd(range.getSecond());
                        try {
                                textToTranslate = textPane.getText(range.getFirst(),
range.getSecond() - range.getFirst());
                        } catch (BadLocationException e) {}

                        translatePopupMenuItem.setText("Translate to " +
languageButtonGroup.getSelection().getActionCommand());
                        //textPane.setSelectionStart() + " - " + textPane.getSelectionEnd() + "
=> " + range.getFirst() + "-" + range.getSecond());

                        if (textToTranslate.length() > 0) {
```

```
                        popupMenu.add(translatePopupMenuItem);
                        popupMenu.show(invoker, x, y);
                }
        }

        // Adds <br> line breaks to the given text
        private String addNewLines(String text)
        {
                final int LINE_WIDTH = 80;

                String output = "";
                //text.replaceAll(" ", "<br>");
                //System.err.println(">>>>>>>>>> " + text.indexOf("\n"));
                while (text.length() > LINE_WIDTH) {
                        int end = LINE_WIDTH;
                        while (end > 0 && isWordChar(text.charAt(end - 1))) {
                                --end;
                        }
                        int nextNewLine = text.indexOf("<br>");
                        if (nextNewLine != -1 && nextNewLine <= end)
                        {
                                output += text.substring(0, nextNewLine
+ "<br>".length());
                                text = text.substring(nextNewLine + "<br>".length());
                                continue;
                        }
                        output += text.substring(0, end) + "<br>";
                        text = text.substring(end);
                }
                return output + text;
        }

        public void actionPerformed(ActionEvent ae)
        {
                if (ae.getSource() == translatePopupMenuItem)
                {
                        // Balloon Tip
                        try {
                                Rectangle position = new Rectangle(0, 0, 0, 0);
                                position =
textPane.modelToView(textPane.getSelectionStart());
                                String translatedText = translate(textToTranslate,
Language.AUTO_DETECT, translateToLanguage, "<br>");
                                //System.out.println("'"+translatedText+"'");
                                showBalloonTip(textPane, position,
addNewLines(translatedText));
                        } catch (Exception e) {}
                }
        }

        private boolean isWordChar(char ch)
        {
                return !Character.isWhitespace(ch);
        }
        private Tuple<Integer> expandRange(String content, int start, int end)
        {
                while (start > 0 && isWordChar(content.charAt(start - 1))) {
                        --start;
                }
```

```
                        while (end < content.length() && isWordChar(content.charAt(end))) {
                                ++end;
                        }

                        return new Tuple<Integer>(start, end);
                }
        }

        private Tagger tagger = new Tagger();

        private File lastDirectory = null;            // Keeps track of the last current
directory
        //private File lastDirectory = new File("C:/Uni/5-2/cse6390E/Project");  // Keeps track
of the last current directory
        //private File lastDirectory = new File("E:/Uni/5-2/cse6390E/Project");  // Keeps track
of the last current directory

        private ButtonGroup modeButtonGroup = new ButtonGroup();
        private ButtonModel lastModeOption = null;
        private ButtonGroup languageButtonGroup = new ButtonGroup();
        private ButtonModel lastLanguageOption = null;

        private Language translateToLanguage = Language.ENGLISH;                // The language
to translate to

        private int pageNumber = 0;                                                  // Current
page number
        private Vector<String> bookPages = new Vector<String>();                // The loaded
book pages

        private MyPopupMenu popupMenu;

        private CustomBalloonTip balloonTip;

    /** Creates new form MainFrame */
    public MainFrame() {
        initComponents();

                setSidebarVisible(false);

                addLanguages();

                try {
                        loadLessons();
                        lessonsList.setSelectedIndex(0);
                } catch (Exception e) {
                        System.err.println("Error: Unable to load lessons description file (\"CL
Lessons.txt\").");
                        //e.printStackTrace();
                        JOptionPane.showMessageDialog(this, "Unable to load lessons description
file (\"CL Lessons.txt\").", "Error", JOptionPane.ERROR_MESSAGE);
                        grammarModeRadioButtonMenuItem.setEnabled(false);
                }

        lastModeOption = modeButtonGroup.getSelection();
                lastLanguageOption = languageButtonGroup.getSelection();

                Translate.setHttpReferrer("http://www.am.ca");    // Set the HTTP referrer to
some website address
```

```
                setLocationRelativeTo(null);

                displayPages();

                popupMenu = new MyPopupMenu();
                jTextPane1.setComponentPopupMenu(popupMenu);
                jTextPane2.setComponentPopupMenu(popupMenu);

                /*SimpleAttributeSet aSet = new SimpleAttributeSet();
        StyleConstants.setAlignment(aSet, StyleConstants.ALIGN_JUSTIFIED);
                StyledDocument doc = jTextPane1.getStyledDocument();
        doc.setParagraphAttributes(0, doc.getLength(), aSet, false);
                doc = jTextPane2.getStyledDocument();
        doc.setParagraphAttributes(0, doc.getLength(), aSet, false);*/

// DEBUG
/*try {
File file = new File("C:/Uni/5-2/cse6390E/Project/alice.txt");
BufferedReader br = new BufferedReader(new FileReader(file));
loadBook(br);
br.close();
} catch (Exception e) {}*/
    }

        // Sets the form size, either expanded with Grammar Lesson Sidebar or not
        private void setSidebarVisible(boolean visible)
        {
                //if (!visible) setSize(1169, getHeight());
                //else setSize(1499, getHeight());
                lessonsScrollPane.setVisible(visible);
                lessonDescriptionScrollPane.setVisible(visible);
                getContentPane().doLayout();
        }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        previousPageButton = new javax.swing.JButton();
        nextPageButton = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextPane1 = new javax.swing.JTextPane();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTextPane2 = new javax.swing.JTextPane();
        goToStartButton = new javax.swing.JButton();
        goToEndButton = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        lessonsScrollPane = new javax.swing.JScrollPane();
        lessonsList = new javax.swing.JList();
        lessonDescriptionScrollPane = new javax.swing.JScrollPane();
        lessonDescriptionTextPane = new javax.swing.JTextPane();
        goToPageButton = new javax.swing.JButton();
```

```
jMenuBar1 = new javax.swing.JMenuBar();
fileMenu = new javax.swing.JMenu();
openMenuItem = new javax.swing.JMenuItem();
closeMenuItem = new javax.swing.JMenuItem();
exitMenuItem = new javax.swing.JMenuItem();
readerModeMenu = new javax.swing.JMenu();
readingModeRadioButtonMenuItem = new javax.swing.JRadioButtonMenuItem();
grammarModeRadioButtonMenuItem = new javax.swing.JRadioButtonMenuItem();
translateModeRadioButtonMenuItem = new javax.swing.JRadioButtonMenuItem();
translateToLanguageMenu = new javax.swing.JMenu();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("CL Book");

previousPageButton.setText("Previous Page");
previousPageButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        previousPageButtonActionPerformed(evt);
    }
});

nextPageButton.setText("Next Page");
nextPageButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        nextPageButtonActionPerformed(evt);
    }
});

jTextPane1.setBackground(new java.awt.Color(251, 245, 229));
jTextPane1.setEditable(false);
jTextPane1.setFont(new java.awt.Font("Palatino Linotype", 0, 16));
jTextPane1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jTextPane1MouseClicked(evt);
    }
});
jScrollPane1.setViewportView(jTextPane1);

jTextPane2.setBackground(new java.awt.Color(251, 245, 229));
jTextPane2.setEditable(false);
jTextPane2.setFont(new java.awt.Font("Palatino Linotype", 0, 16));
jTextPane2.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jTextPane2MouseClicked(evt);
    }
});
jScrollPane2.setViewportView(jTextPane2);

goToStartButton.setText("|<");
goToStartButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        goToStartButtonActionPerformed(evt);
    }
});

goToEndButton.setText(">|");
goToEndButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        goToEndButtonActionPerformed(evt);
```

```
            }
        });

        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("Page 1 of 50");

        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel2.setText("Page 2 of 50");

        lessonsList.setFont(new java.awt.Font("Arial", 0, 12));
        lessonsList.setModel(new javax.swing.AbstractListModel() {
            String[] strings = { " None", " Lesson 1", "   Sublesson 1.1", "   Sublesson 1.2", "
Lesson 2", " Lesson 3", "   Sublesson 3.1", "   Sublesson 3.2", "   Sublesson 3.3", " Lesson
4" };
            public int getSize() { return strings.length; }
            public Object getElementAt(int i) { return strings[i]; }
        });
        lessonsList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        lessonsList.addListSelectionListener(new javax.swing.event.ListSelectionListener() {
            public void valueChanged(javax.swing.event.ListSelectionEvent evt) {
                lessonsListValueChanged(evt);
            }
        });
        lessonsScrollPane.setViewportView(lessonsList);

        lessonDescriptionTextPane.setContentType("text/html");
        lessonDescriptionTextPane.setEditable(false);
        lessonDescriptionTextPane.setText("<font
face=\"Tahoma\" size=3><strong>ADJECTIVES</strong>\n  <ol>\n    <li>Adjectives describe a
noun.</li>\n\n    <li>They do not change their form depending on the gender or number of the
noun.</li>\n\n    <li>Comparative Adjectives: To show adjective in the comparative form <i>more +
adjective</i>.</li>\n\n    <li>Superlative Adjectives: To show adjective in the superlative form
<i>most + adjective</i>.</li>\n  </ol>\n\n  <p><font color=\"#3399CC\">\"Show
Adjectives\"</font><br>\n  <font color=\"#3399CC\">\"Show Comparative Adjectives\"</font><br>\n
<font color=\"#3399CC\">\"Show�Superlative Adjectives\"</font><br></p></font>");
        lessonDescriptionScrollPane.setViewportView(lessonDescriptionTextPane);

        goToPageButton.setText("Go To Page...");
        goToPageButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                goToPageButtonActionPerformed(evt);
            }
        });

        fileMenu.setText("File");


openMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_O,
java.awt.event.InputEvent.CTRL_MASK));
        openMenuItem.setText("Open...");
        openMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                openMenuItemActionPerformed(evt);
            }
        });
        fileMenu.add(openMenuItem);
```

```
closeMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_W,
java.awt.event.InputEvent.CTRL_MASK));
        closeMenuItem.setText("Close");
        closeMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                closeMenuItemActionPerformed(evt);
            }
        });
        fileMenu.add(closeMenuItem);

        exitMenuItem.setText("Exit");
        exitMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                exitMenuItemActionPerformed(evt);
            }
        });
        fileMenu.add(exitMenuItem);

        jMenuBar1.add(fileMenu);

        readerModeMenu.setText("Mode");

        modeButtonGroup.add(readingModeRadioButtonMenuItem);
        readingModeRadioButtonMenuItem.setSelected(true);
        readingModeRadioButtonMenuItem.setText("Reading Mode");
        readingModeRadioButtonMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                modeButtonGroupActionPerformed(evt);
            }
        });
        readerModeMenu.add(readingModeRadioButtonMenuItem);

        modeButtonGroup.add(grammarModeRadioButtonMenuItem);
        grammarModeRadioButtonMenuItem.setText("Grammar Mode");
        grammarModeRadioButtonMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                modeButtonGroupActionPerformed(evt);
            }
        });
        readerModeMenu.add(grammarModeRadioButtonMenuItem);

        modeButtonGroup.add(translateModeRadioButtonMenuItem);
        translateModeRadioButtonMenuItem.setText("Translate Mode");
        translateModeRadioButtonMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                modeButtonGroupActionPerformed(evt);
            }
        });
        readerModeMenu.add(translateModeRadioButtonMenuItem);

        jMenuBar1.add(readerModeMenu);

        translateToLanguageMenu.setText("Translate To Language");
        jMenuBar1.add(translateToLanguageMenu);

        setJMenuBar(jMenuBar1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 497, Short.MAX_VALUE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 498, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(goToStartButton)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(previousPageButton)
                        .addGap(45, 45, 45)
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 121,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 130, Short.MAX_VALUE)
                        .addComponent(goToPageButton)
                        .addGap(125, 125, 125)
                        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 133,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(62, 62, 62)
                        .addComponent(nextPageButton)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(goToEndButton, javax.swing.GroupLayout.PREFERRED_SIZE, 45,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(lessonDescriptionScrollPane,
javax.swing.GroupLayout.DEFAULT_SIZE, 258, Short.MAX_VALUE)
                    .addComponent(lessonsScrollPane,
javax.swing.GroupLayout.DEFAULT_SIZE, 258, Short.MAX_VALUE))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 651, Short.MAX_VALUE)
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 651, Short.MAX_VALUE)
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                        .addComponent(lessonsScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 309, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(lessonDescriptionScrollPane,
javax.swing.GroupLayout.DEFAULT_SIZE, 336, Short.MAX_VALUE)))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(previousPageButton)
                    .addComponent(goToStartButton)
                    .addComponent(nextPageButton)
                    .addComponent(goToEndButton)
                    .addComponent(jLabel2)
                    .addComponent(jLabel1)
                    .addComponent(goToPageButton))
                .addContainerGap())
        );

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

        // Exit
    private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {

                this.dispose();
    }

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

                pageNumber -= (translateModeRadioButtonMenuItem.getModel() !=
modeButtonGroup.getSelection()) ? 2 : 1;
                if (pageNumber < 0) pageNumber = 0;
                displayPages();
        }

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

                performFileOpen();
        }

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

                pageNumber += (translateModeRadioButtonMenuItem.getModel() !=
modeButtonGroup.getSelection()) ? 2 : 1;
                while (pageNumber >= bookPages.size() && pageNumber > 0) pageNumber -= 2;
                displayPages();
        }

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

                pageNumber = 0;
                displayPages();
        }

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

                while (pageNumber < bookPages.size()) pageNumber += 2;
                if (pageNumber >= 2) pageNumber -= 2;
                displayPages();
        }

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

                closeBook();
                displayPages();
        }

        private void jTextPane1MouseClicked(java.awt.event.MouseEvent evt) {

                if (null != balloonTip) balloonTip.closeBalloon();
        }

        private void jTextPane2MouseClicked(java.awt.event.MouseEvent evt) {

                if (null != balloonTip) balloonTip.closeBalloon();
        }

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

                if (modeButtonGroup.getSelection() != lastModeOption) {
                        // Reader Mode Changed
                        if (readingModeRadioButtonMenuItem.getModel() ==
modeButtonGroup.getSelection())
                        {
                                setSidebarVisible(false);
                        }
                        else if (grammarModeRadioButtonMenuItem.getModel() ==
modeButtonGroup.getSelection())
                        {
                                setSidebarVisible(true);
                        }
                        else if (translateModeRadioButtonMenuItem.getModel() ==
modeButtonGroup.getSelection())
                        {
                                jTextPane2.setBackground(new Color(235, 251, 229));
                                setSidebarVisible(false);
                        }

                        if (translateModeRadioButtonMenuItem.getModel() == lastModeOption)
                        {
                                if (pageNumber % 2 == 1) --pageNumber;
                                jTextPane2.setBackground(new Color(251, 245, 229));
                        }

                        displayPages();

                        lastModeOption = modeButtonGroup.getSelection();
                }
        }

        private int previousLessonIndex = -1;
        private void lessonsListValueChanged(javax.swing.event.ListSelectionEvent evt) {

                if (previousLessonIndex != lessonsList.getSelectedIndex())
                {
                        if (-1 != lessonsList.getSelectedIndex())

lessonDescriptionTextPane.setText(lessonDescs.get(lessonsList.getSelectedIndex()));
                        else
                                lessonDescriptionTextPane.setText("");
                        displayPages();

                        previousLessonIndex = lessonsList.getSelectedIndex();
                }
        }

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

                Object answer = JOptionPane.showInputDialog(this, "Go to page...", "CL
Book", JOptionPane.PLAIN_MESSAGE, null, null, (1+pageNumber));

                if (null != answer)
                {
                        System.out.println("answer; " + answer);
                        try {
                                int newPageNumber = (-1) + Integer.valueOf(answer.toString());

                                if (newPageNumber >= 0 && newPageNumber < bookPages.size()) {
                                        pageNumber = newPageNumber;

                                        if (translateModeRadioButtonMenuItem.getModel() !=
modeButtonGroup.getSelection())
                                                if (pageNumber % 2 == 1) --pageNumber;

                                        displayPages();
                                }
                        } catch (Exception e) {}
                }
        }

        private void languageButtonGroupActionPerformed(ActionEvent evt) {
                if (languageButtonGroup.getSelection() != lastLanguageOption) {
                        //System.out.println("Language changed to " +
translateToLanguage.toString());
                        displayPages();

                        lastLanguageOption = languageButtonGroup.getSelection();
                }
        }

        // Creates a fresh new JFileChooser
        // This is needed to reset it if the user presses cancel, which doesn't happen in Java if
you
        // reuse the same instance
        // But that's how all other Windows applications behave, so I wanted it to be consistent
        private FileNameExtensionFilter txtFilter = new FileNameExtensionFilter("Text Files
(*.txt)", "txt");
        private FileFilter lastFileFilter = null;
        private JFileChooser createChooser(int chooserType)
        {
                JFileChooser chooser = new JFileChooser();

                if (lastDirectory != null)
                        chooser.setCurrentDirectory(lastDirectory);

                // This is needed to get the file filters in correct order and the text filter
active by default
                chooser.setAcceptAllFileFilterUsed(false);
                chooser.setFileFilter(txtFilter);
                chooser.setAcceptAllFileFilterUsed(true);
                if (lastFileFilter == null)
                        chooser.setFileFilter(txtFilter);                               // Set Text
Files filter selected by default
                else
                        chooser.setFileFilter(lastFileFilter);

                if (chooserType == JFileChooser.OPEN_DIALOG) {
                        // Nothing to do
```

```
		} else if (chooserType == JFileChooser.SAVE_DIALOG) {
			// Change the text from 'Save' to 'Save As'
			chooser.setDialogTitle("Save As");
			chooser.setApproveButtonText("Save As");
		}

		return chooser;
	}

	// Display and process an Open dialog
	private void performFileOpen() {
		JFileChooser chooser = createChooser(JFileChooser.OPEN_DIALOG);
		int response = chooser.showDialog(this, null);
		if (response == JFileChooser.APPROVE_OPTION) {
			File file = null;
			try {
				file = chooser.getSelectedFile();
				BufferedReader br = new BufferedReader(new FileReader(file));
				//jTextPane1.read(br, null);
				loadBook(br);
				br.close();
				lastFileFilter = chooser.getFileFilter();
				/*if (file.getName().toLowerCase().endsWith(".java")) {

highlightingButtonGroup.setSelected(noneMenuItem.getModel(), true);		// Force
fullCheckSpelling() to get executed

highlightingButtonGroup.setSelected(javaSourceCodeMenuItem.getModel(), true);
				} else {

highlightingButtonGroup.setSelected(noneMenuItem.getModel(), true);		// Force
fullCheckSpelling() to get executed

highlightingButtonGroup.setSelected(englishSpellcheckingMenuItem.getModel(), true);
				}*/
				//textPaneWasReset();

				//documentFile = file;
				//setDocumentHasUnsavedChanges(false);
				lastDirectory = file.getParentFile();
			} catch (java.io.FileNotFoundException fnfe) {
				System.err.println("Warning: File '" + file + "' not found.");
			} catch (Exception e) {
				e.printStackTrace();
				System.exit(1);
			}
		}
	}

	// Loads the given book into memory
	private void loadBook(BufferedReader br)
	{
		final int LINES_PER_PAGE = 24;

		closeBook();

		try {
			while (br.ready()) {
				String page = br.readLine();
```

```
                                for (int line = 0; line < LINES_PER_PAGE && br.ready(); ++line)
                                        page += "\n" + br.readLine().replaceAll("_", "");

                                bookPages.add(page);
                        }
                } catch (Exception e) { e.printStackTrace(); }

                displayPages();
        }

        // Closes the loaded book
        private void closeBook()
        {
                bookPages.clear();
                pageNumber = 0;
        }

        // This function is used to display
        private void displayPages()
        {
                if (null != balloonTip) balloonTip.closeBalloon();

                if (translateModeRadioButtonMenuItem.getModel() !=
modeButtonGroup.getSelection())
                {
                        if (pageNumber <
bookPages.size()) jTextPane1.setText(removeNewLines(bookPages.get(pageNumber)));
                        else jTextPane1.setText("");
                        if (pageNumber + 1 <
bookPages.size()) jTextPane2.setText(removeNewLines(bookPages.get(pageNumber + 1)));
                        else jTextPane2.setText("");

                        highlightWords(jTextPane1);
                        highlightWords(jTextPane2);

                        if (0 != bookPages.size())
                                if (1+pageNumber < bookPages.size()) {
                                        jLabel1.setText("Page " + (1+pageNumber) + " of " +
bookPages.size());
                                        jLabel2.setText("Page " + (2+pageNumber) + " of " +
bookPages.size());
                                } else {
                                        jLabel1.setText("Page " + (1+pageNumber) + " of " +
bookPages.size());
                                        jLabel2.setText("");
                                }
                        else {
                                jLabel1.setText("");
                                jLabel2.setText("");
                        }
                }
                else
                {
                        if (pageNumber <
bookPages.size()) jTextPane1.setText(removeNewLines(bookPages.get(pageNumber)));
                        else jTextPane1.setText("");
                        if (pageNumber <
bookPages.size()) jTextPane2.setText(translate(bookPages.get(pageNumber), Language.AUTO_DETECT,
translateToLanguage, "\n"));
```

```
                        else jTextPane2.setText("");

                        highlightWords(jTextPane1);
                        highlightWords(jTextPane2);

                        if (0 != bookPages.size()) {
                                jLabel1.setText("Page " + (1+pageNumber) + " of " +
bookPages.size());
                                jLabel2.setText("Page " + (1+pageNumber) + " of " +
bookPages.size());
                        } else {
                                jLabel1.setText("");
                                jLabel2.setText("");
                        }
                }

                nextPageButton.setEnabled(0 != bookPages.size());
                previousPageButton.setEnabled(0 != bookPages.size());
                goToStartButton.setEnabled(0 != bookPages.size());
                goToEndButton.setEnabled(0 != bookPages.size());
                goToPageButton.setEnabled(0 != bookPages.size());
        }

        // This is used to highlight words based on the currently selected grammar lesson
        private Highlighter.HighlightPainter painter
= new DefaultHighlighter.DefaultHighlightPainter(Color.yellow);
        private void highlightWords(JTextPane textPane)
        {
                try {
                        // Run the tagger on the text
                        tagger.preprocessPage(textPane.getText());
                } catch (Exception e) { e.printStackTrace(); }

                final Highlighter highlighter = textPane.getHighlighter();

                // Clear all of the previous highlighting
                Highlighter.Highlight[] highlights = highlighter.getHighlights();
                for (int i = 0; i < highlights.length; ++i) {
                        Highlighter.Highlight h = highlights[i];
                        //if (h.getPainter() instanceof Highlighter.HighlightPainter) {
                                highlighter.removeHighlight(h);
                        //}
                }

                // Lesson Highlighting
                if (grammarModeRadioButtonMenuItem.getModel() ==
modeButtonGroup.getSelection() && -1 != lessonsList.getSelectedIndex())
                {
                        String lessonId = lessonIds.get(lessonsList.getSelectedIndex());

                        List<Tuple> highlightedWords = tagger.getHighlightedWords(lessonId);
                        for (Iterator<Tuple> it1 = highlightedWords.iterator(); it1.hasNext(); )
                        {
                                Tuple<Integer> tuple = it1.next();
                                try {
                                        highlighter.addHighlight(tuple.getFirst(),
tuple.getSecond(), painter);
                                } catch (BadLocationException e) {}
                        }
                }
        }

        // Shows a balloon tip (used for translation)
        private void showBalloonTip(JComponent component, Rectangle position, String text)
        {
                if (null != balloonTip) balloonTip.closeBalloon();
                if (component == jTextPane2 && text.length() > 80)
                        position.x = 10;
                balloonTip = new CustomBalloonTip(component,
                                "<html><font face=\"Palatino Linotype\" size=4>" + text
+ "</font></html>",
                                position,

  new RoundedBalloonStyle(5, 5, new java.awt.Color(235, 251, 229) /*Color.yellow*/, Color.black),
                                BalloonTip.Orientation.LEFT_ABOVE,
                                BalloonTip.AttachLocation.ALIGNED,
                                15, 15,
                                false);
        }

        // Removes new lines from within paragraphs
        String removeNewLines(String text)
        {
                return translate(text, Language.ENGLISH, Language.ENGLISH, "\n");
        }

        // Uses Google Translate API to translate the given text from source to target language
        // newLine string is used to indicate what type of newline char is wanted (e.g. "\n" or
"<br>")
        String translate(String textToTranslate, Language languageFrom, Language
languageTo, String newLine)
        {
                // Break up the string into blocks (paragraphs)
                String block = "";
                Vector<String> blocks = new Vector<String>();
                String[] lines = textToTranslate.split("\n");
                for (int line = 0; line < (-1) + lines.length; ++line) {
                        if (lines[line].length() > 0)
                                block += lines[line];
                        else
                                { blocks.add(""); continue; }
                        if (lines[line+1].equals("")) {
                                blocks.add(block);
                                block = "";
                        } else if (lines[line+1].startsWith(" ")) {
                                block += "\n";
                        } else {
                                block += " ";
                        }
                }
                block += lines[lines.length - 1];
                blocks.add(block);

                String output = "";

                // Translate each one separately and combine them together
                for (Iterator<String> it1 = blocks.iterator(); it1.hasNext(); )
                {
                        block = it1.next();
                        //System.err.println("trying to translate: '" + block + "'");
                        String translatedBlock = block;
                        try {
                                if (block.length() > 0 && languageFrom != languageTo)
                                        translatedBlock = Translate.execute(block, languageFrom,
languageTo);
                        } catch (Exception e) {}
                        output += translatedBlock;
                        if (it1.hasNext()) output += newLine;
                }

                return output;
        }

        // Load the lessons from a lesson description file
        private Vector<String> lessonIds = new Vector<String>();
        private Vector<String> lessonDescs = new Vector<String>();
        private void loadLessons() throws Exception
        {
                final Vector<String> lessonNames = new Vector<String>();

                final String fileName = "CL Lessons.txt";
                BufferedReader br = new BufferedReader(new FileReader(fileName));
                while (br.ready())
                {
                        String lessonName = br.readLine();
                        String lessonId = br.readLine();
                        String lessonDesc = br.readLine();

                        for (String line = br.readLine(); !line.equals(""); line = br.readLine())
                        {
                                lessonDesc += " " + line;
                        }

                        lessonNames.add(lessonName);
                        lessonIds.add(lessonId);
                        lessonDescs.add(lessonDesc);
                }
                br.close();

                lessonsList.setModel(new javax.swing.AbstractListModel() {
            public int getSize() { return lessonNames.size(); }
            public Object getElementAt(int i) { return " " + lessonNames.get(i); }
        });
        }

        // Adds the supported languages to the Translate To Language menu
        private void addLanguages() {
                TreeSet<Pair<String, Language>> languages = new TreeSet(Arrays.asList(
                                new Pair<String, Language>("English", Language.ENGLISH),
                                new Pair<String, Language>("Arabic", Language.ARABIC),
                                new Pair<String, Language>("Chinese", Language.CHINESE),
                                new Pair<String, Language>("French", Language.FRENCH),
                                new Pair<String, Language>("German", Language.GERMAN),
                                new Pair<String, Language>("Hindi", Language.HINDI),
                                new Pair<String, Language>("Italian", Language.ITALIAN),
                                new Pair<String, Language>("Persian", Language.PERSIAN),
                                new Pair<String, Language>("Polish", Language.POLISH),
                                new Pair<String, Language>("Russian", Language.RUSSIAN),
                                new Pair<String, Language>("Belarusian", Language.BELARUSIAN),
                                new Pair<String, Language>("Serbian", Language.SERBIAN),
                                new Pair<String, Language>("Spanish", Language.SPANISH),
                                new Pair<String, Language>("Thai", Language.THAI),
                                new Pair<String, Language>("Swahili", Language.SWAHILI)));

                // Add them all to
                for (final Pair<String, Language> language : languages)
                {
                        JRadioButtonMenuItem languageRadioButtonMenuItem
= new JRadioButtonMenuItem();

                        languageButtonGroup.add(languageRadioButtonMenuItem);

  if (language.first().equals("English")) languageRadioButtonMenuItem.setSelected(true);
                        languageRadioButtonMenuItem.setText(language.first());
                        languageRadioButtonMenuItem.setActionCommand(language.first());

languageRadioButtonMenuItem.addActionListener(new java.awt.event.ActionListener() {
                                public void actionPerformed(java.awt.event.ActionEvent evt) {
                                        translateToLanguage = language.second();
                                        languageButtonGroupActionPerformed(evt);
                                }
                        });
                        translateToLanguageMenu.add(languageRadioButtonMenuItem);
                }
        }

        /**
        * @param args the command line arguments
        */
        public static void main(String args[]) {
                java.awt.EventQueue.invokeLater(new Runnable() {
                        public void run() {
                                // Use Win32 look and feel
                                try {

javax.swing.UIManager.setLookAndFeel(javax.swing.UIManager.getSystemLookAndFeelClassName());
                                } catch (Exception e) {}

                                new MainFrame().setVisible(true);
                        }
                });
        }

    // Variables declaration - do not modify
    private javax.swing.JMenuItem closeMenuItem;
    private javax.swing.JMenuItem exitMenuItem;
    private javax.swing.JMenu fileMenu;
    private javax.swing.JButton goToEndButton;
    private javax.swing.JButton goToPageButton;
    private javax.swing.JButton goToStartButton;
    private javax.swing.JRadioButtonMenuItem grammarModeRadioButtonMenuItem;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTextPane jTextPane1;
    private javax.swing.JTextPane jTextPane2;
    private javax.swing.JScrollPane lessonDescriptionScrollPane;
    private javax.swing.JTextPane lessonDescriptionTextPane;
    private javax.swing.JList lessonsList;
    private javax.swing.JScrollPane lessonsScrollPane;
    private javax.swing.JButton nextPageButton;
    private javax.swing.JMenuItem openMenuItem;
    private javax.swing.JButton previousPageButton;
    private javax.swing.JMenu readerModeMenu;
    private javax.swing.JRadioButtonMenuItem readingModeRadioButtonMenuItem;
    private javax.swing.JRadioButtonMenuItem translateModeRadioButtonMenuItem;
    private javax.swing.JMenu translateToLanguageMenu;
    // End of variables declaration
}
```

Tagger.java

```
package CL_Book;

import edu.stanford.nlp.tagger.maxent.MaxentTagger;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.StringTokenizer;

public class Tagger {

        String modelFile = "models/left3words-wsj-0-18.tagger";
        //String modelFile = "models/bidirectional-distsim-wsj-0-18.tagger";
        LinkedHashMap<Tuple, String> m = new LinkedHashMap<Tuple, String>();
        ArrayList<String> arrTags = new ArrayList<String>();
        ArrayList<Tuple> arrTuples = new ArrayList<Tuple>();


        /* method: preprocessPage
         * input: String of the text to be processed
         * output: it's a void method, doesn't return anything
         * description: creates an instance of the MaxentTagger from stanford-postagger.jar
library; tags the input text; tokenizes the tagged String into word and tags; computes the first
and last index of each word in the input text and stores it in a instance of a "Tuple"; stores
the Tuple and the tag in a map.
         */
        public void preprocessPage(String textBlock) throws Exception // Creates the word tagging
database
        {
                m.clear();

                int first = 0;
                int last = 0;

                //create instead of a Maxent Tagger; the modelFile is "left3words-wsj-0-
18.tagger"
                MaxentTagger tagger = new MaxentTagger(modelFile);
                @SuppressWarnings("unchecked")

                String taggedString = tagger.tagString(textBlock);
                //System.out.println(taggedString);

                StringTokenizer st1 = new StringTokenizer(taggedString);
                while (st1.hasMoreTokens()) {
                        StringTokenizer st2 = new StringTokenizer(st1.nextToken(), "_");
                        String taggedWord = st2.nextToken();
                        String tag = st2.nextToken();
                        first = textBlock.indexOf(taggedWord, last);
                        last = first + taggedWord.length();
                        Tuple t = new Tuple(first, last);
                        m.put(t, tag); //store Tuple, tag in a Map
                        arrTags.add(tag);
                        arrTuples.add(t);
                }

        }//end preprocess


        /*method: getHighlightedWords
         *      input: String lesson selected
         *      output: List of tuples of all tagged Word Positions
         *      description: retrieve the map entry for the current word; check this entry's
value to lesson; return true if matches
         */
        public List getHighlightedWords(String lesson) {
                List<Tuple> t = new ArrayList<Tuple>();
                boolean flagAdverbComp = false;
                int arrIndex = -1;

                for (Map.Entry<Tuple, String> e : m.entrySet()) {
                        arrIndex++;

                        if (lesson.equals("adjective")) {
                                if (e.getValue().equals("JJ") || e.getValue().equals("JJR") ||
e.getValue().equals("JJS")) {
                                        t.add(e.getKey());
                                }
                        }
                        //if JJR or if RBR + JJ
                        if (lesson.equals("adjectiveComp")) {
                                if (e.getValue().equals("JJR")) {
                                        t.add(e.getKey());
                                }
                                if (arrTags.get(arrIndex).equals("JJ") && arrTags.get(arrIndex -
 1).equals("RBR")) {
                                        t.add(arrTuples.get(arrIndex));
                                        t.add(arrTuples.get(arrIndex - 1));
                                }
                        }
                        //if JJS or if RBS + JJ
                        if (lesson.equals("adjectiveSuper")) {
                                if (e.getValue().equals("JJS")) {
                                        t.add(e.getKey());
                                }
                                if (arrTags.get(arrIndex).equals("JJ") && arrTags.get(arrIndex -
 1).equals("RBS")) {
                                        t.add(arrTuples.get(arrIndex));
                                        t.add(arrTuples.get(arrIndex - 1));
                                }
                        }
                        if (lesson.equals("adverb")) {
                                if (e.getValue().equals("RB") || e.getValue().equals("RBR") ||
e.getValue().equals("RBS")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("adverbComp")) {
                                //check if the words RBR and RB are consecutive
                                if (flagAdverbComp) {
                                        if (e.getValue().equals("RB")) {
                                                t.add(e.getKey());
                                                flagAdverbComp = false;
                                        }
                                }
                                if (e.getValue().equals("RBR")) {
                                        flagAdverbComp = true;
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("adverbSuper")) {
                                if (e.getValue().equals("RBS")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("determiner")) {
                                if (e.getValue().equals("DT")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("noun")) {
                                if (e.getValue().equals("NN") || e.getValue().equals("NNS") ||
e.getValue().equals("NNP") || e.getValue().equals("NNPS")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("nounPlural")) {
                                if (e.getValue().equals("NNS")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("nounProperSingular")) {
                                if (e.getValue().equals("NNP")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("nounProperPlural")) {
                                if (e.getValue().equals("NNPS")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("verb")) {
                                if (e.getValue().equals("VB") || e.getValue().equals("VBG") ||
e.getValue().equals("VBD") || e.getValue().equals("VBN") || e.getValue().equals("VBP")||
e.getValue().equals("VBZ")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("verbPast")) {
                                if (e.getValue().equals("VBD")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("verbPresent")) {
                                if (e.getValue().equals("VBG")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("verbPastParticiple")) {
                                if (e.getValue().equals("VBN")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("verbThirdPersonPresentSingular")) {
                                if (e.getValue().equals("VBZ")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("verbNonThirdPersonPresentSingular")) {
                                if (e.getValue().equals("VBP")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("pronoun")) {
                                if (e.getValue().equals("PRP") || e.getValue().equals("PRP$")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("pronounPersonal")) {
                                if (e.getValue().equals("PRP")) {
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("pronounPossessive")) {
                                if (e.getValue().equals("PRP$")) {
                                        t.add(e.getKey());
                                }
                        }
                        //checks if VBN is preceded by VBG or VBZ or VBD
                        if (lesson.equals("passive")) {
                                if (e.getValue().equals("VBN")) {
                                        if (arrTags.get(arrIndex - 1).equals("VBG") ||
arrTags.get(arrIndex - 1).equals("VBZ") || arrTags.get(arrIndex - 1).equals("VBD") ||
arrTags.get(arrIndex - 1).equals("VBN")) {
                                                t.add(arrTuples.get(arrIndex - 1));
                                                t.add(e.getKey());
                                        }
                                }
                                if (e.getValue().equals("MD") && arrTags.get(arrIndex
+ 1).equals("VB") && (arrTags.get(arrIndex + 2).equals("VBN") || (arrTags.get(arrIndex
+2).equals("VBG") && arrTags.get(arrIndex + 3).equals("VBN")))) {
                                        t.add(e.getKey());
                                        t.add(arrTuples.get(arrIndex + 1));
                                        t.add(arrTuples.get(arrIndex + 2));
                                }
                        }
                        if (lesson.equals("possessive")) {
                                if (e.getValue().equals("POS") && (arrTags.get(arrIndex -
 1).equals("NN") || arrTags.get(arrIndex - 1).equals("NNP"))) {
                                        t.add(arrTuples.get(arrIndex - 1));
                                        t.add(e.getKey());
                                }
                        }
                        if (lesson.equals("conjunction")) {
                                if (e.getValue().equals("CC")) {
                                        t.add(e.getKey());
                                }
                        }

                }       //end for
                return t;
        }       //end shouldHighlightWord
}       //end class
```

Tuple.java

```
/*
 * Tuple.java
 *
 * Created on November 29, 2007, 2:54 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package CL_Book;

/**
 *
 * @author shuralydm
 */
public class Tuple<E>
{
        private E first;                // First element
        private E second;              // Second element

        /** Creates a new instance of Tuple */
        public Tuple(E first, E second) {
```

```java
		this.first = first;
		this.second = second;
	}

	// Gets the first element
	public E getFirst() {
		return this.first;
	}

	// Gets the second element
	public E getSecond() {
		return this.second;
	}

	// Sets the first element
	public void setFirst(E first) {
		this.first = first;
	}

	// Sets the second element
	public void setSecond(E second) {
		this.second = second;
	}

	// Returns true of two tuple intervals overlap
	public boolean doesOverlap(Tuple<Integer> other)
	{
		return !((other.getSecond() <= (Integer)getFirst()) || ((Integer)getSecond() <=
other.getFirst()));
	}

	// Returns a string repesentation of the tuple
	@Override
	public String toString() {
		return "(" + getFirst() + ", " + getSecond() + ")";
	}

	// Converts a string representation of the tuple into a tuple (assuming Integer type)
	public static Tuple<Integer> valueOfInteger(String tupleString) {
		String[] tuple = tupleString.split("\\D");
		return new Tuple<Integer>(Integer.valueOf(tuple[1]), Integer.valueOf(tuple[3]));
	}
}
```