ASSIGNMENT # 2

SE 6356 Software Maintenance, Evolution and Re-Engineering

Mahabubul Alam
Abdullah Moyeen

Table of Contents

1 ANALYZING CLASS COHESION IN OO SOFTWARE

1.1 SYSTEM ANALYZED: JEDIT

1.1.1 The top 2 most cohesive classes, based on SourceMeter results:

Class Name	Lack of Cohesion in Methods 5 (LCOM5)
org.gjt.sp.jedit.textarea.TextAreaException	0
org.gjt.sp.jedit.syntax.SyntaxStyle	0

- The **TextAreaException** class implements a single functionality - Exception that the TextArea can throw when an error occurs. It can be catched and an error dialog can be displayed.

- The **SyntaxStyle** class is a simple text style class. It implements the functionality of specifying the color, italic flag, and bold flag of a run of text and is self-contained.

1.1.2 The top 2 least cohesive classes, based on SourceMeter results:

Class Name	Lack of Cohesion in Methods 5 (LCOM5)
org.gjt.sp.jedit.MiscUtilities	38
org.gjt.sp.jedit.GUIUtilities	22

- The **MiscUtilities** class can be split into 38 coherent classes which is also apparent by examining the code which shows a lot of methods busy, implementing functionalities such as path manipulation, string manipulation, URL name manipulation and more. Also, the class is coupled with 19 other classes and there are 135 incoming invocations along with 52 outgoing invocations.

- According to SourceMeter analysis, The **GUIUtilities** class can be split into 22 coherent classes. The class is coupled with 25 other classes and has 228 incoming and 43 outgoing invocations. The class implements various GUI utility functions related to icons, menus, toolbars, keyboard shortcuts, etc. The most frequently used members of this class are:

```
loadIcon(String)
confirm(Component,String,Object[],int,int)
error(Component,String,Object[])
message(Component,String,Object[])
showVFSFileDialog(View,String,int,boolean)
loadGeometry(Window,String)
saveGeometry(Window,String)
showPopupMenu(JPopupMenu,Component,int,int)
```

1.2 System Analyzed: aTunes

1.2.1 The top 2 most Cohesive Classes, based on SourceMeter results:

Class Name	Lack of Cohesion in Methods 5 (LCOM5)
net.sourceforge.atunes.kernel.modules.repository.tags.reader.TagDetector	0
net.sourceforge.atunes.kernel.modules.amazon.AmazonAlbum	0

- The **TagDetector** class is limited to a single functionality of getting MP3 tags from the audio file. The class has 3 private attributes (ID3v2TagReader, ID3v1_1TagReader, ID3v1TagReader) and the single method, getTags() inside this class is only using these 3 attributes.

- The **AmazonAlbum** class has 4 attributes that it uses to implement the responsibility of getting the album's artist, album and url information and is self-contained in that.

1.2.2 The top 2 least Cohesive Classes, based on SourceMeter results:

Class Name	Lack of Cohesion in Methods 5 (LCOM5)
net.sourceforge.atunes.kernel.modules.state.ApplicationState	19
net.sourceforge.atunes.kernel.handlers.PlayListHandler	8

- The **ApplicationState** class, according to SourceMeter could be divided into 19 distinct cohesive classes, which is also apparent by looking at the code. We can easily see that it has methods to do a lot things which are not necessarily related to each other. For example,

```
setSongProperties()
setWindowLocation()
setShowAlbumInPlayList()
```

- The **PlayListHandler** class is implementing quite a few functionalities and some of which are not related. Such as, sorting and reordering PlayList and editing tags.

The main difference between the classes with the highest and lowest cohesion lies in the number of functionalities of the class, i.e. how well or badly the class adheres to the single responsibility principle. To identify the most and least cohesive classes, we depended on the LCOM5 metric analyzed by source meter which measures the lack of cohesion and computes into how many coherent classes the class could be split. We also looked at the coupling metrics to see how many incoming and outgoing invocations the classes had. A high value indicated a low cohesive class and a lower value indicated a class with higher cohesion.

2 CODE SMELL DETECTION USING JDEODORANT AND INCODE

2.1 CODE SMELLS IN ATUNES

2.1.1 Smell: Feature Envy - net.sourceforge.atunes.kernel.controllers.playListControls

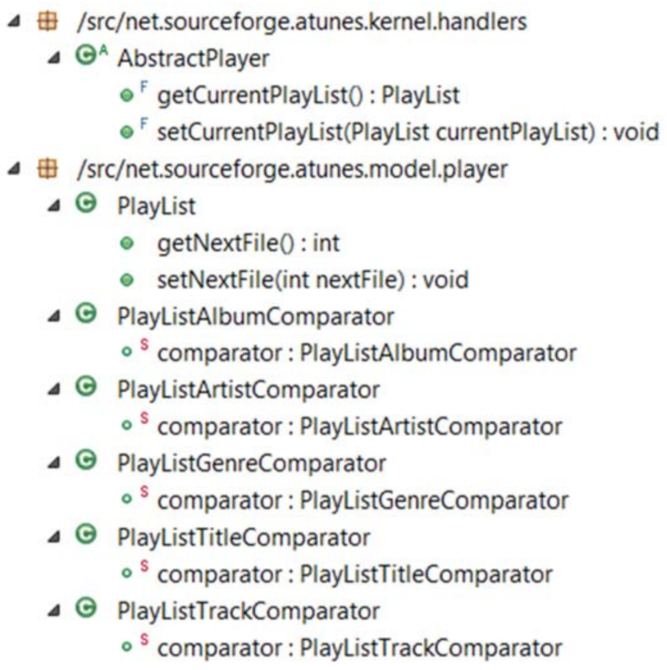
Protected method **addBindings()** in **PlayListControlsController.java** is heavily using data from external class **PlayListControlsPanel.java** to add bindings to **PlaylistControlsListener.java class.**

```
	protected void addBindings() {
		final PlayListControlsPanel panel = (PlayListControlsPanel)
panelControlled;

		PlayListControlsListener listener = new PlayListControlsListener(panel);

		panel.getSortByTrack().addActionListener(listener);
		panel.getSortByTitle().addActionListener(listener);
		panel.getSortByArtist().addActionListener(listener);
		panel.getSortByAlbum().addActionListener(listener);
		panel.getSortByGenre().addActionListener(listener);
		panel.getSavePlaylistButton().addActionListener(listener);
		panel.getLoadPlaylistButton().addActionListener(listener);
		panel.getTopButton().addActionListener(listener);
		panel.getUpButton().addActionListener(listener);
		panel.getDeleteButton().addActionListener(listener);
		panel.getDownButton().addActionListener(listener);
		panel.getBottomButton().addActionListener(listener);
		panel.getInfoButton().addActionListener(listener);
		panel.getClearButton().addActionListener(listener);
		panel.getFavoriteSong().addActionListener(listener);
		panel.getFavoriteAlbum().addActionListener(listener);
		panel.getFavoriteArtist().addActionListener(listener);
		panel.getShowTrack().addActionListener(listener);
		panel.getShowArtist().addActionListener(listener);
		panel.getShowGenre().addActionListener(listener);
		panel.getShowAlbum().addActionListener(listener);
		panel.getArtistButton().addActionListener(listener);
		panel.getAlbumButton().addActionListener(listener);
	}
```

It is a smell because a method is accessing the data of another object more than its own data.

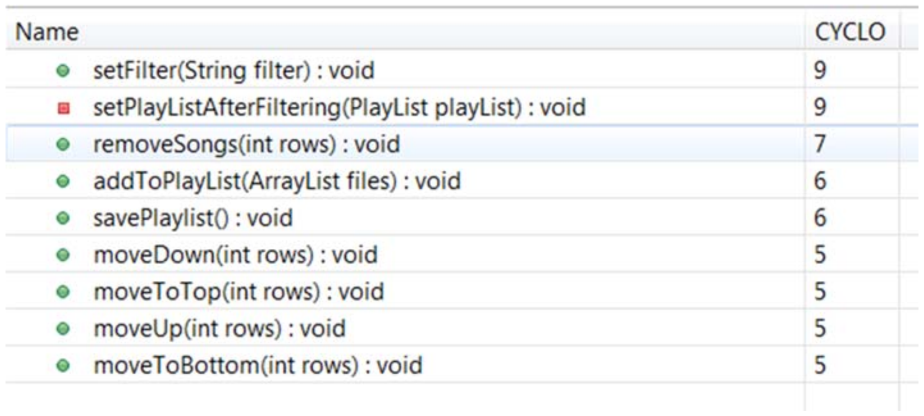
2.1.2 Smell: God Class - net.sourceforge.atunes.kernel.handlers.PlayListHandler

The PlayListHandler class uses many attributes from external classes –



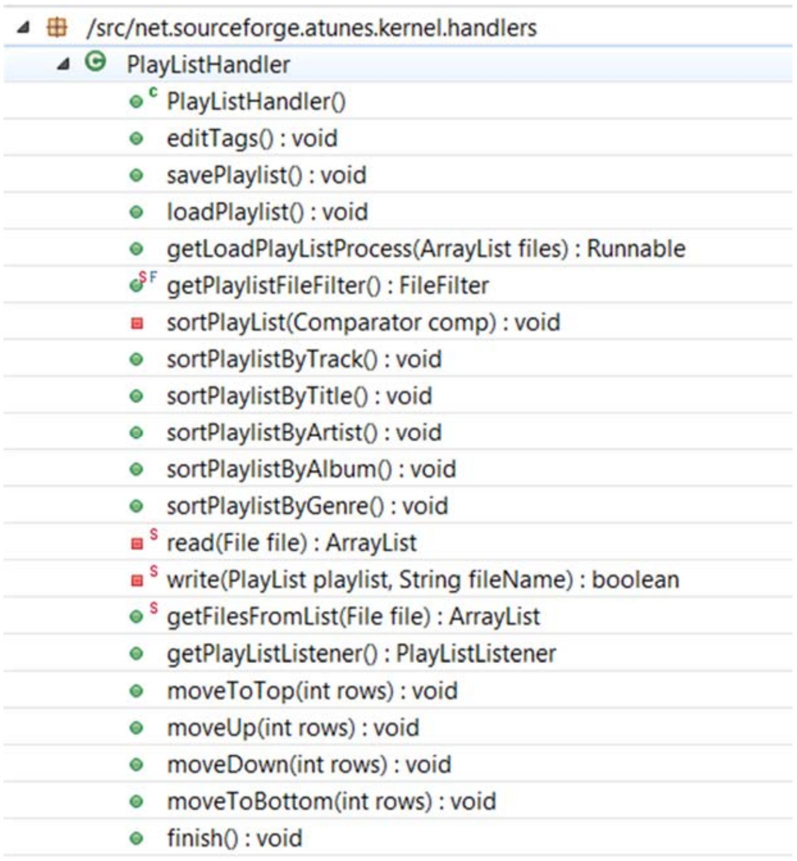
```
/src/net.sourceforge.atunes.kernel.handlers
    AbstractPlayer
        getCurrentPlayList() : PlayList
        setCurrentPlayList(PlayList currentPlayList) : void
/src/net.sourceforge.atunes.model.player
    PlayList
        getNextFile() : int
        setNextFile(int nextFile) : void
    PlayListAlbumComparator
        comparator : PlayListAlbumComparator
    PlayListArtistComparator
        comparator : PlayListArtistComparator
    PlayListGenreComparator
        comparator : PlayListGenreComparator
    PlayListTitleComparator
        comparator : PlayListTitleComparator
    PlayListTrackComparator
        comparator : PlayListTrackComparator
```

The PlayListHandler class is excessively large and complex, due to its methods having a high cyclomatic complexity and nesting level



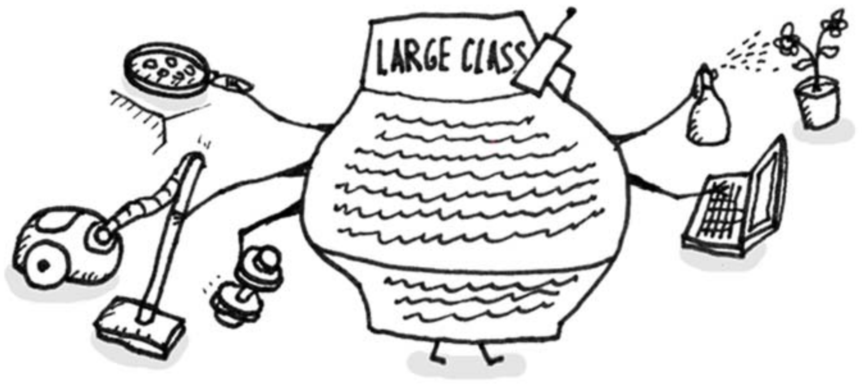
Name	CYCLO
setFilter(String filter) : void	9
setPlayListAfterFiltering(PlayList playList) : void	9
removeSongs(int rows) : void	7
addToPlayList(ArrayList files) : void	6
savePlaylist() : void	6
moveDown(int rows) : void	5
moveToTop(int rows) : void	5
moveUp(int rows) : void	5
moveToBottom(int rows) : void	5

This PlayListHandler class is very non-cohesive, in terms of how class attributes are used by its methods.



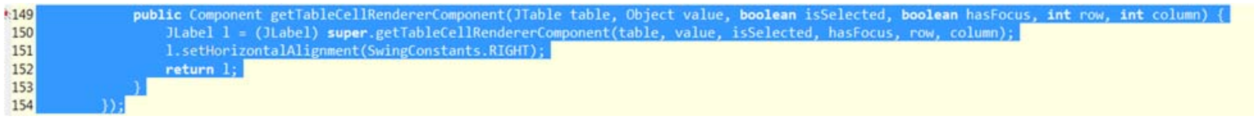
```
/src/net.sourceforge.atunes.kernel.handlers
  PlayListHandler
      PlayListHandler()
      editTags() : void
      savePlaylist() : void
      loadPlaylist() : void
      getLoadPlayListProcess(ArrayList files) : Runnable
      getPlaylistFileFilter() : FileFilter
      sortPlayList(Comparator comp) : void
      sortPlaylistByTrack() : void
      sortPlaylistByTitle() : void
      sortPlaylistByArtist() : void
      sortPlaylistByAlbum() : void
      sortPlaylistByGenre() : void
      read(File file) : ArrayList
      write(PlayList playlist, String fileName) : boolean
      getFilesFromList(File file) : ArrayList
      getPlayListListener() : PlayListListener
      moveToTop(int rows) : void
      moveUp(int rows) : void
      moveDown(int rows) : void
      moveToBottom(int rows) : void
      finish() : void
```

The PlayListHandler class is affected by the “God Class” smell because it contains many fields/methods/lines of code and can be considered a violation of the single responsibility principle of Object Oriented design.



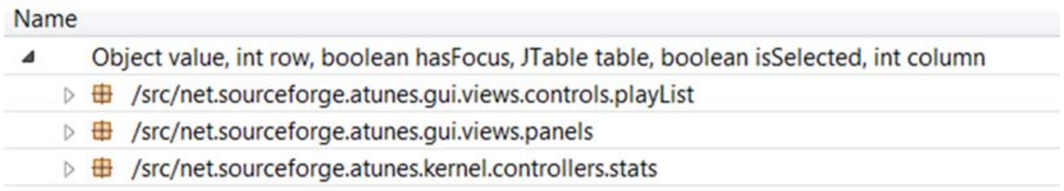
2.1.3 Smell: Data Clumps - net.sourceforge.atunes.kernel.controllers.stats

getTableCellRendererComponent in the StatsDialogController class is affected by Data Clumps because the method has a long parameter list, and its signature or a significant fragment thereof is duplicated by other methods. This is a sign that the group of parameters, being passed around collectively to multiple methods in the system, could form a new abstraction that could be extracted to a new class.



```
            public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {
                JLabel l = (JLabel) super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);
                l.setHorizontalAlignment(SwingConstants.RIGHT);
                return l;
            }
        });
```

The detected parameter clusters show individual parameter list fragments that are repeatedly used all around the system.

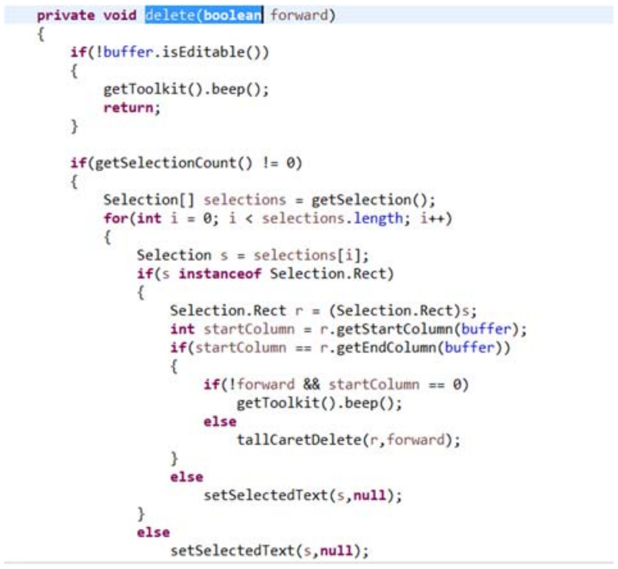


Name
Object value, int row, boolean hasFocus, JTable table, boolean isSelected, int column
/src/net.sourceforge.atunes.gui.views.controls.playList
/src/net.sourceforge.atunes.gui.views.panels
/src/net.sourceforge.atunes.kernel.controllers.stats

2.2 Code smells in JEdit

2.2.1 Smell: Type Checking - org.gjt.sp.jedit.textarea

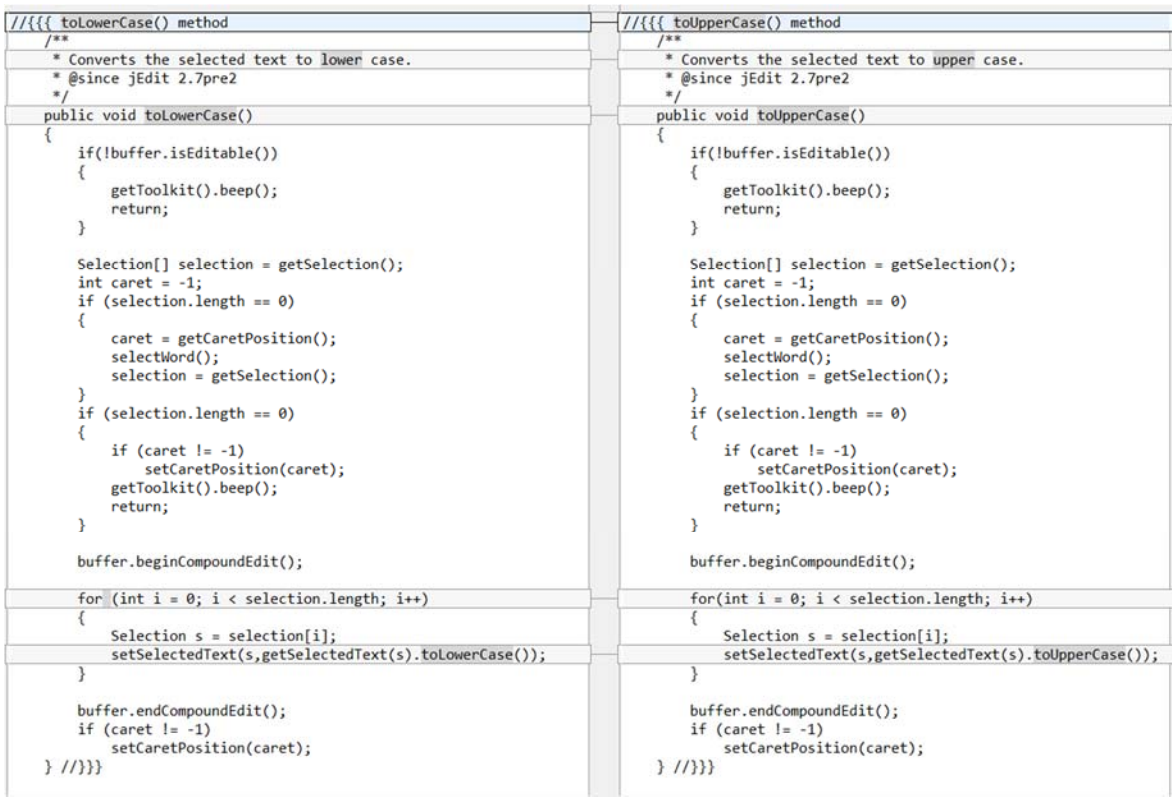
A sequence of if statements in delete(boolean forward) method of TextArea.java class, where the delete operation is being performed based on different type of text selections,



```
private void delete(boolean forward)
{
    if(!buffer.isEditable())
    {
        getToolkit().beep();
        return;
    }

    if(getSelectionCount() != 0)
    {
        Selection[] selections = getSelection();
        for(int i = 0; i < selections.length; i++)
        {
            Selection s = selections[i];
            if(s instanceof Selection.Rect)
            {
                Selection.Rect r = (Selection.Rect)s;
                int startColumn = r.getStartColumn(buffer);
                if(startColumn == r.getEndColumn(buffer))
                {
                    if(!forward && startColumn == 0)
                        getToolkit().beep();
                    else
                        tallCaretDelete(r,forward);
                }
                else
                    setSelectedText(s,null);
            }
            else
                setSelectedText(s,null);
```

2.2.2 Smell: Internal Duplication - org.gjt.sp.jedit.textarea

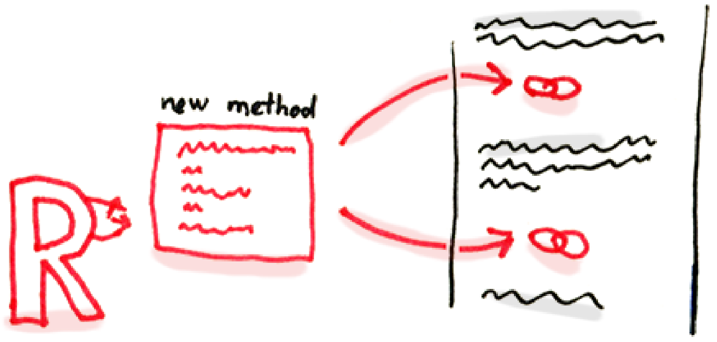
Public methods toUpperCase() and toLowerCase() in the **TextArea** class are exactly identical except just in one line where they are calling different String extensions of toUpperCase() and toLowerCase() respectively.



```
//{{{ toLowerCase() method
    /**
     * Converts the selected text to lower case.
     * @since jEdit 2.7pre2
     */
    public void toLowerCase()
    {
        if(!buffer.isEditable())
        {
            getToolkit().beep();
            return;
        }

        Selection[] selection = getSelection();
        int caret = -1;
        if (selection.length == 0)
        {
            caret = getCaretPosition();
            selectWord();
            selection = getSelection();
        }
        if (selection.length == 0)
        {
            if (caret != -1)
                setCaretPosition(caret);
            getToolkit().beep();
            return;
        }

        buffer.beginCompoundEdit();

        for (int i = 0; i < selection.length; i++)
        {
            Selection s = selection[i];
            setSelectedText(s,getSelectedText(s).toLowerCase());
        }

        buffer.endCompoundEdit();
        if (caret != -1)
            setCaretPosition(caret);
    } //}}}
```

```
//{{{ toUpperCase() method
    /**
     * Converts the selected text to upper case.
     * @since jEdit 2.7pre2
     */
    public void toUpperCase()
    {
        if(!buffer.isEditable())
        {
            getToolkit().beep();
            return;
        }

        Selection[] selection = getSelection();
        int caret = -1;
        if (selection.length == 0)
        {
            caret = getCaretPosition();
            selectWord();
            selection = getSelection();
        }
        if (selection.length == 0)
        {
            if (caret != -1)
                setCaretPosition(caret);
            getToolkit().beep();
            return;
        }

        buffer.beginCompoundEdit();

        for(int i = 0; i < selection.length; i++)
        {
            Selection s = selection[i];
            setSelectedText(s,getSelectedText(s).toUpperCase());
        }

        buffer.endCompoundEdit();
        if (caret != -1)
            setCaretPosition(caret);
    } //}}}
```

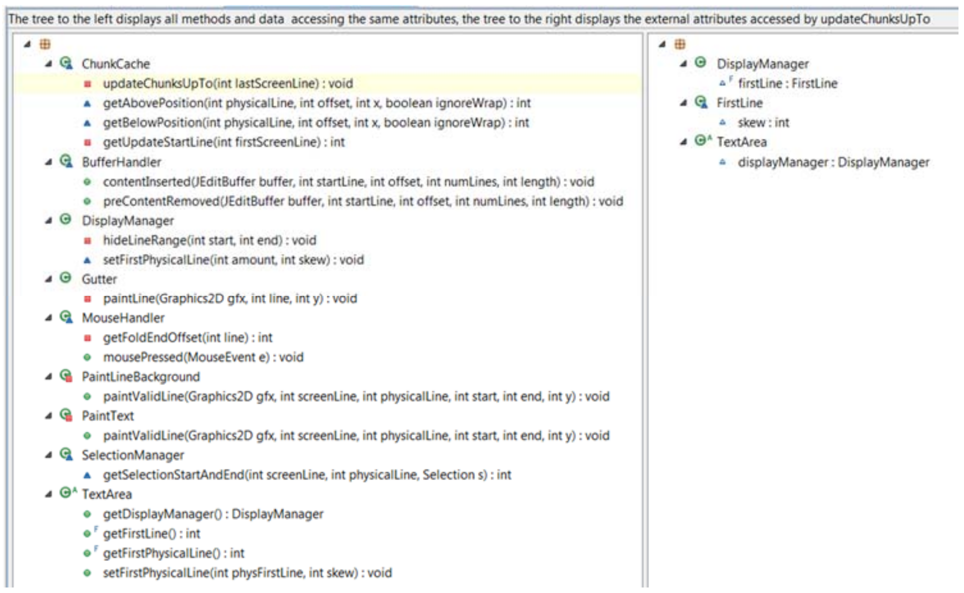
This is internal duplication smell, because the same code is found in two or more methods in the same class and the suggestion is to use Extract Method and place calls for the new method in both places.



2.2.3 Smell: Message Chains - org.gjt.sp.jedit.textarea

The updateChunksUpTo method in the ChunkCache class is affected by Message Chains because the method uses one object to access another object, then uses the obtained object to access another object, and so on, all objects having different types.

The screengrab below from InCode shows the used external Attributes, and it helps better understand what external data is used, and who else is using it.

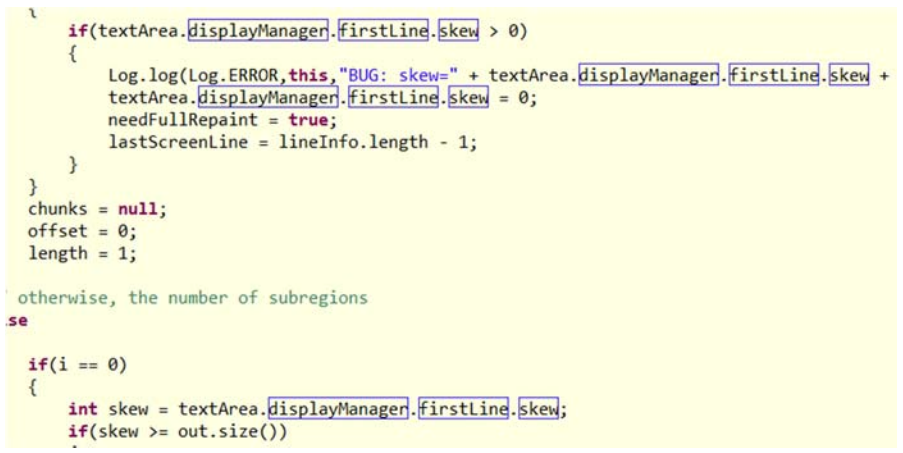


The tree to the left displays all methods and data accessing the same attributes, the tree to the right displays the external attributes accessed by updateChunksUpTo

```
ChunkCache
    updateChunksUpTo(int lastScreenLine) : void
    getAbovePosition(int physicalLine, int offset, int x, boolean ignoreWrap) : int
    getBelowPosition(int physicalLine, int offset, int x, boolean ignoreWrap) : int
    getUpdateStartLine(int firstScreenLine) : int
BufferHandler
    contentInserted(JEditBuffer buffer, int startLine, int offset, int numLines, int length) : void
    preContentRemoved(JEditBuffer buffer, int startLine, int offset, int numLines, int length) : void
DisplayManager
    hideLineRange(int start, int end) : void
    setFirstPhysicalLine(int amount, int skew) : void
Gutter
    paintLine(Graphics2D gfx, int line, int y) : void
MouseHandler
    getFoldEndOffset(int line) : int
    mousePressed(MouseEvent e) : void
PaintLineBackground
    paintValidLine(Graphics2D gfx, int screenLine, int physicalLine, int start, int end, int y) : void
PaintText
    paintValidLine(Graphics2D gfx, int screenLine, int physicalLine, int start, int end, int y) : void
SelectionManager
    getSelectionStartAndEnd(int screenLine, int physicalLine, Selection s) : int
TextArea
    getDisplayManager() : DisplayManager
    getFirstLine() : int
    getFirstPhysicalLine() : int
    setFirstPhysicalLine(int physFirstLine, int skew) : void
```

```
DisplayManager
    firstLine : FirstLine
FirstLine
    skew : int
TextArea
    displayManager : DisplayManager
```

The highlighted attributes below are involved in Message Chains and is a good indication of where the message chain originates:



```
    if(textArea.displayManager.firstLine.skew > 0)
    {
        Log.log(Log.ERROR,this,"BUG: skew=" + textArea.displayManager.firstLine.skew +
        textArea.displayManager.firstLine.skew = 0;
        needFullRepaint = true;
        lastScreenLine = lineInfo.length - 1;
    }
}
chunks = null;
offset = 0;
length = 1;

otherwise, the number of subregions
se

if(i == 0)
{
    int skew = textArea.displayManager.firstLine.skew;
    if(skew >= out.size())
```

3 REFACTORING (TO REMOVE CODE SMELL) USING TOOL SUPPORT

3.1 ATUNES – REMOVING “GOD CLASS” SMELL

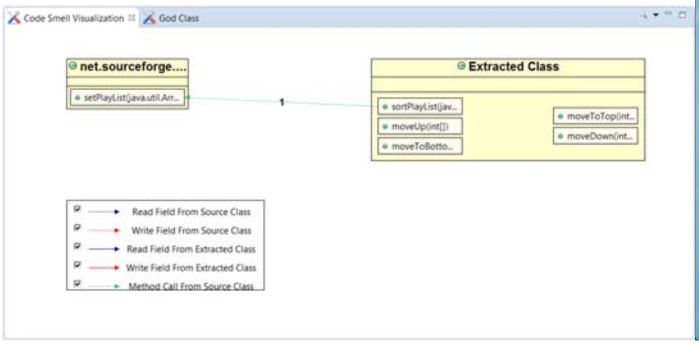
3.1.1 Justification for refactoring

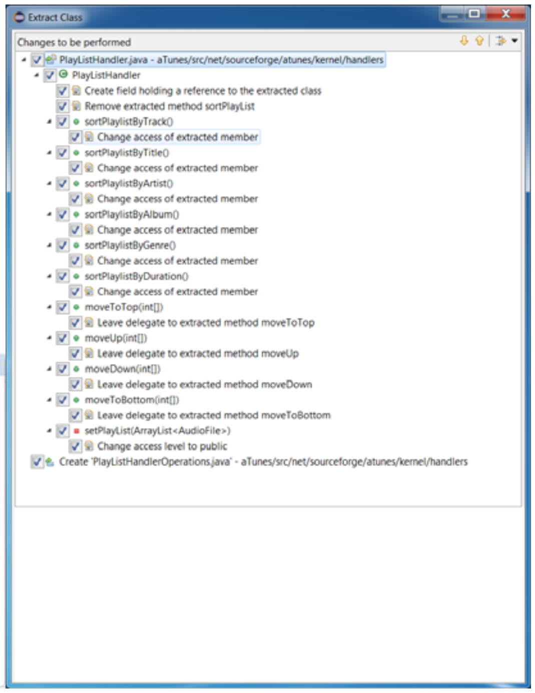
Refactoring of these classes spares developers from needing to remember a large number of attributes for a class.
In many cases, splitting large classes into parts avoids duplication of code and functionality.

3.1.2 Description and Rationale

We have extracted some of the operations on playlist from God Class PlayListHandler.java (namely - sortPlaylistByTrack(), sortPlaylistByTitle(), sortPlaylistByArtist(), sortPlaylistByAlbum(), sortPlaylistByGenre(), sortPlaylistByDuration(), moveToTop(), moveUp(), moveDown(), moveToBottom()) to a new class PlayListHandlerOperations.java to simplify the former.

3.1.3 Code Smell visualization

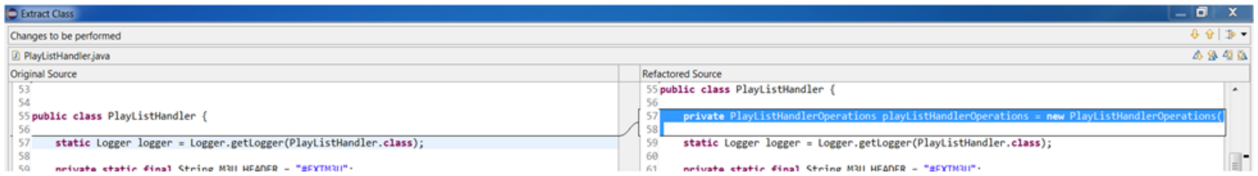

Code Smell Visualization
God Class
net.sourceforge....
setPlayList(java.util.Arr...
1
Extracted Class
sortPlayList(jav...
moveUp(int[])
moveToBotto...
moveToTop(int...
moveDown(int...
Read Field From Source Class
Write Field From Source Class
Read Field From Extracted Class
Write Field From Extracted Class
Method Call From Source Class


Extract Class

Changes to be performed

- ✓ PlayListHandler.java - aTunes/src/net/sourceforge/atunes/kernel/handlers
  - ✓ PlayListHandler
    - ✓ Create field holding a reference to the extracted class
    - ✓ Remove extracted method sortPlayList
    - ✓ sortPlaylistByTrack()
      - ✓ Change access of extracted member
    - ✓ sortPlaylistByTitle()
      - ✓ Change access of extracted member
    - ✓ sortPlaylistByArtist()
      - ✓ Change access of extracted member
    - ✓ sortPlaylistByAlbum()
      - ✓ Change access of extracted member
    - ✓ sortPlaylistByGenre()
      - ✓ Change access of extracted member
    - ✓ sortPlaylistByDuration()
      - ✓ Change access of extracted member
    - ✓ moveToTop(int[])
      - ✓ Leave delegate to extracted method moveToTop
    - ✓ moveUp(int[])
      - ✓ Leave delegate to extracted method moveUp
    - ✓ moveDown(int[])
      - ✓ Leave delegate to extracted method moveDown
    - ✓ moveToBottom(int[])
      - ✓ Leave delegate to extracted method moveToBottom
    - ✓ setPlayList(ArrayList<AudioFile>)
      - ✓ Change access level to public
- ✓ Create 'PlayListHandlerOperations.java' - aTunes/src/net/sourceforge/atunes/kernel/handlers

3.1.4 Changes to be performed



Extract Class

Changes to be performed

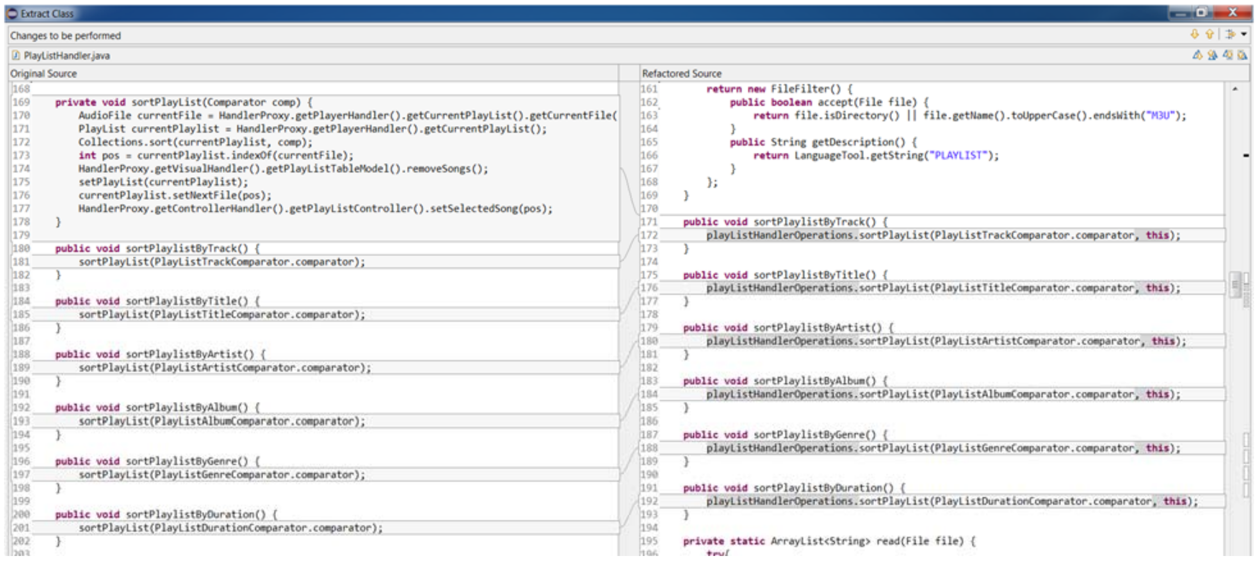
PlayListHandler.java

Original Source

```java

public class PlayListHandler {

    static Logger logger = Logger.getLogger(PlayListHandler.class);

    private static final String M3U_HEADER = "#EXTM3U";
```

Refactored Source

```java
public class PlayListHandler {

    private PlayListHandlerOperations playListHandlerOperations = new PlayListHandlerOperations(

    static Logger logger = Logger.getLogger(PlayListHandler.class);

    private static final String M3U_HEADER = "#EXTM3U";
```



Extract Class

Changes to be performed

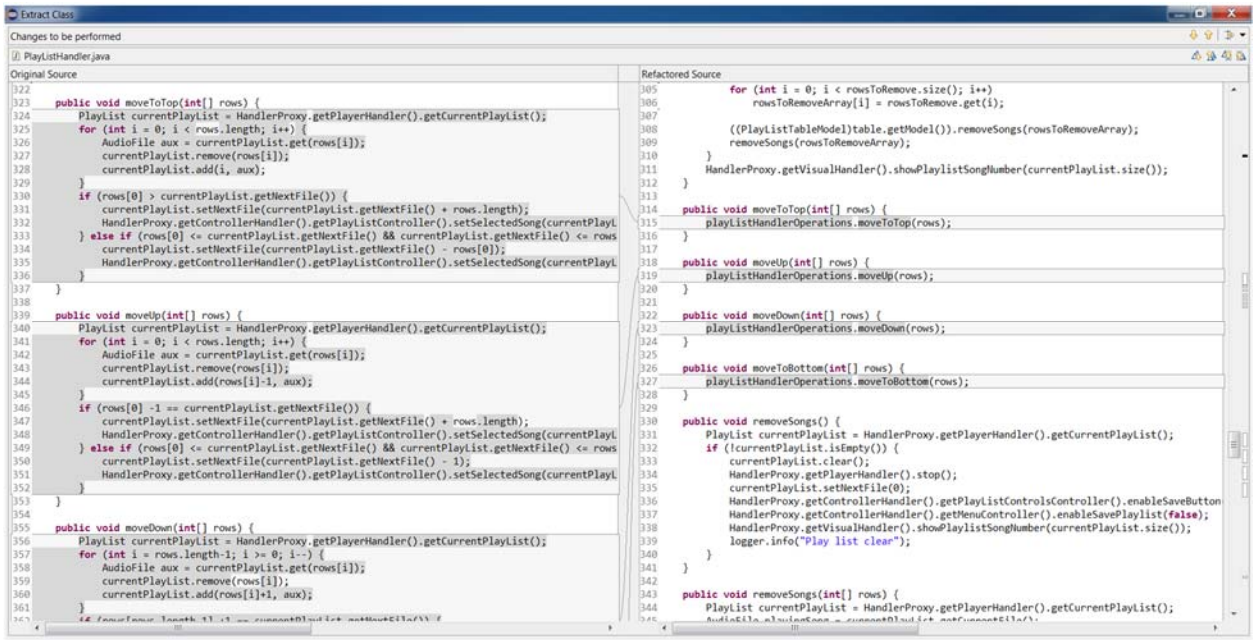
PlayListHandler.java

Original Source

```java

    private void sortPlayList(Comparator comp) {
        AudioFile currentFile = HandlerProxy.getPlayerHandler().getCurrentPlayList().getCurrentFile(
        PlayList currentPlaylist = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        Collections.sort(currentPlaylist, comp);
        int pos = currentPlaylist.indexOf(currentFile);
        HandlerProxy.getVisualHandler().getPlayListTableModel().removeSongs();
        setPlayList(currentPlaylist);
        currentPlaylist.setNextFile(pos);
        HandlerProxy.getControllerHandler().getPlayListController().setSelectedSong(pos);
    }

    public void sortPlaylistByTrack() {
        sortPlayList(PlayListTrackComparator.comparator);
    }

    public void sortPlaylistByTitle() {
        sortPlayList(PlayListTitleComparator.comparator);
    }

    public void sortPlaylistByArtist() {
        sortPlayList(PlayListArtistComparator.comparator);
    }

    public void sortPlaylistByAlbum() {
        sortPlayList(PlayListAlbumComparator.comparator);
    }

    public void sortPlaylistByGenre() {
        sortPlayList(PlayListGenreComparator.comparator);
    }

    public void sortPlaylistByDuration() {
        sortPlayList(PlayListDurationComparator.comparator);
    }
```

Refactored Source

```java
        return new FileFilter() {
            public boolean accept(File file) {
                return file.isDirectory() || file.getName().toUpperCase().endsWith("M3U");
            }
            public String getDescription() {
                return LanguageTool.getString("PLAYLIST");
            }
        };
    }

    public void sortPlaylistByTrack() {
        playListHandlerOperations.sortPlayList(PlayListTrackComparator.comparator, this);
    }

    public void sortPlaylistByTitle() {
        playListHandlerOperations.sortPlayList(PlayListTitleComparator.comparator, this);
    }

    public void sortPlaylistByArtist() {
        playListHandlerOperations.sortPlayList(PlayListArtistComparator.comparator, this);
    }

    public void sortPlaylistByAlbum() {
        playListHandlerOperations.sortPlayList(PlayListAlbumComparator.comparator, this);
    }

    public void sortPlaylistByGenre() {
        playListHandlerOperations.sortPlayList(PlayListGenreComparator.comparator, this);
    }

    public void sortPlaylistByDuration() {
        playListHandlerOperations.sortPlayList(PlayListDurationComparator.comparator, this);
    }

    private static ArrayList<String> read(File file) {
        try{
```



Extract Class

Changes to be performed

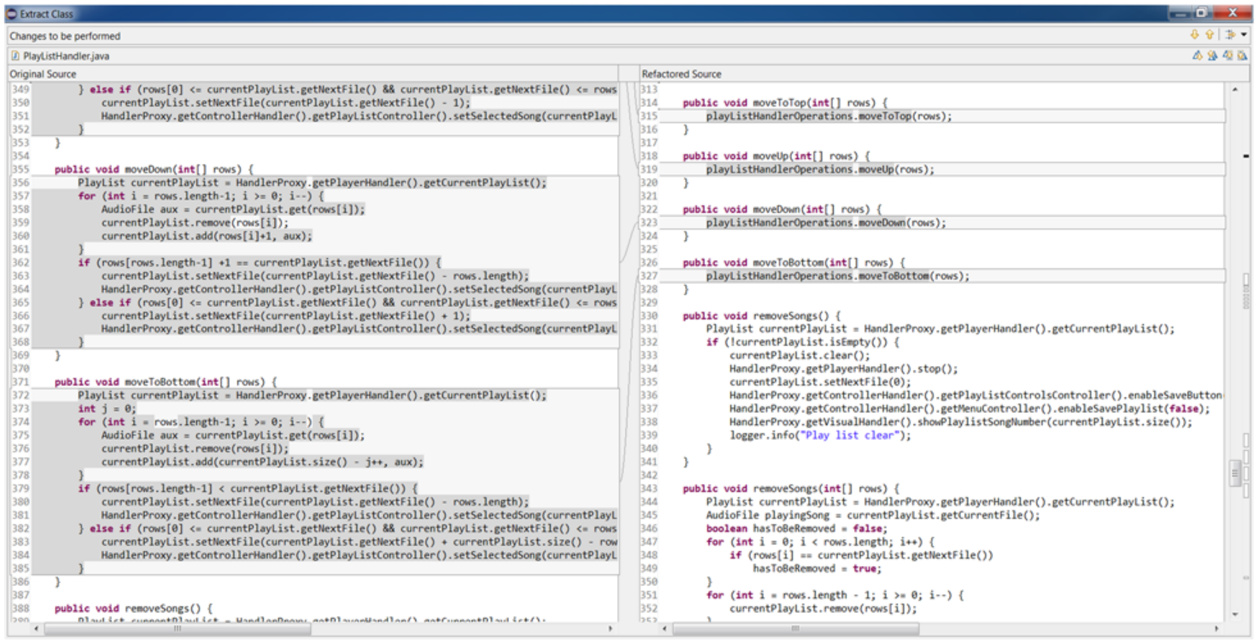
PlayListHandler.java

Original Source

```java

    public void moveToTop(int[] rows) {
        PlayList currentPlayList = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        for (int i = 0; i < rows.length; i++) {
            AudioFile aux = currentPlayList.get(rows[i]);
            currentPlayList.remove(rows[i]);
            currentPlayList.add(i, aux);
        }
        if (rows[0] > currentPlayList.getNextFile()) {
            currentPlayList.setNextFile(currentPlayList.getNextFile() + rows.length);
            HandlerProxy.getControllerHandler().getPlayListController().setSelectedSong(currentPlayL
        } else if (rows[0] <= currentPlayList.getNextFile() && currentPlayList.getNextFile() <= rows
            currentPlayList.setNextFile(currentPlayList.getNextFile() - rows[0]);
            HandlerProxy.getControllerHandler().getPlayListController().setSelectedSong(currentPlayL
        }
    }

    public void moveUp(int[] rows) {
        PlayList currentPlayList = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        for (int i = 0; i < rows.length; i++) {
            AudioFile aux = currentPlayList.get(rows[i]);
            currentPlayList.remove(rows[i]);
            currentPlayList.add(rows[i]-1, aux);
        }
        if (rows[0] -1 == currentPlayList.getNextFile()) {
            currentPlayList.setNextFile(currentPlayList.getNextFile() + rows.length);
            HandlerProxy.getControllerHandler().getPlayListController().setSelectedSong(currentPlayL
        } else if (rows[0] <= currentPlayList.getNextFile() && currentPlayList.getNextFile() <= rows
            currentPlayList.setNextFile(currentPlayList.getNextFile() - 1);
            HandlerProxy.getControllerHandler().getPlayListController().setSelectedSong(currentPlayL
        }
    }

    public void moveDown(int[] rows) {
        PlayList currentPlayList = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        for (int i = rows.length-1; i >= 0; i--) {
            AudioFile aux = currentPlayList.get(rows[i]);
            currentPlayList.remove(rows[i]);
            currentPlayList.add(rows[i]+1, aux);
        }
        if (rows[rows.length-1] +1 == currentPlayList.getNextFile()) {
```

Refactored Source

```java
            for (int i = 0; i < rowsToRemove.size(); i++)
                rowsToRemoveArray[i] = rowsToRemove.get(i);

            ((PlayListTableModel)table.getModel()).removeSongs(rowsToRemoveArray);
            removeSongs(rowsToRemoveArray);
        }
        HandlerProxy.getVisualHandler().showPlaylistSongNumber(currentPlayList.size());
    }

    public void moveToTop(int[] rows) {
        playListHandlerOperations.moveToTop(rows);
    }

    public void moveUp(int[] rows) {
        playListHandlerOperations.moveUp(rows);
    }

    public void moveDown(int[] rows) {
        playListHandlerOperations.moveDown(rows);
    }

    public void moveToBottom(int[] rows) {
        playListHandlerOperations.moveToBottom(rows);
    }

    public void removeSongs() {
        PlayList currentPlayList = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        if (!currentPlayList.isEmpty()) {
            currentPlayList.clear();
            HandlerProxy.getPlayerHandler().stop();
            currentPlayList.setNextFile(0);
            HandlerProxy.getControllerHandler().getPlayListControlsController().enableSaveButton
            HandlerProxy.getControllerHandler().getMenuController().enableSavePlaylist(false);
            HandlerProxy.getVisualHandler().showPlaylistSongNumber(currentPlayList.size());
            logger.info("Play list clear");
        }
    }

    public void removeSongs(int[] rows) {
        PlayList currentPlayList = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        AudioFile playingSong = currentPlayList.getCurrentFile();
```

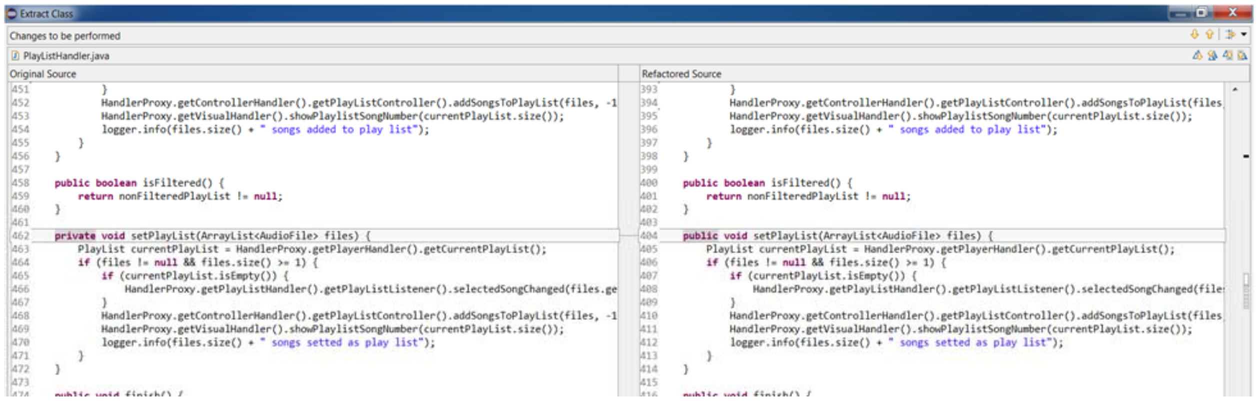


Original Source:

```java
        } else if (rows[0] <= currentPlayList.getNextFile() && currentPlayList.getNextFile() <= rows
            currentPlayList.setNextFile(currentPlayList.getNextFile() - 1);
            HandlerProxy.getControllerHandler().getPlayListController().setSelectedSong(currentPlayL
        }
    }

    public void moveDown(int[] rows) {
        PlayList currentPlayList = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        for (int i = rows.length-1; i >= 0; i--) {
            AudioFile aux = currentPlayList.get(rows[i]);
            currentPlayList.remove(rows[i]);
            currentPlayList.add(rows[i]+1, aux);
        }
        if (rows[rows.length-1] +1 == currentPlayList.getNextFile()) {
            currentPlayList.setNextFile(currentPlayList.getNextFile() - rows.length);
            HandlerProxy.getControllerHandler().getPlayListController().setSelectedSong(currentPlayL
        } else if (rows[0] <= currentPlayList.getNextFile() && currentPlayList.getNextFile() <= rows
            currentPlayList.setNextFile(currentPlayList.getNextFile() + 1);
            HandlerProxy.getControllerHandler().getPlayListController().setSelectedSong(currentPlayL
        }
    }

    public void moveToBottom(int[] rows) {
        PlayList currentPlayList = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        int j = 0;
        for (int i = rows.length-1; i >= 0; i--) {
            AudioFile aux = currentPlayList.get(rows[i]);
            currentPlayList.remove(rows[i]);
            currentPlayList.add(currentPlayList.size() - j++, aux);
        }
        if (rows[rows.length-1] < currentPlayList.getNextFile()) {
            currentPlayList.setNextFile(currentPlayList.getNextFile() - rows.length);
            HandlerProxy.getControllerHandler().getPlayListController().setSelectedSong(currentPlayL
        } else if (rows[0] <= currentPlayList.getNextFile() && currentPlayList.getNextFile() <= rows
            currentPlayList.setNextFile(currentPlayList.getNextFile() + currentPlayList.size() - row
            HandlerProxy.getControllerHandler().getPlayListController().setSelectedSong(currentPlayL
        }
    }

    public void removeSongs() {
```

Refactored Source:

```java

    public void moveToTop(int[] rows) {
        playListHandlerOperations.moveToTop(rows);
    }

    public void moveUp(int[] rows) {
        playListHandlerOperations.moveUp(rows);
    }

    public void moveDown(int[] rows) {
        playListHandlerOperations.moveDown(rows);
    }

    public void moveToBottom(int[] rows) {
        playListHandlerOperations.moveToBottom(rows);
    }

    public void removeSongs() {
        PlayList currentPlayList = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        if (!currentPlayList.isEmpty()) {
            currentPlayList.clear();
            HandlerProxy.getPlayerHandler().stop();
            currentPlayList.setNextFile(0);
            HandlerProxy.getControllerHandler().getPlayListControlsController().enableSaveButton
            HandlerProxy.getControllerHandler().getMenuController().enableSavePlaylist(false);
            HandlerProxy.getVisualHandler().showPlaylistSongNumber(currentPlayList.size());
            logger.info("Play list clear");
        }
    }

    public void removeSongs(int[] rows) {
        PlayList currentPlayList = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        AudioFile playingSong = currentPlayList.getCurrentFile();
        boolean hasToBeRemoved = false;
        for (int i = 0; i < rows.length; i++) {
            if (rows[i] == currentPlayList.getNextFile())
                hasToBeRemoved = true;
        }
        for (int i = rows.length - 1; i >= 0; i--) {
            currentPlayList.remove(rows[i]);
```



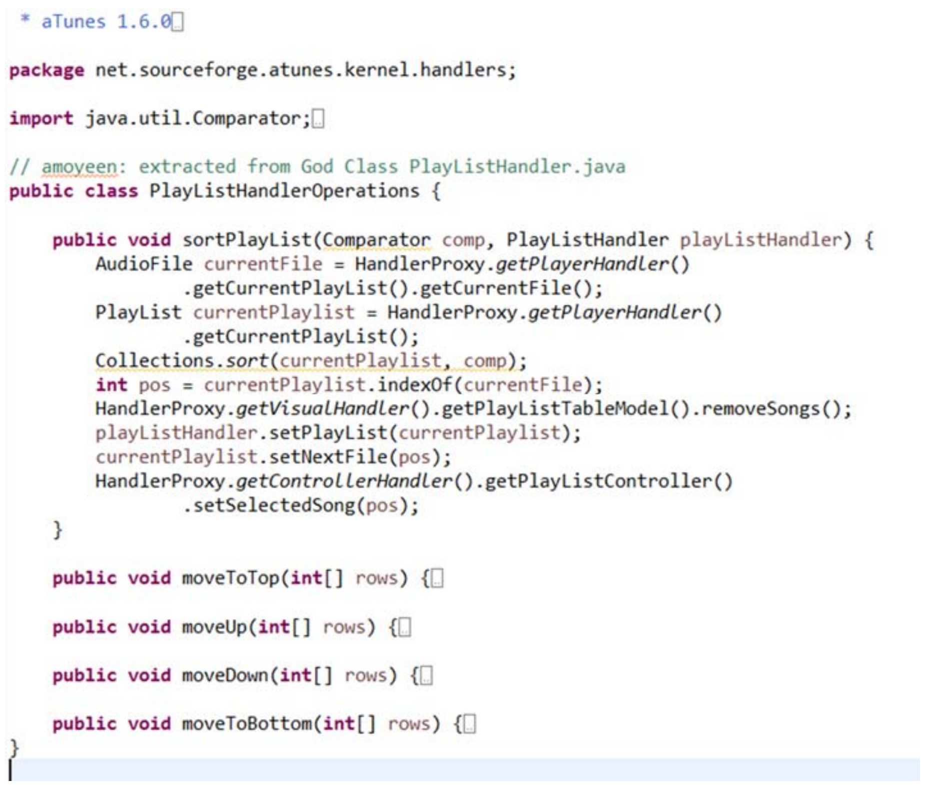
Original Source:

```java
            }
            HandlerProxy.getControllerHandler().getPlayListController().addSongsToPlayList(files, -1
            HandlerProxy.getVisualHandler().showPlaylistSongNumber(currentPlayList.size());
            logger.info(files.size() + " songs added to play list");
        }
    }

    public boolean isFiltered() {
        return nonFilteredPlayList != null;
    }

    private void setPlayList(ArrayList<AudioFile> files) {
        PlayList currentPlayList = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        if (files != null && files.size() >= 1) {
            if (currentPlayList.isEmpty()) {
                HandlerProxy.getPlayListHandler().getPlayListListener().selectedSongChanged(files.ge
            }
            HandlerProxy.getControllerHandler().getPlayListController().addSongsToPlayList(files, -1
            HandlerProxy.getVisualHandler().showPlaylistSongNumber(currentPlayList.size());
            logger.info(files.size() + " songs setted as play list");
        }
    }

    public void finish() {
```

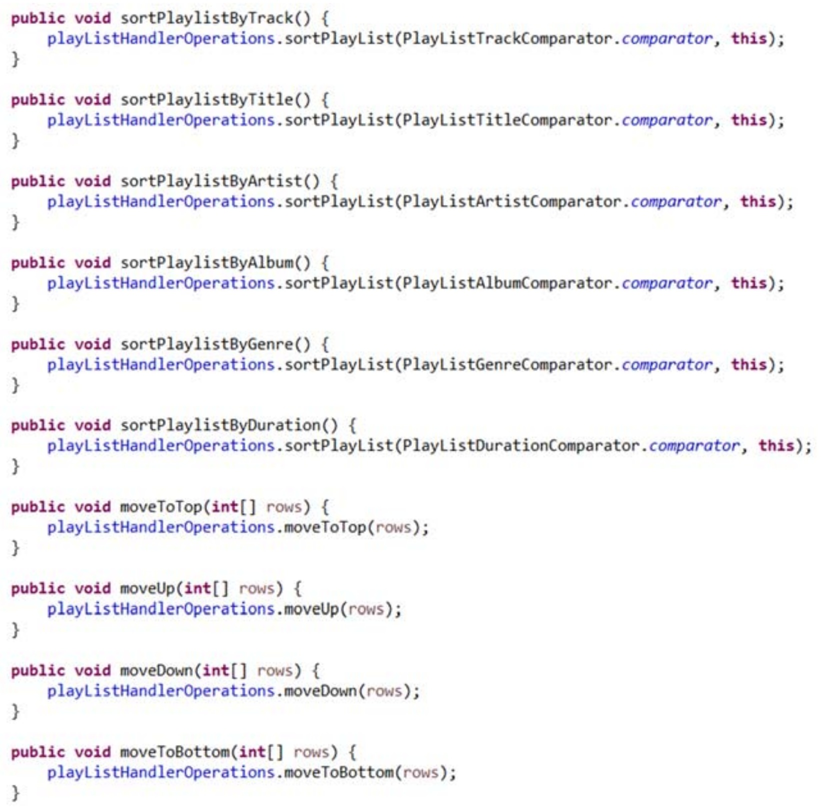
Refactored Source:

```java
            }
            HandlerProxy.getControllerHandler().getPlayListController().addSongsToPlayList(files
            HandlerProxy.getVisualHandler().showPlaylistSongNumber(currentPlayList.size());
            logger.info(files.size() + " songs added to play list");
        }
    }

    public boolean isFiltered() {
        return nonFilteredPlayList != null;
    }

    public void setPlayList(ArrayList<AudioFile> files) {
        PlayList currentPlayList = HandlerProxy.getPlayerHandler().getCurrentPlayList();
        if (files != null && files.size() >= 1) {
            if (currentPlayList.isEmpty()) {
                HandlerProxy.getPlayListHandler().getPlayListListener().selectedSongChanged(file
            }
            HandlerProxy.getControllerHandler().getPlayListController().addSongsToPlayList(files
            HandlerProxy.getVisualHandler().showPlaylistSongNumber(currentPlayList.size());
            logger.info(files.size() + " songs setted as play list");
        }
    }

    public void finish() {
```

3.1.5 Changed code



```java
 * aTunes 1.6.0

package net.sourceforge.atunes.kernel.handlers;

import java.util.Comparator;

// amoyeen: extracted from God Class PlayListHandler.java
public class PlayListHandlerOperations {

    public void sortPlayList(Comparator comp, PlayListHandler playListHandler) {
        AudioFile currentFile = HandlerProxy.getPlayerHandler()
                .getCurrentPlayList().getCurrentFile();
        PlayList currentPlaylist = HandlerProxy.getPlayerHandler()
                .getCurrentPlayList();
        Collections.sort(currentPlaylist, comp);
        int pos = currentPlaylist.indexOf(currentFile);
        HandlerProxy.getVisualHandler().getPlayListTableModel().removeSongs();
        playListHandler.setPlayList(currentPlaylist);
        currentPlaylist.setNextFile(pos);
        HandlerProxy.getControllerHandler().getPlayListController()
                .setSelectedSong(pos);
    }

    public void moveToTop(int[] rows) {

    public void moveUp(int[] rows) {

    public void moveDown(int[] rows) {

    public void moveToBottom(int[] rows) {
}
```

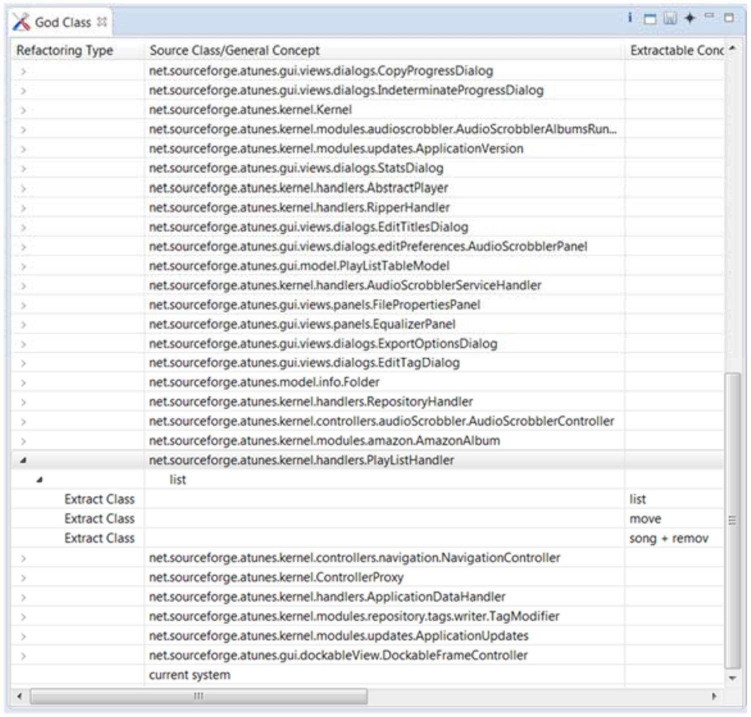


```java
    public void sortPlaylistByTrack() {
        playListHandlerOperations.sortPlayList(PlayListTrackComparator.comparator, this);
    }

    public void sortPlaylistByTitle() {
        playListHandlerOperations.sortPlayList(PlayListTitleComparator.comparator, this);
    }

    public void sortPlaylistByArtist() {
        playListHandlerOperations.sortPlayList(PlayListArtistComparator.comparator, this);
    }

    public void sortPlaylistByAlbum() {
        playListHandlerOperations.sortPlayList(PlayListAlbumComparator.comparator, this);
    }

    public void sortPlaylistByGenre() {
        playListHandlerOperations.sortPlayList(PlayListGenreComparator.comparator, this);
    }

    public void sortPlaylistByDuration() {
        playListHandlerOperations.sortPlayList(PlayListDurationComparator.comparator, this);
    }

    public void moveToTop(int[] rows) {
        playListHandlerOperations.moveToTop(rows);
    }

    public void moveUp(int[] rows) {
        playListHandlerOperations.moveUp(rows);
    }

    public void moveDown(int[] rows) {
        playListHandlerOperations.moveDown(rows);
    }

    public void moveToBottom(int[] rows) {
        playListHandlerOperations.moveToBottom(rows);
    }
```

3.1.6 Tests

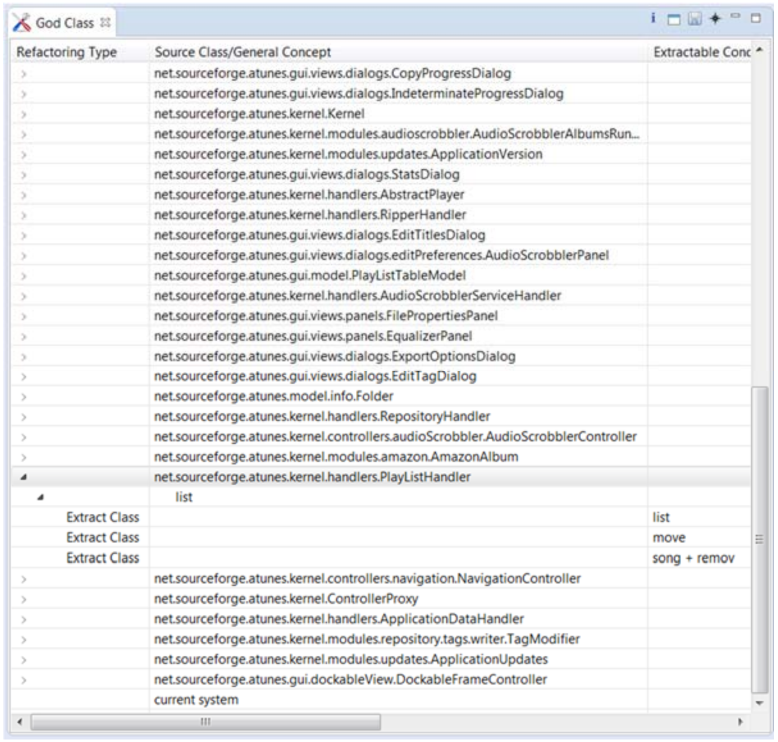
Test Case #	Description	Execution before Refactoring	Execution after Refactoring
T1	Demonstrate that program can be invoked successfully	Pass	Pass
T2	Demonstrate that songs can be added to playlist	Pass	Pass
T3	Demonstrate that playlist can be sorted by track	Pass	Pass
T4	Demonstrate that playlist can be sorted by title	Pass	Pass
T5	Demonstrate that playlist can be sorted by artist	Pass	Pass
T6	Demonstrate that playlist can be sorted by album	Pass	Pass
T7	Demonstrate that playlist can be sorted by genere	Pass	Pass
T8	Demonstrate that Move to Top moves the selected song to the top of playlist	Pass	Pass
T9	Demonstrate that Move Up moves the selected song up one place	Pass	Pass
T10	Demonstrate that Move Down moves the selected song down one place	Pass	Pass
T11	Demonstrate that Move to Bottom moves the selected song to the bottom of playlist	Pass	Pass

3.1.7 Tool result before refactoring



Refactoring Type	Source Class/General Concept	Extractable Conc
	net.sourceforge.atunes.gui.views.dialogs.CopyProgressDialog	
	net.sourceforge.atunes.gui.views.dialogs.IndeterminateProgressDialog	
	net.sourceforge.atunes.kernel.Kernel	
	net.sourceforge.atunes.kernel.modules.audioscrobbler.AudioScrobblerAlbumsRun...	
	net.sourceforge.atunes.kernel.modules.updates.ApplicationVersion	
	net.sourceforge.atunes.gui.views.dialogs.StatsDialog	
	net.sourceforge.atunes.kernel.handlers.AbstractPlayer	
	net.sourceforge.atunes.kernel.handlers.RipperHandler	
	net.sourceforge.atunes.gui.views.dialogs.EditTitlesDialog	
	net.sourceforge.atunes.gui.views.dialogs.editPreferences.AudioScrobblerPanel	
	net.sourceforge.atunes.gui.model.PlayListTableModel	
	net.sourceforge.atunes.kernel.handlers.AudioScrobblerServiceHandler	
	net.sourceforge.atunes.gui.views.panels.FilePropertiesPanel	
	net.sourceforge.atunes.gui.views.panels.EqualizerPanel	
	net.sourceforge.atunes.gui.views.dialogs.ExportOptionsDialog	
	net.sourceforge.atunes.gui.views.dialogs.EditTagDialog	
	net.sourceforge.atunes.model.info.Folder	
	net.sourceforge.atunes.kernel.handlers.RepositoryHandler	
	net.sourceforge.atunes.kernel.controllers.audioScrobbler.AudioScrobblerController	
	net.sourceforge.atunes.kernel.modules.amazon.AmazonAlbum	
	net.sourceforge.atunes.kernel.handlers.PlayListHandler	
	list	
Extract Class		list
Extract Class		move
Extract Class		song + remov
	net.sourceforge.atunes.kernel.controllers.navigation.NavigationController	
	net.sourceforge.atunes.kernel.ControllerProxy	
	net.sourceforge.atunes.kernel.handlers.ApplicationDataHandler	
	net.sourceforge.atunes.kernel.modules.repository.tags.writer.TagModifier	
	net.sourceforge.atunes.kernel.modules.updates.ApplicationUpdates	
	net.sourceforge.atunes.gui.dockableView.DockableFrameController	
	current system	

3.1.8 Tool results after refactoring



God Class

Refactoring Type	Source Class/General Concept	Extractable Conc
›	net.sourceforge.atunes.gui.views.dialogs.CopyProgressDialog	
›	net.sourceforge.atunes.gui.views.dialogs.IndeterminateProgressDialog	
›	net.sourceforge.atunes.kernel.Kernel	
›	net.sourceforge.atunes.kernel.modules.audioscrobbler.AudioScrobblerAlbumsRun...	
›	net.sourceforge.atunes.kernel.modules.updates.ApplicationVersion	
›	net.sourceforge.atunes.gui.views.dialogs.StatsDialog	
›	net.sourceforge.atunes.kernel.handlers.AbstractPlayer	
›	net.sourceforge.atunes.kernel.handlers.RipperHandler	
›	net.sourceforge.atunes.gui.views.dialogs.EditTitlesDialog	
›	net.sourceforge.atunes.gui.views.dialogs.editPreferences.AudioScrobblerPanel	
›	net.sourceforge.atunes.gui.model.PlayListTableModel	
›	net.sourceforge.atunes.kernel.handlers.AudioScrobblerServiceHandler	
›	net.sourceforge.atunes.gui.views.panels.FilePropertiesPanel	
›	net.sourceforge.atunes.gui.views.panels.EqualizerPanel	
›	net.sourceforge.atunes.gui.views.dialogs.ExportOptionsDialog	
›	net.sourceforge.atunes.gui.views.dialogs.EditTagDialog	
›	net.sourceforge.atunes.model.info.Folder	
›	net.sourceforge.atunes.kernel.handlers.RepositoryHandler	
›	net.sourceforge.atunes.kernel.controllers.audioScrobbler.AudioScrobblerController	
›	net.sourceforge.atunes.kernel.modules.amazon.AmazonAlbum	
◢	net.sourceforge.atunes.kernel.handlers.PlayListHandler	
◢	list	
Extract Class		list
Extract Class		move
Extract Class		song + remov
›	net.sourceforge.atunes.kernel.controllers.navigation.NavigationController	
›	net.sourceforge.atunes.kernel.ControllerProxy	
›	net.sourceforge.atunes.kernel.handlers.ApplicationDataHandler	
›	net.sourceforge.atunes.kernel.modules.repository.tags.writer.TagModifier	
›	net.sourceforge.atunes.kernel.modules.updates.ApplicationUpdates	
›	net.sourceforge.atunes.gui.dockableView.DockableFrameController	
	current system	

3.2 JEDIT – REMOVING "TYPE CHECKING" SMELL

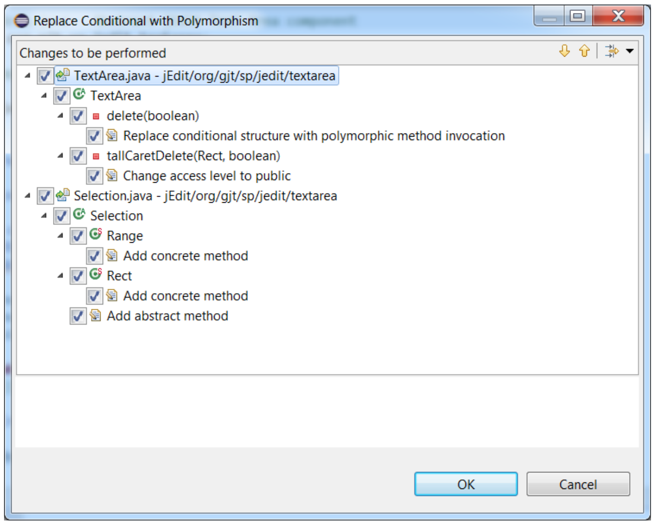
3.2.1 Justification for refactoring

- Improved code organization.
- Utilization of Object Orientation by removing Procedural elements.
- Reduction of complicated branching - simplifying the code base, making it easier to read and test.

3.2.2 Description and Rationale

A sequence of if statements in delete(boolean forward) method of TextArea.java class, where the delete operation was being performed based on different type of text selections, were replaced and removed the Type Checking/Switch Statements smell using polymorphism by adding an abstract delete() method in Selection.java abstract class and adding concrete delete() methods in both Range and Rect concrete classes that extend Selection class.

3.2.3 Code smell visualization

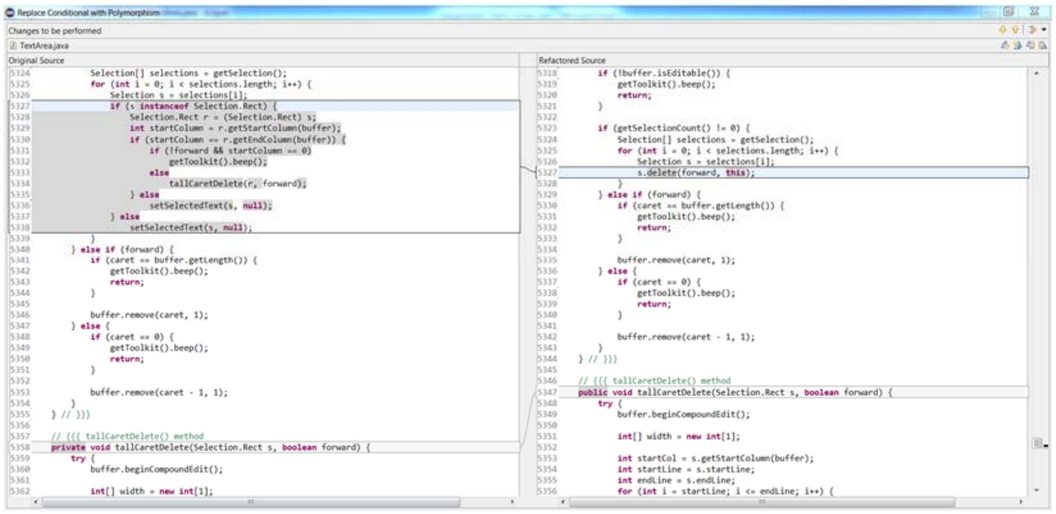


Replace Conditional with Polymorphism

Changes to be performed

- ☑ TextArea.java - jEdit/org/gjt/sp/jedit/textarea
  - ☑ TextArea
    - ☑ delete(boolean)
      - ☑ Replace conditional structure with polymorphic method invocation
    - ☑ tallCaretDelete(Rect, boolean)
      - ☑ Change access level to public
- ☑ Selection.java - jEdit/org/gjt/sp/jedit/textarea
  - ☑ Selection
    - ☑ Range
      - ☑ Add concrete method
    - ☑ Rect
      - ☑ Add concrete method
    - ☑ Add abstract method

3.2.4 Changes to be performed



Replace Conditional with Polymorphism

Changes to be performed

TextArea.java

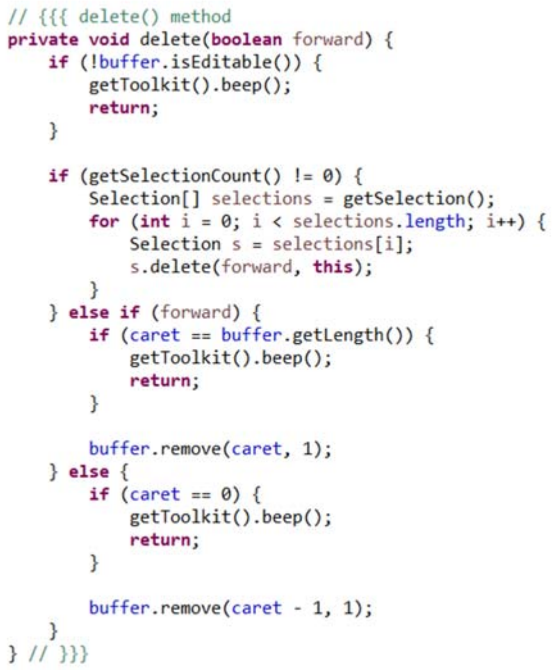
Original Source

```
            Selection[] selections = getSelection();
            for (int i = 0; i < selections.length; i++) {
                Selection s = selections[i];
                if (s instanceof Selection.Rect) {
                    Selection.Rect r = (Selection.Rect) s;
                    int startColumn = r.getStartColumn(buffer);
                    if (startColumn == r.getEndColumn(buffer)) {
                        if (!forward && startColumn == 0)
                            getToolkit().beep();
                        else
                            tallCaretDelete(r, forward);
                    } else
                        setSelectedText(s, null);
                } else
                    setSelectedText(s, null);
            }
        } else if (forward) {
            if (caret == buffer.getLength()) {
                getToolkit().beep();
                return;
            }

            buffer.remove(caret, 1);
        } else {
            if (caret == 0) {
                getToolkit().beep();
                return;
            }

            buffer.remove(caret - 1, 1);
        }
    } // }}}

    // {{{ tallCaretDelete() method
    private void tallCaretDelete(Selection.Rect s, boolean forward) {
        try {
            buffer.beginCompoundEdit();

            int[] width = new int[1];
```

Refactored Source

```
        if (!buffer.isEditable()) {
            getToolkit().beep();
            return;
        }

        if (getSelectionCount() != 0) {
            Selection[] selections = getSelection();
            for (int i = 0; i < selections.length; i++) {
                Selection s = selections[i];
                s.delete(forward, this);
            }
        } else if (forward) {
            if (caret == buffer.getLength()) {
                getToolkit().beep();
                return;
            }

            buffer.remove(caret, 1);
        } else {
            if (caret == 0) {
                getToolkit().beep();
                return;
            }

            buffer.remove(caret - 1, 1);
        }
    } // }}}

    // {{{ tallCaretDelete() method
    public void tallCaretDelete(Selection.Rect s, boolean forward) {
        try {
            buffer.beginCompoundEdit();

            int[] width = new int[1];

            int startCol = s.getStartColumn(buffer);
            int startLine = s.startLine;
            int endLine = s.endLine;
            for (int i = startLine; i <= endLine; i++) {
```

3.2.5 Changed code

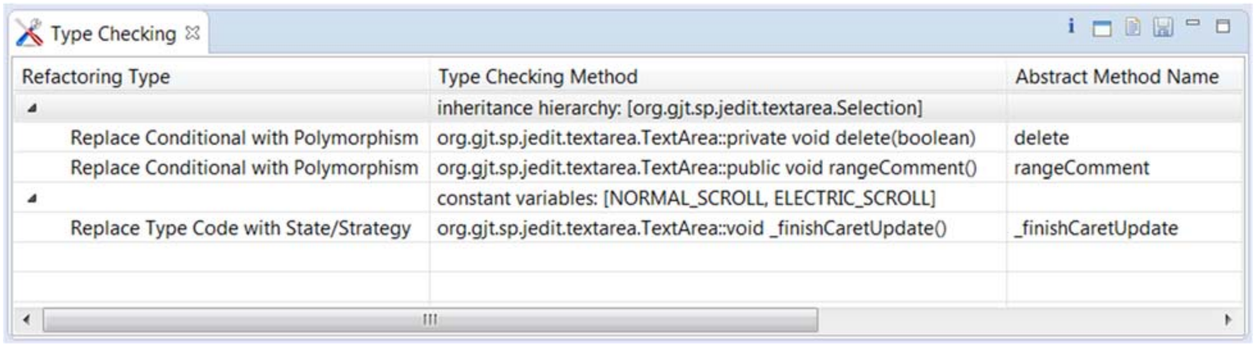


```
// {{{ delete() method
private void delete(boolean forward) {
    if (!buffer.isEditable()) {
        getToolkit().beep();
        return;
    }

    if (getSelectionCount() != 0) {
        Selection[] selections = getSelection();
        for (int i = 0; i < selections.length; i++) {
            Selection s = selections[i];
            s.delete(forward, this);
        }
    } else if (forward) {
        if (caret == buffer.getLength()) {
            getToolkit().beep();
            return;
        }

        buffer.remove(caret, 1);
    } else {
        if (caret == 0) {
            getToolkit().beep();
            return;
        }

        buffer.remove(caret - 1, 1);
    }
} // }}}
```

3.2.6 Tests

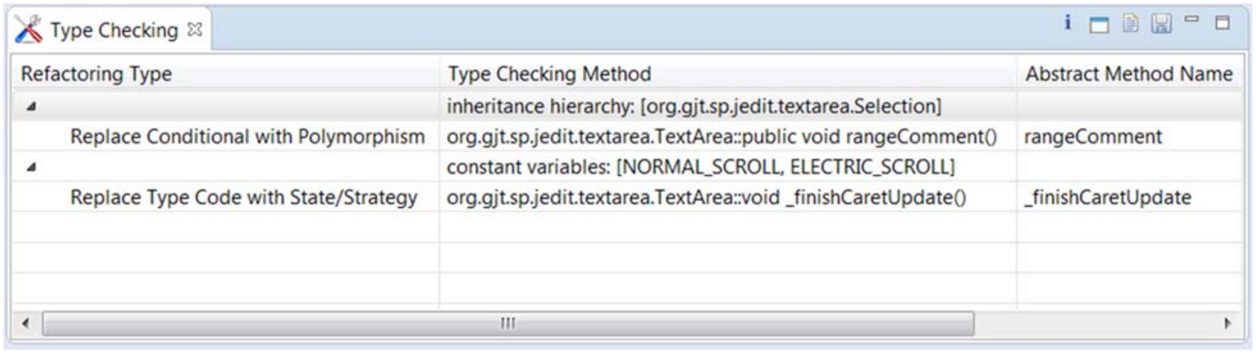
Test Case #	Description	Execution before Refactoring	Execution after Refactoring
T1	Demonstrate that program can be invoked successfully	Pass	Pass
T2	Demonstrate that an existing test file can be opened	Pass	Pass
T3	Demonstrate that part of text from open text file can be selected	Pass	Pass
T4	Demonstrate that selected text can be deleted using [Delete] key on the keyboard	Pass	Pass
T5	Demonstrate that selected text can be deleted using toolbar icon	Pass	Pass
T6	Demonstrate that selected lines can be deleted using Edit > Text > Delete Lines	Pass	Pass
T7	Demonstrate that text from current cursor position to start of line can be deleted using Edit > Text > Delete to Start of Line	Pass	Pass
T8	Demonstrate that text from current cursor position to end of line can be deleted using Edit > Text > Delete to End of Line	Pass	Pass
T9	Demonstrate that whole paragraph where cursor currently is can be deleted using Edit > Text > Delete Paragraph	Pass	Pass

3.2.7 Tool result before refactoring



Refactoring Type	Type Checking Method	Abstract Method Name
◢	inheritance hierarchy: [org.gjt.sp.jedit.textarea.Selection]	
Replace Conditional with Polymorphism	org.gjt.sp.jedit.textarea.TextArea::private void delete(boolean)	delete
Replace Conditional with Polymorphism	org.gjt.sp.jedit.textarea.TextArea::public void rangeComment()	rangeComment
◢	constant variables: [NORMAL_SCROLL, ELECTRIC_SCROLL]	
Replace Type Code with State/Strategy	org.gjt.sp.jedit.textarea.TextArea::void _finishCaretUpdate()	_finishCaretUpdate

3.2.8 Tool result after refactoring



Refactoring Type	Type Checking Method	Abstract Method Name
◢	inheritance hierarchy: [org.gjt.sp.jedit.textarea.Selection]	
Replace Conditional with Polymorphism	org.gjt.sp.jedit.textarea.TextArea::public void rangeComment()	rangeComment
◢	constant variables: [NORMAL_SCROLL, ELECTRIC_SCROLL]	
Replace Type Code with State/Strategy	org.gjt.sp.jedit.textarea.TextArea::void _finishCaretUpdate()	_finishCaretUpdate

4 MANUAL REFACTORING (TO REMOVE CODE SMELLS)

4.1 ATUNES – REMOVING “FEATURE ENVY” SMELL

4.1.1 Justification for refactoring

• Less code duplication (if the data handling code is put in a central place).

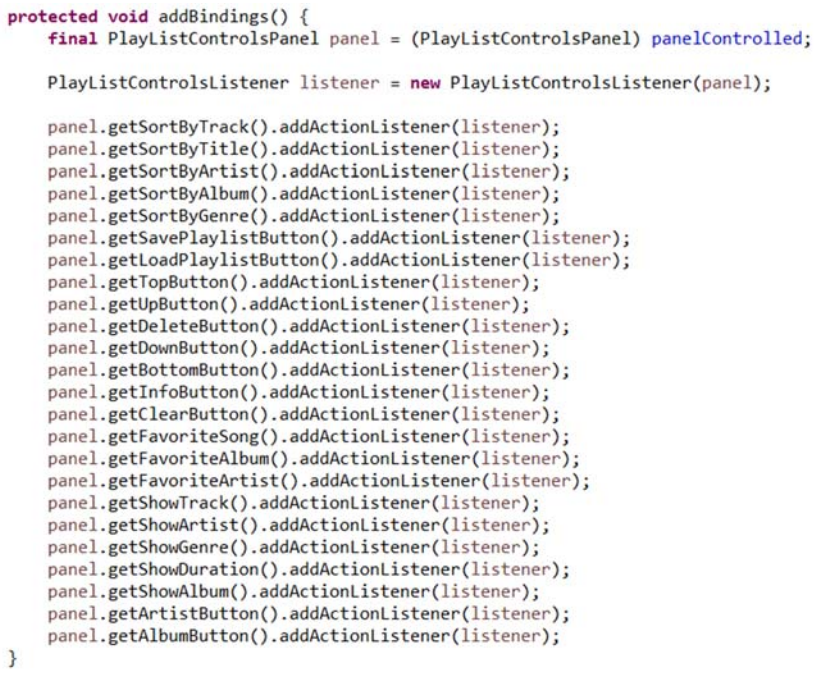
• Better code organization (methods for handling data are next to the actual data).

4.1.2 Description and rationale

Protected method addBindings() in PlayListControlsController.java was heavily using data from external class PlayListControlsPanel.java to add bindings to PlaylistControlsListener.java class. So, we have created a public method addBindings(PlaylistControlsListener listener) in PlayListControlsPanel.java class that takes an instance of PlaylistControlsListener as an input and adds the bindings to its properties accordingly - which then is called from PlayListControlsController.java.

4.1.3 Code before refactoring

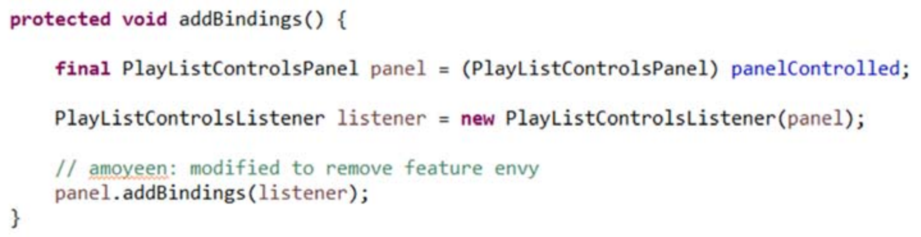
/aTunes/src/net/sourceforge/atunes/kernel/controllers/playListControls/PlayListControlsController.java



```
protected void addBindings() {
    final PlayListControlsPanel panel = (PlayListControlsPanel) panelControlled;

    PlayListControlsListener listener = new PlayListControlsListener(panel);

    panel.getSortByTrack().addActionListener(listener);
    panel.getSortByTitle().addActionListener(listener);
    panel.getSortByArtist().addActionListener(listener);
    panel.getSortByAlbum().addActionListener(listener);
    panel.getSortByGenre().addActionListener(listener);
    panel.getSavePlaylistButton().addActionListener(listener);
    panel.getLoadPlaylistButton().addActionListener(listener);
    panel.getTopButton().addActionListener(listener);
    panel.getUpButton().addActionListener(listener);
    panel.getDeleteButton().addActionListener(listener);
    panel.getDownButton().addActionListener(listener);
    panel.getBottomButton().addActionListener(listener);
    panel.getInfoButton().addActionListener(listener);
    panel.getClearButton().addActionListener(listener);
    panel.getFavoriteSong().addActionListener(listener);
    panel.getFavoriteAlbum().addActionListener(listener);
    panel.getFavoriteArtist().addActionListener(listener);
    panel.getShowTrack().addActionListener(listener);
    panel.getShowArtist().addActionListener(listener);
    panel.getShowGenre().addActionListener(listener);
    panel.getShowDuration().addActionListener(listener);
    panel.getShowAlbum().addActionListener(listener);
    panel.getArtistButton().addActionListener(listener);
    panel.getAlbumButton().addActionListener(listener);
}
```

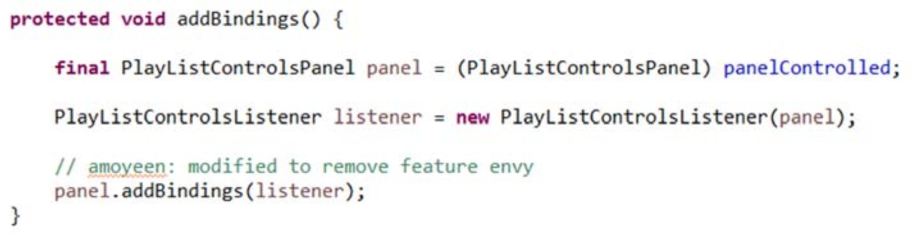
4.1.4 Refactored code

/aTunes/src/net/sourceforge/atunes/kernel/controllers/playListControls/PlayListControlsController.java



```
protected void addBindings() {

    final PlayListControlsPanel panel = (PlayListControlsPanel) panelControlled;

    PlayListControlsListener listener = new PlayListControlsListener(panel);

    // amoyeen: modified to remove feature envy
    panel.addBindings(listener);
}
```

/aTunes/src/net/sourceforge/atunes/gui/views/panels/PlayListControlsPanel.java



```
protected void addBindings() {

    final PlayListControlsPanel panel = (PlayListControlsPanel) panelControlled;

    PlayListControlsListener listener = new PlayListControlsListener(panel);

    // amoyeen: modified to remove feature envy
    panel.addBindings(listener);
}
```

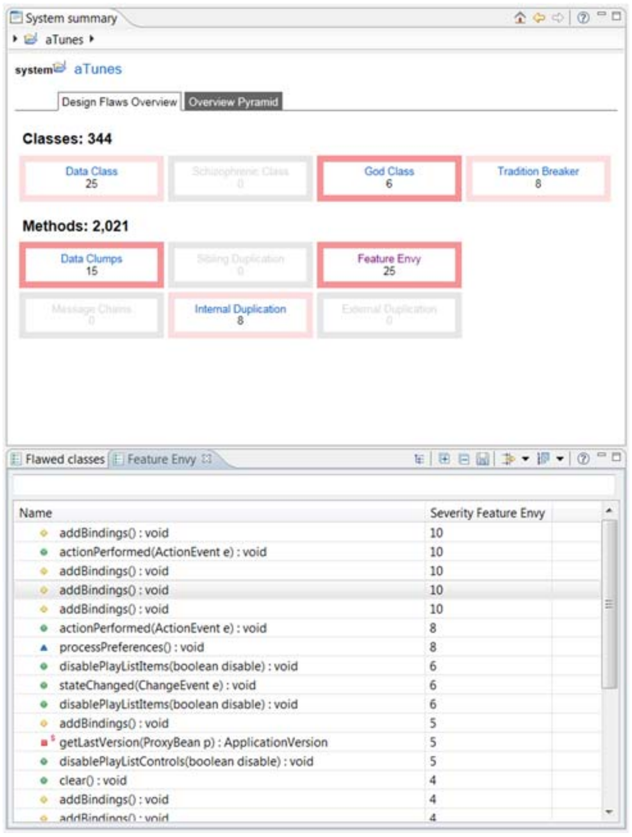
4.1.5 Tests

Test Case #	Description	Execution before Refactoring	Execution before Refactoring
T1	Demonstrate that program can be invoked successfully	Pass	Pass
T2	Demonstrate that songs can be added to playlist	Pass	Pass
T3	Demonstrate that playlist can be sorted by track	Pass	Pass
T4	Demonstrate that playlist can be sorted by title	Pass	Pass
T5	Demonstrate that playlist can be sorted by artist	Pass	Pass
T6	Demonstrate that playlist can be sorted by album	Pass	Pass
T7	Demonstrate that playlist can be sorted by genere	Pass	Pass
T8	Demonstrate that Save Playlist invokes save dialog box	Pass	Pass
T9	Demonstrate that Load Playlist invokes load dialog box	Pass	Pass
T10	Demonstrate that Move to Top moves the selected song to the top of playlist	Pass	Pass
T11	Demonstrate that Move Up moves the selected song up one place	Pass	Pass
T12	Demonstrate that Remove deletes the selected song from playlist	Pass	Pass
T13	Demonstrate that Move Down moves the selected song down one place	Pass	Pass
T14	Demonstrate that Move to Bottom moves the selected song to the bottom of playlist	Pass	Pass
T15	Demonstrate that Info invokes info dialog box	Pass	Pass
T16	Demonstrate that Clear Playlist deletes all songs from playlist	Pass	Pass
T17	Demonstrate that Set as Favorite Song sets selected song as favorite song	Pass	Pass
T18	Demonstrate that Set as Favorite Album sets album of selected song as favorite album	Pass	Pass
T19	Demonstrate that Set as Favorite Artist sets artist of selected song as favorite artist	Pass	Pass

Test Case #	Description	Execution before Refactoring	Execution before Refactoring
T20	Demonstrate that Options > Show Track Number checkbox can show/hide track number column in playlist	Pass	Pass
T21	Demonstrate that Options > Show Artist checkbox can show/hide artist column in playlist	Pass	Pass
T22	Demonstrate that Options > Show Genre checkbox can show/hide genre column in playlist	Pass	Pass
T23	Demonstrate that Options > Show Duration checkbox can show/hide duration column in playlist	Pass	Pass
T24	Demonstrate that Options > Show Album checkbox can show/hide album column in playlist	Pass	Pass
T25	Demonstrate that Set Artist as Playlist sets only songs from artist of selected song as playlist	Pass	Pass
T26	Demonstrate that Set Album as Playlist sets only album of selected song as playlist	Pass	Pass

4.1.6 Tool results

Tool result before refactoring


System summary
aTunes
system aTunes
Design Flaws Overview | Overview Pyramid

Classes: 344

Data Class 25 | Schizophrenic Class 0 | God Class 6 | Tradition Breaker 8

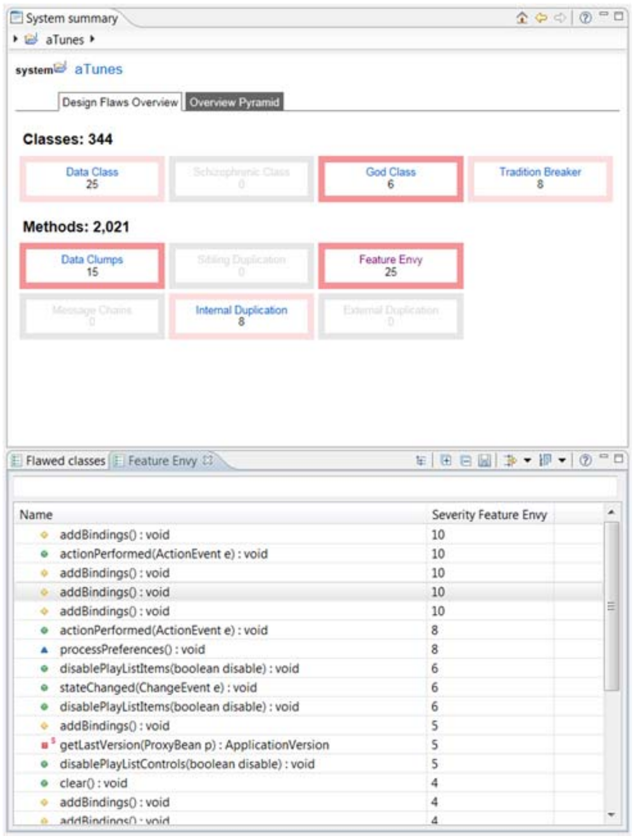
Methods: 2,021

Data Clumps 15 | Sibling Duplication 0 | Feature Envy 25
Message Chains 0 | Internal Duplication 8 | External Duplication 0

Flawed classes | Feature Envy

Name	Severity Feature Envy
addBindings() : void	10
actionPerformed(ActionEvent e) : void	10
addBindings() : void	10
addBindings() : void	10
addBindings() : void	10
actionPerformed(ActionEvent e) : void	8
processPreferences() : void	8
disablePlayListItems(boolean disable) : void	6
stateChanged(ChangeEvent e) : void	6
disablePlayListItems(boolean disable) : void	6
addBindings() : void	5
getLastVersion(ProxyBean p) : ApplicationVersion	5
disablePlayListControls(boolean disable) : void	5
clear() : void	4
addBindings() : void	4
addBindings() : void	4

Tool result after refactoring


System summary
aTunes
system aTunes
Design Flaws Overview | Overview Pyramid

Classes: 344

Data Class 25 | Schizophrenic Class 0 | God Class 6 | Tradition Breaker 8

Methods: 2,021

Data Clumps 15 | Sibling Duplication 0 | Feature Envy 25
Message Chains 0 | Internal Duplication 8 | External Duplication 0

Flawed classes | Feature Envy

Name	Severity Feature Envy
addBindings() : void	10
actionPerformed(ActionEvent e) : void	10
addBindings() : void	10
addBindings() : void	10
addBindings() : void	10
actionPerformed(ActionEvent e) : void	8
processPreferences() : void	8
disablePlayListItems(boolean disable) : void	6
stateChanged(ChangeEvent e) : void	6
disablePlayListItems(boolean disable) : void	6
addBindings() : void	5
getLastVersion(ProxyBean p) : ApplicationVersion	5
disablePlayListControls(boolean disable) : void	5
clear() : void	4
addBindings() : void	4
addBindings() : void	4

4.2 jEdit – Removing "Internal duplication" smell

4.2.1 Justification for refactoring

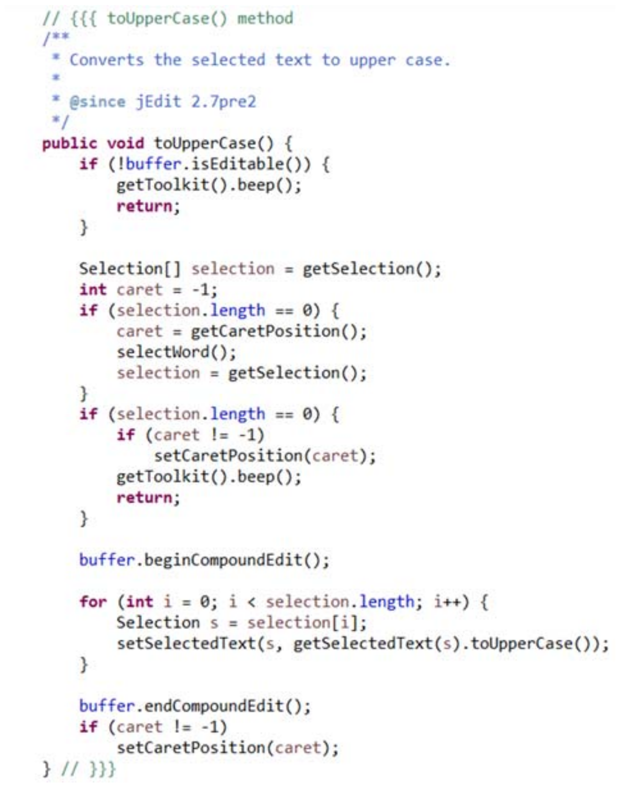
- Merging duplicate code simplifies the structure of the code and makes it shorter.
- Simplification + Shortness = code that is easier to simplify and cheaper to support.

4.2.2 Description and Rationale

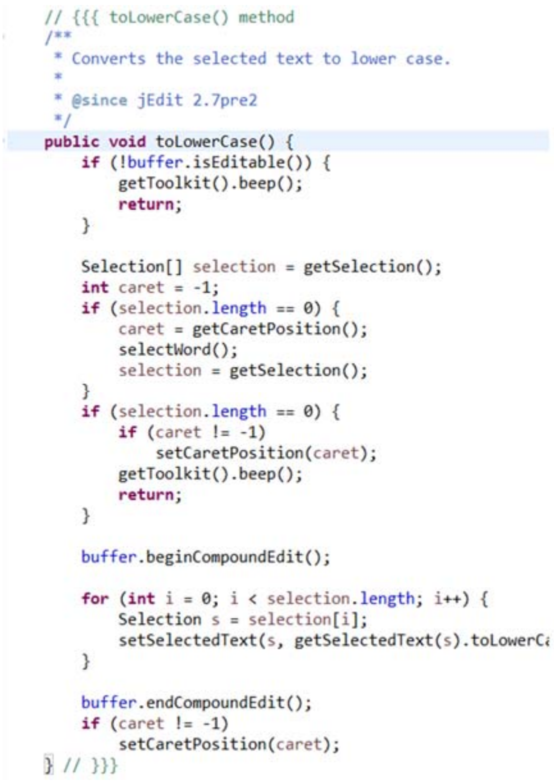
Public methods toUpperCase() and toLowerCase() in TextArea.java were exactly identical except just in one line where they were calling different String extensions of toUpperCase() and toLowerCase() respectively. So, we created a private method changeCase(String toCase) with an input parameter that tells which case to change to - and called that private method from the formers with appropriate inputs.

4.2.3 Code before refactoring

/jEdit/org/gjt/sp/jedit/textarea/TextArea.java

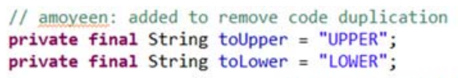


```
// {{{ toUpperCase() method
/**
 * Converts the selected text to upper case.
 *
 * @since jEdit 2.7pre2
 */
public void toUpperCase() {
    if (!buffer.isEditable()) {
        getToolkit().beep();
        return;
    }

    Selection[] selection = getSelection();
    int caret = -1;
    if (selection.length == 0) {
        caret = getCaretPosition();
        selectWord();
        selection = getSelection();
    }
    if (selection.length == 0) {
        if (caret != -1)
            setCaretPosition(caret);
        getToolkit().beep();
        return;
    }

    buffer.beginCompoundEdit();

    for (int i = 0; i < selection.length; i++) {
        Selection s = selection[i];
        setSelectedText(s, getSelectedText(s).toUpperCase());
    }

    buffer.endCompoundEdit();
    if (caret != -1)
        setCaretPosition(caret);
} // }}}
```

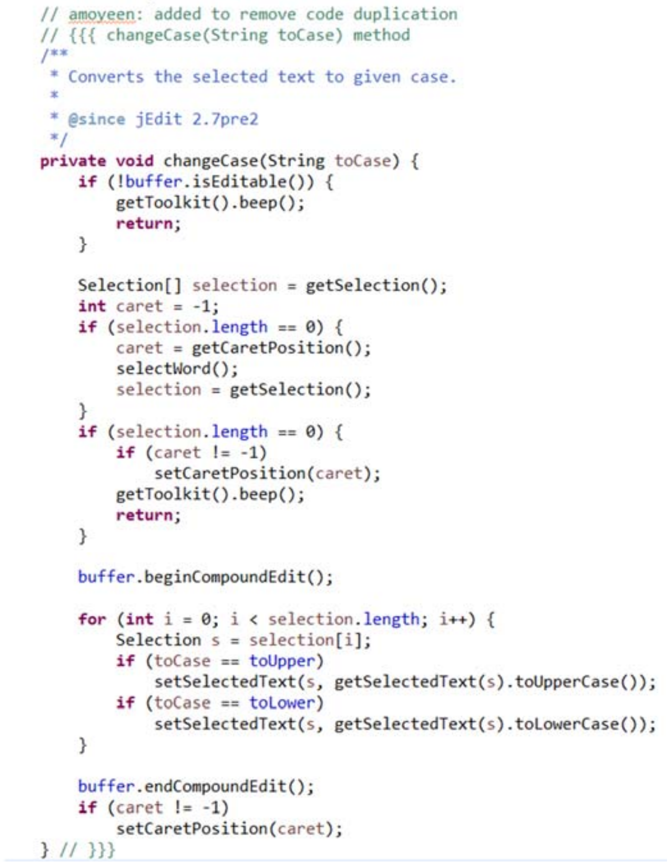


```
// {{{ toLowerCase() method
/**
 * Converts the selected text to lower case.
 *
 * @since jEdit 2.7pre2
 */
public void toLowerCase() {
    if (!buffer.isEditable()) {
        getToolkit().beep();
        return;
    }

    Selection[] selection = getSelection();
    int caret = -1;
    if (selection.length == 0) {
        caret = getCaretPosition();
        selectWord();
        selection = getSelection();
    }
    if (selection.length == 0) {
        if (caret != -1)
            setCaretPosition(caret);
        getToolkit().beep();
        return;
    }

    buffer.beginCompoundEdit();

    for (int i = 0; i < selection.length; i++) {
        Selection s = selection[i];
        setSelectedText(s, getSelectedText(s).toLowerC
    }

    buffer.endCompoundEdit();
    if (caret != -1)
        setCaretPosition(caret);
} // }}}
```

4.2.4 Refactored code



```
// amoyeen: added to remove code duplication
private final String toUpper = "UPPER";
private final String toLower = "LOWER";
```



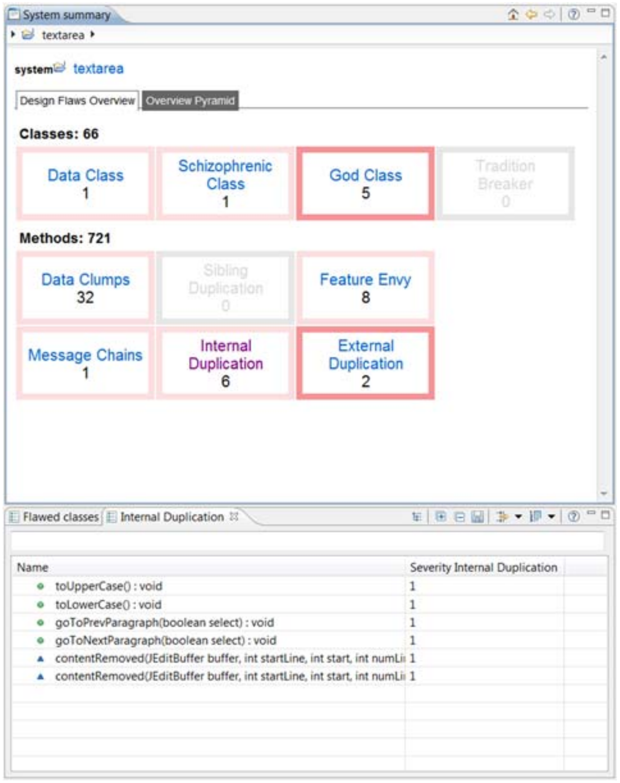
```
// amoyeen: added to remove code duplication
// {{{ changeCase(String toCase) method
/**
 * Converts the selected text to given case.
 *
 * @since jEdit 2.7pre2
 */
private void changeCase(String toCase) {
    if (!buffer.isEditable()) {
        getToolkit().beep();
        return;
    }

    Selection[] selection = getSelection();
    int caret = -1;
    if (selection.length == 0) {
        caret = getCaretPosition();
        selectWord();
        selection = getSelection();
    }
    if (selection.length == 0) {
        if (caret != -1)
            setCaretPosition(caret);
        getToolkit().beep();
        return;
    }

    buffer.beginCompoundEdit();

    for (int i = 0; i < selection.length; i++) {
        Selection s = selection[i];
        if (toCase == toUpper)
            setSelectedText(s, getSelectedText(s).toUpperCase());
        if (toCase == toLower)
            setSelectedText(s, getSelectedText(s).toLowerCase());
    }

    buffer.endCompoundEdit();
    if (caret != -1)
        setCaretPosition(caret);
} // }}}
```

4.2.5 Tests

Test Case #	Description	Execution before Refactoring	Execution before Refactoring
T1	Demonstrate that program can be invoked successfully	Pass	Pass
T2	Demonstrate that an existing test file can be opened	Pass	Pass
T3	Demonstrate that part of text from open text file can be selected	Pass	Pass
T4	Demonstrate that selected text can be converted to Upper Case using Edit > Text > To Upper Case menu item	Pass	Pass
T5	Demonstrate that selected text can be converted to Lower Case using Edit > Text > To Lower Case menu item	Pass	Pass

4.2.6 Tool results

Tool result before refactoring



Classes: 66

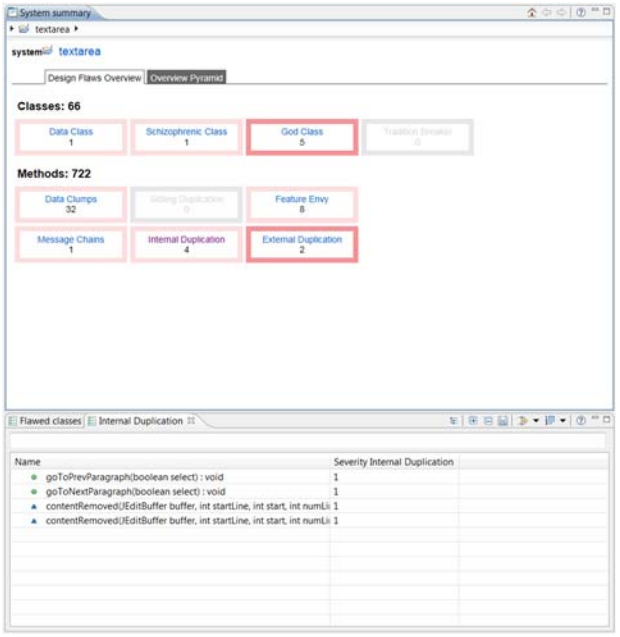
Data Class 1
Schizophrenic Class 1
God Class 5
Tradition Breaker 0

Methods: 721

Data Clumps 32
Sibling Duplication 0
Feature Envy 8
Message Chains 1
Internal Duplication 6
External Duplication 2

Name	Severity Internal Duplication
toUpperCase() : void	1
toLowerCase() : void	1
goToPrevParagraph(boolean select) : void	1
goToNextParagraph(boolean select) : void	1
contentRemoved(JEditBuffer buffer, int startLine, int start, int numLi	1
contentRemoved(JEditBuffer buffer, int startLine, int start, int numLi	1

Tool result after refactoring



Classes: 66

Data Class 1
Schizophrenic Class 1
God Class 5
Tradition Breaker 0

Methods: 722

Data Clumps 32
Sibling Duplication 0
Feature Envy 8
Message Chains 1
Internal Duplication 4
External Duplication 2

Name	Severity Internal Duplication
goToPrevParagraph(boolean select) : void	1
goToNextParagraph(boolean select) : void	1
contentRemoved(JEditBuffer buffer, int startLine, int start, int numLi	1
contentRemoved(JEditBuffer buffer, int startLine, int start, int numLi	1

5 REFERENCES

Bad Smells detailed descriptions and solutions. (n.d.). Retrieved from https://sourcemaking.com/refactoring/smells

InCode User Documentation. (n.d.). Retrieved from https://www.intooitus.com/products/incode

Source Meter for Java User Guide. (n.d.). Retrieved from https://www.sourcemeter.com/resources/java/