NORTHWEST NAZARENE UNIVERSITY

foxKeeper File Management System

THESIS
Submitted to the Department of Mathematics and Computer Science
In partial fulfillment of the requirements for the degree of
BACHELOR OF ARTS

Amanda Joy Panell
2017
THESIS
Submitted to the Department of Mathematics and Computer Science
In partial fulfillment of the requirements for the degree of
BACHELOR OF ARTS

Amanda Joy Panell
2017

foxKeeper File Management System

Author: Amanda Joy Panell

Approved: Barry Myers, Ph.D., Chair,
Department of Mathematics and Computer Science, Faculty Advisor

Approved: Michael Harrold-Panell, Product and Quality Manager,
Gompf Brackets, Second Reader

Approved: Barry Myers, Ph.D., Chair,
Department of Mathematics and Computer Science
Abstract

foxKeeper File Management System.

PANELL, AMANDA (Department of Mathematics & Computer Science), MYERS, DR.
BARRY (Department of Mathematics & Computer Science).

foxKeeper File Management System is a program developed for the use of Gompf Brackets Inc. It creates a directory of the files and subfolders within a user provided base folder and provides the user access to them through a Graphical User Interface. Through this interface, the user may select files to open from a list of customizable categories. PDFs and text files may be opened within foxKeeper if the user chooses; otherwise, they are opened in their default applications along with all other types of files. foxKeeper was written in Java using IntelliJ IDEA. PDF files are displayed using an external open source Java library called PDF Render. The goal of this project is to make it easier for employees of Gompf Brackets to navigate through and documents, and also to track changes made to those documents. Planned improvements include adding the ability to sort file lists, to export a log of file changes to Excel spreadsheets, and the elimination of rendering errors when displaying PDFs within the program.
Acknowledgments

I would like to thank my friends and family who have supported me and kept me sane while working on this project, especially my mom, Janette Panell. I like to thank my Dad for providing me with a project idea and a way to implement it, and for guiding me through the continued development of foxKeeper. I would also like to thank my study partners Aaron Ewing and Patrick Richardson, and our TA Liam Johnston for helping me learn to program in Java in Event Driven Programming, and Jonathan Hamilton for inspiring me to change my major to Computer Science. Finally, I would like to thank my CS professors, Scott Shimanski, Dale Hamilton, and Dr. Barry Myers for mentoring me, being patient with me, and for making Computer Science the best major that NNU has to offer.
# Table of Contents

Title Page...................................................................................................................................................... i
Thesis ............................................................................................................................................................ ii
Abstract........................................................................................................................................................ iii
Acknowledgments ........................................................................................................................................ iv
Overview..................................................................................................................................................... 1
Background................................................................................................................................................... 1
Research........................................................................................................................................................ 2
Implementation............................................................................................................................................. 2
    GUI.java.................................................................................................................................................... 2
    Figure 1- The main menu...................................................................................................................... 4
PDF Renderer............................................................................................................................................ 5
    Settings.java.............................................................................................................................................. 5
    Figure 2- The setup dialog box ............................................................................................................. 6
FileInOut.java ........................................................................................................................................... 6
Challenges and Future Work ........................................................................................................................ 7
    Challenges................................................................................................................................................. 7
    Figure 3- PDF Rendering Error on An Older Operating System.......................................................... 8
    Figure 4 – Installation Instructions for foxKeeper 1.0........................................................................... 9
Future Work ............................................................................................................................................. 9
    Figure 5 – Documents Menu (Version 1.0) ........................................................................................ 10
    Figure 6 – Documents Menu Design-Still-In-Progress (Version 2.0)................................................ 10
Conclusion ................................................................................................................................................ 11
Citations .................................................................................................................................................... 11
Appendix A: Glossary................................................................................................................... 13
Appendix B: foxKeeper source code ............................................................................................ 15
    mainMenu.java........................................................................................................................................ 15
    menuOptions.java................................................................................................................................... 15
    menuFiles.java......................................................................................................................................... 15
    FileInOut.java ........................................................................................................................................ 16
    PDFViewer.java...................................................................................................................................... 16
    Settings.java............................................................................................................................................ 17
    GUI.java.................................................................................................................................................. 22
Overview

foxKeeper File Management System was developed for Gompf Brackets Inc. in Mukilteo, Washington. It is a Java desktop application that allows users to access files sorted into various categories. Additional features include a PDF viewer and a basic text editor. foxKeeper is a work in progress. However, this paper will detail the research, planning, and implementation process from the beginning of the project in the Spring of 2016 to January 2017.

Background

Gompf Brackets, Inc is a small company located in Mukilteo, Washington which creates brackets and other products. Gompf has around 20 to 25 permanent employees, and they also hire temporary employees who can potentially work for the company as little as two or three hours. They currently use Adobe Acrobat Reader to view and send work instructions and job aids in the form of PDF files to their employees.

This system has some flaws. A menu that lists available files and links to open those files within Adobe Reader exists, but it is not automatically generated. Some of the employees find that there are too many steps involved in opening a file using Acrobat Reader and are frustrated by it. There are no security measures in place to make sure that employees only have access to appropriate files, and there is nothing to prevent people who are not employees from reading or downloading files.

The goal of foxKeeper File Management System is to provide employees of Gompf Brackets with a user-friendly, secure, and trackable way to access appropriate files and to limit access to files that are unnecessary. This project focused mainly on the first of these goals.
Research

The research phase of this project included a tour of Gompf Brackets, a review of their current system, and a study of event-driven programming in Java. The study of event-driven programming was ongoing throughout the project. Stack Overflow proved to be a valuable resource both for the research stage and for the debugging stage of the project.

Implementation

GUI.java

GUI is a class that creates the Graphical User Interface for foxKeeper and handles all events. Its constructor takes a Settings object as a parameter and uses it to set up the main menu, all the category menus, the menu bar and the banner image by calling the settings.setSettings() function for the Settings object.

Elements of the GUI include

1. The banner image, displaying a logo or banner that represents the company that is tuning foxKeeper.
3. The main menu, which displays a list of every document category. When a user clicks on a list item, the document menu for that category is launched.
4. Document menus, which display a list every document in a category. When a user clicks on a list item that document can be opened for viewing.

Creating the menu bar and displaying the banner image are simple tasks that require no string manipulation and do not require much code. In the menu bar, the home button is a picture
of a little house and returns the user to the main menu when pressed. The "File" submenu contains options to create a new .pdf or .txt file, open a file, or view/edit file settings. If the file a user chose to open is a pdf file, they will be given the option to open it in foxKeeper, or in its default program, and if it is a txt file, the file will be opened in foxKeeper. All other file types will be opened by their default program. "Settings" allows you to enable or disable full-screen mode. "About" launches a dialog box that provides some information about when and why foxKeeper was created, and by who. The "Help" submenu launches the "Contact" dialog box, which provides contact information for technical support, and the "Help" dialog box, with instructions and troubleshooting tips. Finally, "Exit" lets you exit the program. Functionality for the buttons to create a new pdf or txt file, and for the "Help" button, will be added in future versions of foxKeeper.

Generating the main menu is also straightforward, but more code is needed to make it look nice and add functionality to its buttons than is required for the menu bar. First, a JPanel is created to hold the menu. Then, for each category specified by the settings object a new button is created, certain design elements are manipulated to give it the right look, and an action listener is created for the button. That action listener opens the category menu associated with the new button in a separate tab if that button is clicked. Finally, each button is added to the main menu's JPanel.
Creating the category menus is more complicated. A JPanel is created for each category, but before any buttons can be added to it, the category and its "rules" need to be defined. How do we know what files belong to which category? In the current version of foxKeeper, files must be named according to a very specific format using a hyphen as a delimiter. Users specify what that naming format is when the program is first set up (see Settings.Java). Each category must have an abbreviation associated with it, and that must be included in the file name. The user will also provide a list of abbreviations during set up. foxKeeper will parse each file name to find its abbreviation and then use the abbreviations list to determine to which category it belongs. Next, a button is created for each file and labeled with a shortened file name, parsed from the full one.
The design elements are manipulated for each button, and an action listener is added that will call the `openPDFWithOptions()` function if a button is pressed.

`openPDFWithOptions()` requires a file path and the title of its document as parameters. It launches a dialog box that asks the user if they would like to open the file within foxKeeper's GUI, or in the user's default application for PDF files. If the default option is chosen, that application is launched. If the open within foxKeeper option is chosen, then a new tab is opened to display the file (assumed to be a PDF in this version of foxKeeper) using PDF renderer.

**PDF Renderer**

PDF Renderer is an open source program that allows foxKeeper to open and display PDF files. Unfortunately, this free service has some problems which will be listed later in the "Challenges and Future Work" section.

**Settings.java**

foxKeeper gets all information needed to display the GUI and provide access to files from a file called `settings.txt`. When a new GUI object is created, it accepts a Settings object as its only parameter, and one of the first actions executed by its constructor is to call the `setSettings()` function of that object.

`setSettings()` contains a try-catch block that attempts to parse the user's information from a file called `settings.txt`. If `settings.txt` cannot be found, or it is empty, the `setUp()` function is called. `setUp()` creates and launches the "Setup dialog" box, which contains a form for the user to fill out and provide foxKeeper with all the information it needs to generate menus for the GUI.
Figure 2 - The setup dialog box

After the `setup()` function is finished executing, it calls the `update()` method. First, `update()` checks to see if `settings.txt` exists. If it does exist, `update()` deletes that file, replaces it with a new one and writes all the users information to that file for quick access the next time someone opens the program. If it does not exist, `update()` creates the new file and writes all the users information to it the same way it would have if the file had already existed. Once `settings.txt` has been created or read, parsed, and updated, the program now has all the information it needs to begin generating the GUI.

**FilelnOut.java**

The first step in creating the software behind foxKeeper's GUI was to find a way to obtain a list of every file the user wanted foxKeeper to have access to and store them for later use. `FilelnOut` is a class that creates a linked list of every file within a specified folder, regardless of any sub-folders.

The `FilelnOut` class has two variables - the "folders" linked list and the "files" linked list. Its constructor takes one parameter, a File object called "folder." When `FilelnOut`'s constructor
is initiated, "folder" is added to the "folders" list and then the recursive method checkFile () is called. The first time checkFile () is called, "folder" is the user specified base-folder that was provided to the constructor.

First, "folder" is added to the "folders" linked list. Next, each File object within "folder" is tested to determine if it is a true file, or if it is really a directory (folder). This is done using the Boolean File class methods isFile () and isDirectory (). If the File is a file, it is added to the "file" linked list. If the File is a directory, then checkFile () is recursively called with the File as its parameter.

After the initial call of checkFile () has been executed, the "folders" linked list contains each of the base folder's subfolders, and the "files" linked list contains each of the base folder's files, including those within subfolders.

Challenges and Future Work

Challenges

One of the first major challenges of this project was a question ban from Stack Overflow. The developer's account was temporally banned from asking questions after asking a question related to displaying images in a Java GUI. They disagreed with another user about a solution to their problem and that user responded by serial down-voting many of the developer's other questions and answers, which resulted in the temporary ban. This prevented the developer from getting personalized support for her questions from Stack Overflow for several weeks until the ban was lifted.

An issue that appeared later in the implementation process was the quality of images displayed using the PDF Renderer library. Sometimes when files are being displayed, parts of the file appear to be missing or are scattered or warped in some way. Often, when going through the
pages of a PDF, some will not be rendered at all. It is a rare event that any one file is ever
displayed properly with no visual issues. This makes one of the main functions of foxKeeper
almost entirely useless. Rendering issues appear to be more common, and more obvious on older
operating systems.

![Figure 3- PDF Rendering Error on An Older Operating System](image)

Finally, foxKeeper does not currently have any installation system. Currently, each user
must create a folder called "foxKeeperO. l" in his or her program files folder and alter its security
permissions so that foxKeeper can have access to it. That is a potential security risk and is
inconvenient to the user, especially if foxKeeper is to be installed on more than one computer
within an organization.
Steps to "Installing" this version of FoxKeeper:
1. Make sure you have Java installed on your computer
2. Change the file extension of FoxKeeper 1.0.txt to .jar
3. Create a folder in Program Files called "foxKeeper 0.1"
4. Within "C:\Program Files\foxKeeper 0.1" right click and select properties
5. Under the "security" tab allow full control to all users (I'm not sure which "user" foxKeeper counts as)
6. Launch the application. The setup menu should pop up, and you may need to re-launch the application.
7. After the initial set-up you should be able to run the application.

Example settings folder:
Gompf Brackets .inc
C:\Gompf_files\Current\Work Instructions\PDF
OMS,Attachments,External Docs,Forms,HR,Procedures.
References,Templates,Work Instructions,Manufacturing Instructions
OMS,Attachments,E:\Internal Docs,Forms,HR,Procedures.
References,Templates,Work Instructions,Manufacturing Instructions
ALL C:\Gompf_files\Current\Work Instructions\PDF
- category parent doc title .type
C:\GompC_files\images\gompBanner.jpg

Figure 4 - Installation Instructions for foxKeeper 1.0

Future Work

foxKeeper is currently in its first version, version 1.0. At least two major updates are planned for the near future. Version 2.0 will be released in May 2017, and version 3.0 will be released later in the year. A mobile application is being planned, as well as a web portal for limited document access outside of a company's Local Area Network.

Version 2.0 is currently in development will fix the problems addressed in the Challenges section by finding a new PDF library to display PDF files and adding a proper installation system. The incomplete category menu design for the GUI will be finished, and sorting features will be added. Finally, a login system will be added to limit user access to sensitive files. The login system is a continuation of the work of a group project for the Database Design and Programming class at NNU, as taught by Dr. Barry Myers. The project was completed in April 2017 by Patrick Richardson, Jonathan Branham, and Amanda Panell.
Version 3.0 will fully integrate foxKeepers ability to connect to a database by eliminating the settings.txt file and instead, pulling information from a database. New file information will also be stored, including when a file was last modified, when it was last modified by foxKeeper, and, in the case of the latter, by which user. An administrative or audit-level user will be able to request user activity and file modification reports, and the administrative user will be able to access the database using SQL queries directly.

Using a database system rather than a file based one will open the door for a mobile application, and for the web portal. The web portal will allow employees access to documents that do not have security restrictions, even when they are not at work. The mobile app will allow users to log into foxKeeper's database and access files on their phones. It will include a barcode reader and QR code scanner, allowing information from a product or new materials to be quickly added to the database. A release date for an Android application is to be determined, and an IOS one will be quick to follow.
Conclusion

In conclusion, much work still needs to be done to make foxKeeper functional, but working on it has provided me with a valuable opportunity to develop my skills in event driven programming and to apply them to a real-world situation. I enjoyed working on this project and look forward to its future improvements.

Citations

Coderanch Web site: https://coderanch.com/

Class. (n.d.). Retrieved from Tech Terms: https://techterms.com/definition/class


Dani Web site: https://www.daniweb.com/


Google Search: Linked list. (n.d.). Retrieved from Google: https://www.google.com/search?q=graphical+user+interface&oq=Graphical+u&aqs=chrome.1.69i57j015.6380j0j4&sourceid=chrome&ie=UTF-8#safe=active&q=linked+list


Java2Novice Web site: http://java2novice.com/


Stack Overflow Web site: http://www.StackOverflow.com


Tutorials Point Web site: https://www.tutorialspoint.com/
Appendix A: Glossary

**Action Listener** - An interface containing a single function which waits for a user event, then executes some code in response. (Kjell, n.d.)

**Bar Code Reader** - A device or application which can scan and retrieve data from a barcode, or an image containing a barcode.

**Boolean** - A variable which can have one of two values- true or false.

**Class** - In object oriented programming, this is code that can create and instantiate objects. (Class, n.d.)

**Class Method** - This is a method/function belonging to a particular class. For example, the function `String.split()` belongs to the String class.
**Constructor** - In object-oriented programming, this is a method/function that is automatically called when a new object of a class is created.

**Database** - In computer science this is a structure which stores data and enables the reading, writing, and manipulation of that data. (Coronel & Morris)

**Delimiter** - A character used to parse data.

**Desktop Application** - A software program that is run from a desktop. Examples include Microsoft Word, Notepad++, and Paint.

**Events** - In event-driven programming, this is an action performed by a user such as a mouse click or a key press. (Webopedia Staff, n.d.)

**Event-Driven Programming** - In Computer Science, this is writing code to respond to events. (Event-Driven Program, n.d.)

**Function** - A set of instructions that can be called perform a specific function and usually returns a value or some sort. (Subroutine, n.d.)

**Graphical User Interface** - A way to visually interact with a computer or computer program. (Google search: graphical user interface, n.d.)

**Java** - An object based programming language.

**JPanel** - A Java element which can display other elements, images and texts.

**Library** - In Java, this is a group of classes which can be publicly shared and downloaded to be used by different developers.

**Linked List** - In computer science, this is a data structure where data is connected together by references to the proceeding element, the preceding element, or both. (Google Search: Linked list, n.d.)

**Local Area Network** - Often abbreviated as LAN, this is a network that connects a limited number of nodes together that are within close proximity of each other. (Mitchell, 2017)

**Method** - See definition for Function.

**Mobile Application** - A software program that is run from a mobile device. Examples include Pokemon Go, Instagram and Snapchat.

**Object** - In computer science this is a procedure that contains data and instructions relating to a specific class. (Object Definitions, n.d.)

**Object Orientated Programming** - A programming language model focused around objects and the instantiation and manipulation of them. (Rouse, n.d.)

**Parameter** - In computer science, these are variables passed to a method/function to be used or altered by it.
Parse - In computer science, this means to separate data into pieces.

QR Code Scanner - A device or application which can scan and retrieve date from a QR code, or an image containing a QR code.

Recursive Method - A method/function that can call itself.

SQL Queries - Pieces of code written in Standard Query Language.

Text editor - Software that allows users to read, write, and edit text.

Try-Catch Block - A data structure that tests code for errors before executing it, and can respond to any errors that are found.

Web Portal - A web page that provides access to information, often is the single access point for that information. (Web Pmial, n.d.)

Appendix B: foxKeeper source code

mainMenu.java

```java
public class mainMenu
{
    public static void main (String [ ] args) {
        Settings settings= new Settings();
        GUI labelFrame = new GUI ();
        labelFrame.viewGUI (settings);
        labelFrame. setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE );
        labelFrame.setTitle(settings.getCompanyName());
        labelFrame.getContentPane().setBackground (Color.white);
        labelFrame.setVisible(true);
        labelFrame.setSize(610, 440);
        labelFrame.setExtendedState(JFrame.MAXIMIZED_BOTH);
    }
}
```

menuOptions.java

```java
public class menuOptions
{
    String rules;
    String abbreviation;
    String title;
    LinkedList myFiles   new LinkedList();
```
void addFile (MenuFiles newFile) {
    myFiles. add (newFile. filePath);
}

public class MenuFiles {
    String fileName;
    String filePath;
    String! fileElements;

    void parseFilePath (String delimiter) {
        fileElements = fileName. split(delimiter);
    }
}

public class FileInOut {
    LinkedList folders = new LinkedList();
    LinkedList files = new LinkedList();

    FileInOut (File folder) {
        folders.add(folder);
        checkFile(folder);
    }

    void checkFile(File folder) {
        for(File file: folder. listFiles()) {
            if(file. isFile()) {
                files.add(file);
            }
        }
    }
}
else if(file.isDirectory()) {
    folders.add(file);
    checkFile(file);
}
}
}

PDFViewer.java

public class PDFViewer extends JPanel{
    private PDFFile pdfFile;
    private PagePanel panel;
    private GUI gooey;
    private String filePath;

    private JMenuBar toolBar = new JMenuBar();
    private JTextField pageNum = new JTextField();
    private JButton goLeft = new JButton("<<");
    private JButton goRight = new JButton(">>");

    public PagePanel open(String path) {
        filePath = path;
        JFrame frame = new JFrame();
        panel = new PagePanel();

        frame.add(panel);
        frame.pack();
        setPDF();
        return panel;
    }

    private void setPDF() {
        try {
            File file = new File(filePath);
            RandomAccessFile raf = new RandomAccessFile(file, "r");
            FileChannel channel = raf.getChannel();
            ByteBuffer buf = channel.map(FileChannel.MapMode.READ_ONLY, 0, channel.size());
            final PDFFile pdfFile = new PDFFile(buf);
            PDFPage page = pdfFile.getPage(1);
            panel.showPage(page);
        }catch(Exception e) {
        }
    }

    public void switchPage(int num) {
        try{
            File file = new File(filePath);
            RandomAccessFile raf = new RandomAccessFile(file, "r");
            FileChannel channel = raf.getChannel();
            ByteBuffer buf = channel.map(FileChannel.MapMode.READ_ONLY, 0, channel.size());
            final PDFFile pdfFile = new PDFFile(buf);
            PDFPage page = pdfFile.getPage(num);
            panel.showPage(page);
        }catch(Exception e) {
        }
    }
}
RandomAccessFile raf = new RandomAccessFile (file, "r");
FileChannel channel = raf.getChannel();
ByteBuffer buf = channel.map(FileChannel.MapMode.READ_ONLY, 0, channel.size());
final PDFFile pdfFile = new PDFFile(buf);
PDFPage page = pdfFile.getPage(num);
panel.showPage(page);
} catch (Exception e) {

}

PagePanel getPDF() {
    return panel;
}

Settings.java

public class Settings {
    private String companyName = null;
    private String folder = null;
    private String menuOptionsList = null;
    private String[] menu = null;
    private String[] abbreviations = null;
    private String[] abbreviationList = null;
    private String[] menuLocations = null;
    private String theOneLocation = null;
    private String filePathFormat = null;
    private String[] pathFormats = null;
    private String bannerImage = null;
    private Boolean folderExists = false;
    private String delimeter = "--";
    BufferedReader input = null;
    BufferedWriter output = null;
    JFileChooser openFiles = new JFileChooser();
    JFileChooser open1 = new JFileChooser();
    JTextField field1 = new JTextField();
    JButton button1 = new JButton("Select file");
    JTextField confirmField1 = new JTextField();
    JTextField field2 = new JTextField();
    JTextField field3 = new JTextField();
    JFileChooser open2 = new JFileChooser();
    JButton field4 = new JButton("Select file");
    JTextField confirmField4 = new JTextField();
    JTextField field5 = new JTextField();
    JTextField field6 = new JTextField();
    JFileChooser open3 = new JFileChooser();
    JTextField confirmField6 = new JTextField();
public void setUp() {

    setUpEventHandler handle = new setUpEventHandler();
    field1.addActionListener(handle);
    field4.addActionListener(handle);
    field6.addActionListener(handle);

    confirmField1.setText("Enter file path");
    confirmField1.setForeground(Color.gray);
    confirmField4.setText("Enter file path");
    confirmField4.setForeground(Color-gray);
    confirmField6.setText("Enter file path");
    confirmField6.setForeground(Color.gray);

    if(companyName != null) {
        field0.setText(getCompanyName());
    }
    if(folder != null) {
        confirmField1.setText(getFolder());
        confirmField1.setForeground(Color.black);
    }
    if(menuOptionsList != null) {
        field2.setText(getMenuOptionsList());
    }
    if(abbreviations != null) {
        field3.setText(getAbbreviations());
    }
    if(menuOptionsLocation != null) {
        confirmField4.setText(getMenuOptionsLocation());
        confirmField4.setForeground(Color.black);
    }
    if(filePathFormat != null) {
        field5.setText(getFilePathFormat());
        parseFilePathFormat();
    }
    if(bannerimage != null) {
        confirmField6.setText(getBannerimage());
        confirmField6.setForeground(Color.black);
    }

    Object[] message = {
        "Company name:", field0,
        "File path:", confirmField1,
        "", field1,
        "Menu options (separate with commas)":", field2,
        "Abbreviations (separate with commas)":", field3,
        "Folder location:", confirmField4,
        "", field4,
        "File path format:", field5,
        "Banner image:", confirmField6,
        "", field6
    };

    int option = JOptionPane.showConfirmDialog(null, message, "Set up dialog",
        JOptionPane.OK_CANCEL_OPTION);

    if(option == JOptionPane.OK_OPTION) {
        companyName = field0.getText();
        folder = confirmField1.getText();
        menuOptionsList = field2.getText();
        abbreviations = field3.getText();
menuOptionsLocation = confirmField4.getText();
filePathFormat = field5.getText();
bannerimage = confirmField6.getText();
}

update();
}

public class setUpEventHandler implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent event) {
        JFrame dialog = new JFrame();
        Object source = event.getSource();
        if(source == field1) {
            open1.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
            open1.showOpenDialog(new JFrame());
            File userFolder1 = open1.getSelectedFile();
            confirmField1.setForeground(Color.black);
            confirmField1.setText(userFolder1.getAbsolutePath());
            confirmField1.setCaretPosition(0);
        } else if(source == field4) {
            open2.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
            open2.showOpenDialog(new JFrame());
            File userFolder2 = open2.getSelectedFile();
            confirmField4.setForeground(Color.black);
            confirmField4.setText(userFolder2.getAbsolutePath());
            confirmField4.setCaretPosition(0);
        } else if (source == field6) {
            open3.showOpenDialog(new JFrame());
            File userFolder3 = open3.getSelectedFile();
            confirmField6.setForeground(Color.black);
            confirmField6.setText(userFolder3.getAbsolutePath());
            confirmField6.setCaretPosition(0);
        }
    }
}

public void setSettings() {
    try {
        input = new BufferedReader(new FileReader("C:\Program Files\foxKeeper 0.1\settings.txt"));
        companyName = input.readLine();
        folder = input.readLine();
        menuOptionsList = input.readLine();
        parseMenuOptions();
        abbreviations = input.readLine();
        parseAbbreviations();
        menuOptionsLocation = input.readLine();
        filePathFormat = input.readLine();
        bannerimage = input.readLine();
    } catch (Exception e) {
        setup II;
    }
}

public void update() {
    try {
        File file = new File("C:\Program Files\foxKeeper 0.1\settings.txt");
    } catch (Exception e) {
        // Setup II
    }
}
if (!folderExists) {
    file.createNewFile();
} else {
    file.delete();
    file.createNewFile();
    FileWriter fw = new FileWriter(file);
    output = new BufferedWriter(fw);
    output.write(companyName);
    output.newLine();
    output.write(folder);
    output.newLine();
    output.write(menuOptionsList);
    output.newLine();
    output.write(abbreviations);
    output.newLine();
    output.write("ALL " + menuOptionsLocation);
    output.newLine();
    output.write(filePathFormat);
    output.newLine();
    output.write(bannerimage);
}

try {
    try {
        if (output != null) {
            output.close();
        }
    } catch (Exception e) {
        System.err.print("Something went wrong");
    }
} catch (IOException e) {
    System.err.print("IO EXCEPTION");
} finally {
    try {
        try {
            if (output != null) {
                output.close();
            }
        } catch (Exception e) {
            System.err.print("Something went wrong");
        }
    } catch (IOException e) {
        System.err.print("IO EXCEPTION");
    }
}

public void parseMenuOptions() {
    menu = menuOptionsList.split(",");
    category = new menuOptions[menu.length];
    ourFiles = new MenuFiles[menu.length];
}

public String parseFileStream() {
    delimeter = String.valueOf(filePathFormat.charAt(0));
    return delimeter;
}

public void parseMenuLocation() {
    menuLocations = menuOptionsLocation.split(" ");
    if (menuLocations[0] == "ALL") {
        theOneLocation = menuLocations[0];
    }
}

public void parseAbbreviations() {
    abbreviationList = abbreviations.split(",");
}

public void setCategories() {
    for (int i = 1; i < menu.length; i++) {
        menuOptions temp = new menuOptions();
        temp.title = menu[i];
        temp.abbreviation = abbreviationList[i];
        category[i] = temp;
    }
}
```java
public void setFiles() {
    FileInOut getFiles = new FileInOut(new File(folder)); // Get all files in user folder

    for(int i = 0; i < getFiles.files.size(); i++) {
        // Get all files in the folder
        MenuFiles tempFiles = new MenuFiles();
        File tempFile = (File) getFiles.files.get(i);
        tempFiles.fileName = tempFile.getName(); // Get file name
        tempFiles.filePath = tempFile.getAbsolutePath();
        tempFiles.parseFilePath(delimeter);

        for(int j = 0; j < category.length; j++) {
            if (tempFiles.fileElements[0].equals(category[j].abbreviation)) { // If category[0] equals category[j].abbreviation
                category[j].addFile(tempFiles); // Add file to that category
            }
        }
        ourFiles[i] = tempFiles;
    }
}

public String getCompanyName() { return companyName; }
public String getFolder() { return folder; }
public String getOptionsMenuList() { return menuOptionsList; }
public String getAbbreviations() { return abbreviations; }
public String getOptionsMenuLocation() { return menuOptionsLocation; }
public String getFilePathFormat() { return filePathFormat; }
public String getBannerImage() { return bannerimage; }
public String[] getMenu() { return menu; }
```
PDFViewer openThis = new PDFViewer();

private JPanel toolBar = new JPanel();
private JTextField pageNum = new JTextField ("", 2); // Place page number
private JButton goLeft = new JButton("<<"); // Go left button
private JButton goRight = new JButton(">>"); // Go right button.

void viewGUI(Settings userSettings) {
    eventHandler handle = new eventHandler();
    mainTab = new JTabbedPane();
    mainPanel = new JPanel(); // Panel to display the pages
    settings = userSettings;
    settings.setSettings(); // Provide all info in settings file and possibly create new settings file

    mainPanel.setLayout(new GridBagLayout()); // For right layout
    GridBagConstraints c = new GridBagConstraints();
    c.weightx = 0.5;
    c.fill = GridBagConstraints.NORTH;

    menuBar = new JMenuBar(); // Menu bar

    // Home button
    char tinyHouseIcon = Oxd302; // Hexadecimal code value for tiny house icon
    String tinyHouse = String.valueOf(tinyHouseIcon); // Tiny house as a string
    homeButton = new JButton(tinyHouse); // Create the action button
    homeButton.setBackground(Color.lightGray); // Set the background color to match menu bar
    menuBar.add(homeButton); // Add tiny house button to menu bar
    menuBar.setLayout(new FlowLayout(FlowLayout.LEFT)); // Set layout and align to the left
    menuBar.setPreferredSize(new Dimension(30, 30)); // Set the size of the button
    homeButton.addActionListener(handle); // Add action listener
    homeButton.setBorderPainted(false);
    homeButton.setFocusPainted(false);
    homeButton.setOpaque(false);

    // File menu
    fileMenu = new JMenu("File"); // Create a new menu
    menuBar.add(fileMenu); // Add "File" to the menu bar

    newMenu = new JMenu("New"); // Create a new menu
    fileMenu.add(newMenu); // Add "New" to the "File" menu
    newMenu.addActionListener(handle); // Add action listener
    newTxt = new JMenuItem("New Text File"); // Create menu item "New Text File"
    newMenu.add(newTxt); // Add "New Text File" to "New" menu item
    newTxt.addActionListener(handle); // Add action listener for "New Text File"

    newPDF = new JMenuItem("New PDF"); // Create menu item "New PDF"
    newMenu.add(newPDF); // Add "New PDF" to "New" menu item
    newPDF.addActionListener(handle); // Add action listener for "New PDF" menu item
openFile = new JMenuItem("Open"); //Create sub menu "Open"
fileMenu.add(openFile); //Add "Open" to the "File" menu
openFile.addActionListener(handle); //Add action listener for "Open" menu

fileMenu.addSeparator(); //Create separator for "File" menu

settingsEdit = new JMenuItem("File settings"); //Create menu item "File settings"
fileMenu.add(settingsEdit); //Add "File settings" to "File" menu
settingsEdit.addActionListener(handle); //Add action listener for "File settings" menu item

settingsMenu = new JMenu("Settings"); //Create menu option "Settings"
menuBar.add(settingsMenu);

fullscreen = new JMenuItem("Full screen on/off"); //Create menu item "Full screen on/off"
settingsMenu.add(fullscreen);
fullscreen.addActionListener(handle);

aboutMenu = new JMenu("About"); //Create menu option "About"
menuBar.add(aboutMenu); //Add "About" to the menu bar

about = new JMenuItem("About"); //Create menu item "About"
aboutMenu.add(about); //Add "About" to the "About" menu
about.addActionListener(handle); //Add action listener for "About" menu item

helpMenu = new JMenu("Help"); //Create menu option "Help"
menuBar.add(helpMenu); //Add "Help" to the menu bar

contact = new JMenuItem("Contact"); //Create menu item "Contact"
helpMenu.add(contact); //Add "Contact" to the "Help" menu
contact.addActionListener(handle); //Add action listener for "Contact" menu item

help = new JMenuItem("Help"); //Create menu item "Help"
helpMenu.add(help); //Add "Help" to the "Help" menu
help.addActionListener(handle); //Add action listener for "Help" menu item

exitMenu = new JMenu("Exit"); //Create menu option "Exit"
menuBar.add(exitMenu); //Add "Exit" to the menu bar

exit = new JMenuItem("Exit"); //Create menu item "Exit"
exitMenu.add(exit); //Add "Exit" to the "Exit" menu
exit.addActionListener(handle); //Add action listener for "Exit" menu item

setLabel = new JLabel(imageLabel); //Add image to the main panel
item2 = new JLabel(label); //Add label to the main panel
mainPanel.add(item2, c); //Add item to the main panel

settings.setCategories(); //Set categories

for(int i = 0; i < settings.category.length; i++) //For every category

JPanel tempPanel = new JPanel();
categoryLists[i] = new JPanel();
categoryLists[i].setLayout(new BorderLayout());
categoryLists[i] = tempPanel;
categoryLists[i].setBackground(Color.white);
JTextArea titleOfCatMenu = new JTextArea(settings.getMenu()[i]);
titleOfCatMenu.setEditable(false);
titleOfCatMenu.setPreferredSize(new Dimension(110, 11));
categoryLists[i].add(titleOfCatMenu, BorderLayout.NORTH);

for(int j = 0; j < settings.category[i].myFiles.size(); j++) {
    String filePath = settings.category[i].myFiles.get(j).toString();
    String fileName = new File(filePath).getName();
    String[] fileElements = fileName.split(settings.delimeter);
    JButton tempButton = new JButton(fileElements[0]);
categoryFile[i][j] = tempButton;
categoryFile[i][j].setForeground(Color.black);
categoryFile[i][j].setHorizontalAlignment(SwingConstants.LEFT);
categoryFile[i][j].setOpaque(true);
categoryFile[i][j].setFocusable(false);
categoryFile[i][j].setBorderPainted(false);
categoryFile[i][j].setVerticalAlignment(SwingConstants.TOP);
categoryFile[i][j].addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {
        openPDFWithOptions(filePath, fileName);
    }
});
}

JPanel listRow = new JPanel();
listRow.setBackground(Color.white);
listRow.setLayout(new BorderLayout());
listRow.setPreferredSize(new Dimension(100, 30));
JTextField category = new JTextField(fileElements[1]);
category.setEditable(false);  
category.setBackground(Color.white);  
JTextField parent = new JTextField(fileElements[2]);
parent.setEditable(false);  
parent.setBackground(Color.white);  
listRow.add(parent, BorderLayout.WEST);
listRow.add(categoryFile[i][j], BorderLayout.CENTER);
listRow.add(category, BorderLayout.EAST);
categoryLists[i].add(listRow, BorderLayout.CENTER);
}
```java
JPanel list = new JPanel();
list.setLayout(new GridLayout(settings.getMenu().length, 1));
for (int i = D; i < settings.getMenu().length; i++) {
    final JPanel tempPanel = categoryLists[i];
    menuOption[i] = new JButton(settings.getMenu()[i]);
    menuOption[i].setPreferredSize(new Dimension(imageLabel.getIconWidth(), 2016));
    menuOption[i].setBackground(Color.black);
    menuOption[i].setForeground(Color.white);
    menuOption[i].setOpaque(true);
    menuOption[i].setFocusable(false);
    menuOption[i].setBorderPainted(false);
    menuOption[i].addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (e.getSource() instanceof JButton) {
                mainTab.addTab((JButton) e.getSource()).getText(), tempPanel);
                mainTab.setSelectedComponent(tempPanel);
            }
        }
    });
    list.add(menuOption[i]);
}  
BufferedImage banner = null;
try {
    banner = ImageIO.read(new File(settings.getBannerimage()));
} catch (IOException e) {
}
settings.getBannerimage();
c.gridy = 4;
mainPanel.add(list); 
JTextArea attributedTo = new JTextArea("foxKeeper 0.1
Amanda Joy Panell 2016");
attributedTo.setEditable(false);
mainPanel.setBackground(Color.white);
mainTab.addTab("Home", mainPanel);
mainTab.setSelectedComponent(mainPanel);
add(mainTab);
setJMenuBar(menuBar);
}

public JPanel getTXT(String path) {
    BufferedReader input = null;
    JPanel displayTXT = new JPanel();
    JTextArea fullText = new JTextArea("\n");
    String fileContents = "\n",
    try {
        input = new BufferedReader(new FileReader(path));
        while (input.readLine() != null) {
            fileContents += input.readLine() + "\n";
            displayTXT.setText(fileContents);
        }
    }
    finally {
        input.close();
    }
    return displayTXT;
}
```
fullText.append(input.readLine() + "\n");
}
displayTXT.add(fullText);
} catch(Exception e) {
    
}
return displayTXT;
}

public void openPDFWithOptions(String file, String title) {
    String extension = file.substring(file.lastIndexOf('.'), file.length());
    title = title.replace(extension, "");
    int response;
    if (extension.equals(".pdf") ) {
        Object[] pdfOptions = {"Open in foxKeeper", "Open in default application"};
        response = JOptionPane.showOptionDialog(null, "How would you like to open the file?", "Quick question", JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE, null, pdfOptions, pdfOptions[0]);
    } else{
        response = 1;
    }
    if (response == 1) {
        if (extension.equals(".pdf") && response==1) {
            PagePanel pdfPanel = openThis.open(file);
            JPanel showPdfPanel = new JPanel();
            showPdfPanel.setLayout(new BorderLayout());
            showPdfPanel.add(toolBar, BorderLayout.NORTH);
            showPdfPanel.add(pdfPanel, BorderLayout.CENTER);
            toolBar.add(goLeft);
            goLeft.addActionListener(new ActionListener() {
                override
                public void actionPerformed(ActionEvent e) {
                    Object source = e.getSource();
                    int pageNumber;
                    if(source == goLeft) {
                        pageNumber = openThis.getPDF().getPage().getPageNumber();
                        openThis.switchPage(pageNumber - 1);
                        pageNum.setText(String.valueOf(pageNumber));
                    }
                }
            });
            toolBar.add(pageNum);
            pageNum.setText("0");
            pageNum.addActionListener(new ActionListener() {
                override
                public void actionPerformed(ActionEvent e) {
                    Object source = e.getSource();
                    int nextPage = Integer.parseInt(pageNum.getText());
                    openThis.switchPage(nextPage - 1);
                }
            });
        } else{
        }
    }
toolBar.add(goRight);
goRight.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Object source = e.getSource();
        int pageNumber;
        if (source == goRight) {
            pageNumber = openThis.getPDF().getPageNumber();
            openThis.switchPage(pageNumber + 1);
            pageNum.setText(String.valueOf(pageNumber));
        }
    }
});

mainTab.addTab(title, showPdfPanel);
mainTab.setSelectedComponent(showPdfPanel);

else if (extension.equals(".txt")) {
    JPanel txtReady = getTXT(file);
    mainTab.addTab(file, txtReady);
    mainTab.setSelectedComponent(txtReady);
} else {
    try {
        Desktop.getDesktop().open(new File(file));
    } catch (Exception e) {
    }
}

public class eventHandler implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent event) {
        Object source = event.getSource();
        if (source == newTxt) {
            JOptionPane.showMessageDialog(new JFrame(), "Coming soon!");
        } else if (source == newPDF) {
            JOptionPane.showMessageDialog(new JFrame(), "Coming soon!");
        } else if (source == openFile) {
            JFileChooser openFile = new JFileChooser();
            openFile.showOpenDialog(new JFrame());
            openPDFWithOptions(openFile.getSelectedFile().getAbsolutePath(),
                openFile.getSelectedFile().getName());
        } else if (source == settingsEdit) {
            settings.setup();
        } else if (source == fullScreen) {
            if (GUI.this.getExtendedState() == JFrame.MAXIMIZED_BOTH &&
                GUI.this.isUndecorated() == false) {
                GUI.this.dispose();
                GUI.this.setExtendedState(JFrame.MAXIMIZED_BOTH);
            }
        }
    }
}
GUI.this.setUndecorated(true);
GUI.this.pack();
GUI.this.setVisible(true);
}
else{
    GUI.this.dispose();
    GUI.this.setExtendedState(JFrame.MAXIMIZED_BOTH);
    GUI.this.setUndecorated(false);
    GUI.this.pack();
    GUI.this.setVisible(true);
}
}
else if (source == about) {
    JOptionPane.showMessageDialog(new JFrame(), "foxKeeper File Management System v1.0\n" + "Amanda Joy Panell \nWinter 2016" + "\nSoftware Development Project"+ "\nNorthwest Nazarene University"+ "\nComputer Science");
}
else if (source == contact) {
    JOptionPane.showMessageDialog(new JFrame(), "CONTACT"+ "\nPhone: (425)789-6888" + "\nEmail: mpanell@nnu.edu");
}
else if (source == help) {
    JOptionPane.showMessageDialog(new JFrame(), "Coming soon!");
}
else if (source== homeButton) {
    mainTab.addTab("Home", mainPanel);
    mainTab.setSelectedComponent(mainPanel); //Sets the currently tabbed tab
}
else if (source == exit) {
    int response = JOptionPane.showConfirmDialog(null, "Exit program now?", "Confirm close program", JOptionPane.YES_NO_OPTION);
    if(response == 0) {
        System.exit(0); //Close program
    }
}