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

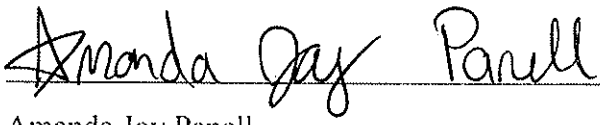
In partial fulfillment of the requirements for the degree of

BACHELOR OF ARTS

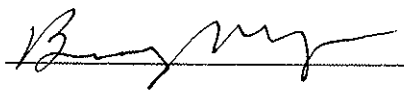
Amanda Joy Panell

2017

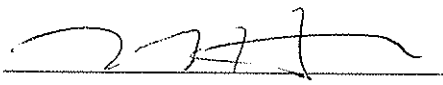
foxKeeper File Management System

Author: 

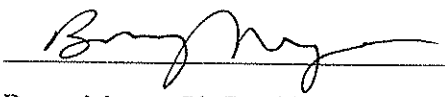
Amanda Joy Panell

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

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 the 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 pe1manent 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 offoxKeeper File Management System is to provide employees of Gompf Brackets with a user-f'iendly, 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 constrnctor 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 tunning foxKeeper.
2. The menu bar, with home, "File," "Settings," "About," "Help," and "Exit" buttons.
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 infonnation 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.

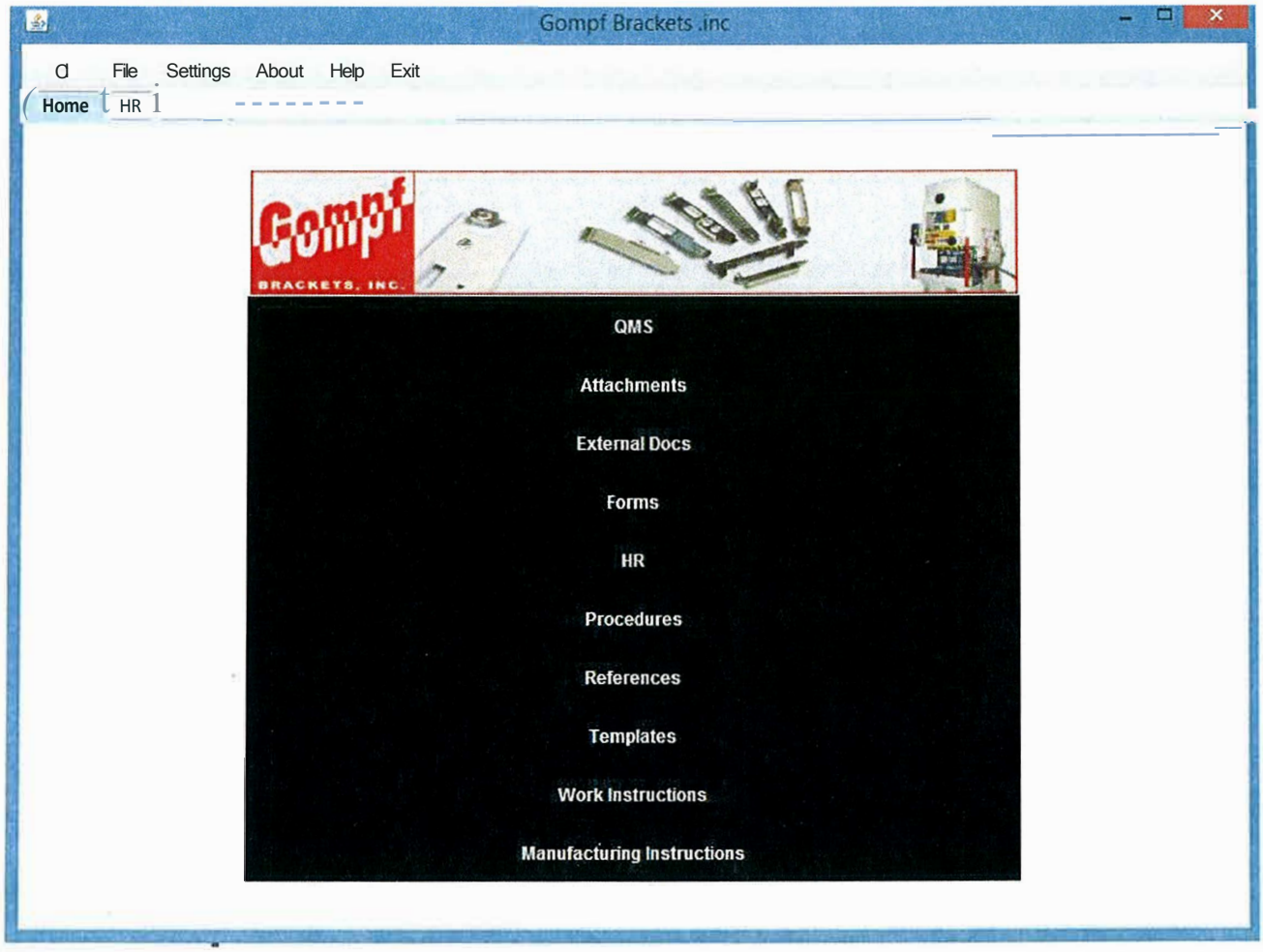


Figure 1- The main menu

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 it's "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 lannched. 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. Unfornmately, 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.

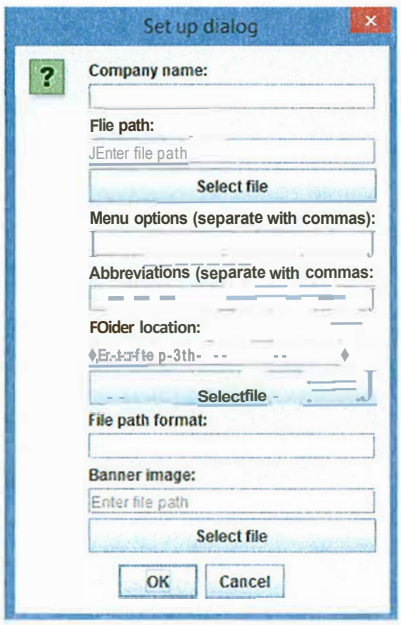


Figure 2- The setup dialog box

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 constmctor 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 constrnctor.

First, "folder" is added to the "folders" linked list. Next, each File object within "folder" is tested to determine ifit is a trne file, or ifit 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. Otien, 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.

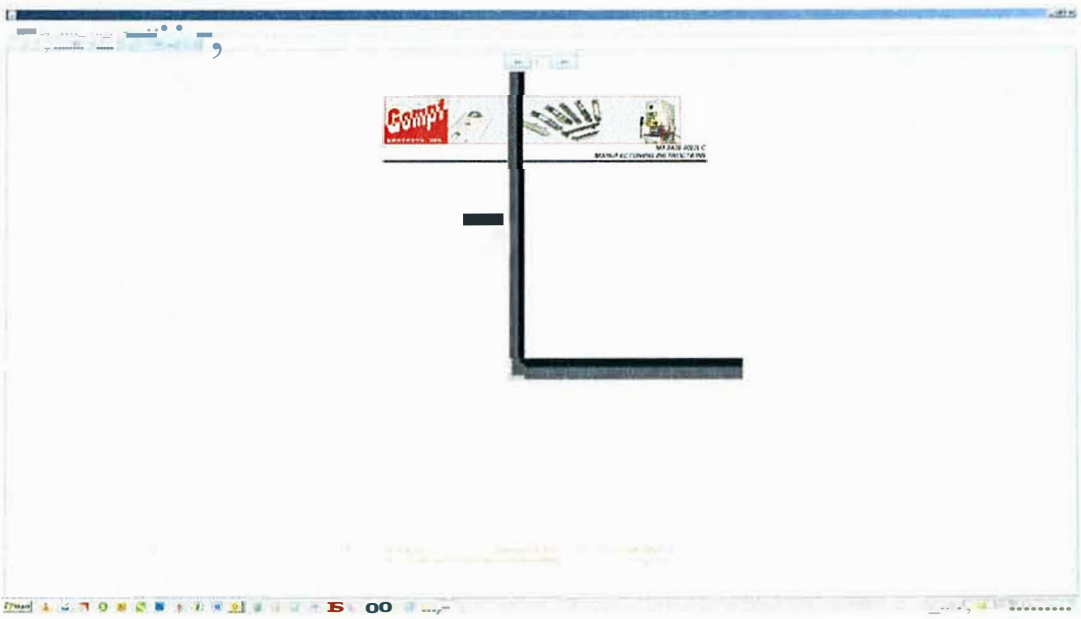


Figure 3- PDF Rendering Error on An Older Operating System

Finally, foxKeeper does not currently have any installation system. Currently, each user must create a folder called "foxKeeperO. 1" 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.

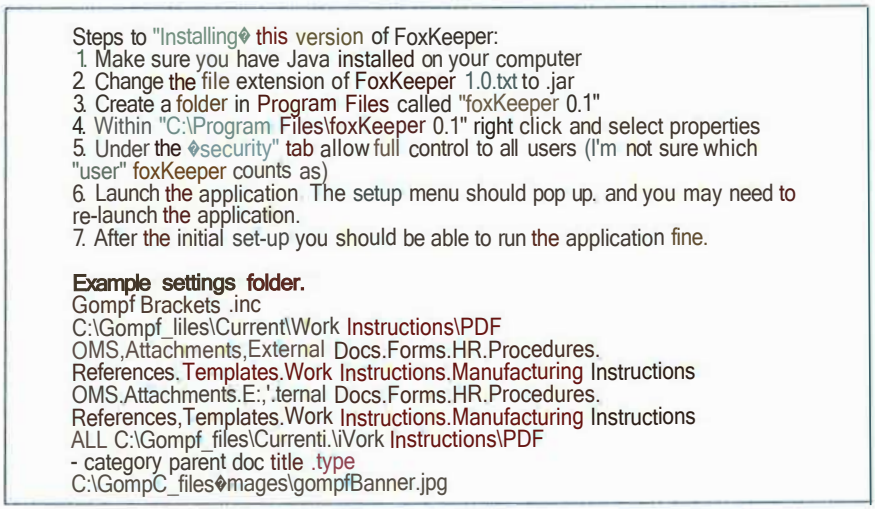


Figure 4 - Installation Instructions for foxKeeper 1.0

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.

Figure 5 - Documents Menu (Version 1.0)

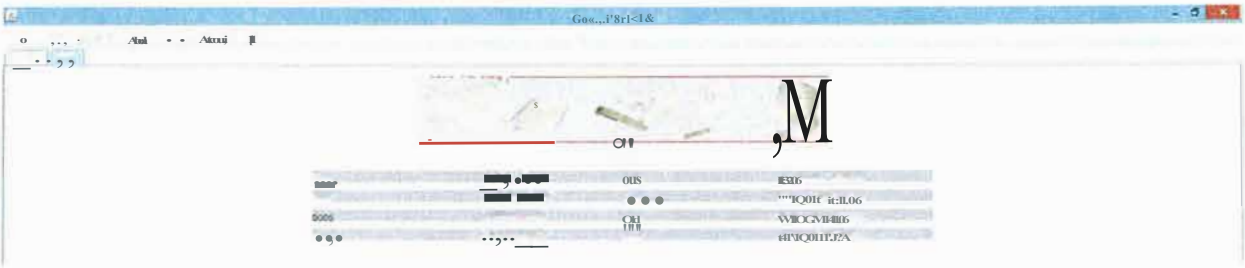


Figure 6 - Documents Menu Design-Still-In-Progress (Version 2.0)

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 administative or audit-level user will be able to request user activity and file modification reports, and the administative 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 Telms: https://techterms.com/definition/class

Coronel, C., & Morris, S. (n.d.). *Database Systems Design Implementation, and Management, 11th edition.* Cengage Leaming.

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

Deitel, P., & Deitel, H. (n.d.). *Java How to Program, 10th edition.* Deitel.

Event-Driven Program. (n.d.). Retrieved from Techopedia: https://www.techopedia.com/definition/7083/event-driven-program

Ganon, R. (n.d.). *Select a directory with a JFileChooser.* Retrieved from Real's How To: http://www.rgagnon.com/javadetails/java-0370.html

GompfBrackets, Inc. [GompfBrackets Logo). (n.d.). Retrieved from http://bracket.com/

Google search: graphical user inte1jace. (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

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

Hamilton, D. (2016). Event Driven Programming Course. *Northwest Nazarene University.*

How to Write a File Line by Line in Java? (n.d.). Retrieved from Program Creek: http://www.programcreek.com/2011/03/java-write-to-a-file-code-example/

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

Java.io.File.delete() Method. (n.d.). Retrieved from Tutorials Point Web Site: https://www.tutorialspoint.com/java/io/file_delete.htm

Kjell, B. (n.d.). *ActionListener*. Retrieved from Centeral Connecticut State University Intro to Compute Science Using Java Web Site: http://chortle.ccsu.edu/java5/notes/chap57/ch57_IO.html

Lievsay, W. (2010, May 14). *Java GUI Tutorial 3 -Adding images*. Retrieved from YouTube: https://www.youtube.com/watch?v= FdQX8sUNO-s&t=1s

Mitchell, B. (2017, March 20). *What's a LAN (Local Area Network)?* Retrieved from Lifewire: https://www.lifewire.com/local-area-network-816382

mkyong. (2010, January IO). *How to get thefile last modified date in Java*. Retrieved from Mkyong: https://www.mkyong.com/java/how-to-get-the-file-last-modified-date-in-java/

Object Definitions. (n.d.). Retrieved from Computing Students: http://www.computingstudents.com/dictionary/?word=Object

Oracle. (n.d.). Retrieved from MySQL Documentation: https://dev.mysql.com/doc/

Oracle. (n.d.). Java Documentation. https://docs.oracle.com/en/java/.

Reyes, A. (2015, May 7). *JFileChooser Swing Example*. Retrieved from Java Code Geeks: https://examples.javacodegeeks.com/desktop-java/swing/jfilechooser/jfilechooser-swing-example/

Rouse, M. (n.d.). *Object-oriented Programming (OOP)*. Retrieved from Search Microservices Web Site: http://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP

Sean. (2012, December 3). *Java Swing PDF Viewer*. Retrieved from A Cup of Tea! Blog: http://seanshou.blogspot.com/2012/IO/java-swing-pdf-viewer.html#ixzz4S7kQiiVp

Shah, A. (2016, August I2). *How to Iterate through LinkedList Instance in Java?* Retrieved from Crnnchify Web site: https://crnnchify.com/how-to-iterate-through-linkedlist-instance-in-java/

Singh, C. (n.d.). *How to write tofile in Java using BufleredWriter*. Retrieved from Beginners Book: http://beginnersbook.com/2014/0I/how-to-write-to-file-in-java-using-bufferedwriter/

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

Subroutine. (n.d.). Retrieved from The Free Dictionary web site: http://www.thefreedictionary.com/Function+(computer+science)

thenewboston. (2009, September 19). *Java Programming Tutorial - 51 - GUI with)Frame*. Retrieved from Youtube: https://www.youtube.com/watch?v=jUdlAgJ7JKo

Tutorials Point Web site: https://www.tutorialspoint.com/

Unicode Character 'HOUSE' (U+2302). (n.d.). Retrieved from FileFormat.Jnfo: http://www.fileformat.info/info/unicode/char/2302/index.htm

Web Portal. (n.d.). Retrieved from Techopedia: l1ttps://www.techopedia.com/definition/l 7352/web-portal

Webopedia Staff. (n.d.). *Event.* Retrieved from Webopedia: http://www.webopedia.com/TERM/E/event.html

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/fonction 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 perfmmed 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 fonction and usually ren1rns 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 Snap Chat.

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

```
public class mainMenu
I
    public static void main (String [J args) {
        Settings settings= new Settings{);

        GUI label Frame = new GUI ();
        labelFrame.viewGUI (settings);
        labelFrame. setDefaul tCloseOpera tion (JFrame. EXIT_ON_CLOSE) ;
        labelFrame.setTitle{settings.getCompanyName());
        label Frame. getContentPane () .setBackground (Color.white);
        labelFrame.setVisible{true);
        labelFrame.setSize(6J0, ,;:lQ);
        label Frame. setExtendedState (JFrame .MAXIMIZED_BOTH);

    }
}
```

menu Optious.java

```
public class menuOptions {
    String rules;
    String abbreviation;
    String title;
    LinkedList myFiles    new LinkedList {);
```

```
    void addFile (MenuFiles newFile) {
        myFiles. add (newFi.le .filePath);
    }
}
```

menuFiles.java

```
public class MenuFi.les {
    String fileName;
    String filePath;
    String!] fileElements;

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

FileInOut.java

```
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.J.istFiles()) {

            if(file.isFi.le()) {

                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 set.PDF{) {
        try {
            File file = new File(filePath);
            RandomAccessFU.e raf = new RandomAccessFile(file, "r");
            FileChannel channel= raf.getChannel();
            ByteBuffer buf = channel .map (F.U.eChannel. MapMode. READ_ONLY, ii,
channel.size());
            final PDFFile pdffile = new PDFFi.le(buf);
            PDFPage page = pdffile.getPage(J);
            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, CJ,
channel.size());
            final PDFFile pdffile = new PDFFile(buf);
            PDFPage page = pdffile.getPage(num);
            panel. showPage (page) ;
        }catch(Exception e) {
        }
    }

    PagePanel getPDF() {
        return panel;
    }
}
```

Settings.j ava

```
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 menuOptionsLocation = 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;
    String delimeter = "—";
    BufferedReader input= null;
    BufferedWriter output= null;
    IVlenuFiles [] ourFiles;
    menuOptions[J category;
    JTextField fieldO = new JTextField();
    JFileChooser openl = new JFileChooser();
    ,JButton fieldl = new JButton("Select file");
    JTextField confirmFieldl = new JTextField ();
    JTextField field2 = new JTextField();
    JTextField field3 = new JTextField();
    JFileChoo.ser open2 = new JFileChooser();
    JButton field4 = new JButton ("Select file");
    JTextField confirmField4 = new JTextField{);
    JTextField fields= new JTextField();
    JButton field6 = new JButton("Select file");
    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 == fieldl} {
                openl. setFileSelectionMode (JFileChooser. DIRECTORIES_ONLY);
                openl.showOpenDialog(new JFrame());
                File userFolderl = openl.getSelectedFile(};
                confirmFieldl.setForeground(Color.black);
                confirmFieldl.setText(userFolderl.getAbsolutePath());
                confirmFieldl. setCaretPosition ('._));
            }
            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. setCaretPosi tion ('l);
            }
            else if (source == field6) {
                open3. showOpenDialog (new JFrame ());
                File userFolder3 = open3.getSelectedFile();
                confirmField6.setForeground(Color.black);
                confirmField6.setText(userFolder3.getAbsolutePath());
                confirmField6.setCaretPosition(O);
            }
        }
    }

    public void setSettings{) {
        try I
            input = new BufferedReader{new F.ileReader{"C:\\Program Files\\foxKeeper
0.1\\settings.txt''));
            companyName = input.readLine{);
            folder = input.readL.i.ne();
            menuOptionsList = input.readL.ine();
            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'');
```

```
            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);

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

    public void parseMenuOptions () {
        menu= menuOptionsList.split(",");
        category = new menuOptions [menu. length];
        ourFiles = new MenuFiles [ J:':,';J;
    }
    public String parseF.i.lePathFormat () {
        delimeter = String.valueOf(filePathFormat.charAt(:'.1));
        return delimeter;
    }
    public void parseMenuLocation() {
        menuLocations = menuOptionsLocation.split(" ");
        if (menuLocations [.:J1 == "ALL") {
            theOneLocation = menuLocations [ ,J;
        }
    }
    public void parseAbbrevi.ations () {abbreviationList = abbreviations.split(",");}

    public void setCategories () {
        for(int i = :'); i < menu.length; i++) {
            menuOptions temp= new menuOptions();

            temp.title = menu[i];
            temp.abbreviation = abbreviationList [iJ;

            category[iJ = temp;
        }
```

```
        setFiles(); //Gets all the files in the folder
    }

    //Gets all the files in the folder
    public void setFiles() {
        FileinOut getFiles = new FileinOut(new File(folder)); //Gets all files in user
folder

        for(int i = 0; i < getFiles.files.size(); i++){ //Goes every file in the folder
            //System.out.println(getFiles.files.get(i));
            MenuFiles tempFiles = new MenuFiles();
            File tempFile = (File)getFiles.files.get(i);
            tempFiles.fileName = tempFile.getName(); //Gets name of file
            tempFiles.filePath = tempFile.getAbsolutePath();
            tempFiles.parseFilePath(delimeter);

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

    public String getCompanyName() {return companyName;}
    public String getFolder() {return folder;}
    public String getMenuOptionsList() {return menuOptionsList;}
    public String getAbbreviations() {return abbreviations;}
    public String getMenuOptionsLocation() {return menuOptionsLocation;}
    public String getFilePathFormat() {return filePathFormat;}
    public String getBannerimage() {return bannerimage;}
    public String[] getMenu() {return menu;}
}
```

GUI.java

```
/**
 * Class to create and display a Graphical User Interface, and handle its events and
interaction.
 */

public class GUI extends JFrame{
    JTabbedPane mainTab;
    private JPanel mainPanel;
    private Imageicon imageLabel;
    private JLabel item2;
    private JMenuBar menuBar;
    private JMenu fileMenu, settingsMenu, aboutMenu, helpMenu, newMenu, exitMenu;
    private JButton homeButton;
    private JMenuitem newTxt, newPDF;
    private JMenuitem fullScreen, settingsEdit;
    private JMenuitem openFile;
    private JMenuitem about;
    private JMenuitem contact, help;
    private JMenuitem exit;
    private Settings settings;
    private JButton menuOption[] = new JButton[100]; //Sets the layout
    JPanel categoryLists[] = new JPanel[100];
```

```
PDFViewer openThis = new PDFViewer();

private JPanel toolBar = new JPanel();
private JTextField pageNum = new JTextField ("", 2);
private JButton goLeft = new JButton ("<<");
private JButton goRight = new JButton (">>");

void viewGUI(Settings userSettings) {
    eventHandler handle = new eventHandler ();

    mainTab = new JTabbedPane ();
    mainPanel = new JPanel();

    settings = userSettings;
    settings.setSettings();

    mainPanel.setLayout(new GridBagLayout());
    GridBagConstraints c = new GridBagConstraints();
    c.weightx = 0.5;
    c.fill = GridBagConstraints.NORTH;

    menuBar = new JMenuBar();

    char tinyHouseicon = 0x2302;
    String tinyHouse = String.valueOf(tinyHouseicon);
    homeButton = new JButton (tinyHouse);
    homeButton.setBackground(Color.lightGray);
    menuBar.add(homeButton);
    menuBar.setLayout(new FlowLayout(FlowLayout.LEFT));
    menuBar.setPreferredSize(new Dimension(30,30));
    homeButton.addActionListener(handle);

    homeButton.setBorderPainted(false);
    homeButton.setFocusPainted(false);
    homeButton.setOpaque(false);

    fileMenu = new JMenu("File");
    menuBar.add (fileMenu);

    newMenu = new JMenu ("New") ;
    fileMenu.add (newMenu);
    newMenu.addActionListener(handle);

    newTxt = new JMenuitem ("New Text File");
    newMenu.add(newTxt);
    newTxt.addActionListener (handle);

    newPDF = new JMenuitem ("New PDF");
    newMenu.add (newPDF) ;
    newPDF.addActionListener(handle);
```

```java
        openFile = new JMenuitem("Open"); //Create sub menu "open"
        fileMenu. add (openFile); //Add "Open" to the "File" menu
        openFile.add.ActionListener(handle); //Add action listener for "Open" menu
action

        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

        //"Settings" menu
        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);

        //"About" menu
        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

        //"Help" menu
        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.add.ActionListener(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

        //"Exit" menu
        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

        //Set banner
        imageLabel = new Imageicon(settings.getBannerimage());
        item2 = new JLabel(imageLabel);
        c.gridx = 1;
        mainPanel.add(item2,c);

        settings.setCategories();

        JButton[J [] categoryFile = new JButton[settings.category.length] [100];

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

```
            JPanel tempPanel = new JPanel();
            categoryListsfi] = new JPanel ();
            categoryLists[i] .setLayout(new BorderLayout());
            categoryLists [iJ = tempPanel;
            categoryLists [iJ•setBackground (Color.white);
            JTextArea titleOfCat:Menu = new JTextArea (settings. getMenu () [iJ);
            titleOfCatMenu.setEditable(false);
            titleOfCatMenu.setPreferredSize(new Dimension( 1LJJO, 1 ));

            categoryLists [i] .add (title OfCatMenu, BorderLayout. NORTH) ;

            for(int j = 'J; j < settings.category{i] .myFiles.size(); j++) {

                String filePath = settings. category [iJ•myFiles. get (j) .toString ();
                String fileName = new
File(settings.category[i] .myFiles.get(j) .toString()) .getName ();
                String[] fileElements = fileName.split(settings.delimeter);

                JButton tempButt.on = new JButton (fileElement.s [)]);

                categoryFile [iJ [j] = tempButton;
                categoryFile [iJ [jJ•setBackground (Color.white);

                categoryFile (iJ [jJ•set Foreground (Color .b.lack);

                categoryFile [iJ [j] .setlforizontalAlignment {SwingConstants. LEFT) ;
                categoryFile[iJ I .setOpaque(true);
                categoryFile[i] [j] .setFocusable(false);
                categoryFile[i] [j] .setBorderPainted(false);
                categoryFile [iJ [j] .setVerticalAlignment (SwingConstants. TOP);
                categoryFile[i] [j] .addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed{ActionEvent e) {
                        openPDFWithOptions(f'leP 11,f 11!N0 1');
                    }
                II;

                JPanel listRow = new JPanel();
                listRow. set.Background (Color.white);
                listRow.setLayout(new BorderLayout());
                1.istRow.setPreferredSize(new Dimension(HiJ·1, 2y.,));

                JTextField category= new JTextField(fileElements[·l]);
                category.setEditable(false);
                category. setBorder (new EmptyBorder (::.,(1,0,Ci));
                category. setBackground (Color.white) ;
                JTextField parent= new JTextField(fileElements[: ]) ;
                parent.setBorder(new EmptyB0rder(:,1 :1 ~, 1));
                parent.setBackground{Color.white);
                parent.setEditabJ.e(false);

                listRow.add(parent,BorderLayout.WEST);
                listRow. add (categoryFile [iJ [j J,BorderLayout. CENTER) ;
                1is tRow. add (category, BorderLayout.. EAST) ;

                categoryLists [JJ. add (listRow, BorderLayout. CENTER);
            }

        }
```

```
        JPanel list= new JPanel();
        list.setLayout (new GridLayout (settings.getMenu ().length,)));

        for(int i = D; i < settings.getMenu() .length; i++) {
            final JPanel tempPanel = categoryLists [iJ;
            menuOption [i) = new JBu tton (settings. getMenu () [iJ);
            menuOption[iJ .setPreferredSize(new
Dimension (imageLabel .geticonWidth (), r;()));
            menuOption[i] .setBackground(Color.black);
            menuOption [iJ•setForeground (Color. white);
            rnenuOption[i] .setOpaque(true);
            menuOption [iJ•set Focusable {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 (), temrr·c:r:e_:);
                        main Tab. setSelectedComponent (t>;npF :n,-;l);
                    }
                }
            ));
            list.add(menuOption[.i]);
        }

        Bufferedimage banner= null;
        try {
            banner = ImageIO.read(new File(settings.getBannerimage{)));
        }catch(I0Exception e) {

        }
        settings.getBannerimage();

        c.gridy = 4;
        mainPanel.add(list, c);

        JTextArea attributedTo = new JTextArea("foxKeeper 0.1\nAmanda 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('''');
        String fileContents = ""!

        try {
            input= new BufferedReader(new FileReader(path});
            while(input.readLine{) != null) {
```

```
                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.s/JowOptionDiaJog(null,  "How would you like to open
the file?",
                  "Quick question",  JOptionPane. DEFAULT OPTION,
JOptionPane. INFORMATION_MESSAGE,  null,  pdfOptions,  pdfOpt·ions [Cl]);
      } else{
          response= );
      }

      if (response  ==  ') II response  ==  ) {

          if  (extension.equals{''.pdf'')  && response==;)) {

              PagePanel pdfPanel  = openThis.open(file);

              JPanel  showPdfPanel  = new JPanel ();
              showPdfPanel.setLayout(new  BorderLayout());

              showPdfPanel. add (toolBar,  BorderLayout. NORTH);
              showPdf Panel. add (pdf Panel,  BorderLayout. CENTER);
              toolBar.add(goLeft);
              goLeft.addActionListener(new ActionListener{)  {
                  @Override
                  public void actionPerformed{ActionEvent e)  {
                      Object  source= e.getSource{);
                      int pageNumber;

                      if(source  ==  goLeft) {
                          page Number  = open This. get PDF () .get Page () .getPageNumber ();
                          openThis.switchPage(pageNumber  -  1);

                          pageNum. set Text (String. valueOf(pageNumber));
                      }
                  }
              II;
              too1Bar.add(pageNum);
              pageNum.setText("O");
              pageNum.addActionListener(new ActionListener()  {
                  @Override
                  public void actionPerformed(ActionEvent e)  {
                      Object  source  =  e.getSource();
                      int  ne,\Page = Integer .parseint (pageNum. get Text ());

                      openThis. swi tchPage (ne1\Page  -  :);
                  }
```

```
            II;
            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().getPage().getPageNumber();
                        openThis.switchPage(pageNumber + 1);

                        pageNum.setText(String.valueOf(pageNumber));
                    }
                }
            II;

            mainTab.addTab(title, showPdfPanel);
            mainTab.setSelectedComponent(showPdfPaneJ.);
        } else if (extension.equals(". txt")) {
            JPanel txtReady = getTX'r(file);
            mainTab.add'I'ab(file, txtReady);
            mainTab.setSelectedComponent{txtReady);
        } else {
            try I
                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 1");
            } else if (source == newPDF) {
                JOptionPane.sho1.11J,JessageDialog(new JFrame(), "Coming soon!");
            } else if (source== openFile) {

                JFileChooser openFile = new JFileCt1ooser();
                openFile.showOpenDialog(new JFrame());

openPDFWithOptions(openFile.getSelectedFi.le() .getAbsolutePath(),openFile.getSelectedFi
le 11.getName 11I;

            } else if (source == settingsEdit) {
                settings.setup();
            } else if (source== fullScreen) {
                if(GUI.this.getExtendedState() == 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 vl.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 [illegible] to
the recently added tab
            } else if (source == exit) {
                int response = JOptionPane. sho,1ConfirmDialog (null, "Exit program now?",
"Confirm close program",JOptionPane.YES_NO_OPTION);
                if(response == 0) {
                    System.exit(O); //Close program.
                }
            }
        }
    }
}
```