```
import java.io.*;
/*************************************************************************
 * CmdProcess: class for processing commands from text editor
 * @author Michael Hartung, Matthew Armand
 */
public class CmdProcess {
    //LineList of input lines, boolean saved instance variables
    private LineList tedList;
    private boolean saved;
    private CutPaste clipboard;

    /*******************************************************************
     * Standard constructor, instantiates new process object
     */
    public CmdProcess() {
        tedList = new LineList();
        clipboard = new CutPaste();
        saved = false;
    }

    /*******************************************************************
     * Checks whether the file has been saved since last alteration
     * @return true if unchanged since last saved, false if altered
     */
    public boolean isSaved() {
        return saved;
    }

    /*******************************************************************
     * Gets linelist object within Process class
     * @return LineList object (tedList)
     */
    public LineList getList () {
        return tedList;
    }

    /*******************************************************************
     * Inserts a new line before the current line in the list
     * @param newLine String of text to comprise new line
     */
    public void insertBefore(String newLine) {
        tedList.insertBefore(newLine);
        saved = false;
    }

    /*******************************************************************
     * Inserts a new line after the current line in the list
     * @param newLine String of text to comprise new line
     */
    public void insertAfter(String newLine) {
        tedList.insertAfter(newLine);
        saved = false;
    }

    /*********************************************************************
     * Inserts a new line at the end of the file
     * @param newLine String of text to comprise new line
     */
    public void insertLast(String newLine) {
        tedList.insertLast(newLine);
        saved = false;
    }

    /*********************************************************************
     * Moves current line indicator down a certain number of positions
     * @param numDown Number of lines to move down
     */
    public void down(int numDown) {
        while(numDown > 0) {
            downOnePos();
            numDown--;
        }
    }

    /*********************************************************************
     * Moves the current line indicator down one position
     */
    public void downOnePos() {
        tedList.down();
    }

    /*********************************************************************
     * Moves the current line indicator up a certain number of positions
     * @param numUp Number of lines to move up
     */
    public void up(int numUp) {
        while(numUp > 0) {
            upOnePos();
            numUp--;
        }
    }

    /*********************************************************************
     * Moves the current line indicator up one position
     */
    public void upOnePos() {
```

```
    tedList.up();
}

/*******************************************************************
 * Removes the current line from the list
 */
public void removeCurrentLine() {
    tedList.remove();
    saved = false;
}

/*******************************************************************
 * Removes a number of Lines from the list
 */
public void remove(int numLines) {
    for(int i = 0; i < numLines; i++) {
        removeCurrentLine();
    }
}
/*******************************************************************
 * Displays all the lines in the list in formatted order
 */
public void displayFile() {
    System.out.println(tedList.toString());
}

/*******************************************************************
 * Displays lines x to y in the list in formatted order
 * @param x starting line to display
 * @param y ending line to display
 */
public void display (int x, int y) {
    if (x<1 || y<x) {
        System.out.println("Invalid Command!");
    }
    else {
        System.out.println(tedList.display(x,y));
    }
}

/*******************************************************************
 * Clears file and removes all existing lines
 */
public void clearFile() {
    tedList = new LineList();
    saved = false;
}
```

```java
/******************************************************************
 * Saves contents of file to a file in directory structure
 * @param fileName Name of the file to be saved
 */
public void saveFile(String fileName) {
    PrintWriter p = null;
    try {
        p=new PrintWriter(new BufferedWriter(new FileWriter(fileName)));
        Line tmp = tedList.getHead();
        while (tmp!=null){
            p.print(tmp);
            tmp=tmp.getNext();
        }
        saved = true;
        p.close();
    } catch (IOException e) {
        System.out.println("Error while writing file!");
    }
}

/******************************************************************
 * Loads contents of the file into current buffer
 * @param fileName Name of the file to be loaded
 */
public void loadFile(String fileName) {
    try {
        Scanner sc = new Scanner (new File(fileName));
        clearFile();
        while (sc.hasNextLine()) {
            tedList.insertAfter(sc.nextLine());
        }
        tedList.setCurrent(tedList.getHead());
        saved = true;
        sc.close();
    } catch (FileNotFoundException e) {
        System.out.println("File not found");
    }
    saved = false;
}

/******************************************************************
 * Displays the help dialog with list of commands and functions
 */
public String showHelp() {
    String out = "";
    out += ("Welcome to Text Editor Help!\n");
    out += ("Command:      Function:\n");
    out += ("b 'sentence'  Insert sentence before current");
```

```
        out += (" line, make inserted line current\n");
        out += ("i 'sentence'  Insert sentence after current");
        out += (" line, make inserted line current\n");
        out += ("e 'sentence'  Insert sentence after last line, make inserted 
line current\n");
        out += ("m             Move cursor down 1 position\n");
        out += ("m #           Move cursor down # positions\n");
        out += ("u             Move cursor up 1 position\n");
        out += ("u #           Move cursor down # positions\n");
        out += ("r             Remove current line. Next line becomes ");
        out += ("current, unless no next \n              line, then previous 
becomes ");
        out += ("current\n");
        out += ("r #           Remove # lines, starting at current\n");
        out += ("d             Display all lines with line numbers\n");
        out += ("d # *         Display lines # to * with line numbers\n");
        out += ("cut # $ *     Cut lines # to $ to clipboard *\n");
        out += ("pas *         Paste clipboard * before current position\n");
        out += ("c             Clear all lines in the file\n");
        out += ("!c            Force clear all lines in the file\n");
        out += ("s 'filename'  Save contents to specified text file\n");
        out += ("l 'filename'  Load contents of file into current buffer\n");
        out += ("!l 'filename' Force load contents of file into current buffer
\n");
        out += ("h             Display this help page\n");
        out += ("x             Exit the editor\n");
        out += ("!x            Force exit the editor");
        return out;
    }

    /*******************************************************************
     * Cuts selection onto a clipboard to be pasted
     */
    public void cutSelection(int startLine, int endLine, int clipboardNum) {
        LineList temp = new LineList();
        tedList.setCurrent(tedList.getHead());
            // Finding Starting Position
            for(int i = 1; i < startLine; i++) {
                if(tedList.getCurrent().getNext() != null) {
                    tedList.setCurrent(tedList.getCurrent().getNext());
                }
                else {
                    System.out.println("Invalid Starting Point!");
                    i = endLine;
                }
            }
            // After Start has been found, copy lines into temp line list and 
remove lines
```

```
        for(int i = startLine; i <= endLine; i++) {
            if(tedList.getCurrent() != null) {
                temp.insertAfter(tedList.getCurrent().toString());
                removeCurrentLine();
            }
            else {
                System.out.println("Invalid Ending Point!");
                i = endLine;
            }
        }
    if(temp.getHead() != null) {
        clipboard.add(temp, clipboardNum);
    }
    else {
        System.out.println("Cut failed!");
    }
    saved = false;
}

/*******************************************************************
 * Pastes selection from clipboard into the file
 */
public void pasteClipboard(int clipboardNum) {
    LineList temp = clipboard.getBoard(clipboardNum);
    String mod;
    temp.setCurrent(temp.getHead());
    if(tedList.getHead() == null) {
        while(temp.getCurrent() != null) {
            mod = temp.getCurrent().toString();
            mod = mod.substring(0,mod.length()-2);
            tedList.insertAfter(mod);
            temp.setCurrent(temp.getCurrent().getNext());
        }
    }
    else if(tedList.getCurrent().getPrev() != null) {
        upOnePos();
        while(temp.getCurrent() != null) {
            mod = temp.getCurrent().toString();
            mod = mod.substring(0,mod.length()-2);
            tedList.insertAfter(mod);
            temp.setCurrent(temp.getCurrent().getNext());
        }
    }
    else {
        mod = temp.getCurrent().toString();
        mod = mod.substring(0,mod.length()-2);
        tedList.insertBefore(mod);
        temp.setCurrent(temp.getCurrent().getNext());
```

```java
                while(temp.getCurrent() != null) {
                    mod = temp.getCurrent().toString();
                    mod = mod.substring(0,mod.length()-2);
                    tedList.insertAfter(mod);
                    temp.setCurrent(temp.getCurrent().getNext());
                }
            }
            if (tedList.getCurrent().getNext() != null)
                downOnePos();
            saved = false;
        }
    }
```