

Project Report

***CSE 2100:***

***Software Development Project - I***

**Developed by:-**

Md.Asaf-Uddowla Golap

Roll No:1207005

Nasrin Akter Rima

Roll No:1207048

## **Under Supervision of:**

## Prof. Dr. K. M. Azahrul Hasan

## Department of Computer Science & Engineering

Khulna University of Engineering and Technology Signature

File System Wizard

Acknowledgment

With the blessings and limitless mercy of Almighty, we are able to do this. We express our heartiest gratitude to Almighty Allah for this.

A word of special thanks must go to our highly esteemed teacher and this project’s supervisor, **Prof. Dr. K. M. Azharul Hasan**, Department of computer Science & Engineering, KUET, for his excellent advices and right directions without which our project may not have reached a state it is in now.

We would also like to thank our seniors, who have tried to teach us java and arranged workshop to teach us this language. We like to thank our friends for their association also.

Any constructive comments, suggestions, criticism from teachers as well as seniors will be highly appreciated and gratefully acknowledged.

**INDEX**

|  |  |  |
| --- | --- | --- |
| 1 | Introduction | 3 |
| 2 | Objective | 3 |
| 3 | The Deployments | 4 |
| 4 | Project Structure | 5 |
| 5 | Control Flow in the Project: The Web | 6 |
| 6 | A Brief Discussion on How “File System Wizard” Works | 7-8 |
| 7 | A short Explanation of code | 9-13 |
| 8 | Features & Usefulness Rendered | 13-14 |
| 9 | Limitations | 14 |
| 10 | Improvement | 14 |
| 11 | Conclusion | 15 |
| 12 | Reference | 15 |
|  |  |  |

1. **Introduction:**

Often we might want to have our files in the computer Copy/Move/Delete to a certain Condition, be it for long-term storage purpose. Suppose we only copy/move/delete only pdf file or only last used at 12/12/2012 file or shortcut file or size less than 5 MB or has name photo from certain location to a certain location. When number of files to copy/move/delete is small, the task is not very hard, but might be wearisome in some cases. However, when large number of files is concerned the operation becomes as hard as impossible, if not tedious. As the organisation of files in the computer is to be done in the computer itself, the task could best be accomplished by producing an application that does the job for us. That’s where our application ***File System Wizard*** comes in!

1. **Objective:**

To help users copy/move/delete their computer files in a very intuitive and efficient manner, even when large number of files is concerned. This application provides facilities not usually provided by the operating system, be it Microsoft’s Windows or Linux.

1. **The Deployments:**

This application is developed as a 2nd year 1st semester software development project, a course titled *“Software Development Project-1”*. As a requisite we used the Java programming language to develop the application. The development environment used was “*NetBeans IDE 8.0”*.

**3-1. Why use Java:**

Java is Object Oriented Programming language that contains very rich API, specifically designed towards faster and more efficient application development. The API is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. As a platform-independent environment, although the Java platform can be a bit slower than native code, its powerful API is the primary reason for it to be used to develop this application, as in many others being developed by a large number of developers around the globe. Moreover, advances in compiler and virtual machine technologies are bringing performance of Java applications close to that of native code without threatening portability.

**3-2. About NetBeans IDE:**

The Integrated Development Environment most favoured by many Java application developers is *“NetBeans”.* It is also the IDE endorsed by Oracle Corporation, the organisation that currently owns Java. The IDE provides very rich and handy features for developing Java applications including its GUI Design feature that brings down the overhead of GUI designing to a minimum. Its auto-generation of codes for GUI components, intelligent tips and hints, powerful code re-factoring capability are just a few of the many eye-stunning abilities the IDE is adorned with.

**4.Project Structure:**

The project is divided into two packages, each containing a number of classes. The packages, along with the classes they contain, are shown below:

Fig 1: Packages and classes in the project

There are a total of 10 classes, total number of lines of code adding up to approximately 2600.

**5.Control Flow in the Project: The Web**

Homepage

The starting point of the application

Table Model

Files selected by user

File Chooser

CopyMove Windows

Main Window

The hub of the application

Delete Windows

Files

Select Source?

Start

Files Sets on table Table

Copy/Move ? ?

Delete?

Copy/move selected

files

Copy ?

Move ?

Delete selected

files

Deleter

Copier/Mover

Copy/Move/delete

User selected Files.

Fig 2: Control Flow of File System wizard

**6. A brief Discussion on How “File System Wizard” Works:**

From user’s point of view, the application provides a very user-friendly interface, in which the user first select files. Then the Copy/move/delete operation will be done with those Files. The Main Window of the application is shown below for easy reference.

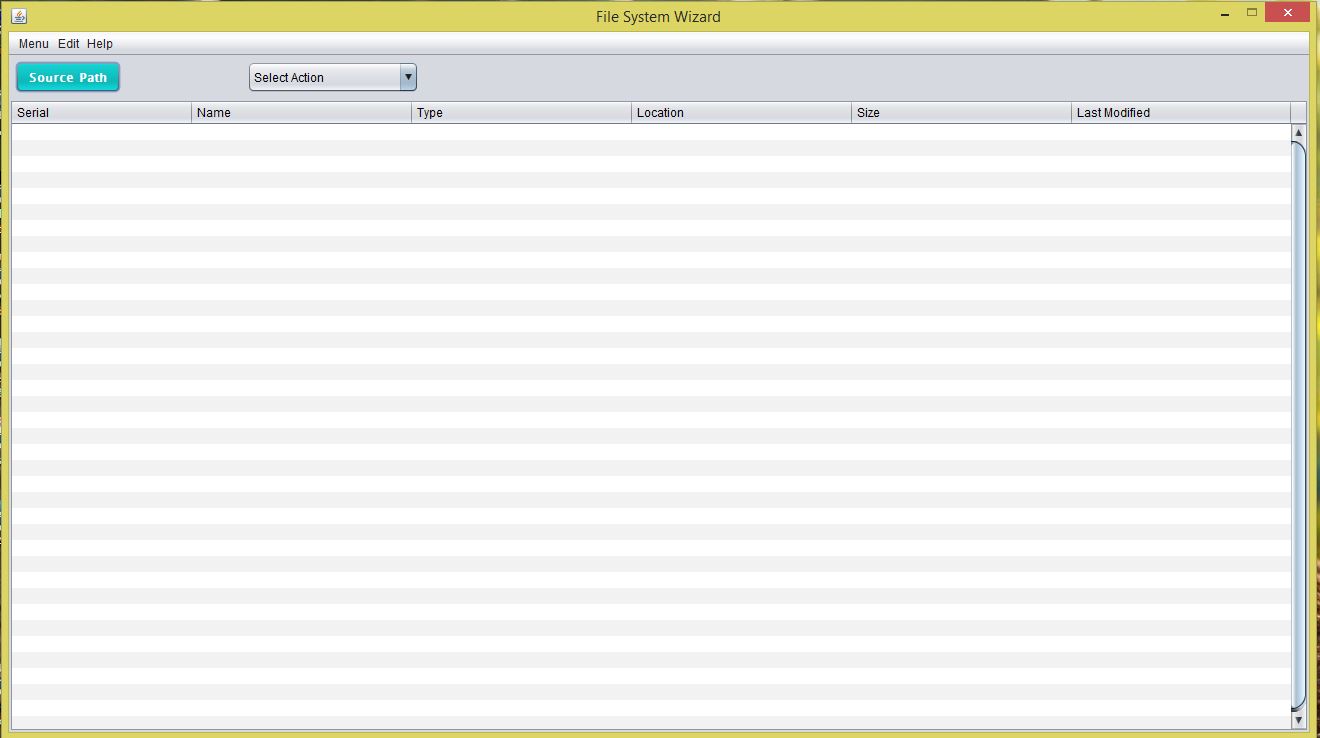


Fig3: The Main Window of ***File System Wizard***

The application Mainwindow has buttons and combo box that speak for themselves, in the sense that their names show what they do. Additionally, the items in menu bar have keyboard shortcuts assigned to most of them. The ones floating on the Main Window have mnemonics for easy access. The table in the Main Window have column names that say what they contain. The preview of the selected files shown on the table. The files are not changed in any way as yet, as this is just a preview. Clicking the combo box open copy/move/delete windows. Then Fill-up the necessary field on copy/move/delete windows, if we click confirm button copy/move/delete action will be done and show successful message. However, when coping/moving/deleting files there show a progress bar that show the progress of copying/moving/deleting.

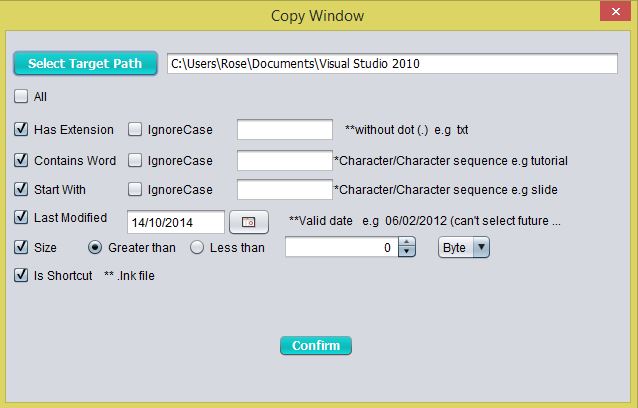


Fig4: Copy window of ***File System Wizard***

1. **A short Explanation of code:**

The application starts as an instance of the Mainwindow class, being instantiated from the main() method of Homepage class. The files selected by user are stored as an array of files in a public static File array field inside the Mainwindow class. As this field, most of the fields and methods of Mainwindow class are public and static. The underlying reason for the mainwindow class to have its fields and methods as static is that, it is obvious that only one instance of this class needs to be instantiated, meaning that both static and non-static members will be same to this instance. However, making them static allows other classes to gain access to these fields and methods without having to instantiate another Mainwindow class. In order for other classes to get access to the members of Mainwindow class, the members must not be private. This explains why they are public.

Almost all other classes have similar fields, but none extends the Mainwindow class. The design is made this way for two reasons:

1. When a copy/move/delete is processed, the public static field public static File[] files in Mainwindow class is processed and this files are also used in FilesTableModel . If the copier/mover/deleter,FilesTableModel classes, were to inherit from Mainwindow class, this would never be possible.
2. We need not be concerned about the overhead of all these files being copied to the caller classes – JAVA handles this very cleverly. Only shallow copy takes place, i.e. copy of references is sent.

Now follows some brief explanations of some of the major classes in the project. As already mentioned, the application starts from the Homepage class and it comes first.

* 1. Homepage

It’s a small class with no field. In its main() method, the LookAndFeel of the application is set to “Nimbus” and an instance of Mainwindow class is instantiated, the control flows to this object of Mainwindow class. The control is meant never to return to Homepage class.

* 1. Mainwindow

As the name suggests, this class is the hub of the application. From this control flows to the other classes and eventually returns to it (see Section: 5 to get to know how the control flows from and to this class). This class extends the built-in Java class javax.swing.JFrame and is the Main Window of the application. For Selecting files the control flows to FileChooser class, where the user selects directory of files. These files are returned to Mainwindow class. In Mainwindow class, these files are stored in File[] files, picking the names from the files stored in String[] fileNames. The files are then passed to the class FileTableModel(see below). This class returns the table model required to setup the table. When the user chooses from combo box (copy service/move service /delete service) new jDialog box of CopyMoveActionWindows/DeleteActionWindows is opend. We can return to Mainwindow by closing the CopyMoveActionWindows/DeleteActionWindows.

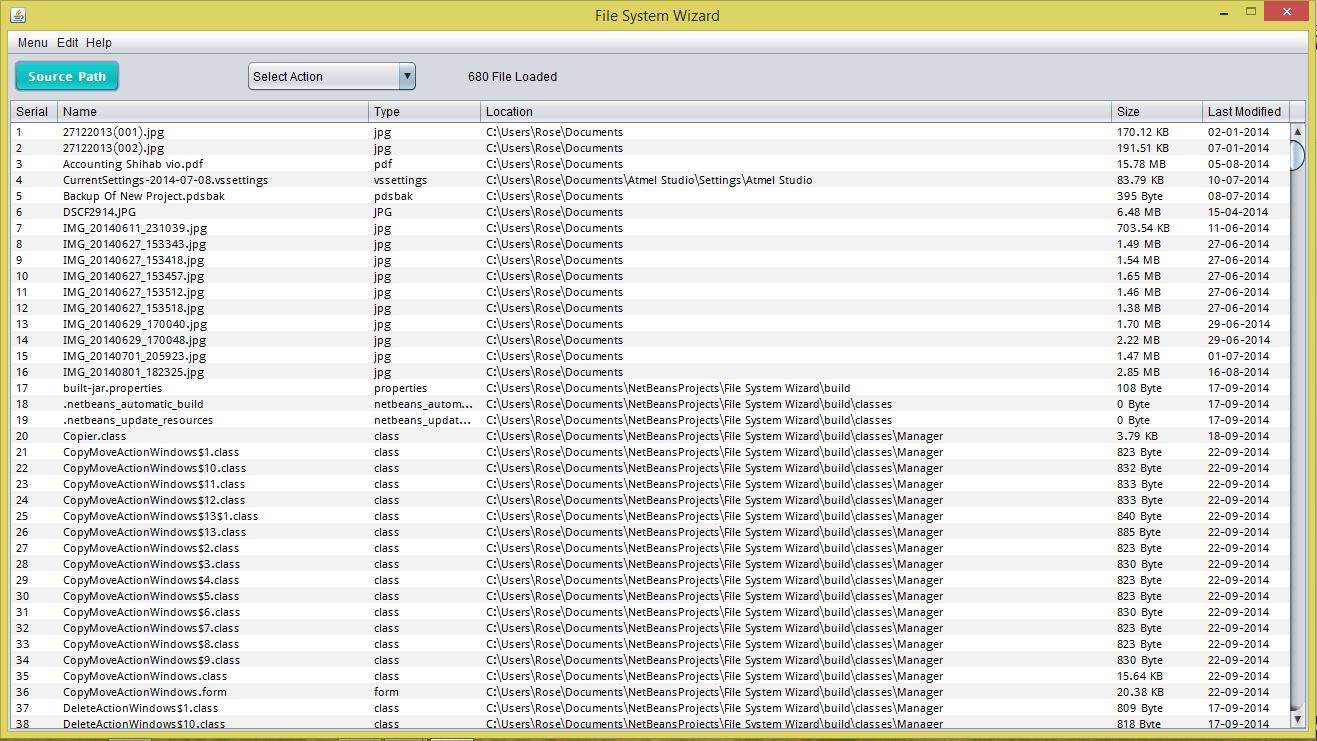


Fig:5 Mainwindow

* 1. File Chooser

This class is create to select files. When this class is called from Mainwindow, give the opportunity to choose file browsing the whole computer and return the selected files to the Mainwindow.

private void extractFiles(Path dir) {

try {

DirectoryStream<Path> stream = Files.newDirectoryStream(dir);

for (Path path : stream) {

if (path.toFile().isDirectory()) {

if ((path.toFile().isHidden() && path.toFile().canRead()) == false) {

extractFiles(path);

}

} else {

if ((path.toFile().isHidden() && path.toFile().canRead()) == false) {

progress++;

allFiles.add(path.toFile());

}

}

}

stream.close();

} catch (IOException e) {

System.err.println("ERROR");

e.printStackTrace();

}

}

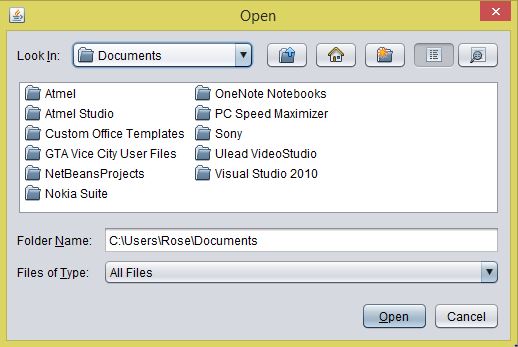


Fig:6 Screenshots showing FileChooser

* 1. CopyMoveActionWindows/DeleteActionWindows

This is bulkiest of all classes in the project, and is primarily concerned for coping/moving/deleting, as I shall call them. This class also extends the built-in java class javax.swing.JDialog, and is meant to act as a dialog window. From here the user can choose option all, extension, contains word, start with, last modified, size etc. for coping/moving/deleting files(see screenshots). This class also has a progress bar and a label to show the progress of copy/move/delete actions. After done the work the window show successful message and after pressing ok button the window dispose.

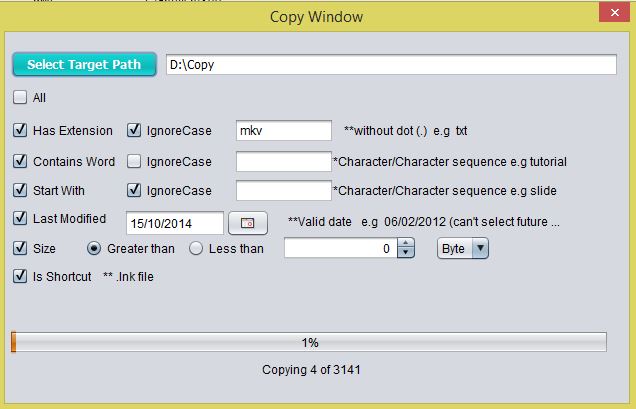


Fig:7 Screenshots showing Copywindow

* 1. Table Model class

There is a table in the Main Window of the application. The tables get their models created from the class FileTableModel which are in package “Tablemodel”. Each time a change is meant to occur to any of the tables, the model classes are invoked to create a model that represents the current contents.

public FilesTableModel() {

dataObject = new Object[numberOfFiles][6];

if (Names != null) {

for (int i = 0; i < numberOfFiles; i++) {

dataObject[i][0] = i + 1;

dataObject[i][1] = Names[i];

dataObject[i][2]= Names[i].substring(Names[i].lastIndexOf('.')+1);

dataObject[i][3] = MainWindows.files[i].toString().substring(0, MainWindows.files[i].toString().lastIndexOf('\\'));

double size = (double)(MainWindows.files[i].length());

if (size > 1073741824) {

size /= 1073741824;

dataObject[i][4] = String.format("%.2f", size) + " GB";

} else if (size > 1048576) {

size /= 1048576;

dataObject[i][4] = String.format("%.2f", size) + " MB";

} else if (size > 1024) {

size /= 1024;

dataObject[i][4] = String.format("%.2f", size) + " KB";

} else {

dataObject[i][4] = String.format("%.0f", size) + " Byte";

}

DateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");

dataObject[i][5] = dateFormat.format(MainWindows.files[i].lastModified());

}

} else {

dataObject = null;

}

}

@Override

public int getRowCount() {

if (dataObject != null) {

return dataObject.length;

} else {

return 36;

}

}

@Override

public int getColumnCount() {

return columnNames.length;

}

@Override

public String getColumnName(int col) {

return columnNames[col];

}

@Override

public Object getValueAt(int rowIndex, int columnIndex) {

if (dataObject != null) {

return dataObject[rowIndex][columnIndex];

} else {

return null;

}

}

}

* 1. Copier/Mover/Deleter

Those three class is create to perform copy/move/delete action. When those class called from CopyMoveActionWindows/DeleteActionWindow first select files to copy/move/delete as user given category then copy/move/delete those files to user selected destination.

for (int i = 0; i < paths.length; i++) {

Files.copy(paths[i], target[i], StandardCopyOption.REPLACE\_EXISTING);

double pr = ((double) i / paths.length) \* 100;

progressBar.setValue((int) (pr + 1));

label.setText("Copying"+Integer.toString(i)+"of"+Integer.toString(paths.length));

clk = i;

}

1. **Features and Usefulness**

The purpose of the application was to enable users to copy/move/delete their files in an efficient way. The application File System Wizard just copy/move/delete a batch of Files Under user given condition. The usefulness rendered by this application is described in brief below.

User can copy/move/delete File under those option

1. **Extension**: If the extension given by the user matches with the selected files extension then the action performed .The extension given by the user can be made case-sensitive or case-insensitive.
2. **Contains character or character sequence**: If the character/ character sequence given by the user matches with the selected files name extension then the action performed .The extension given from user can be made case-sensitive or case-insensitive.
3. **Start With**: If the character/ character sequence given from the user matches with the starting of selected files name then the action performed .The extension given from user can be made case-sensitive or case-insensitive.
4. **Last Modified**: If the date selected by the user matches with the selected files last modified date then the action performed .
5. **Size**: The user can set a lower-limit or upper-limit on file-size for those to be copy/move/delete.
6. **Shortcut**: If the files are shortcut the copy/move/delete action performed.
7. **Limitations**

As no single program can fulfil all requirements, our application **File system Wizard**, hence, has some shortcomings. These, along with the improvements that can be made to this application (in Section-10), are discussed below:

1. The action once done can not be undone.
2. If the file already exists in target directory or same name file exists in target directory that file will be replaced with new file.
3. If there exist some selected files in the tablemodel and we again select some new files, then the existing files will be replaced by the new files.
4. More time consuming .
5. **Improvement**

The following improvements could be made to the application against the limitations stated above:

1. Undo option should be added to undo action performed before.
2. If the file already exists in target directory or same name file exists in target directory There should be an option to the user whether he/she wants to replace or not.
3. If there exist some selected files and user selects some new files again, then there be an option whether the user wants to replace the existing or add with the existing.
4. Renaming option should be added with this so that, user can rename files.
5. **Conclusion**

**File System Wizard** as an application does what it is supposed and meant to do. Although, user range for such an application is quite small, those who need to have an application that does coping/moving/deleting for them as easily and efficiently as **File System Wizard** does, this application, we hope, will be of great value.

1. **Reference**

* The Java SE 8 Documentation by Oracle Corporation
* The Java Tutorial by Oracle Corporation
* Java – How to Program, 9th Edition, Paul Deitel & Harvey Deitel
* <http://en.wikipedia.org/wiki/Integrated_development_environment>
* <https://netbeans.org/features/>
* <http://searchsoa.techtarget.com/definition/Java-Runtime-Environment>
* <http://java2everyone.blogspot.com>
* <http://en.wikipedia.org/wiki/Java_Development_Kit>
* <http://zetcode.com/tutorials/javaswingtutorial/>
* [www.javatutorialhub.com](http://www.javatutorialhub.com)
* [www.youtube.com](http://www.youtube.com)
* <http://en.wikipedia.org/wiki/Java>
* <http://docs.oracle.com/javafx>