# A NOVEL APPROACH FOR MRI IMAGE CLASSIFICATION AND IMAGE RETRIEVAL

By

ANKIT V MASTER

Presented to the Faculty of the Graduate School of

The University of Texas Arlington in Partial Fulfillment

of the Requirements

for the degree of

MASTER OF SCIENCE BIOMEDICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2008

## ACKNOWLEDGEMENTS

First and foremost, I would like to pay my obeisance to my mentor; Dr. Michael D. Devous Sr. He has been a great source of encouragement and support all throughout the course of my master’s degree. I have learned many important lessons from this lab. This lab has taught me the importance of working in group and independently, critical thinking and time management. I thank Dr. Devous for guiding me and making this thesis possible.

Next, I am grateful to my committee members Dr. Linda Hynan and Dr. Hanli Liu for sparing their precious time and taking interest in my thesis work.

I would like to thank all my lab members for guiding me; by giving their inputs in my thesis. In particular, I am grateful to Tom Harris for his patience and effort in listening to my difficulties and helping me in every possible way he could.

I would also like to thank my friends Ruddhi Chaphekar, Parth Bhave, Jaimin Darbari and Andrew Noronha for their support and encouragement.

Next, I thank my Father, Mother and Anand who are always a constant source of inspiration, encouragement and support.

Last but not the least; to all my teachers from whom I have ever learned, in my life.

July 18, 2008

ABSTRACT

A NOVEL APPROACH FOR MRI IMAGE CLASSIFICATION
AND IMAGE RETRIEVAL

Ankit V Master, MS

The University of Texas at Arlington, 2008

Supervising Professor: Dr. Michael D. Devous Sr.

OBJECTIVE: Classification of MRI image data based on its attribute characteristics is very important for storing image data and its efficient retrieval. This project aims to find a solution to the problem of classification of the images that are stored on a hard drive and image retrieval from the image database with optimum automation in a user friendly way. METHOD: Matlab was used to create a software tool for image classification, and Java and SQL were used to create a software tool for image retrieval. RESULTS: The image classification tool could read multiple files, extract attribute values from the files and store them in a directory hierarchy automatically. The image retrieval tool could query the database and route the user to the actual image data that was stored on the hard drive. CONCLUSIONS: The image classification tool was efficient at classifying images in relatively short time as compared to other tools and with minimum labor. The image retrieval tool could successfully execute queries via a user friendly

Graphical User Interface (GUI), without the user having to know the programming language of the database.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

# CHAPTER 1

# INTRODUCTION AND BACKGROUND

## 1.1 Introduction

Modern neuroimaging techniques (especially Magnetic Resonance Imaging, MRI) produce large volumes of data. There are no well-established methods for storing and categorizing these data in a manner easy to sort or retrieve according to typical research-driven criteria. Consequently the UT Southwestern Alzheimer's Disease Center (ADC) had initially chosen to store MRI data on a locally available hard drive sequentially, that is as data were acquired. The purposes of this project were to develop software that would 1) automatically sort these data according to criteria that would be useful for prospective research investigations and 2) allow investigators to query and retrieve the stored data according to these research criteria, employing a user-friendly graphical interface (GUI).

These MRI images follow the Digital Imaging and Communication in Medicine (DICOM) standard [1]. DICOM is a standard for storing, handling, transmitting and printing information in medical imaging. The data that was stored was difficult to access because it was not organized. It was therefore decided to organize these data into a directory structure. For organizing the data, such that it could be accessed by the neuroimaging investigators, the data needed to be classified according to their characteristics (attributes). Since all the image files followed DICOM standards, all of them had the same internal structure; that is,  all DICOM files store their data in their header (meaning all the attributes of a DICOM file are stored in its header). Accessing these attributes using available neuroimaging software tools was extremely cumbersome because all of the available tools could read just one file at a time. In addition, the user had to manually perform the procedure of opening up a file (in the available neuroimaging tool), accessing the header and search for different attributes within that header. This process was very time consuming and laborious.

The ADC investigators lacked a reliable mechanism for executing the task of data organization. Manually sorting files was infeasible because, typically a subject had around fifteen thousand files. Hence, a tool was required which could bring in some automation, by reading multiple files, automatically accessing header information, creating the directory structure as required and storing the files in their respective folders.

After organizing the data based on its classification, accessing the data was still inconvenient for neuroimaging experts.  A database was created containing all the information of the subjects that was retrieved from the image classification.  However, accessing this subject-related data was a problem because the neuroimaging investigators are not experts at writing a query in the database (which has its own query language known as Structured Query Language (SQL)). A typical query may be of the type “How many FLAIR and DTI scans do we have on files that were collected at 3T?” Further, once the relevant subject names were obtained, investigators still had to manually search for each subject folder on the drive on which the data was stored. In order to address this problem of data retrieval, a tool was created which would allow the neuroimaging investigator to query through the database (without having to know SQL), retrieve data and get routed to image files.

### *1.1.1 A Case Study*

We consider the database of the ADC and try to solve the issues related to data classification and extraction. The DataBase Management System (DBMS) considered is Microsoft Access. Microsoft access uses Structured Query Language for retrieval of information. The data so considered follow DICOM standard.

### 1.1.2 *Challenges*

As mentioned, a DICOM image contains several attributes. In order to classify the images based on their attributes some sort of automation is desirable. Currently available software tools can only read one file at a time and require manual labor to read the attributes and sort them accordingly, so software which reads multiple files as its input, automatically extracts the attributes from the header and sorts the files according to the criteria of modality, magnetic field strength and series description is desired. After having classified the files according to these characteristics, some sort of automation is desired to retrieve the data from the database in a cumulative manner. Different users have different requirements and for every different requirement a new search query needs to be designed using Structured Query Language (SQL) if the database were to be used conventionally (directly).

### *1.1.3 Motivation*

In order to allow of data classification with minimal manual labor and data retrieval without having to write a different SQL query every time, there was a need to develop a new, automated and more efficient system for the process of image classification and data retrieval.

### *1.1.4 Scope*

In order to solve the problems related to image classification and retrieval, two software tools have been developed. Both tools are desktop applications. These tools are independent and are currently not interfaced.

### *1.1.5 Goal*

The primary goal of the project was to develop and deploy the tools that classify and retrieve images successfully.

## 1.2 <u>Background</u>

### 1.2.1 *Software Background*

The tool for the image classification was developed using Matlab. Matlab has an image processing toolbox which enables the user to make use of its built-in functions to read the header and extract necessary information (attributes) from it so as to utilize that information and perform more complex tasks. Matlab was preferred over other programming languages because of the user friendly programming language, online support and easy availability. On the other hand, the tool for image retrieval was built using Java and Structured Query Language (SQL). Java was selected as programming language for several reasons, one of them being Java is a free and an open source language. Java derives much of its syntax from C and C++ but has a simpler object model. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture i.e. Java is platform independent. Platform independence means that programs written in the Java language run similarly on any supported hardware/operating-system platform. One can write a program once, compile it once, and run it anywhere.

### 1.2.2 *Background on Database*

The database management system being used in this system is MS Access. Microsoft access stores data in tabular format. Access can use data stored in Microsoft SQL Server, Oracle, or any Open DataBase Connectivity (ODBC)-compliant data container. It supports some object-oriented techniques but falls short of being a fully object-oriented (database) development tool. Using the ODBC feature of Microsoft Access, Java can connect to it, execute SQL queries and display the result. Microsoft Access uses SQL as its query language and as a result whenever user requirement change a new SQL query needs to be fired. Writing a new SQL query effective for investigators with some background in retrieving data from the database. However, for a person having no knowledge of SQL or the database, writing a SQL query is cumbersome. In addition, if the user were to execute an SQL query every time, the user needs to know the location of the database in order to access it.

### *1.2.3 Alternate approaches for data retrieval*

The data (patient ID, scan types, modality, scan date, manufacturer, etc) can be stored in an Excel spreadsheet and the Excel spreadsheet can be searched manually for the relevant information. This technique works well when patients with only one or two scan types are to be searched, but when there are more than two scan types to search for, this procedure becomes tedious and there is a strong possibility that the user may miss some of the relevant data. Before choosing Java as the programming language experiments were made with .NET and Matlab**.** Since, C#.NET is not platform independent and without support from Microsoft, very few third parties have been willing to implement it. Swing, one of the Application Programming Interface of Java is more powerful than System.Windows.Forms (SWF) of C#.NET due to which creating GUI in Java is more convenient. Also, Java supports better Integrated Development Environment (IDE) for the development of the program and support more features than .NET IDE. However Matlab could have been used. Matlab has a database toolbox which can perform the operations of firing queries, getting the result set and displaying the result effectively. Unfortunately, Matlab is not platform independent. Due to these challenges and shortcomings, two software tools were created using languages that were most suitable for their implementation as platform independent, easy to use and readily available.

# CHAPTER 2

# METHODS

The following subsections describe the methods that were adopted for creating the tool on image classification and image retrieval respectively. Section 2.1 describes the methods, proposed for solving the problem of MR image classification and image retrieval respectively and section 2.2 deals with the design aspect of the proposed solutions.

## 2.1 Method Proposed

### *2.1.1 Software for image classification*

As mentioned, images that follow DICOM standards have all their patient pertaining information (Patient ID, Manufacturer name, Series Description, Series Number, Study Date, Voxel size, Slice thickness etc) stored in the DICOM header in sets. So, in order to access any of the DICOM header information for sorting the images based on their Patient ID, Modality, Scan date and Scan type these attributes need to be searched for in the header. There are two approaches for doing so. First is to make use of available neuroimaging software tools and read the header of a single file at a time and sort the files. The second approach is to write a code which automatically reads the headers of multiple files and classifies the files according to the requirement, using some programming language. The problem of file classification was solved using the second approach. Matlab was chosen as the programming software. Matlab has an image processing toolbox (a toolbox is a library of software codes which supports a specific application in a user friendly way) which supports DICOM format. Matlab stores all the DICOM header information in a one dimensional array. Since the array can be traversed, pulling out attributes from the array becomes convenient. Once the required attributes are pulled out, they are sent through a series of loops that check them for a particular attribute value and sort the files accordingly. This concept works fine with a single input. However, if multiple files were to be read, a concept called structures has to be used. We will discuss in detail the software design in subsequent sections.

### 2.1.2 *Software design for image retrieval*

Once the images have been classified all of their relevant attributes are stored in a database. In order to extract the images based on the information stored in the database, a software tool was developed using Java. Programs that are written in Java run equally well on all platforms. Here a Graphical User Interface (GUI) is created using Swing. Swing is an Application Programming Interface (API) of Java, which is used to provide a sophisticated set of GUI components. After having created the GUI, the second task was to connect the GUI to the database. Connection to the database was accomplished using Java DataBase Connectivity (JDBC), another API of Java. Here, we used Microsoft Access Database Management System (DBMS), which uses SQL in order to query, retrieve and display the result [3]. After the connection to the database is established, SQL queries are written to handle both simple as well as complex tasks. JDBC not only connects to the database but also supports SQL. This means that all of the SQL code can be written and executed in Java. Certainly, the mapping between the data types of Java and SQL needs to be done.  After execution of the query the result so obtained is saved in an Excel spreadsheet. One of the important features of this result is that it contains hyperlinks to the folders of the drive where the actual images are stored. Creating a hyperlink in the Excel spreadsheet using Java is different from just copying and pasting a column having hyperlinks from one table to another. Here we are working with the database and Java which have a different style of interpreting data. The technical details of this process are described in the design section.

## 2.2 Design

### *2.2.1 Design of the image classification software*

The flow chart shown below shows how the image classification program works

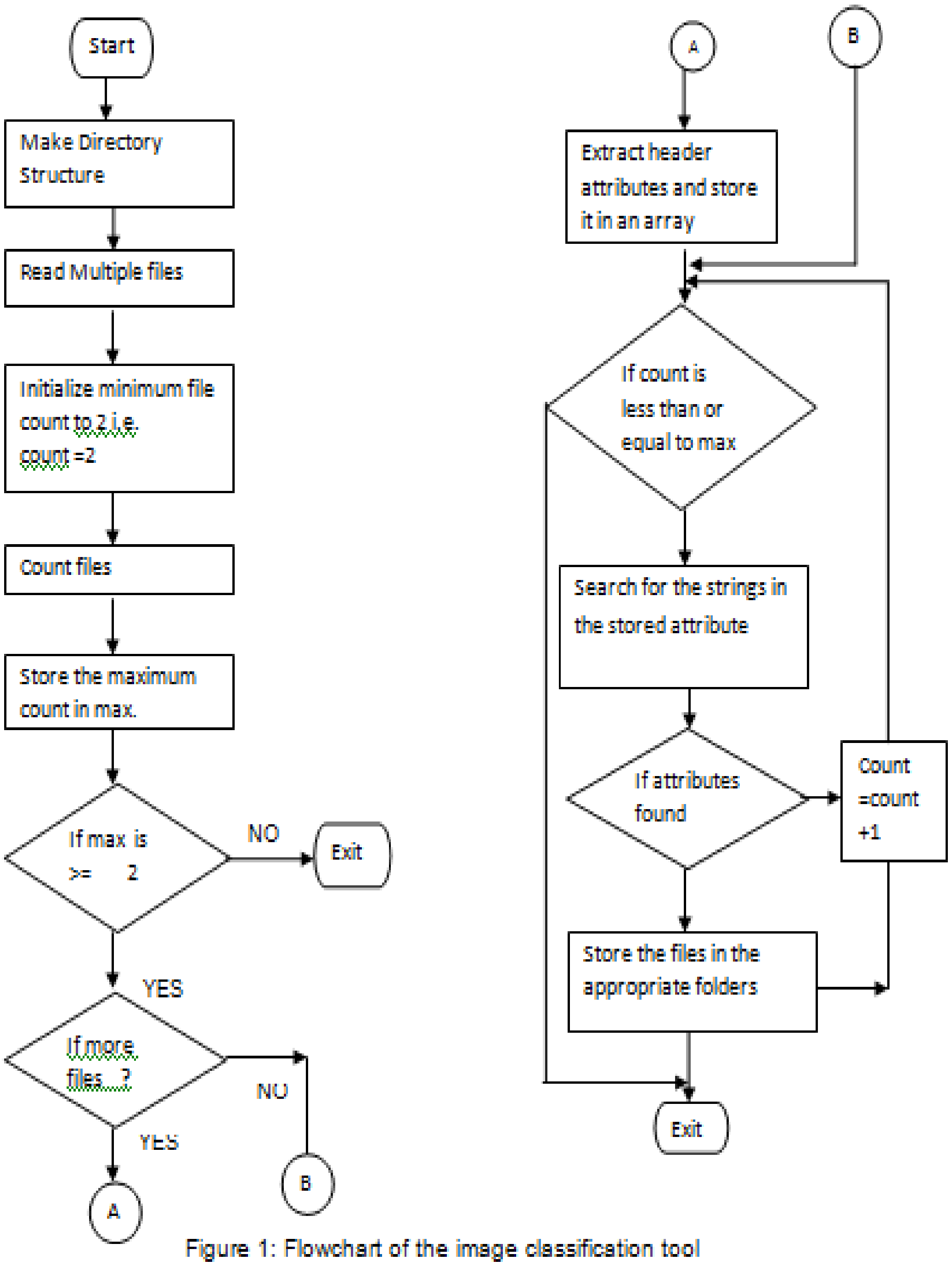
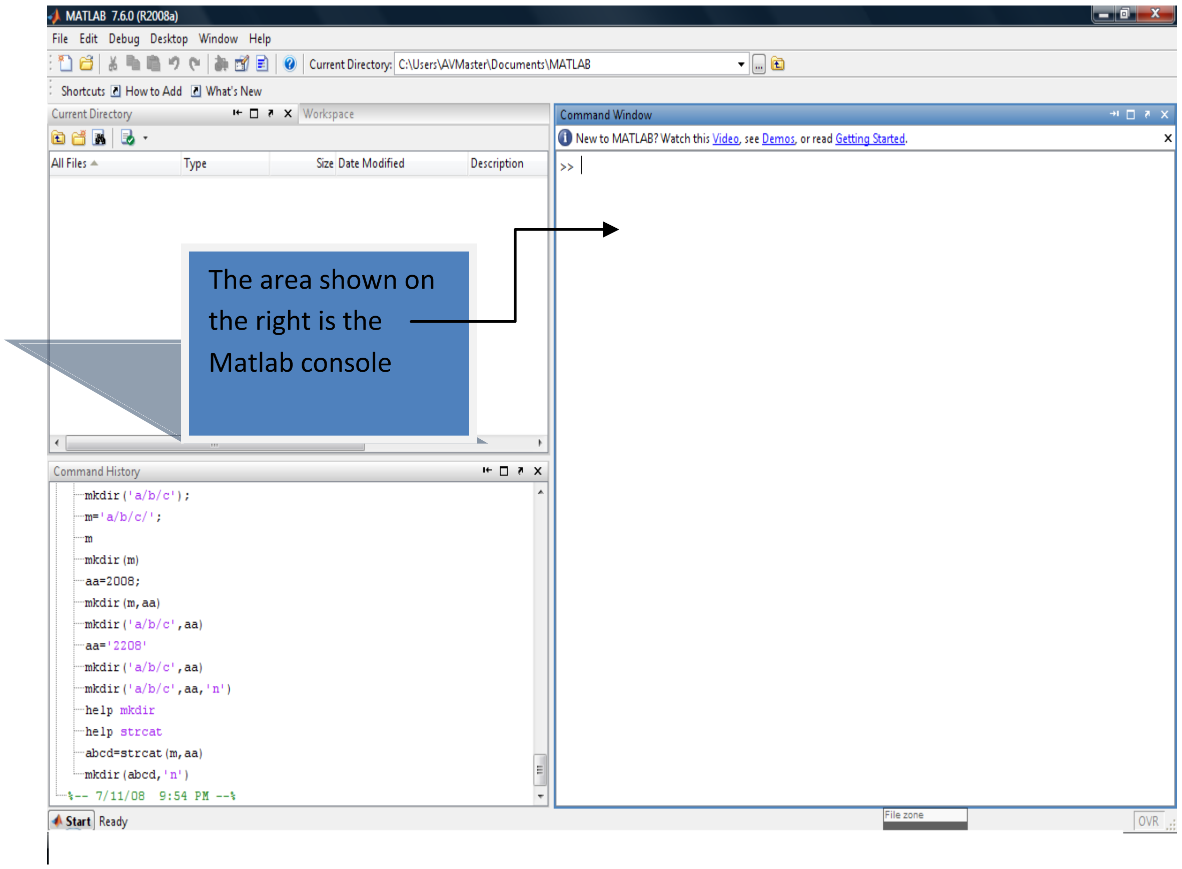

Start
Make Directory Structure
Read Multiple files
Initialize minimum file count to 2 i.e. count =2
Count files
Store the maximum count in max.
If max is >= 2
NO
Exit
YES
If more files...?
NO
YES
A
B
A
B
Extract header attributes and store it in an array
If count is less than or equal to max
Search for the strings in the stored attribute
If attributes found
Count =count +1
Store the files in the appropriate folders
Exit

Figure 1: Flowchart of the image classification tool

Explanation of the flow chart: The program starts with clearing all the data that is stored in the memory and clearing all the contents from the console. A console is window in which commands can be typed in directly for performing a particular action. Figure 2 shows the Matlab console. As soon as the program starts it creates a directory structure. The first folder of this directory is named PID. This folder is used as the buffer folder, to separate all the internal folders from the external environment. This folder (PID) however has to be renamed by the user. The PID folder contains a sub folder which has the name of the modality (specifically MRI). The MRI folder again contains two sub folders by the name of 1.5T and 3.0T. 1.5T and 3.0T subfolders indicate the magnetic field strength at which the MRI data were acquired. After having created this directory structure the program reads multiple files and stores the filenames and pathnames in a two dimensional array. Rows and columns of this array are then read to create two arrays, which stores only the filename and only the pathname. The total number of files can be determined from the count of the array which stores the filenames. Hence, we create a matrix which stores the total count of all the input files. Also, doing so dynamically changes the number of files handled by the program at a given time. Approximately 38 variables are initialized to zero so that they can all be used in the conditional loops to follow. All the files that are read are DICOM files and therefore all of them have the same structure in which they store the information. In order to access the attributes of interest from the headers of multiple DICOM files we make use of the concept of structure. A structure is a user defined data type, which can store multiple data. The data that is stored by the structures may have different data types (e.g. integer, string, double etc). However, once a structure is declared all the variables that are of type structure can store only those data for which the structure has been programmed.

All the files are stored in a variable which is of data type cell. This data type doesn't support the retrieval of attributes from the header. In order to access the attributes from the header we need to change the data type of the files from the cell to structure. This is accomplished using a Matlab command called struct. Once the files are converted into structures the next task is to access the attributes of interest from the header which are modality, magnetic field strength, study data and series description. In the process of searching through the headers, there have been times when the magnetic field strength has been missing in the header file and due to this the program originally failed with an exception.  In order to rectify this problem the program was revised so that it reads the attribute of the magnetic field strength from the header and if magnetic field strength is missing from the header it replaces the null (missing) attribute with a string saying "data not available."


MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: C:\Users\AVMaster\Documents\MATLAB
Shortcuts How to Add What's New
Current Directory
Workspace
All Files Type Size Date Modified Description
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
The area shown on the right is the Matlab console
Command History
Start Ready
OVR

```
>> |
```

```
mkdir('a/b/c');
m='a/b/c/';
m
mkdir(m)
aa=2008;
mkdir(m,aa)
mkdir('a/b/c',aa)
aa='2208'
mkdir('a/b/c',aa)
mkdir('a/b/c',aa,'n')
help mkdir
help strcat
abcd=strcat(m,aa)
mkdir(abcd,'n')
%-- 7/11/08  9:54 PM --%
```

Figure 2: Matlab console. Matlab console can be used for script programming and for other general purposes like checking the type of a variable.

The image classification tool was originally developed to identify the files according to their scan types (e.g., study date, manufacturer, file name, Patient ID, institution and magnetic field strength). All the files belonging to a particular folder had the same characteristics. There were many folders that had files in them but no information was available about them. So it was required to create a search routine that could read several files (one from each folder) and extract the required information from them so that the data can be stored in the directory structure; which is Patient ID\ Modality\ Magnetic Field Strength\ Scan date\ Scan type(s). The criteria behind designing the directory structure mentioned above can be explained as follows. Each sub directory name (for example, Patient ID) in this structure corresponds to a particular attribute in the header of a DICOM image. A header of a DICOM image contains all of its attributes - typically there are over 40 of them. Patient identification number, patient birth date, patient sex, scanning sequence, MR acquisition, repetition time, number of averages, slice thickness, echo train length, percent sampling are some of these attributes. Most of these attributes give an insight into the technical details of MR acquisition. All of these attributes may or may not be of interest to neuroimaging experts at the onset. At the time of searching for subjects, specific attributes must be looked for, in the header. Hence the neuroimaging core of ADC decided to have a directory structure representing a hierarchy consisting of the attributes that are most frequently sought at the time of searching for data. The meaning of each of these most frequently sought attributes is described next. Patient identification number (known as patient id according to the naming convention used by ADC) is used for identifying patients. Modality determines the scan type to which the image belongs, from among types like MR, PET SPECT, etc. ADC possesses neuroimaging data belonging to not only the MR modality but also other modalities like PET and SPECT, although fewer in number. Different MR data could be of different magnetic field strength, determined by the Magnetic field strength attribute value. Value of the Date attribute helps keep a track of subjects that have several scan dates, to distinguish subjects based on the study group and to determine the improvement in a particular subject. Scan type (which has values like T1, T2, FLAIR, DTI, etc.) is used by most investigators as a criterion to search for data based on their scan types. Thus, the informational value of the most important attributes, as described in the preceding sentences, aided our choice of the directory structure.

Since the amount of data was huge, a group was formed in order to organize the data, which was already loaded on the hard drive. This group was further divided into smaller sub-groups, each of which were assigned the responsibility of organizing a particular portion of data. Most of the files that were already loaded on the hard drive were stored in a particular folder and each of these folders contained files that had the same attribute values (that is, the same scan type and scan date and so on for each attribute). Therefore, a single file (from such a folder) could be considered to fully represent all files in that particular folder in terms of their attribute values. Hence, for each such folder, classification of files stored in it, was not required. However, each such folder had no name and identification indicating the scan type of files stored in it. Hence, the task of reading the scan type of a file inside such a folder and renaming the folder with that scan type was still required to be performed.

Due to the indeterminate nature of the data that was stored on the hard drive, there were times when a particular team member had to read only a few files at a time or read all the files that were stored in a folder under consideration. In either case manually reading attributes from the files was time consuming and the group working on organizing the data was in need of some sort of automation in the process. Time assigned for the organization of data that was loaded on the hard drive was three months and therefore the first software code was written to read multiple files, extract the attributes from the header of the files and write the attributes on an Excel spreadsheet. Information related to each file was written on a new sheet. Since each sheet had a file name as one of its attributes, this helped the user to distinguish between different files. Based on the information so extracted from the headers of the image files a database was created. However, attributes of filename, patient ID, manufacturer and institution, have been disabled, as they are no longer required but they can always be enabled by including them as an argument in the body of the main structure of the code. These attributes were disabled because, initially, value of the attribute named filename was used, so that the user of the code could distinguish between different files and manually create the folders and place the files in their respective folders, since the updated tool can perform this procedure automatically this feature is no longer required. The attribute named patient id was removed because the patient ID assigned to the subjects by the ADC was different from the patient Id that was present on the files of the scans. Attributes Manufacturer and Institution name were included with an intention of including these attribute values in the database and since the manual procedure of creating the folders for storing the files was followed initially, these attributes could be copied from the Excel sheet. The latest tool can sort the files and move the files into their respective folder automatically and hence the Excel sheet was removed from the design, therefore there is no place to save these attributes. Values of the attributes named manufacturer and institution can be obtained from the Matlab console. However, there were many folders which did not have files stored in them in any particular format or order.  In order to deal with this situation, new features were added to the tool and once the database was organized unrequired features were removed from the tool.

Coming back to the present version, since we are dealing with multiple files we create an array of cells to store attributes related to each file (e.g. magfs{1} stores the magnetic field strength of the first file that has been read, magfs{2} stores the magnetic field strength of the second file and so on). After creating the arrays and storing different attributes in different arrays we combine all of these arrays to form a structure (vaguely, a structure is a user defined data type, as mentioned earlier) so as to organize the data and make maximum sense out of it. Converting the data from one type to another is called type casting. So now we have a structure whose elements are filename, patient ID, modality, magnetic field strength, series description and study date (). For example, consider a structure (user defined data type) named

‘Main’. Main contains a number of sub parts, each containing data corresponding to a particular file. Main can be indexed using a natural number i whose value ranges from 1 to N (N is any natural number). Thus, Main {i} refers to the ith sub part of Main which contains data corresponding to the ith file.

Thus, Main {1} would contain all the required attributes of the first file, Main {2} would contain all the required attributes of the second file and so on.

If we were to look at the “structure” of the structure this is how it would appear

```
Main{ i} = (   filename {i}

               Patient Id {i },

               Modality {i},

               Magnetic field strength {i},

               Series Description {i},

               Scan date{ i}     )
```

After having extracted all the required attributes from headers of multiple files and organizing it in the structure, such that the data is organized in a usable way, the next task is to find the required attributes from within arrays of attributes. That is, if there are 500 files given as input, there would be an array of series descriptions within the structure with 500 attributes. Initially the value of various attributes is unclear (e.g., consider an attribute of series description, all of those 500 files could have T1, or some of them could be T1 and others could be FLAIR and so on). In order to delineate attribute values we make use of several conditional statements, which identify up to 14 different types of scan types, all of which are enclosed in an iteration statement. The logic in all the conditional statements is similar.

Logic within the iteration statement is as follows; we declare different conditional statements each of which searches for a specific scan type (e.g. T1, FLAIR, DTI, SWI, ASL, etc). If a match is found in any of the conditional statements that particular conditional statement is executed. Within every conditional statement there is one more conditional statement which checks for the magnetic field strength of the file and appropriately creates a folder of that particular scan type, but only once (i.e. for the first time when it encounters the respective scan type), in the directory structure created at the start of the program. Consider for example 2 files both of which are of scan type SWI having magnetic field strength 3.0T. The loop condition is checked; if the count of the file is between 1 and n, closed interval, the loop condition is true and the iteration of the loop is executed. After entering into the iteration loop the file encounters first conditional statement which checks if the file is a T1 file or not. Since the first file is not a T1 file, this conditional loop is not executed. Then the file checks for the second conditional statement, perhaps this time the conditional statement is checking for DWI. Again the conditional statement is not satisfied and hence the file checks the third conditional statement, and for example this time the conditional statement is checking for SWI files. This time the condition statement executes and the file enters into another conditional statement. This conditional statement checks for the magnetic field strength of the file. If the magnetic field strength is 3.0T it creates a new folder by the name of SWI in the directory structure of PID/ 3.0T/, but only for the first time it (conditional loop) is encountered with that file. After having created a folder by the name of SWI the program automatically places the file(s) in that folder. After placing the file in that folder the program increments the count of the file to 2 and passes the control to the next iteration of the loop. Now the file count is 2 and again the same steps are executed until the condition of the conditional statement is satisfied. Once this condition is satisfied, the program again checks for the condition of magnetic field strength, if the magnetic strength is 3.0T this time the file doesn't create a new folder but moves the file into an already existing folder In order to get more insight into the working of the code please refer to Appendix 1.

### 2.2.2 *Design of the software for image retrieval*

The section below describes the tool required to search image files placed in storage and to populate a database with relevant variables describing these images. This tool will primarily be used by the image experts in the Neuroimaging Core of the ADC. However, many of these investigators who need access to image data are not image experts. These investigators needed a tool to query the database created by the first set of tools in order to identify image data relevant to their scientific question. A sample question might be "How many FLAIR scans do we have on file that were collected at 3T in subjects between the ages of 55 and 80?". In addition, that investigator will need to know the exact location of that file for ready access in subsequent analyses. The following tool was developed for that purpose.

The GUI for image retrieval is created using an Integrated Development Environment (IDE) of Java named Netbeans. Netbeans provides certain features which can be utilized for the creation of the GUI. The GUI consists of 3 tabs, each of which contains a panel and each of these panels is programmed to hold a specific number of other Java components like buttons, checkboxes, radio buttons, labels, text field, etc. The first tab contains 2 components; both checkboxes. This tab is used for the selection of magnetic field strength. Since these are checkboxes, either one or both of them can be selected at the same time. The second tab contains 15 components, 9 of which are check boxes, 3 are labels, 2 radio buttons and 1 for the button group. The radio buttons need to be grouped so that only one selection is made at a time. The second tab allows the user to select multiple scan types from the provided list. The radio button in the second tab selects the Boolean operation to perform (AND or OR). The Boolean selection allows the user to search for all the patients with those particular scan types (in accordance with the Boolean operation). The third tab contains 14 components, 4 of which are text fields, 1 is a button and the remaining 8 are labels. Four text fields are provided to search for keywords. The user can search for 4 keywords from the available selection of name, scan date and comments. The user may opt for the third tab if the user is unable to search for subjects by the criteria provided from the first two tabs (e.g. search by name Andrew). We describe the technical details of how these components are stored on the tabs and what these components are in the design of class NewJFrame. The key feature of this GUI is that it allows the user to use all 3 tabs in coordination (e.g. search for the patient with magnetic field strength 3.0T, having scan types T2 and DTI and FLAIR and names Andrew and Newman, or all patients having magnetic field strength of 3.0T and 1.5T and having scan types of T1 OR T2 OR SWI etc). This application is a desktop application. Once the GUI is created it's connected to the database using JDBC-ODBC bridge (a driver). On getting connected to the database, SQL queries can be executed and the result so obtained will be stored in the Excel spreadsheet. In addition to storing and displaying result in an Excel spreadsheet the result set can also route the user to the respective folder of the subjects that have been searched for. This is achieved by including a column of hyperlinks in the result.

Putting everything together, the above process is achieved by different pieces of software. These pieces of software are called classes. In simple terms, a class is a user defined data type, which can contain elements of different data types, implement certain actions, interact with other classes and return some value. The above mentioned design falls into the category of object oriented design which extensively makes use of objects. An object is a name given to the term variable which is of a user defined type. Objects are nothing but variables of data type class (to put it in Object Oriented Programming (OOP) terminology, an object is an instance of the class). For example, consider a class by the name of Vehicle, which can have many features but the most basic and required features are capabilities provided by the acceleration system and the steering wheel represented by methods accelerate() and steer(), respectively,  and attributes color and weight. Vehicles can be of different types, for example, Automobile, Airplane and Boat. However, all of these have all the features that a Vehicle has. Hence, Automobile, Airplane and Boat, are all sub classes of class Vehicle which inherit all attributes and methods of class Vehicle. However, each of them has certain unique attributes and capabilities. For example, an Automobile has tires which a Boat does not. An Airplane has wings which an Automobile and a Boat do not. A Boat has an anchor which an Automobile and an Airplane do not. Hence, we can say that the class Airplane has attribute WingLength, class Boat has an attribute AnchorWeight and class Automobile has attribute NumberOfTires. Everything that can be done by the class can also be done by its objects (e.g., read, write, perform certain actions, copy etc).

Before we delve into the technical details of the program on image retrieval there is one more concept that needs to be introduced with respect to object oriented design, which is inheritance.  Inheritance means that all the public features of a particular class (parent class) are also available to the class (child class) which inherits the original class. Apart from these inherited features, more features can be added to the sub class (e.g., child).

### 2.2.2.1 Design of the Classes

#### 2.2.2.1.1 Class NewJFrame

NewJFrame class is responsible for creating the GUI, connecting the GUI components to other parts of the program and storing the result in an Excel spreadsheet. NewJFrame class inherits a class called JFrame. JFrame class acts as a base class for all the other components. To do so, we create an object of class JFrame so that we can add some more features to it. We then make use of another class called JTabbedPane and do so by creating an object of this class. After having created both of these objects, an object of class JTabbedPane is added onto the object of class JFrame. JTabbedPane is responsible for creating tabs. Tabs as such cannot hold components like radio button, button, checkboxes etc (at least in Java, they do not). So in order to hold the components we make use of another class called JPanel which creates a panel. A panel created using JPanel is also a base component on which other components can anchor themselves. JPanel class is a child class to the class JFrame. Three objects of class JPanel are created and added onto the object of class JTabbedPane; this way we get 3 tabs. What is interesting about this concept is we do not create 3 tabs and add 1 panel on each of them; we create one tab and add three panels as its components. Now, 3 tabs are available which are capable of holding new components. The components that are mentioned in the above sections are nothing but objects of some class. In particular, a button is an object of the Java class JButton, a radio button is an object of the Java class JRadioButton, a label is an object of Java class JLabel, a text field is an object of the Java class JTextField, a checkbox is an object of the Java class JCheckBox, and so on. In order to make use of radio buttons they need to be grouped so that only one button is selected at a time. In order to do so we make use of another object of class

JButtonGroup. Class NewJFrame also makes use of objects of other class that have been used to create the program (described in detail below).

#### 2.2.2.1.2 Class DBStatement

Class DBStatement is responsible for creating a connection between the GUI and the database. The DBMS used is Microsoft access. Java interacts with the database using an API called JDBC (as mentioned earlier). The first step for Java is to request the driver for the connection to the DBMS using JDBC-ODBC bridge, this is accomplished using a method named forName. After getting the required drivers the Java code needs to know the name of the database to which it needs to connect. Finally, after acquiring all the permissions and information that it requires, Java requests the database for the connection using one of its in built classes called Connection. An object of class connection is used to establish the connection between the GUI and the database. However, if for some reason the program fails to connect to the database an error is issued saying, “program not working”.

Figure 3 shown below gives a brief idea of how a Java program interfaces itself with the database. It is just the pictorial representation of what has been explained above.

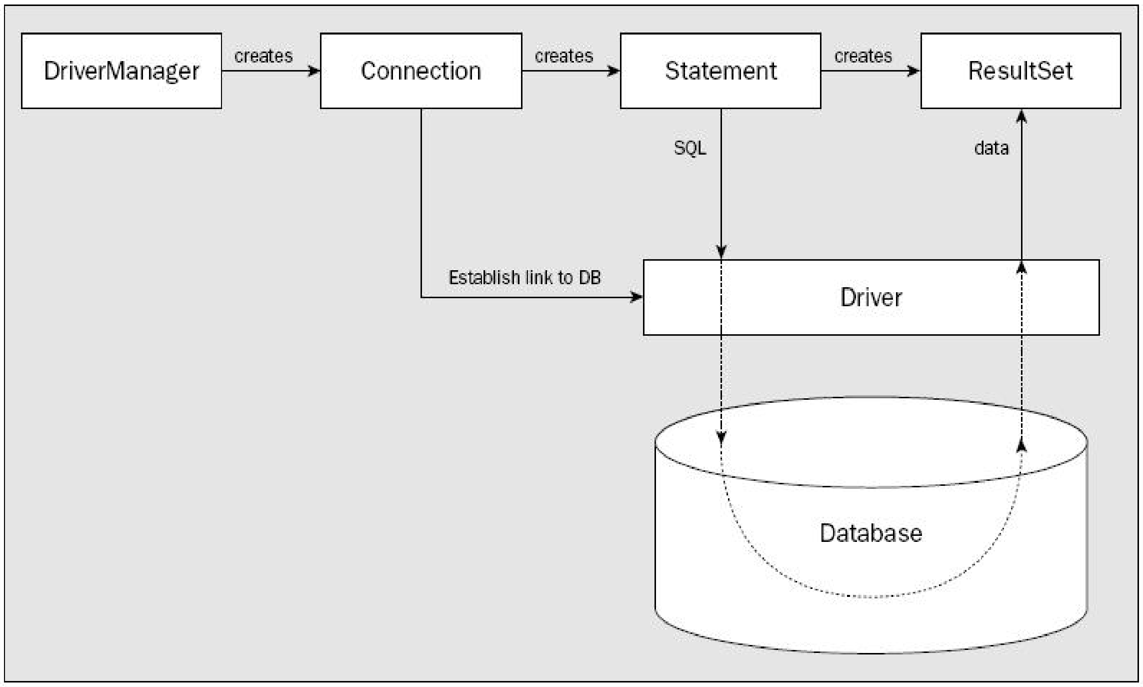


Figure 3: JDBC Architecture. (Adapted from Horton, Ivor(2005). Java 2, Indianapolis, IN, Wiley publications)

In order to execute the SQL queries Java has a class called PreparedStatement. Therefore we create an object of class PreparedStatement and execute an update query. An object of class PreparedStatement updates the columns which store the data of scan types (e.g. Proton Density, T1, T2, FLAIR, etc) by replacing a null entry by zero so that an exception (error) is not issued at the time of retrieving the data. After updating the database, again a query is executed on the object of class PreparedStatement but this time it is used to retrieve data from the database based on the selections that are made in the GUI. The general form of the SQL query (used in Java) is as follows ( Booleans explained AND and OR)

Display (name, ID, PID, scan date, hyperlink) from the (table_name) where (scan type1 AND scan type2 AND scan type3 AND … scan typeN) AND (Magnetic field Strength) AND (Comments) // AND OPERATION

Display (name, ID, PID, scan date, hyperlink) from the (table_name) where ( scan type1 OR scan type2 OR scan type3 OR … scan typeN ) AND (Magnetic field Strength) AND (Comments) // OR OPERATION

Where scan type(i) can further be disintegrated into (scan type(i) =1 OR scan type(i) =?) where the ? is filled at the time of execution of the program depending on the value selected by the user. If the user selects a particular scan type, then ? is replaced by 1; if not it is replaced by 0. If it is filled with a one the result of the OR causes the selects statement to retrieve only those records for which that scan type has a value one. Otherwise the value of that scan type doesn't play a role in selecting which records to display.

The effect of the value of magnetic field strength (mfs) is also determined by a similar OR operation i.e. (mfs = ? OR mfs =1). The effect of the comments field is determined by the values of fields first, last, scan date and comments, where, fist is the first name of the subject, last is the last name of the subject, scan date is the date on which the scanning was performed and comments field may or may not have any data in it. First, Last, Scan date, Magnetic field strength, Comments are the names of the columns in the database, through which the search routine is executed. Each of the four keywords entered by the user is searched for in the fields of first, last, scan date and comments. If any of the keywords is found as a part of any of the column values, entire row corresponding to that column is selected.

The tool searches for its results through the following columns, first, last, scan date, magnetic field strength, Proton density, T1, T2, FLAIR, DTI, FcMRI, ASL, DWI, SWI and Comments in the database.

The next part of the class DBStatement deals with the development of the logic at integrating class NewJFrame with class  DBStatement so that both the classes can communicate with each other by referring to the objects of other classes. This communication is made easier with the help of objects. With the use of objects is it just a matter of declaring an object of one class into the body of another class and accessing each and every member of the other class. For example,

```
Class A {    int a;                    // variable of Class A

             Double b;           // variable of Class A

             String c;             // variable of Class A

          }

Class B {

      Class A objectOfClassA = new Class A( );  // declaring an object of class A

      objectOfClassA.a = 56;   // variable of Class A accessed through the object of
Class A

      objectOfClassA.b = 5.6;   // variable of Class A accessed through the object of
Class A

      objectOfClassA.c = Hello; // variable of Class A accessed through the object of
Class A
```

```
        }
```

An array of integers has been created in class DBStatement, each element of which stores the state of a check box in tab 2 as a Boolean value corresponding to checked or unchecked, respectively. A value of one is stored in them if the check box is selected and a value of 0 is stored if the check boxes are unselected. The default value of all the elements of the array is zero (i.e. the check boxes are not selected). Once the check boxes are checked the values of all the elements corresponding to those checked check boxes change to one. As stated earlier, an array of integers is created in class DBStatement. An object of class DBStatement is created in Class NewJFrame and therefore this object has all the functionality of the class DBStatement, including all the variables and functions. Since an array is a part of class DBStatement, objects of DBStatement can refer to this array and use it to collect data. This is exactly what is done. Object of class DBStatement refers to an array and stores (1's and 0's) in it and since this array is already a part of class DBStatement, no special arrangements need to be made to pass on these values in it. They are automatically stored with it.  Similarly the states of the check boxes in the first tab, which prompt the user to select the magnetic field strength, are collected in the variables declared in DBStatement from an object of DBStatement in class NewJFrame. The default value for the magnetic field strength is 3.0T. Data from the text field in the third tab of the GUI is read using a method provided by Java and after reading the input from the user; the strings (user's input) are collected in class DBStatement via the object of class DBStatement in class NewJFrame. These collected variables are then fit into the SQL query.

Class DBStatement is also responsible for storing the result in the Excel spreadsheet. To do so it makes use of an interesting concept called metadata. A metadata gives data about the data (e.g., consider a table which has three columns, name (which is of type String), ID (which is of type integer) and grade (which is of type character)). Everything that is entered in this table is data, for instance consider a hypothetical row having values like John, 56, A, where John is a string element, 56 is an integer element and A is a character element. Thus, metadata enables the user to get all the data (data types) about the data that is stored in the table. The results obtained after running the SQL query are used to get the column count and column label using the concept of metadata. Once the column labels are extracted from the result set, the program enters into an iteration statement where every row is extracted one by one, at every iteration of the loop. Let’s take a more detailed look at how it is done. By default, the program points at the row that immediately precedes the first entry (row) that is made in the table. So a counter has to be set to increment the cursor to the next row of the table. Once the cursor increments to the next row value, every cell in that particular row is read from the database and written on the result table, when the program encounters the cell corresponding to the sixth column (sixth column to be displayed on the result table), the program makes the necessary arrangements for creating a hyperlink. A pictorial representation of a ResultSet object is shown in Figure 4

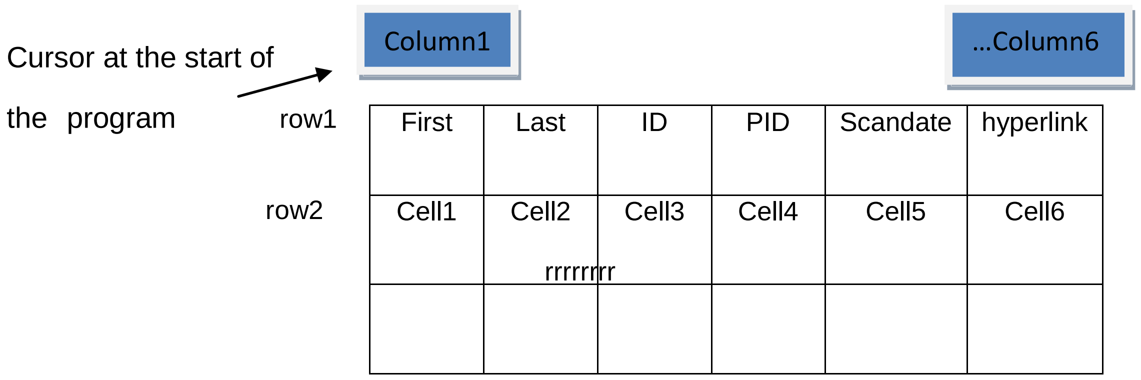


Figure 4: Reading of data from the database and writing it on to the Excel spreadsheet

Once all the data from the result set has been written to the Excel spreadsheet; the Excel spreadsheet is closed (the Excel spreadsheet is opened when the first data element is written on it, in this case, first column label, and remains opened till it is implicitly closed). As mentioned earlier, hyperlink strings are formatted in this class. Shown below is one of the examples of how a hyperlink data element appears in the database

#W:\CBCS\ADC\ADCMRI\031-S-1066(Target)\MRI\3T\2006.

The above written hyperlink is not a valid hyperlink due to the presence of the '#' sign. In order to make the above line, to work as a hyperlink, '#' needs to be replaced by a null string (the one which just removes the '#' and doesn't introduce any space character). The program is written such that always the sixth column of the result set displays the hyperlink, so an iteration statement is written which reads all the hyperlinks from the database and formats the hyperlinks appropriately. Hyperlinks allow the user to directly access the actual image data.

#### 2.2.2.1.3 Classes SQL1 and ResultDisplay

Class SQL1 and Class ResultDisplay were originally created for the purpose of displaying the data on the Java Table. A Java Table can only display the data, but provides no functionality to save it. So we need to create a special mechanism to do so. It is necessary to not only view results but also store them, so that they can be used for future reference.  Since the present tool doesn't display data on Java Table, these classes will not be explained in detail.

#### 2.2.2.1.4 Class SQL1 in a nutshell

Class SQL1 focuses on reading data from the database and writing it onto the Java Table. To do so, Class SQL1 makes use of two interesting concepts, namely vector and metadata. Concept of metadata has already been explained in class DBStatement, so it is not explained here. Vector however is a new concept in terms of data storage. A vector is a Java Class, whose objects can store both the string and 'object' data types. Class SQL1 makes use of two vector objects.  One of the vector objects is used to store the all the data of a single row ( as opposed to reading every cell of a particular  row, buffering them till a particular row comes to an end) and another vector object is used to store the previous vector. A pictorial representation of vectors is shown in figure 5.

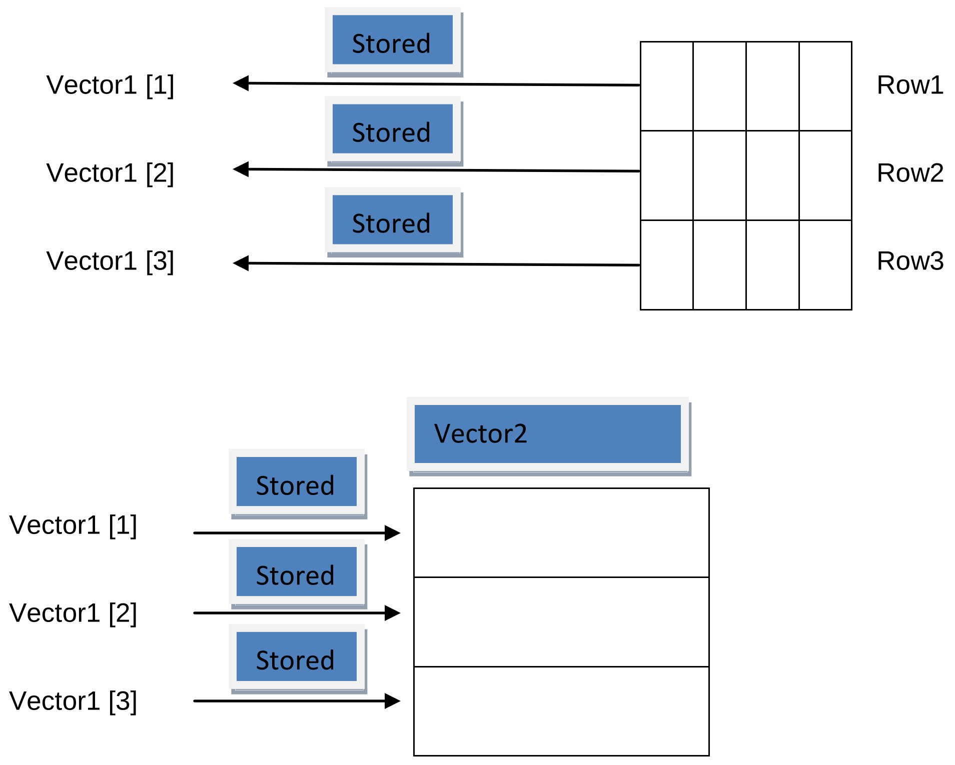


Figure 5: Usage of Vectors in Class SQL1

Both of the above mentioned actions are in an iteration statement, so that data from all the rows of the result set is read and written on the Excel spreadsheet.

2.2.2.1.5 Class ResultDisplay

Class ResultDisplay creates the Java table. A Java table is created using swing API of Java. JFrame is used as the base class. JFrame creates a frame. After creating a frame a component of table is added on it. A table in java is considered as a component, so it cannot be used directly. It has to be added onto the frame, panel or a dialog box called as containers for the display. Any of the mentioned containers could have been used to add to the table as its component. The next task was to make the table read the URL's as hyperlinks, i.e. the program must be able to intercept that a user clicked on the cell containing the URL and launch the file corresponding to that URL. All these issues were also handled by class ResultDisplay. The way it was done is as follows, get the X and Y coordinate of the cell where the click is identified; combine X and Y coordinate to get a point; identify if the point location is in the sixth column. When this is done, the URL is treated as a hyperlink.

After creating the table, the next task was to interface it with the result set such that results of the SQL query could be displayed on it. Since, Java code cannot traverse backwards after traversing through the result set, the result can only be displayed in either a Java table or Excel spreadsheet. In order to display the result set the Java code has to traverse through the database. Once the database has been traversed, the code cannot start reading the result set from the start. Java Table and Excel spreadsheet are two different objects in Java and hence, the result cannot be displayed on both of them simultaneously. However the part of the program which does the necessary math and manipulations is still kept for future use, while the feature of displaying the result on the Java table has been removed.

###### 2.2.2.1.6 Class Renderer

Previous sections explained about the hyperlinks and their formatting in detail. But, both Excel and Java have different styles of creating and reading hyperlinks. In Excel, when the URL is entered, the user needs to right click at the URL and specify the folder where the URL is intended to open. So the challenge was whether it is possible for Excel to read random URL's as hyperlinks without having to route to the folders at the time of creation of hyperlinks. This was resolved by taking advantage of a feature in Excel which permits formats other than that mentioned above to create the hyperlink as follows:

= HYPERLINK (link location, name)

The result set has hyperlinks in its sixth column, so all the data getting entered in the sixth column brought with it the format specified above using several programming steps in this class.

#### 2.2.2.2 Overview

All these classes and their inter relation with each other can be explained with the help of the UML Figure. UML stands for Unified Modeling Language. UML is a standard way of describing interrelationship between classes. Figure 6 shows the relation between all the classes that have been used in this software tool. This UML Figure was created using a software tool called StarUML. The UML Figure shown below falls into the category of class Figure, wherein a class Figure is a type of static structure Figure that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

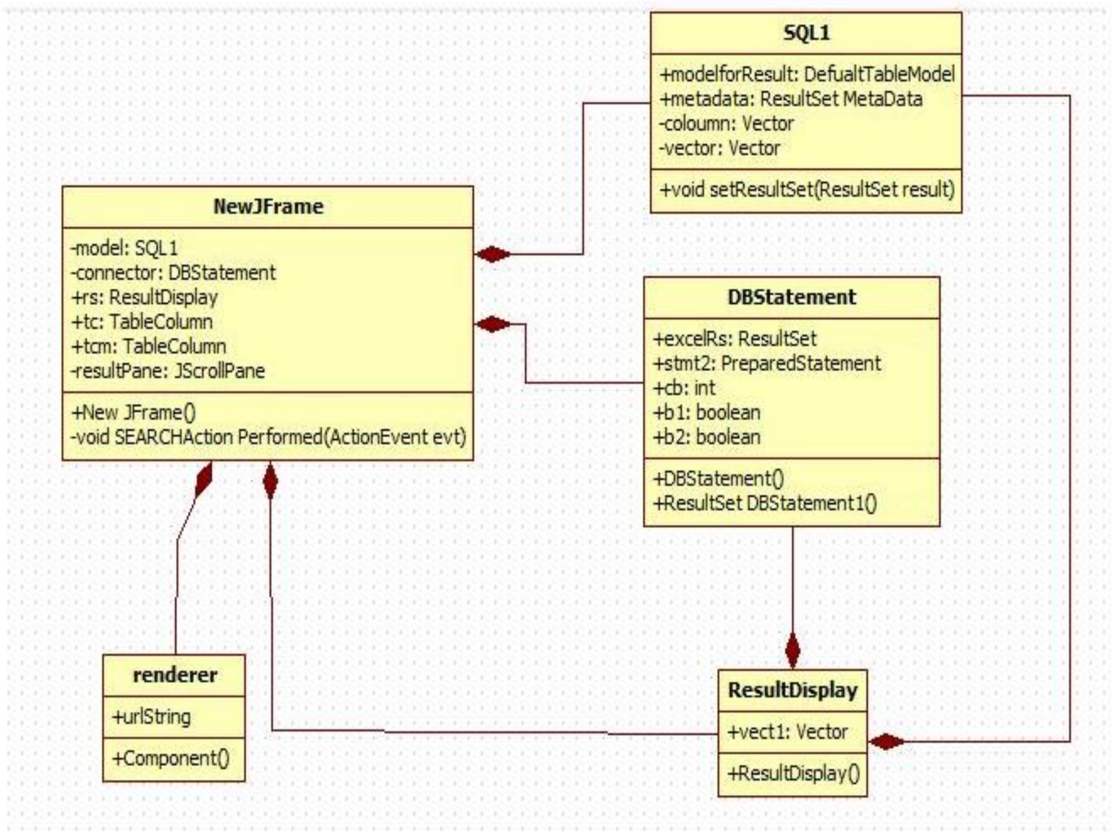


Figure 6: UML class Figure showing the structure of different classes and their relationship with each other.

Class NewJFrame consists of the following variables which are of importance: an object of class SQL1 named model; an object of class DBStatement named connector; an object of class ResultDisplay named rs; an object of class TableColumn named tc; another object of class TableColumn named tcm; and finally an object of class JScrollPane named resultPane. Class NewJFrame has several methods but the ones that are most important have been shown in the box which is named NewJFrame. Important methods of this class are NewjFrame and SEARCHAction Performed.

Class DBStatement consists of the following important variables: an object of class ResultSet named ExcelRs; an object of class PreparedStatement named stmt2; an integer array named cb; and two Boolean variables named b1 and b2 respectively. Important methods of class DBStatement are DBStatement and ResultSetDBStatement.

Class renderer has one important variable named urlString and one important method named Component.

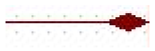
Class ResultDisplay has one important variable named vect1 and one important method named ResultDisplay.

Class SQL1 consists of the following important variables, an object of class DefaultTabelModel named modelforResult; an object of class ResultSetMetaData named metadata; and two objects of class Vector named column and vector. An important method of class SQL1 is setResultSet.

In Figure 6 only the important variables and important methods have been mentioned. For a more comprehensive reference of all the variables and all the methods please refer to the code given in Appendix 2 .

Each of the variables and methods shown in the UML Figure have a '+' or a '-' before them. A '+ before a particular variable or a method means that, that particular component is accessible by all the other classes and all the methods of the same class. i.e. they are public. A '-' before a particular variable means that, that particular component is only accessible by the methods and objects of that particular class.

The filled diamond sign as shown below means composition.



The UML graphical representation of a composition relationship is a filled diamond shape on the containing class end of the tree of lines that connect contained class(es) to the containing class. Class NewJFrame has to have an object of class ResultDisplay, class DBStatement, classSQL1 and class renderer. Class SQL1 has to have an object of class render.

# CHAPTER 3

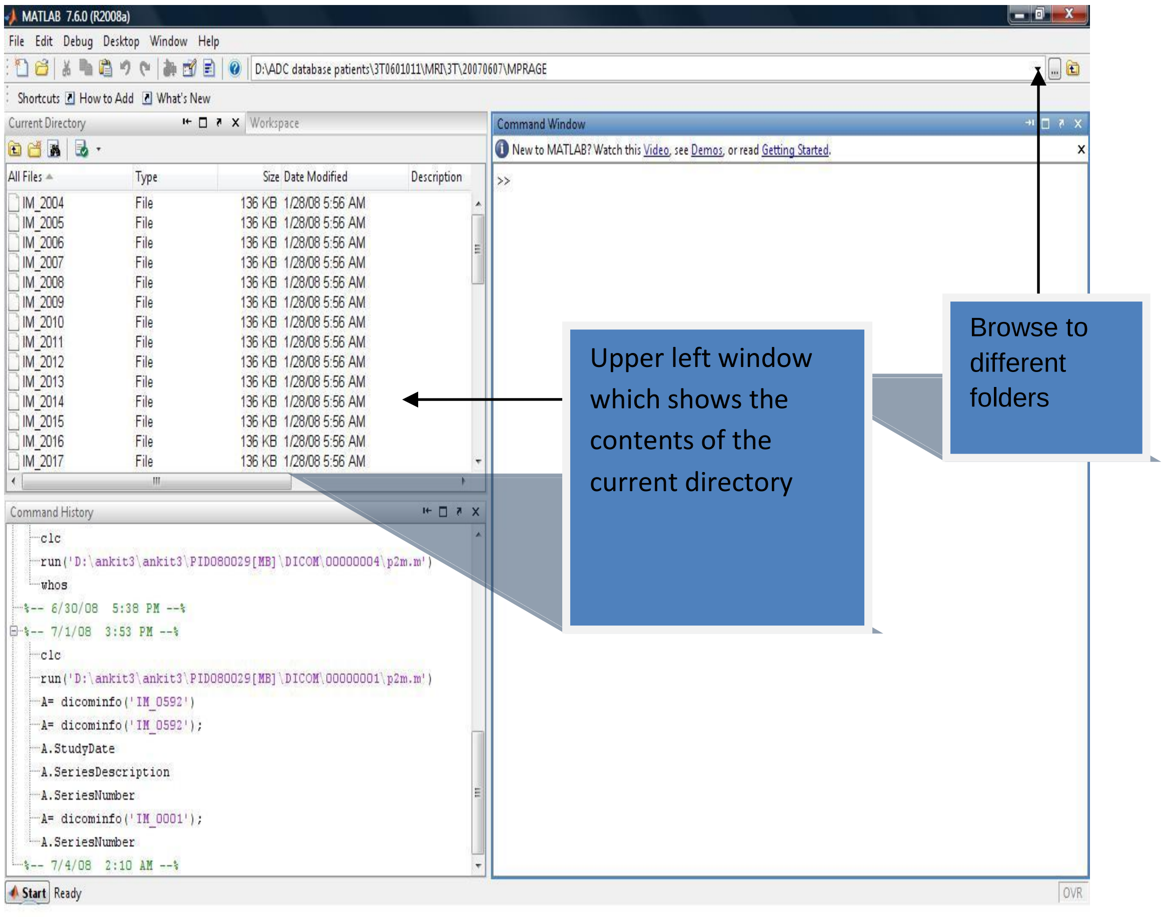
# IMPLEMENTATION AND RESULTS

This chapter deals with the implementation and result of the software tools that have been created to solve the problem of MR image classification and image retrieval, respectively.

## 3.1 Implementation and results for image classification software

The software tool for image classification was developed using Matlab. Therefore the script file is a Matlab file with an extension of “.m”. The name of the file is p2m.m. System requirements for using this tool are as follows. The tool is designed for the windows PC with a RAM of greater than or equal to 512 MB. The tool would run with less RAM, but having more RAM allows the user to select more files without hanging the computer. Matlab versions R2007a or above are required because earlier versions of Matlab (the versions previous to R2007a) do not support all the features and attributes of DICOM files. A part of this tool is created using Matlab 2007a and some more features were developed using Matlab R2008a. However, it has been observed that the tool runs with Matlab R2007a and on other newer versions. For optimum performance the latest version of Matlab (since newer features and supports for DICOM files have been updated in those versions) and RAM of above 1GB is recommended. Presently, there are two computers in the Nuclear Medicine Center that are using the beta version of this tool, both of which are windows PC. One of the two computers has Matlab R2008a and 2GB RAM and the other computer has Matlab R2007a and 1GB RAM.

Implementation of this tool is very simple. The user is required to store the script file “p2m.m” in the same folder as that of the data (DICOM images). Start Matlab and set the path of the current directory to the folder where the data and script file are stored. The folder can be browsed through the browse tab given at the top of the Matlab console. Once the path has been set, all the files, including the script file “p2m.m” which is in that particular folder can be viewed from the first window on the upper left hand side of the console.  Figure 7 shows the pictorial representation of the Matlab window.


MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New
Current Directory
Workspace
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Command History
Start Ready
OVR
Upper left window which shows the contents of the current directory
Browse to different folders

```
D:\ADC database patients\3T0601011\MRI\3T\20070607\MPRAGE
```

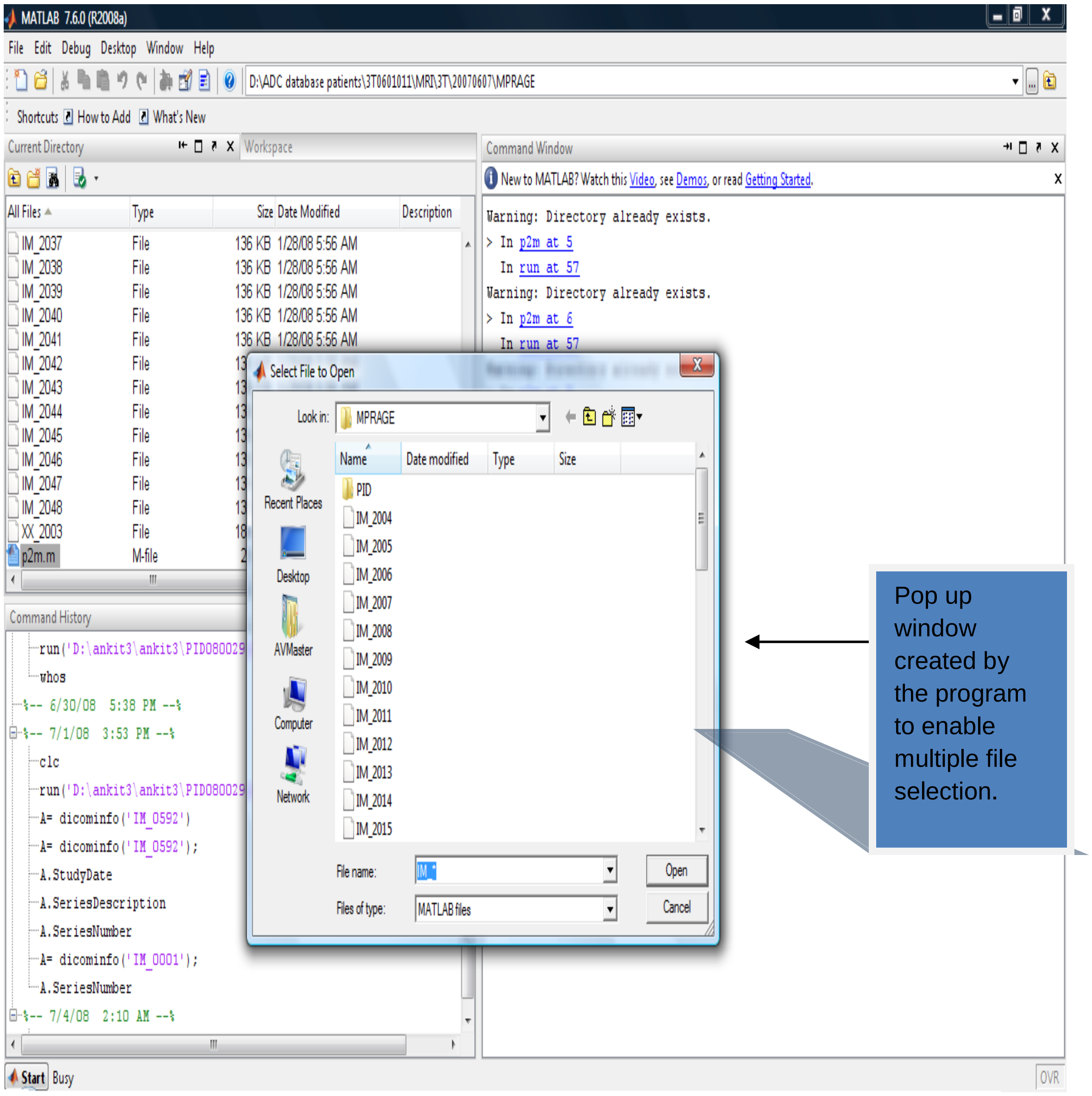
| All Files | Type | Size | Date Modified | Description |
|---|---|---|---|---|
| IM_2004 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2005 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2006 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2007 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2008 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2009 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2010 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2011 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2012 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2013 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2014 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2015 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2016 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2017 | File | 136 KB | 1/28/08 5:56 AM | |

```
>>
```

```
    clc
    run('D:\ankit3\ankit3\PID080029[MB]\DICOM\00000004\p2m.m')
    whos
  %-- 6/30/08  5:38 PM --%
  %-- 7/1/08  3:53 PM --%
    clc
    run('D:\ankit3\ankit3\PID080029[MB]\DICOM\00000001\p2m.m')
    A= dicominfo('IM_0592')
    A= dicominfo('IM_0592');
    A.StudyDate
    A.SeriesDescription
    A.SeriesNumber
    A= dicominfo('IM_0001');
    A.SeriesNumber
  %-- 7/4/08  2:10 AM --%
```

Figure 7: Typical view of the Matlab Console

The user should drag and drop the script file in the command prompt and from there the script starts its work. The script displays a pop up window prompting the user to select multiple files. Multiple files can be selected using the control button. Figure 8 shows the pictorial representation of the pop up window. As mentioned earlier, as soon as the program starts its execution, just before reading the input files the program creates the directory structure, as shown in Figure 8.


MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
D:\ADC database patients\3T0601011\MRI\3T\20070607\MPRAGE
Shortcuts How to Add What's New
Current Directory
Workspace
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Select File to Open
Look in: MPRAGE
Recent Places
Desktop
AVMaster
Computer
Network
File name: IM_*
Files of type: MATLAB files
Open
Cancel
Command History
Start Busy
OVR
Pop up window created by the program to enable multiple file selection.

| All Files | Type | Size | Date Modified | Description |
|---|---|---|---|---|
| IM_2037 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2038 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2039 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2040 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2041 | File | 136 KB | 1/28/08 5:56 AM | |
| IM_2042 | File | 13 | | |
| IM_2043 | File | 13 | | |
| IM_2044 | File | 13 | | |
| IM_2045 | File | 13 | | |
| IM_2046 | File | 13 | | |
| IM_2047 | File | 13 | | |
| IM_2048 | File | 13 | | |
| XX_2003 | File | 18 | | |
| p2m.m | M-file | 2 | | |

```
Warning: Directory already exists.
> In p2m at 5
  In run at 57
Warning: Directory already exists.
> In p2m at 6
  In run at 57
```

```
run('D:\ankit3\ankit3\PID080029
whos
%-- 6/30/08  5:38 PM --%
%-- 7/1/08  3:53 PM --%
clc
run('D:\ankit3\ankit3\PID080029
A= dicominfo('IM_0592')
A= dicominfo('IM_0592');
A.StudyDate
A.SeriesDescription
A.SeriesNumber
A= dicominfo('IM_0001');
A.SeriesNumber
%-- 7/4/08  2:10 AM --%
```

| Name | Date modified | Type | Size |
|---|---|---|---|
| PID | | | |
| IM_2004 | | | |
| IM_2005 | | | |
| IM_2006 | | | |
| IM_2007 | | | |
| IM_2008 | | | |
| IM_2009 | | | |
| IM_2010 | | | |
| IM_2011 | | | |
| IM_2012 | | | |
| IM_2013 | | | |
| IM_2014 | | | |
| IM_2015 | | | |

Figure 8: Appearance of the pop up window created by the program, once the program file is dragged and dropped in the console.

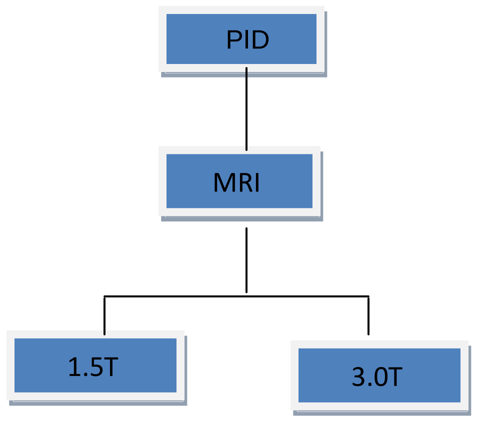
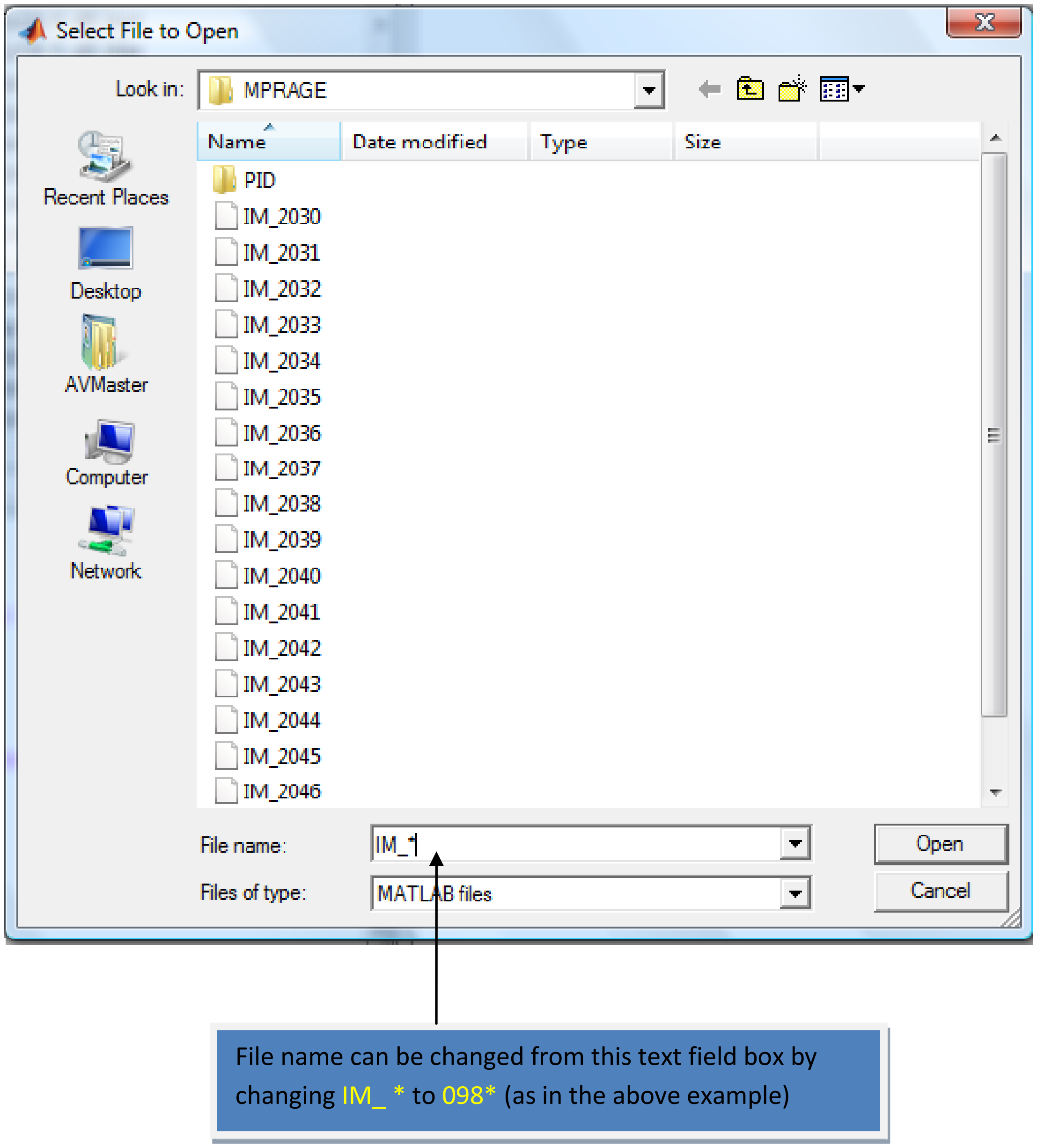


Figure 9: Directory structure created at the start of the program.

Based on the types of files given as input, the program creates subfolders corresponding to the name of the file type (scan type) within the folders named 1.5T and/or 3.0T which correspond to the magnetic field strength.

Typically file names start with IM_XXXX, where XXXX could be any 4 digit number (observed range is 0001 to 2048). However, rarely files with names that do not start with IM_XXXX are encountered. The program can still be used for file classification. Consider 500 files, each of these files have names starting from 098 followed by XXX, where XXX is any three digit number. The complete file name then turns out to be 098XXX. When the script is dragged from the current directory window into the Matlab console, a window pops up prompting the user to make the selection of the file. Please refer Figure 8 for the display for the pop up window. Using this pop up window the user can select files even if they have different file names (please refer to figure 10.)



Look in: MPRAGE

```
PID
IM_2030
IM_2031
IM_2032
IM_2033
IM_2034
IM_2035
IM_2036
IM_2037
IM_2038
IM_2039
IM_2040
IM_2041
IM_2042
IM_2043
IM_2044
IM_2045
IM_2046
```

File name: IM_*

Files of type: MATLAB files

File name can be changed from this text field box by changing IM_ * to 098* (as in the above example)

Figure 10: Dialog box for selecting files.

For other attributes like scan date, manufacturer name, etc, the Matlab console has to be used. Using a command named dicominfo, header information of a particular file can be accessed and hence, other attributes can be extracted from the header.

## 3.2 Implementation and Result for software on image retrieval

Software for image retrieval was developed using Java. Any machine running this software tool is required to have the Java development kit (jdk1.6 or any other higher version) and Java runtime environment (jre1.6 or any other higher version if it installed). This software tool is platform independent, meaning this tool can run on all platforms (Windows, Mac, Linux etc). This software tool can be executed by double clicking on the tool icon, since it is ".jar" file (equivalent to .exe file; .exe files are executable files of .net and other technologies whereas .jar files are executable files of Java technology). ".jar" stands for Java ARchive. For the application to access the database and display results, appropriate drivers for the database need to be acquired. This task is performed by ODBC data source administrator. The Microsoft ODBC data source administrator manages database drivers and data sources. For the computers running on Windows 2000 and XP this application (ODBC administrator) is located in the controls panel under administrative tools. There is an icon named data sources (ODBC). Drivers are the components that process ODBC requests and return data to the application. Data sources are the databases or files accessed by a driver and are identified by a Data Source Name (DSN). The ODBC Data Source Administrator can be used to add, configure, and delete data sources from the system. User and system data sources are collectively known as machine data sources because they are local to a computer. Figure 11 shows the pictorial representation of the ODBC window. The processes are accomplished as follows; Java makes use of the JDBC-ODBC bridge to access data through drivers written to the ODBC standards. The driver bridge is extremely useful to access data from the data sources for which no pure JDBC drivers exist. Following are the steps to use the JDBC-ODBC Bridge driver, which are performed at the time of software installation. One needs to create a DSN (Data Source Name) representing an ODBC connection to a particular database server.

1. Run Control Panel > Administrative Tools > Data Sources (ODBC).

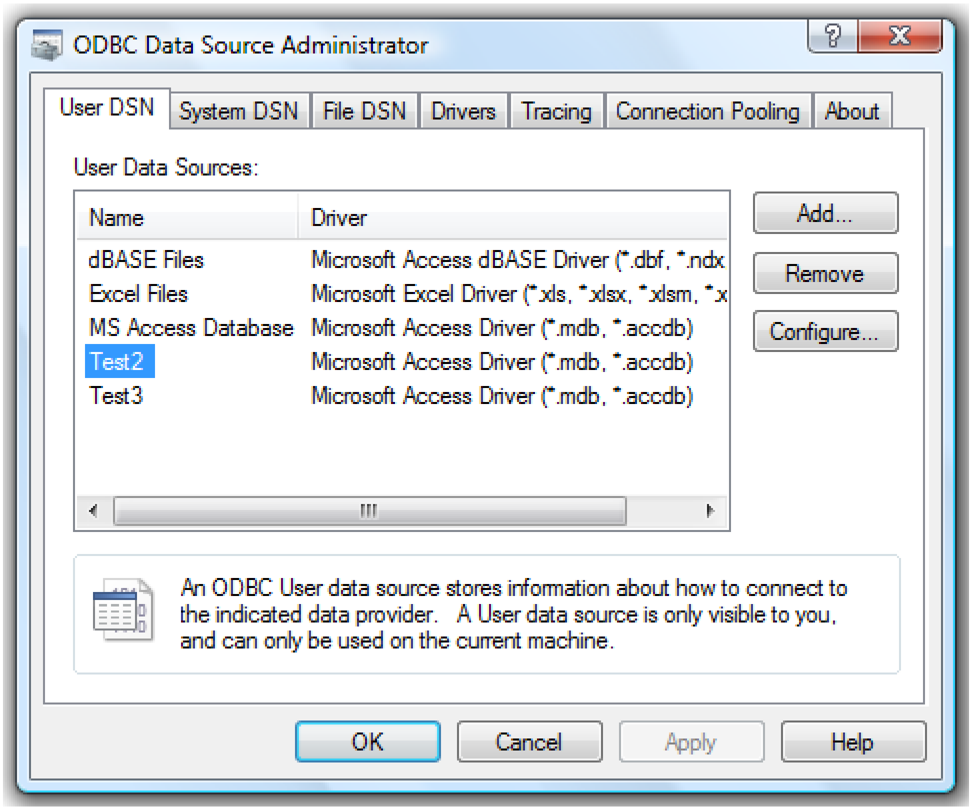
2. Click the System DSN tab. Then click the Add button.

3. From the ODBC driver list, select Microsoft Access Driver (*.mdb)

4. Now fill in the DSN header information as:

   DSN: Test2

5. Description: Test2 (Test2 is the name of the data source)6. Select the “select” button and select the table which stores the data and click ok.Test2 is now ready.



ODBC Data Source Administrator

User DSN | System DSN | File DSN | Drivers | Tracing | Connection Pooling | About

User Data Sources:

| Name | Driver |
|---|---|
| dBASE Files | Microsoft Access dBASE Driver (*.dbf, *.ndx |
| Excel Files | Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.x |
| MS Access Database | Microsoft Access Driver (*.mdb, *.accdb) |
| Test2 | Microsoft Access Driver (*.mdb, *.accdb) |
| Test3 | Microsoft Access Driver (*.mdb, *.accdb) |

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

Figure 11: ODBC Data Source Administrator (for Windows Vista)

The desktop name of the software tool is Gui-prac. When the icon named Gui-prac is double clicked an application is opened with three tabs as shown in Figure 12, 13 and 14, respectively, each of these figure displays tabs of the GUI which allows the user to select the magnetic field strength, scan types and search by keywords.

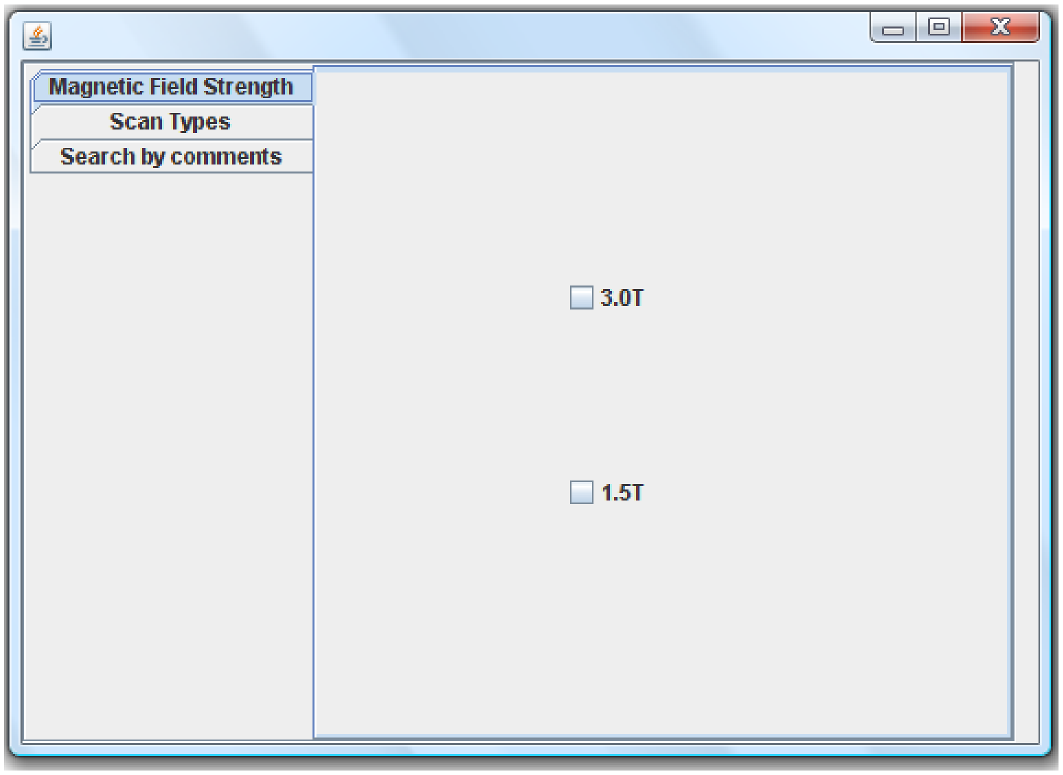
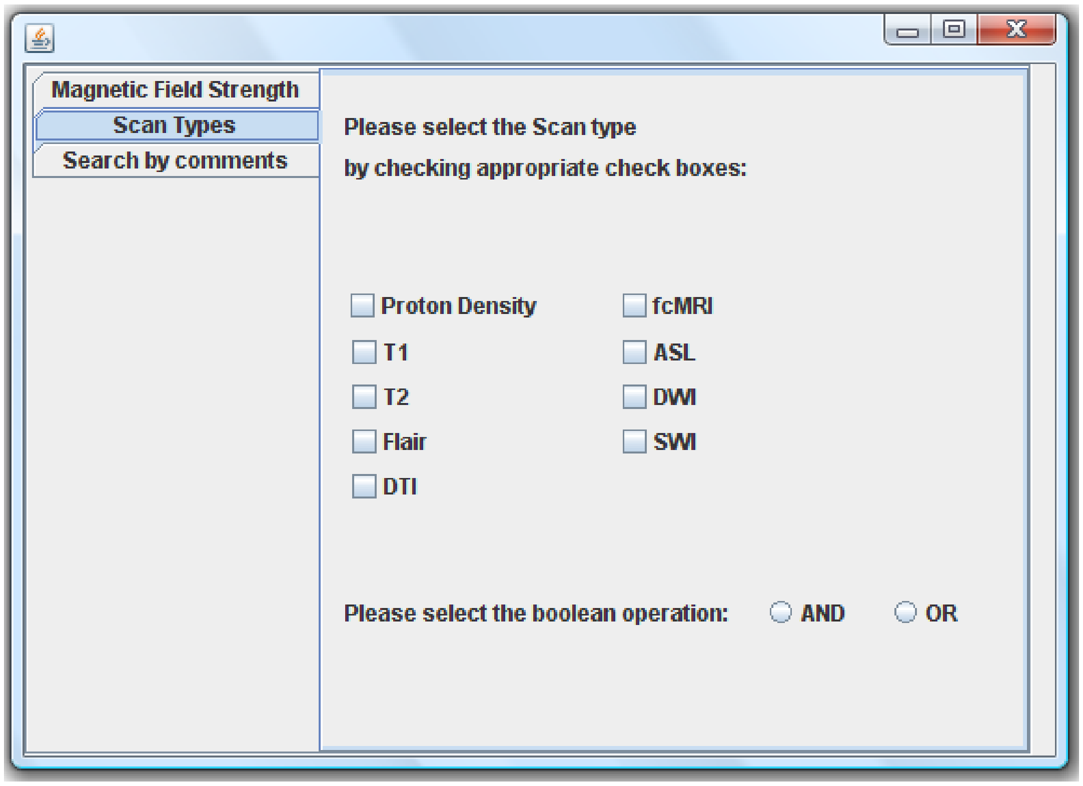


Figure 12: Tab 1 from the Gui-prac allows the user to select the magnetic field strength

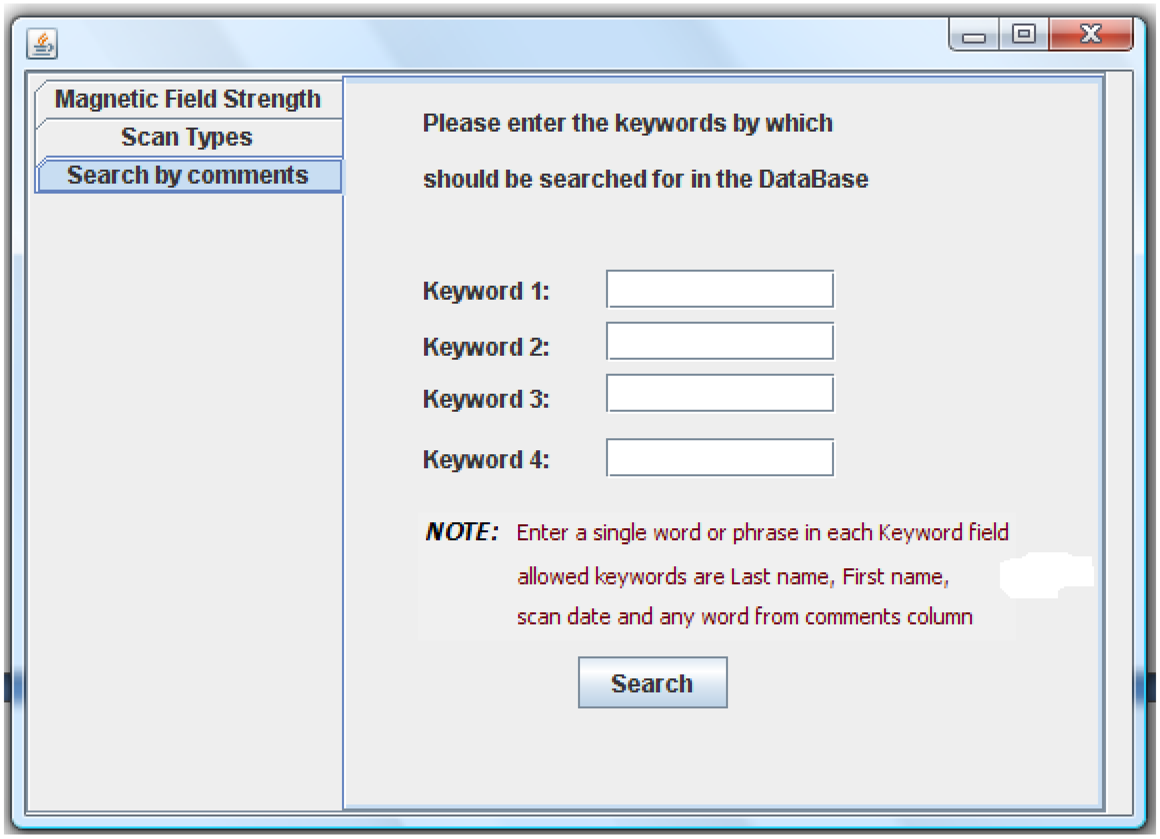


Please select the Scan type
by checking appropriate check boxes:

☐ Proton Density ☐ fcMRI
☐ T1 ☐ ASL
☐ T2 ☐ DWI
☐ Flair ☐ SWI
☐ DTI

Please select the boolean operation: ○ AND ○ OR

Figure 13: Tab 2 of Gui-prac allows the user to select the scan types and Boolean operation to be performed with those scan types.



Please enter the keywords by which

should be searched for in the DataBase

Keyword 1:

Keyword 2:

Keyword 3:

Keyword 4:

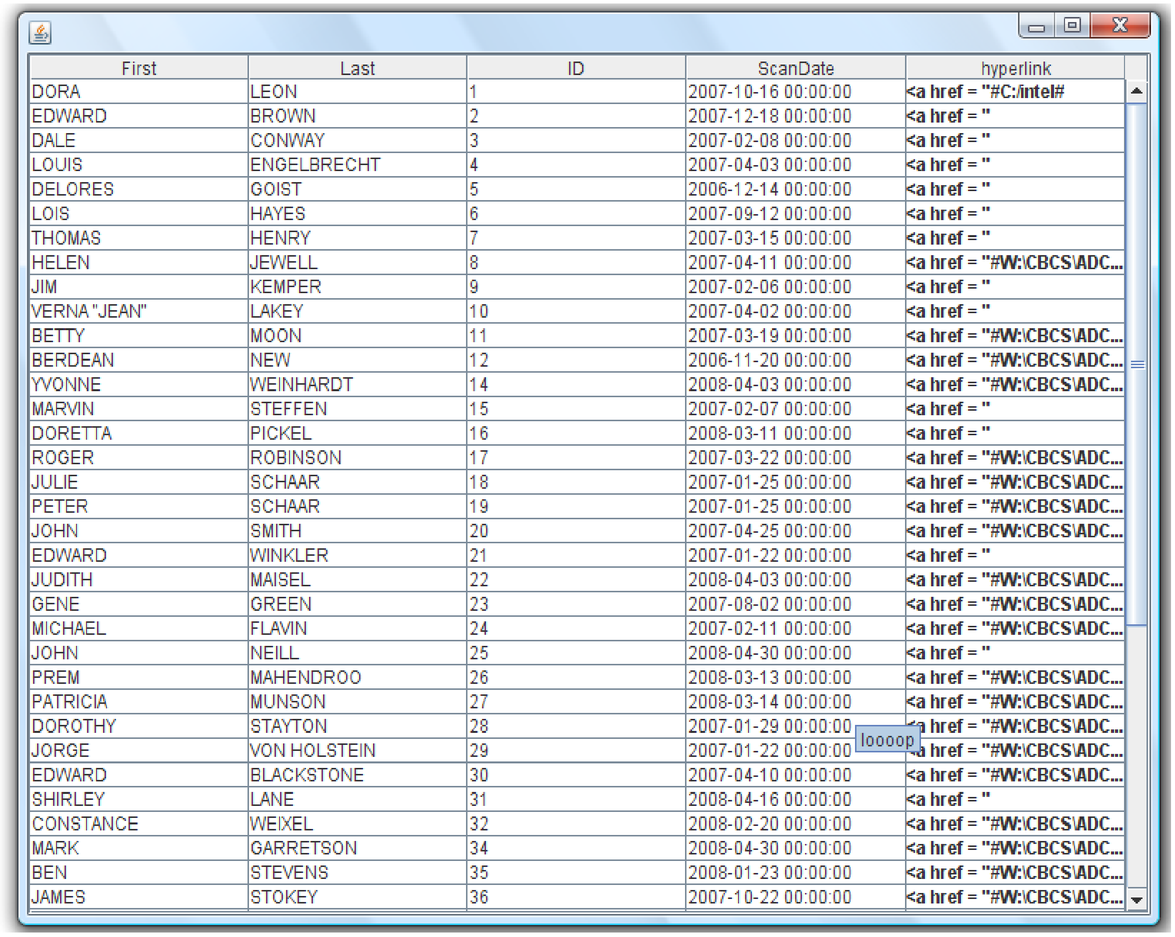
*NOTE:* Enter a single word or phrase in each Keyword field

allowed keywords are Last name, First name,

scan date and any word from comments column

Figure 14: Tab 3 allows the user to search by the keywords through the first name, last name, scan date and comments column of the database.

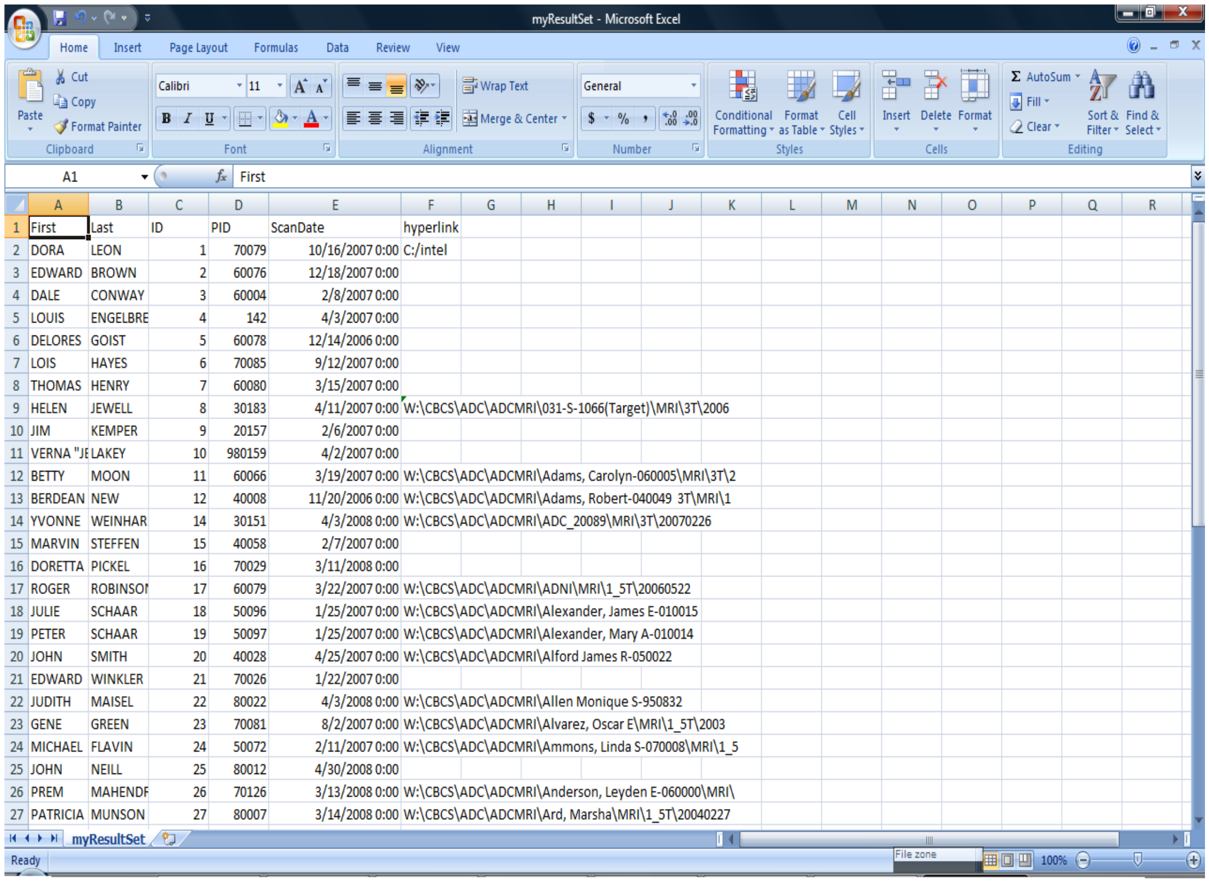
Figure 15 shows a typical Java table result. In Java terminology a Java table is called a JTable. Observe that the Java table has no features for saving the data. Due to this shortcoming, the result set is now displayed on an Excel spreadsheet, which can be used to display as well as save the data. Once the user has selected the search criteria from the three tabs, the user hits the search button provided in the third tab. The Java table is displayed after hitting the search button. The Java table has been included in this manuscript just to show the appearance of the table. Results displayed on the Java Table shown in Figure 13 are in response to the following query “How many T1 AND T2 scans do we have that were collected at 3T?” The rest of the subjects found in this search can be viewed by using the scroll that is provided at the right of the Java table.



| First | Last | ID | ScanDate | hyperlink |
|---|---|---|---|---|
| DORA | LEON | 1 | 2007-10-16 00:00:00 | <a href = "#C:/intel# |
| EDWARD | BROWN | 2 | 2007-12-18 00:00:00 | <a href = " |
| DALE | CONWAY | 3 | 2007-02-08 00:00:00 | <a href = " |
| LOUIS | ENGELBRECHT | 4 | 2007-04-03 00:00:00 | <a href = " |
| DELORES | GOIST | 5 | 2006-12-14 00:00:00 | <a href = " |
| LOIS | HAYES | 6 | 2007-09-12 00:00:00 | <a href = " |
| THOMAS | HENRY | 7 | 2007-03-15 00:00:00 | <a href = " |
| HELEN | JEWELL | 8 | 2007-04-11 00:00:00 | <a href = "#W:\CBCS\ADC... |
| JIM | KEMPER | 9 | 2007-02-06 00:00:00 | <a href = " |
| VERNA "JEAN" | LAKEY | 10 | 2007-04-02 00:00:00 | <a href = " |
| BETTY | MOON | 11 | 2007-03-19 00:00:00 | <a href = "#W:\CBCS\ADC... |
| BERDEAN | NEW | 12 | 2006-11-20 00:00:00 | <a href = "#W:\CBCS\ADC... |
| YVONNE | WEINHARDT | 14 | 2008-04-03 00:00:00 | <a href = "#W:\CBCS\ADC... |
| MARVIN | STEFFEN | 15 | 2007-02-07 00:00:00 | <a href = " |
| DORETTA | PICKEL | 16 | 2008-03-11 00:00:00 | <a href = " |
| ROGER | ROBINSON | 17 | 2007-03-22 00:00:00 | <a href = "#W:\CBCS\ADC... |
| JULIE | SCHAAR | 18 | 2007-01-25 00:00:00 | <a href = "#W:\CBCS\ADC... |
| PETER | SCHAAR | 19 | 2007-01-25 00:00:00 | <a href = "#W:\CBCS\ADC... |
| JOHN | SMITH | 20 | 2007-04-25 00:00:00 | <a href = "#W:\CBCS\ADC... |
| EDWARD | WINKLER | 21 | 2007-01-22 00:00:00 | <a href = " |
| JUDITH | MAISEL | 22 | 2008-04-03 00:00:00 | <a href = "#W:\CBCS\ADC... |
| GENE | GREEN | 23 | 2007-08-02 00:00:00 | <a href = "#W:\CBCS\ADC... |
| MICHAEL | FLAVIN | 24 | 2007-02-11 00:00:00 | <a href = "#W:\CBCS\ADC... |
| JOHN | NEILL | 25 | 2008-04-30 00:00:00 | <a href = " |
| PREM | MAHENDROO | 26 | 2008-03-13 00:00:00 | <a href = "#W:\CBCS\ADC... |
| PATRICIA | MUNSON | 27 | 2008-03-14 00:00:00 | <a href = "#W:\CBCS\ADC... |
| DOROTHY | STAYTON | 28 | 2007-01-29 00:00:00 | a href = "#W:\CBCS\ADC... |
| JORGE | VON HOLSTEIN | 29 | 2007-01-22 00:00:00 | a href = "#W:\CBCS\ADC... |
| EDWARD | BLACKSTONE | 30 | 2007-04-10 00:00:00 | <a href = "#W:\CBCS\ADC... |
| SHIRLEY | LANE | 31 | 2008-04-16 00:00:00 | <a href = " |
| CONSTANCE | WEIXEL | 32 | 2008-02-20 00:00:00 | <a href = "#W:\CBCS\ADC... |
| MARK | GARRETSON | 34 | 2008-04-30 00:00:00 | <a href = "#W:\CBCS\ADC... |
| BEN | STEVENS | 35 | 2008-01-23 00:00:00 | <a href = "#W:\CBCS\ADC... |
| JAMES | STOKEY | 36 | 2007-10-22 00:00:00 | <a href = "#W:\CBCS\ADC... |

Figure 15: JTable

Columns first and last refer to the first and last names of the subjects, respectively. Column ID is included for the purpose of checking if all the data has been acquired and is used only for test purposes. The column scan date provides the date at which the images were scanned and finally the column of hyperlink provides the hyperlink that allows the user to directly access the actual image data. Observe the unusual format in which the hyperlinks are displayed. This unusual hyperlink format seen in Figure 13 is the way in which Java interprets hyperlinks. Figure 16 shows the Excel spreadsheet. Results displayed on the Excel spreadsheet are answers to the following query "how many T1 AND T2 scans do we have on file that were collected at 3T?" Where the columns named first and last are the first and last names of the subjects, respectively. Column named ID is a test column. Column named ScanDate represents the date of scan and the column named hyperlink routes the user to the actual images.



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | First | Last | ID | PID | ScanDate | hyperlink |
| 2 | DORA | LEON | 1 | 70079 | 10/16/2007 0:00 | C:/intel |
| 3 | EDWARD | BROWN | 2 | 60076 | 12/18/2007 0:00 | |
| 4 | DALE | CONWAY | 3 | 60004 | 2/8/2007 0:00 | |
| 5 | LOUIS | ENGELBRE | 4 | 142 | 4/3/2007 0:00 | |
| 6 | DELORES | GOIST | 5 | 60078 | 12/14/2006 0:00 | |
| 7 | LOIS | HAYES | 6 | 70085 | 9/12/2007 0:00 | |
| 8 | THOMAS | HENRY | 7 | 60080 | 3/15/2007 0:00 | |
| 9 | HELEN | JEWELL | 8 | 30183 | 4/11/2007 0:00 | W:\CBCS\ADC\ADCMRI\031-S-1066(Target)\MRI\3T\2006 |
| 10 | JIM | KEMPER | 9 | 20157 | 2/6/2007 0:00 | |
| 11 | VERNA "JE | LAKEY | 10 | 980159 | 4/2/2007 0:00 | |
| 12 | BETTY | MOON | 11 | 60066 | 3/19/2007 0:00 | W:\CBCS\ADC\ADCMRI\Adams, Carolyn-060005\MRI\3T\2 |
| 13 | BERDEAN | NEW | 12 | 40008 | 11/20/2006 0:00 | W:\CBCS\ADC\ADCMRI\Adams, Robert-040049 3T\MRI\1 |
| 14 | YVONNE | WEINHAR | 14 | 30151 | 4/3/2008 0:00 | W:\CBCS\ADC\ADCMRI\ADC_20089\MRI\3T\20070226 |
| 15 | MARVIN | STEFFEN | 15 | 40058 | 2/7/2007 0:00 | |
| 16 | DORETTA | PICKEL | 16 | 70029 | 3/11/2008 0:00 | |
| 17 | ROGER | ROBINSOI | 17 | 60079 | 3/22/2007 0:00 | W:\CBCS\ADC\ADCMRI\ADNI\MRI\1_5T\20060522 |
| 18 | JULIE | SCHAAR | 18 | 50096 | 1/25/2007 0:00 | W:\CBCS\ADC\ADCMRI\Alexander, James E-010015 |
| 19 | PETER | SCHAAR | 19 | 50097 | 1/25/2007 0:00 | W:\CBCS\ADC\ADCMRI\Alexander, Mary A-010014 |
| 20 | JOHN | SMITH | 20 | 40028 | 4/25/2007 0:00 | W:\CBCS\ADC\ADCMRI\Alford James R-050022 |
| 21 | EDWARD | WINKLER | 21 | 70026 | 1/22/2007 0:00 | |
| 22 | JUDITH | MAISEL | 22 | 80022 | 4/3/2008 0:00 | W:\CBCS\ADC\ADCMRI\Allen Monique S-950832 |
| 23 | GENE | GREEN | 23 | 70081 | 8/2/2007 0:00 | W:\CBCS\ADC\ADCMRI\Alvarez, Oscar E\MRI\1_5T\2003 |
| 24 | MICHAEL | FLAVIN | 24 | 50072 | 2/11/2007 0:00 | W:\CBCS\ADC\ADCMRI\Ammons, Linda S-070008\MRI\1_5 |
| 25 | JOHN | NEILL | 25 | 80012 | 4/30/2008 0:00 | |
| 26 | PREM | MAHENDF | 26 | 70126 | 3/13/2008 0:00 | W:\CBCS\ADC\ADCMRI\Anderson, Leyden E-060000\MRI\ |
| 27 | PATRICIA | MUNSON | 27 | 80007 | 3/14/2008 0:00 | W:\CBCS\ADC\ADCMRI\Ard, Marsha\MRI\1_5T\20040227 |

Figure 16: Results displayed on an Excel spreadsheet.

Also note that the formatting of the hyperlinks is not the same as it was in Java Table. This is again due to the difference in the way Excel interprets hyperlinks from the way Java interprets hyperlinks. In order for Excel to read a particular URL as a hyperlink, hyperlink standards of Excel have been followed. All Excel spreadsheets are saved in the folder where the Java code has been installed and are named "myResultSet.xls."

Hence the user must rename the file before every new search.  Failing to do so, the Excel file is overwritten and the previous result set is lost.

### *3.2.1 Test Method*

The query results that were obtained using the image retrieving tool were verified using the “SQL design view” of Microsoft Access and by random manual checks (meaning some of the names were randomly selected from the result set and were checked for in the in the database, bearing the search criteria in mind). The converse procedure was also followed, after intersecting all the search criteria patients who met those criteria were found in the database and were randomly checked with the result set that was obtained using the image retrieval tool for those search conditions.

### *3.2.2 Test Criteria*

The set of queries executed satisfied the following coverage requirements. For each “question mark {?}” in the queries (AND condition and OR condition) there was at least one query which substituted a zero in place of that {?} and a one in place of that {?}. This set of queries was executed and the result of each query was compared with that obtained by running the same query in MS access SQL design.  Thus each Boolean criterion in the query was tested for both its possible values of 0 and 1 several times. The result for each query was found to be correct.

# CHAPTER 4

# DISCUSSION

The image classification tool was tested rigorously to check the maximum number of files that can be handled by the computers available without creating errors. Based on the experiments that were performed it was observed that a computer having a RAM of 512 MB could comfortably read 400-500 files, a computer having a RAM of 1GB could read 600-700 files (since other factors like clock speed also led to variations) and a computer having a RAM of 2GB could read over 900 files. The time taken for the file classification depends on the clock speed and the processor of the computer. But roughly 500 files can be classified in 3 to 8 minutes on any computer with a RAM greater than or equal to 512MB and having a clock speed of 1.4 GHz or more.

Almost all the classified files on the drive (the drive that stores the DICOM images) have been classified using this tool. It has also been observed that Matlab versions lower than R2007a can still execute the results as desired but they give errors for some of the attributes (i.e. they fail to identify that particular attribute). In order to classify the files in accordance to their scan type an attribute called Series Description has to be read from the header. Series description may consists of many other characters in addition to the #, %, ‘ ‘ ,etc. The program written in Matlab can ignore these non required characters and route the images to their respective folder. Available neuroimaging tools like Dicom works, MRICro, Philips Dicom viewer, etc can also read the headers of the files, reading one file at a given time,  However they cannot automatically read the attributes. In this process, the user has to open the header of every single file using either Dicom works, MRICro, Philips dicom viewer, etc, manually read the attributes of interest and classify the files accordingly. It was observed that it takes approximately 50 seconds to open up a file and read the required header attributes from them. There are approximately fifteen thousands files under each subject folder, which need to be classified. If every single file were to be read manually and then classified, it would take approximately 15000X50 seconds (more than 200 hours!) for its classification. The software tool for image classification saves both, the time and the effort of the user by automatically classifying the images into the required directory structure. Approximate time for classifying all the files (over 15000 files) of a particular patient, on a computer with a RAM of 1GB or above is 45 mins. This approximate time is based on extensive trials that have been made on the data of hundreds of subjects, and satisfy the intentions of developing the image classification tool. Hence, a system was developed, that could read multiple files, access their header information without manual effort, traverse through the header and read multiple attributes and move the files according to their attributes into their respective folders. This tool can be used by any institution for classifying their MRI image data, which follow DICOM standards.

The image retrieval tool interactively searches for images of particular characteristics as needed. The image retrieval tool is capable of executing both simple and complex queries. The tool searches for its results through the following columns, first, last, scan date, magnetic field strength, Proton density, T1, T2, FLAIR, DTI, FcMRI, ASL, DWI, SWI and Comments in the database without having to write the queries in the database using SQL. Typically it takes around 20 minutes to manually search for the subjects of a single scan type (e.g. FLAIR) through the database or a spreadsheet. The complexity of the problem increases as the number of variables increase, (e.g., Search for all subjects having T1 AND Flair scans on a 3T magnet or Search for all subjects having T1 OR DTI scans on a 1.5T magnet). With the increase in the complexity of the problem, the time and hence the effort of the user increases for manually searching for the required subjects from the database. An alternative approach is to query the database using SQL to search for records satisfying specific criteria. The second approach is most feasible, since it saves both the time and the effort of the user. Also, the second approach is less laborious. But SQL is not known to all. Neuroimaging investigators, who do not know the usage of SQL have to either search through the database manually or contact the database administrators with their specific requirements. The image retrieval tool solves this problem by incorporating predefined parameterized SQL statements, which assume a suitable form at the run time.

For example, if the user is trying to answer a question like “How many subjects do we have with scan types T1 and T2 and DTI having magnetic field strength of 1.5T?”,

The SQL query changes if the search criteria changes, for example if the question were “how many FLAIR AND DTI scans do we have on files that were collected at 1.5T OR 3.0T?”

Writing an SQL query is infeasible for someone who doesn’t have experience of using SQL.

The image retrieval tool provides the user with a simple and convenient interface to the database and hence the user is not required to write a new query for a new question. Working of the query has been explained in section 2.2.1-b(page 16-17). Users, can easily search for their specific requirements through the database (without having to know the SQL) and can also be routed to the actual image data. Limitations of the search are discussed in the following chapter. The user doesn’t have to write a new query for a new question and doesn’t have to get into the math of writing the query(consider a situation when the user forgets to put the parenthesis, when performing several Boolean operations). Hence, the objective of creating an image retrieval tool has been satisfied.

# CHAPTER 5

# LIMITATIONS AND FUTURE WORK

## 5.1 Limitations of the software on image classification

The tool developed has no provisions for extracting series number from the header of the files. Series number is an important variable for determining the continuation of the slices each of which is stored in a file. The current folder format stores two thousand forty eight files (for the Philips 3.0T scanner, infact the problem arises only with a Philip 3.0T scanner) in each folder and therefore if there are more than two thousand forty eight files they are stored in a new subfolder with the same indexing as in the previous folder meaning, files are stored with the following numbering system IM_0001 to IM_2048. Now if there are more than two thousand forty eight files a new folder will be created with the first file having the number IM_0001 instead of IM_2049. So there is potentially no way of knowing if the slices stored in folder 2 are the slices which are in continuation of the first folder or whether they belong to a new set of scan. This problem can be addressed with the help of an attribute called series number in the header of DICOM files. Presently the problem can be solved by having the user type in two commands in the Matlab console, but the code itself cannot make this decision. Also, to retrieve the scan date the user has to type in two commands in the Matlab console

Future work for the image classification tool; the code can be modified to classify the images based on the series number and the series description. Automatic extraction of scan date can also be implemented; A GUI can be created to read the files.

## 5.2 Limitations of the software tool on image retrieval

The two tools are not interfaced. Since, the two tools are not interfaced, after classification of the files; user is required to update the database manually. This can be a major issue if some other user is relying on the database for recent images; because the new images cannot be retrieved until the database has been updated. Also, the user cannot execute queries which contain combinations of AND and OR operations for the images; for example is the user has a question as follows, “how many T1 AND (T2 OR FLAIR) scans do we have on file that were collected at 3.0T?”.

Future work for the software tool for Image retrieval; the present tool is a desktop based application, but it can be extended to a web based application. A new tool can be written for searching through the directories as opposed to the database using J2EE (Enterprise version of Java). As mentioned above, the two tools are not interfaced. Hence, after data classification, all the information related to a particular subject has to be manually entered into the database. If there could be a software tool, which could directly read the directory structure, dependence on the database can be avoided. Also, if that is infeasible, a software tool which could update the database every time a new file is stored on the drive would be helpful.

# APPENDIX A

# MATLAB CODE

```
clc
clear

f1=fullfile('C:','Documents and Settings','amaste','Desktop', 'DICOM');
%%%%%set the path of the desktop
mkdir('PID');
mkdir('PID/MRI');
mkdir('PID/MRI/1.5T');
mkdir('PID/MRI/3T');


[filename, pathname]=uigetfile('IM_*','multiselect','on')
A=filename;
B=pathname;
[m,n]=size(A);

a1=0;
a2=0;
a3=0;
a4=0;
a5=0;
a6=0;
a7=0;
a8=0;
a9=0;
a10=0;
a11=0;
a12=0;
a13=0;
a14=0;
a15=0;
a16=0;
a17=0;
a18=0;
a19=0;
a20=0;
a21=0;
a22=0;
a23=0;
a24=0;%%%%%%%%%%%fMRI1
a25=0;
a26=0;%%%%%%fmri2
a27=0;
a28=0;%%%%%%%%%%%%% asl
a29=0;
a30=0;
a31=0;
a32=0;
a33=0;
a34=0;
a35=0;
a36=0;
```

```
a37=0;
a38=0;

save=0;


for i=1:n
    T=dicominfo(A{1,i});
    S(i)=struct('Ti',T);
end

for i=1:n
    Fn{i}=getfield(S(1,i).Ti,{1,1}, 'Filename');
    Pat{i}=getfield(S(1,i).Ti,{1,1}, 'PatientID');
    Mod{i}=getfield(S(1,i).Ti,{1,1}, 'Modality');

    %%%%%%%%%%%%%%%%%%%%%%%%

   tf=isfield(S(1,i).Ti,'MagneticFieldStrength');
    if tf==1
    MagFS{i}=getfield(S(1,i).Ti,{1,1},'MagneticFieldStrength');
    else MagFS{i}=char('data not available');
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%
    SeriesDcpn{i}=getfield(S(1,i).Ti,{1,1},'SeriesDescription');
    studydate{i}=getfield(S(1,i).Ti,{1,1},'StudyDate');
    Manufac{i}=getfield(S(1,i).Ti,{1,1},'Manufacturer');
    instname{i}=getfield(S(1,i).Ti,{1,1},'InstitutionName');
    studydes{i}=getfield(S(1,i).Ti,{1,1},'StudyDescription');

main(i)=struct('File',Fn{i},'PID',Pat{i},'mod',Mod{i},'MFS',MagFS{i},'S
D',SeriesDcpn{i},'studydate',studydate{i});

%mainy(i)=struct('Manufact',Manufac{i},'InstName',instname{i},'Studydes
pcription',studydes{i});
     var{i}=struct2cell(main(i));
    % vary{i}=struct2cell(mainy(i));


end


%%%%%%%%%%%%%%%%%%%%%%%% the code not to be messed with

%%%%%%%%%%%%%%%%%% manipulated code%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


for i=1:n
```

```
 %%%%%%%%%%%%

flair=findstr('flair', SeriesDcpn{i});
   if (flair>0)
       if (MagFS{i}==1.5)
           if (a7>0)

               movefile(A{1,i},'PID/MRI/1.5T/FLAIR');
               save=save+1;
           else mkdir('PID/MRI/1.5T/FLAIR');

               movefile(A{1,i},'PID/MRI/1.5T/FLAIR');
               a7=a7+1;
               save=save+1;
           end
       end
   end

   if (save==1)
       save=0;
       continue
   end



   if (flair>0)
       if (MagFS{i}==3)
           if (a8>0)
               movefile(A{1,i},'PID/MRI/3T/FLAIR');
               save=save+1;

           else mkdir('PID/MRI/3T/FLAIR');
               movefile(A{1,i},'PID/MRI/3T/FLAIR');
               a8=a8+1;
               save=save+1;
           end
       end
   end


   if (save==1)
       save=0;
       continue
   end

    FLAIR=findstr('FLAIR',SeriesDcpn{i});
    if (FLAIR>0)
       if (MagFS{i}==1.5)
           if (a7>0)
               movefile(A{1,i},'PID/MRI/1.5T/FLAIR');
               save=save+1;
           else mkdir('PID/MRI/1.5T/FLAIR');
               movefile(A{1,i},'PID/MRI/1.5T/FLAIR');
```

```
            a7=a7+1;
            save=save+1;
        end
    end
end
if (save==1)
    save=0;
    continue
end

if (FLAIR>0)
    if (MagFS{i}==3)
        if (a8>0)
            movefile(A{1,i},'PID/MRI/3T/FLAIR');
            save=save+1;
        else mkdir('PID/MRI/3T/FLAIR');
            movefile(A{1,i},'PID/MRI/3T/FLAIR');
            a8=a8+1;
            save=save+1;
        end
    end
end

if (save==1)
    save=0;
    continue
end

Flair=findstr('Flair',SeriesDcpn{i});
 if (Flair>0)
    if (MagFS{i}==1.5)
        if (a7>0)
            movefile(A{1,i},'PID/MRI/1.5T/FLAIR');
            save=save+1;
        else mkdir('PID/MRI/1.5T/FLAIR');
            movefile(A{1,i},'PID/MRI/1.5T/FLAIR');
            a7=a7+1;
            save=save+1;
        end
    end
 end

 if (save==1)
    save=0;
    continue
end

if (Flair>0)
    if (MagFS{i}==3)
        if (a8>0)
            movefile(A{1,i},'PID/MRI/3T/FLAIR');
            save=save+1;
        else mkdir('PID/MRI/3T/FLAIR');
            movefile(A{1,i},'PID/MRI/3T/FLAIR');
```

```
                a8=a8+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FLAIR ends
DW
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%starts
 DW=findstr('DW',SeriesDcpn{i});


    if (DW>0)
        if (MagFS{i}==1.5)
            if (a5>0)
                movefile(A{1,i},'PID/MRI/1.5T/DWI');
                save=save+1;
            else mkdir('PID/MRI/1.5T/DWI');
                movefile(A{1,i},'PID/MRI/1.5T/DWI');
                a5=a5+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
     end

     if (DW>0)
        if (MagFS{i}==15000)
            if (a5>0)
                movefile(A{1,i},'PID/MRI/1.5T/DWI');
                save=save+1;
            else mkdir('PID/MRI/1.5T/DWI');
                movefile(A{1,i},'PID/MRI/1.5T/DWI');
                a5=a5+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end



    if (DW>0)
        if (MagFS{i}==3)
            if (a6>0)
                movefile(A{1,i},'PID/MRI/3T/DWI');
                save=save+1;
            else mkdir('PID/MRI/3T/DWI');
                movefile(A{1,i},'PID/MRI/3T/DWI');
                a6=a6+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end

dw=findstr('dw',SeriesDcpn{i});


    if (dw>0)
        if (MagFS{i}==1.5)
            if (a5>0)
                movefile(A{1,i},'PID/MRI/1.5T/DWI');
                save=save+1;
            else mkdir('PID/MRI/1.5T/DWI');
                movefile(A{1,i},'PID/MRI/1.5T/DWI');
                a5=a5+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (dw>0)
        if (MagFS{i}==3)
            if (a6>0)
                movefile(A{1,i},'PID/MRI/3T/DWI');
                save=save+1;
            else mkdir('PID/MRI/3T/DWI');
                movefile(A{1,i},'PID/MRI/3T/DWI');
                a6=a6+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%DW
ends DT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
starts


DT=findstr('DT',SeriesDcpn{i});


    if (DT>0)
        if (MagFS{i}==1.5)
            if (a9>0)
                movefile(A{1,i},'PID/MRI/1.5T/DTI');
                save=save+1;
            else mkdir('PID/MRI/1.5T/DTI');
                movefile(A{1,i},'PID/MRI/1.5T/DTI');
                a9=a9+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
     end

     if (DT>0)
        if (MagFS{i}==15000)
            if (a9>0)
                movefile(A{1,i},'PID/MRI/1.5T/DTI');
                save=save+1;
            else mkdir('PID/MRI/1.5T/DTI');
                movefile(A{1,i},'PID/MRI/1.5T/DTI');
                a9=a9+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end



    if (DT>0)
        if (MagFS{i}==3)
            if (a10>0)
                movefile(A{1,i},'PID/MRI/3T/DTI');
                save=save+1;
            else mkdir('PID/MRI/3T/DTI');
                movefile(A{1,i},'PID/MRI/3T/DTI');
                a10=a10+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end

dt=findstr('dt',SeriesDcpn{i});


    if (dt>0)
        if (MagFS{i}==1.5)
            if (a9>0)
                movefile(A{1,i},'PID/MRI/1.5T/DTI');
                save=save+1;
            else mkdir('PID/MRI/1.5T/DTI');
                movefile(A{1,i},'PID/MRI/1.5T/DTI');
                a9=a9+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (dt>0)
        if (MagFS{i}==3)
            if (a10>0)
                movefile(A{1,i},'PID/MRI/3T/DTI');
                save=save+1;
            else mkdir('PID/MRI/3T/DTI');
                movefile(A{1,i},'PID/MRI/3T/DTI');
                a10=a10+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%DTI ends
T1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%starts
    T1=findstr('T1',SeriesDcpn{i});


    if (T1>0)
        if (MagFS{i}==1.5)
            if (a1>0)
                movefile(A{1,i},'PID/MRI/1.5T/T1');
                save=save+1;
            else mkdir('PID/MRI/1.5T/T1');
                movefile(A{1,i},'PID/MRI/1.5T/T1');
                a1=a1+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (T1>0)
        if (MagFS{i}==3)
            if (a2>0)
                movefile(A{1,i},'PID/MRI/3T/T1');
                save=save+1;
            else mkdir('PID/MRI/3T/T1');
                movefile(A{1,i},'PID/MRI/3T/T1');
                a2=a2+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end
```

```
t1=findstr('t1',SeriesDcpn{i});


    if (t1>0)
        if (MagFS{i}==1.5)
            if (a1>0)
                movefile(A{1,i},'PID/MRI/1.5T/T1');
                save=save+1;
            else mkdir('PID/MRI/1.5T/T1');
                movefile(A{1,i},'PID/MRI/1.5T/T1');
                a1=a1+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (t1>0)
        if (MagFS{i}==3)
            if (a2>0)
                movefile(A{1,i},'PID/MRI/3T/T1');
                save=save+1;
            else mkdir('PID/MRI/3T/T1');
                movefile(A{1,i},'PID/MRI/3T/T1');
                a2=a2+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%T1
ends T2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%star
ts

 T2=findstr('T2',SeriesDcpn{i});


    if (T2>0)
        if (MagFS{i}==1.5)
            if (a3>0)
                movefile(A{1,i},'PID/MRI/1.5T/T2');
                save=save+1;
            else mkdir('PID/MRI/1.5T/T2');
                movefile(A{1,i},'PID/MRI/1.5T/T2');
                a3=a3+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (T2>0)
        if (MagFS{i}==3)
            if (a4>0)
                movefile(A{1,i},'PID/MRI/3T/T2');
                save=save+1;
            else mkdir('PID/MRI/3T/T2');
                movefile(A{1,i},'PID/MRI/3T/T2');
                a4=a4+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end

t2=findstr('t2',SeriesDcpn{i});


    if (t2>0)
        if (MagFS{i}==1.5)
            if (a3>0)
                movefile(A{1,i},'PID/MRI/1.5T/T2');
                save=save+1;
            else mkdir('PID/MRI/1.5T/T2');
                movefile(A{1,i},'PID/MRI/1.5T/T2');
                a3=a3+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end
    if (t2>0)
        if (MagFS{i}==3)
            if (a4>0)
                movefile(A{1,i},'PID/MRI/3T/T2');
                save=save+1;
            else mkdir('PID/MRI/3T/T2');
                movefile(A{1,i},'PID/MRI/3T/T2');
                a4=a4+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%T2 ends
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%MPRAGE
starts

mprage=findstr('MPRAGE',SeriesDcpn{i});


    if (mprage>0)
        if (MagFS{i}==1.5)
            if (a11>0)
                movefile(A{1,i},'PID/MRI/1.5T/MPRAGE');
                save=save+1;
            else mkdir('PID/MRI/1.5T/MPRAGE');
                movefile(A{1,i},'PID/MRI/1.5T/MPRAGE');
                a11=a11+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (mprage>0)
        if (MagFS{i}==3)
            if (a12>0)
                movefile(A{1,i},'PID/MRI/3T/MPRAGE');
                save=save+1;
            else mkdir('PID/MRI/3T/MPRAGE');
                movefile(A{1,i},'PID/MRI/3T/MPRAGE');
                a12=a12+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end

mpr=findstr('mprage',SeriesDcpn{i});


    if (mpr>0)
        if (MagFS{i}==1.5)
            if (a11>0)
                movefile(A{1,i},'PID/MRI/1.5T/MPRAGE');
                save=save+1;
            else mkdir('PID/MRI/1.5T/MPRAGE');
                movefile(A{1,i},'PID/MRI/1.5T/MPRAGE');
                a11=a11+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (mpr>0)
        if (MagFS{i}==3)
            if (a12>0)
                movefile(A{1,i},'PID/MRI/3T/MPRAGE');
                save=save+1;
            else mkdir('PID/MRI/3T/MPRAGE');
                movefile(A{1,i},'PID/MRI/3T/MPRAGE');
                a12=a12+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%%%%%%%%%%%%%% a12 ends
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%MPRAGE ends Survey starts


Su=findstr('Survey',SeriesDcpn{i});


    if (Su>0)
        if (MagFS{i}==1.5)
            if (a19>0)
                movefile(A{1,i},'PID/MRI/1.5T/Survey');
                save=save+1;
            else mkdir('PID/MRI/1.5T/Survey');
                movefile(A{1,i},'PID/MRI/1.5T/Survey');
                a19=a19+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
     end

     if (Su>0)
        if (MagFS{i}==15000)
            if (a19>0)
                movefile(A{1,i},'PID/MRI/1.5T/Survey');
                save=save+1;
            else mkdir('PID/MRI/1.5T/Survey');
                movefile(A{1,i},'PID/MRI/1.5T/Survey');
                a19=a19+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end



    if (Su>0)
        if (MagFS{i}==3)
            if (a20>0)
                movefile(A{1,i},'PID/MRI/3T/Survey');
                save=save+1;
            else mkdir('PID/MRI/3T/Survey');
```

```
                movefile(A{1,i},'PID/MRI/3T/Survey');
                a20=a20+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end

su=findstr('su',SeriesDcpn{i});


    if (su>0)
        if (MagFS{i}==1.5)
            if (a19>0)
                movefile(A{1,i},'PID/MRI/1.5T/Survey');
                save=save+1;
            else mkdir('PID/MRI/1.5T/Survey');
                movefile(A{1,i},'PID/MRI/1.5T/Survey');
                a19=a19+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (su>0)
        if (MagFS{i}==3)
            if (a20>0)
                movefile(A{1,i},'PID/MRI/3T/Survey');
                save=save+1;
            else mkdir('PID/MRI/3T/Survey');
                movefile(A{1,i},'PID/MRI/3T/Survey');
                a20=a20+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Survey ends
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%BOLD Starts

BOLD=findstr('BOLD',SeriesDcpn{i});


    if (BOLD>0)
        if (MagFS{i}==1.5)
            if (a21>0)
                movefile(A{1,i},'PID/MRI/1.5T/BOLD');
                save=save+1;
            else mkdir('PID/MRI/1.5T/BOLD');
                movefile(A{1,i},'PID/MRI/1.5T/BOLD');
                a21=a21+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
     end

     if (BOLD>0)
        if (MagFS{i}==15000)
            if (a21>0)
                movefile(A{1,i},'PID/MRI/1.5T/BOLD');
                save=save+1;
            else mkdir('PID/MRI/1.5T/BOLD');
                movefile(A{1,i},'PID/MRI/1.5T/BOLD');
                a21=a21+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end



    if (BOLD>0)
        if (MagFS{i}==3)
            if (a22>0)
                movefile(A{1,i},'PID/MRI/3T/BOLD');
                save=save+1;
            else mkdir('PID/MRI/3T/BOLD');
                movefile(A{1,i},'PID/MRI/3T/BOLD');
                a22=a22+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end

bold=findstr('bold',SeriesDcpn{i});


    if (bold>0)
        if (MagFS{i}==1.5)
            if (a21>0)
                movefile(A{1,i},'PID/MRI/1.5T/BOLD');
                save=save+1;
            else mkdir('PID/MRI/1.5T/BOLD');
                movefile(A{1,i},'PID/MRI/1.5T/BOLD');
                a21=a21+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (bold>0)
        if (MagFS{i}==3)
            if (a22>0)
                movefile(A{1,i},'PID/MRI/3T/BOLD');
                save=save+1;
            else mkdir('PID/MRI/3T/BOLD');
                movefile(A{1,i},'PID/MRI/3T/BOLD');
                a22=a22+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% bold ends
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%fMRI1 starts

f1=findstr('fMRI2',SeriesDcpn{i});


    if (f1>0)
        if (MagFS{i}==1.5)
            if (a23>0)
                movefile(A{1,i},'PID/MRI/1.5T/fMRI1');
                save=save+1;
            else mkdir('PID/MRI/1.5T/fMRI1');
                movefile(A{1,i},'PID/MRI/1.5T/fMRI1');
                a23=a23+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
     end

     if (f1>0)
        if (MagFS{i}==15000)
            if (a23>0)
                movefile(A{1,i},'PID/MRI/1.5T/fMRI1');
                save=save+1;
            else mkdir('PID/MRI/1.5T/fMRI1');
                movefile(A{1,i},'PID/MRI/1.5T/fMRI1');
                a23=a23+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end



    if (f1>0)
        if (MagFS{i}==3)
            if (a24>0)
                movefile(A{1,i},'PID/MRI/3T/fMRI1');
                save=save+1;
            else mkdir('PID/MRI/3T/fMRI1');
                movefile(A{1,i},'PID/MRI/3T/fMRI1');
                a24=a24+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fMRI1 ends

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%fMRI2 starts
f2=findstr('fMRI2',SeriesDcpn{i});


    if (f2>0)
        if (MagFS{i}==1.5)
            if (a25>0)
                movefile(A{1,i},'PID/MRI/1.5T/fMRI2');
                save=save+1;
            else mkdir('PID/MRI/1.5T/fMRI2');
                movefile(A{1,i},'PID/MRI/1.5T/fMRI2');
                a25=a25+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (f2>0)
        if (MagFS{i}==3)
            if (a26>0)
                movefile(A{1,i},'PID/MRI/3T/fMRI2');
                save=save+1;
            else mkdir('PID/MRI/3T/fMRI2');
                movefile(A{1,i},'PID/MRI/3T/fMRI2');
                a26=a26+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%fMRI2
ends
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%ASL
starts

asl=findstr('ASL',SeriesDcpn{i});


    if (asl>0)
        if (MagFS{i}==1.5)
            if (a27>0)
                movefile(A{1,i},'PID/MRI/1.5T/ASL');
                save=save+1;
            else mkdir('PID/MRI/1.5T/ASL');
                movefile(A{1,i},'PID/MRI/1.5T/ASL');
                a27=a27+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (asl>0)
        if (MagFS{i}==3)
            if (a28>0)
                movefile(A{1,i},'PID/MRI/3T/ASL');
                save=save+1;
            else mkdir('PID/MRI/3T/ASL');
                movefile(A{1,i},'PID/MRI/3T/ASL');
                a28=a28+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%ASL ends
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%TBI starts

tbi=findstr('TBI',SeriesDcpn{i});
```

```
    if (tbi>0)
        if (MagFS{i}==1.5)
            if (a29>0)
                movefile(A{1,i},'PID/MRI/1.5T/TBI');
                save=save+1;
            else mkdir('PID/MRI/1.5T/TBI');
                movefile(A{1,i},'PID/MRI/1.5T/TBI');
                a29=a29+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (tbi>0)
        if (MagFS{i}==3)
            if (a30>0)
                movefile(A{1,i},'PID/MRI/3T/TBI');
                save=save+1;
            else mkdir('PID/MRI/3T/TBI');
                movefile(A{1,i},'PID/MRI/3T/TBI');
                a30=a30+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%TBI
ends
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FcMR
starts

fcm=findstr('FcMR',SeriesDcpn{i});


    if (fcm>0)
        if (MagFS{i}==1.5)
            if (a31>0)
                movefile(A{1,i},'PID/MRI/1.5T/FcMR');
                save=save+1;
            else mkdir('PID/MRI/1.5T/FcMR');
                movefile(A{1,i},'PID/MRI/1.5T/FcMR');
                a31=a31+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
     end

     if (fcm>0)
        if (MagFS{i}==15000)
            if (a31>0)
                movefile(A{1,i},'PID/MRI/1.5T/FcMR');
                save=save+1;
            else mkdir('PID/MRI/1.5T/FcMR');
                movefile(A{1,i},'PID/MRI/1.5T/FcMR');
                a31=a31+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end



    if (fcm>0)
        if (MagFS{i}==3)
            if (a32>0)
                movefile(A{1,i},'PID/MRI/3T/FcMR');
                save=save+1;
            else mkdir('PID/MRI/3T/FcMR');
                movefile(A{1,i},'PID/MRI/3T/FcMR');
                a32=a32+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end

fc=findstr('fcmr',SeriesDcpn{i});
```

```
    if (fc>0)
        if (MagFS{i}==1.5)
            if (a33>0)
                movefile(A{1,i},'PID/MRI/1.5T/FcMR');
                save=save+1;
            else mkdir('PID/MRI/1.5T/FcMR');
                movefile(A{1,i},'PID/MRI/1.5T/FcMR');
                a33=a33+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (fc>0)
        if (MagFS{i}==3)
            if (a34>0)
                movefile(A{1,i},'PID/MRI/3T/FcMR');
                save=save+1;
            else mkdir('PID/MRI/3T/FcMR');
                movefile(A{1,i},'PID/MRI/3T/FcMR');
                a34=a34+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FcMR ends
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%SWI starts
SWI=findstr('SWI',SeriesDcpn{i});


    if (SWI>0)
        if (MagFS{i}==1.5)
            if (a35>0)
                movefile(A{1,i},'PID/MRI/1.5T/SWI');
                save=save+1;
            else mkdir('PID/MRI/1.5T/SWI');
                movefile(A{1,i},'PID/MRI/1.5T/SWI');
                a35=a35+1;
                save=save+1;
            end
        end
     end
```

```
      if (save==1)
         save=0;
         continue
      end

      if (SWI>0)
         if (MagFS{i}==15000)
             if (a35>0)
                 movefile(A{1,i},'PID/MRI/1.5T/SWI');
                 save=save+1;
             else mkdir('PID/MRI/1.5T/SWI');
                 movefile(A{1,i},'PID/MRI/1.5T/SWI');
                 a35=a35+1;
                 save=save+1;
             end
         end
      end

      if (save==1)
         save=0;
         continue
     end



     if (SWI>0)
         if (MagFS{i}==3)
             if (a36>0)
                 movefile(A{1,i},'PID/MRI/3T/SWI');
                 save=save+1;
             else mkdir('PID/MRI/3T/SWI');
                 movefile(A{1,i},'PID/MRI/3T/SWI');
                 a36=a36+1;
                 save=save+1;
             end
         end
     end


 if (save==1)
         save=0;
         continue
 end


 swi=findstr('swi',SeriesDcpn{i});


     if (swi>0)
         if (MagFS{i}==1.5)
             if (a35>0)
                 movefile(A{1,i},'PID/MRI/1.5T/SWI');
                 save=save+1;
```

```
            else mkdir('PID/MRI/1.5T/SWI');
                movefile(A{1,i},'PID/MRI/1.5T/SWI');
                a35=a35+1;
                save=save+1;
            end
        end
     end

     if (save==1)
        save=0;
        continue
    end

    if (swi>0)
        if (MagFS{i}==3)
            if (a36>0)
                movefile(A{1,i},'PID/MRI/3T/SWI');
                save=save+1;
            else mkdir('PID/MRI/3T/SWI');
                movefile(A{1,i},'PID/MRI/3T/SWI');
                a36=a36+1;
                save=save+1;
            end
        end
    end


if (save==1)
        save=0;
        continue
end



end         %%%%%%%%%%%%%% for loop ends
```

## APPENDIX B

## JAVA CODE

Class NewJFrame

```
/*
 * NewJFrame.java
 *
 * Created on May 27, 2008, 8:03 PM
 */

package guiprac;


import java.awt.BorderLayout;
import java.awt.Stroke;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileWriter;
import java.util.*;
import java.sql.*;
import javax.swing.*;

import javax.swing.table.*;


/**

 *

 * @author  AVMaster

 */

public class NewJFrame extends javax.swing.JFrame {




    String choice=new String();

   // int [] cb = new int [13];

    //anarray =new int [12];





    /** Creates new form NewJFrame */

    public NewJFrame() {

        initComponents();

    }
```

```
/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */

// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jPopupMenu1 = new javax.swing.JPopupMenu();
    buttonGroup1 = new javax.swing.ButtonGroup();
    buttonGroup2 = new javax.swing.ButtonGroup();
    jTabbedPane1 = new javax.swing.JTabbedPane();
    jPanel1 = new javax.swing.JPanel();
    jCheckBox1 = new javax.swing.JCheckBox();
    jCheckBox2 = new javax.swing.JCheckBox();
    jPanel2 = new javax.swing.JPanel();
    protonDensity = new javax.swing.JCheckBox();
```

```
T1 = new javax.swing.JCheckBox();

T2 = new javax.swing.JCheckBox();

fLAIR = new javax.swing.JCheckBox();

dTI = new javax.swing.JCheckBox();

aSL = new javax.swing.JCheckBox();

dWI = new javax.swing.JCheckBox();

sWI = new javax.swing.JCheckBox();

jLabel1 = new javax.swing.JLabel();

fcMRI = new javax.swing.JCheckBox();

jLabel3 = new javax.swing.JLabel();

AND = new javax.swing.JRadioButton();

OR = new javax.swing.JRadioButton();

jLabel5 = new javax.swing.JLabel();

jPanel3 = new javax.swing.JPanel();

SEARCH = new javax.swing.JButton();

jLabel2 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jLabel6 = new javax.swing.JLabel();

jLabel7 = new javax.swing.JLabel();
```

```
jLabel8 = new javax.swing.JLabel();

jLabel9 = new javax.swing.JLabel();

jLabel10 = new javax.swing.JLabel();

jLabel11 = new javax.swing.JLabel();

jTextField1 = new javax.swing.JTextField();

jTextField2 = new javax.swing.JTextField();

jTextField3 = new javax.swing.JTextField();

jTextField4 = new javax.swing.JTextField();

jLabel12 = new javax.swing.JLabel();

jLabel13 = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);


jTabbedPane1.setBorder(javax.swing.BorderFactory.createEtchedBorder());

jTabbedPane1.setTabPlacement(javax.swing.JTabbedPane.LEFT);


jCheckBox1.setText("3.0T");

jCheckBox1.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {
            jCheckBox1ActionPerformed(evt);
        }
    });

    jCheckBox2.setText("1.5T");
    jCheckBox2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jCheckBox2ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(134, 134, 134)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jCheckBox2)
                .addComponent(jCheckBox1))
            .addContainerGap(195, Short.MAX_VALUE))
    );
    jPanel1Layout.setVerticalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
            .addContainerGap(74, Short.MAX_VALUE)
            .addComponent(jCheckBox1)
            .addGap(82, 82, 82)
            .addComponent(jCheckBox2)
            .addGap(120, 120, 120))
    );


    jTabbedPane1.addTab("Magnetic Field Strength", jPanel1);


    protonDensity.setText("Proton Density");
    protonDensity.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
    protonDensity.addItemListener(new java.awt.event.ItemListener() {
```

```
    public void itemStateChanged(java.awt.event.ItemEvent evt) {
        protonDensityItemStateChanged(evt);
    }
});
protonDensity.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        protonDensityActionPerformed(evt);
    }
});


T1.setText("T1");
T1.addItemListener(new java.awt.event.ItemListener() {
    public void itemStateChanged(java.awt.event.ItemEvent evt) {
        T1ItemStateChanged(evt);
    }
});
T1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        T1ActionPerformed(evt);
    }
});

T2.setText("T2");
T2.addItemListener(new java.awt.event.ItemListener() {
    public void itemStateChanged(java.awt.event.ItemEvent evt) {
        T2ItemStateChanged(evt);
    }
});
T2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        T2ActionPerformed(evt);
    }
});

fLAIR.setText("Flair");
fLAIR.addItemListener(new java.awt.event.ItemListener() {
    public void itemStateChanged(java.awt.event.ItemEvent evt) {
        fLAIRItemStateChanged(evt);
    }
});

dTI.setText("DTI");
dTI.addItemListener(new java.awt.event.ItemListener() {
    public void itemStateChanged(java.awt.event.ItemEvent evt) {
        dTIItemStateChanged(evt);
    }
});

aSL.setText("ASL");
aSL.addItemListener(new java.awt.event.ItemListener() {
    public void itemStateChanged(java.awt.event.ItemEvent evt) {
        aSLItemStateChanged(evt);
    }
});

dWI.setText("DWI");
dWI.addItemListener(new java.awt.event.ItemListener() {
    public void itemStateChanged(java.awt.event.ItemEvent evt) {

        dWIItemStateChanged(evt);

    }

});

dWI.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        dWIActionPerformed(evt);

    }

});



sWI.setText("SWI");

sWI.addItemListener(new java.awt.event.ItemListener() {

    public void itemStateChanged(java.awt.event.ItemEvent evt) {

        sWIItemStateChanged(evt);

    }

});



jLabel1.setText("Please select the Scan type");
```

```
fcMRI.setText("fcMRI");
fcMRI.addItemListener(new java.awt.event.ItemListener() {
    public void itemStateChanged(java.awt.event.ItemEvent evt) {
        fcMRIItemStateChanged(evt);
    }
});
fcMRI.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        fcMRIActionPerformed(evt);
    }
});


jLabel3.setText("by checking appropriate check boxes:");



buttonGroup1.add(AND);
AND.setText("AND");
AND.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ANDActionPerformed(evt);
    }
});

buttonGroup1.add(OR);
OR.setText("OR");
OR.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ORActionPerformed(evt);
    }
});

jLabel5.setText("Please select the boolean operation:");

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel2Layout.createSequentialGroup()
        .addContainerGap()
```

```
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel3)
                .addComponent(jLabel1)
                .addGroup(jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(protonDensity)
                    .addComponent(T2)
                    .addComponent(dTI)
                    .addComponent(T1)
                    .addComponent(fLAIR))
                  .addGap(39, 39, 39)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(aSL)
                    .addComponent(dWI)
```

```
                                .addComponent(sWI)

                                .addComponent(fcMRI))))

                        .addContainerGap(179, Short.MAX_VALUE))

                      .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()

                        .addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                        .addGap(18, 18, 18)

                        .addComponent(AND)

                        .addGap(18, 18, 18)

                        .addComponent(OR)

                        .addGap(281, 281, 281))))

    );

    jPanel2Layout.setVerticalGroup(

      jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

      .addGroup(jPanel2Layout.createSequentialGroup()

        .addGap(19, 19, 19)

        .addComponent(jLabel1)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(jLabel3)
```

```
                .addGap(54, 54, 54)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(protonDensity)
                    .addComponent(fcMRI))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(T1)
                    .addComponent(aSL))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(T2)
                    .addComponent(dWI))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(fLAIR)
                    .addComponent(sWI))
```

```
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(dTI)
            .addGap(44, 44, 44)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
              .addComponent(jLabel5)
              .addComponent(AND)
              .addComponent(OR))
            .addContainerGap(32, Short.MAX_VALUE))
        );

        jTabbedPane1.addTab("Scan Types", jPanel2);


        jPanel3.setBorder(new javax.swing.border.MatteBorder(null));
        jPanel3.setFont(new java.awt.Font("Calibri", 0, 11)); // NOI18N
        jPanel3.setPreferredSize(new java.awt.Dimension(0, 322));


        SEARCH.setText("Search");
        SEARCH.addActionListener(new java.awt.event.ActionListener() {
          public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        SEARCHActionPerformed(evt);
    }
});

jLabel2.setText("Keyword 1:");

jLabel4.setText("Keyword 2:");

jLabel6.setText("Keyword 3:");

jLabel7.setText("Keyword 4:");

jLabel8.setText("Please enter the keywords by which");

jLabel9.setText("should be searched for in the DataBase");

jLabel10.setFont(new java.awt.Font("Calibri", 3, 14)); // NOI18N
jLabel10.setText("NOTE:");
```

```
jLabel11.setForeground(new java.awt.Color(102, 0, 0));

jLabel11.setText("Enter a single word or phrase in each Keyword field");



jTextField1.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jTextField1ActionPerformed(evt);

    }

});



jLabel12.setForeground(new java.awt.Color(102, 0, 0));

jLabel12.setText("allowed keywords are Last name, First name,");



jLabel13.setForeground(new java.awt.Color(102, 0, 0));

jLabel13.setText("scan date and any word from comments column");



javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);

jPanel3.setLayout(jPanel3Layout);

jPanel3Layout.setHorizontalGroup(

    jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
        .addGroup(jPanel3Layout.createSequentialGroup()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
              .addContainerGap()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
              .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel3Layout.createSequentialGroup()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                  .addComponent(jLabel9)
                  .addComponent(jLabel8))
                .addGap(39, 39, 39))
              .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel3Layout.createSequentialGroup()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                  .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel3Layout.createSequentialGroup()
                    .addComponent(jLabel7)
```

```
                                        .addGap(18, 18, 18)
                                        .addComponent(jTextField4, javax.swing.GroupLayout.DEFAULT_SIZE,
132, Short.MAX_VALUE))
                                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel3Layout.createSequentialGroup()
                                        .addComponent(jLabel6)
                                        .addGap(18, 18, 18)
                                        .addComponent(jTextField3, javax.swing.GroupLayout.DEFAULT_SIZE,
132, Short.MAX_VALUE))
                                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel3Layout.createSequentialGroup()
                                        .addComponent(jLabel4)
                                        .addGap(18, 18, 18)
                                        .addComponent(jTextField2, javax.swing.GroupLayout.DEFAULT_SIZE,
132, Short.MAX_VALUE))
                                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel3Layout.createSequentialGroup()
                                        .addComponent(jLabel2)
                                        .addGap(18, 18, 18)
                                        .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 132,
javax.swing.GroupLayout.PREFERRED_SIZE)))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 159,
Short.MAX_VALUE))
                              .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel3Layout.createSequentialGroup()
                                  .addComponent(jLabel10)


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)


.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                      .addComponent(jLabel11, javax.swing.GroupLayout.DEFAULT_SIZE, 322,
Short.MAX_VALUE)
                                      .addComponent(jLabel12, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 233,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                      .addComponent(jLabel13,
javax.swing.GroupLayout.Alignment.LEADING)))))
                      .addGroup(jPanel3Layout.createSequentialGroup()
                          .addGap(79, 79, 79)
                          .addComponent(SEARCH)))
                  .addContainerGap())
        );
```

```
    jPanel3Layout.setVerticalGroup(

        jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel3Layout.createSequentialGroup()

            .addContainerGap()

            .addComponent(jLabel8)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

            .addComponent(jLabel9)

            .addGap(18, 18, 18)


.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                .addComponent(jLabel2)

                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)


.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                .addComponent(jLabel4)

                .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

            .addGap(8, 8, 8)
```

```
.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
          .addComponent(jLabel6)
          .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(8, 8, 8)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
          .addComponent(jLabel7)
          .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
          .addComponent(jLabel10)
          .addComponent(jLabel11))
        .addGap(1, 1, 1)
        .addComponent(jLabel12)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel13)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
                .addComponent(SEARCH)

                .addGap(332, 332, 332))

        );



        jTabbedPane1.addTab("Search by comments", jPanel3);



        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

        getContentPane().setLayout(layout);

        layout.setHorizontalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 514,
javax.swing.GroupLayout.PREFERRED_SIZE)

        );

        layout.setVerticalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

        );



        pack();
```

```
}// </editor-fold>


public void buttonGroup() {




buttonGroup1 = new ButtonGroup();

buttonGroup1.add(AND);

AND.addActionListener((ActionListener) this);

AND.setActionCommand ("AND");

buttonGroup1.add(OR);

OR.addActionListener((ActionListener) this);

OR.setActionCommand("OR");




}


private void actionPerformed(ActionEvent e)

{

    choice = buttonGroup1.getSelection().getActionCommand();
    }

    private void protonDensityItemStateChanged(java.awt.event.ItemEvent evt) {
        if (connector.cb[1]==1)
      {   connector.cb[1]=0;
      }
      else {
          connector.cb[1]=1;
      }
        System.out.println(connector.cb[1]);
         // TODO add your handling code here:
}

    private void T1ItemStateChanged(java.awt.event.ItemEvent evt) {
        if (connector.cb[2]==1)
      {   connector.cb[2]=0;
      }
      else {
          connector.cb[2]=1;
      }// TODO add your handling code here:
        //System.out.append("t1");
}

    private void T2ItemStateChanged(java.awt.event.ItemEvent evt) {
       if (connector.cb[3]==1)
      {   connector.cb[3]=0;
      }
      else {
          connector.cb[3]=1;
      } // TODO add your handling code here:
       //System.out.append("t2");
}

    private void fLAIRItemStateChanged(java.awt.event.ItemEvent evt) {
     if (connector.cb[4]==1)
      {   connector.cb[4]=0;
      }
      else {
            connector.cb[4]=1;

        } // TODO add your handling code here:

}



    private void dTIItemStateChanged(java.awt.event.ItemEvent evt) {

        if (connector.cb[5]==1)

      {   connector.cb[5]=0;

      }

      else {

            connector.cb[5]=1;

      }// TODO add your handling code here:

}



    private void fcMRIItemStateChanged(java.awt.event.ItemEvent evt) {

       if (connector.cb[6]==1)

      {   connector.cb[6]=0;

      }

      else {

            connector.cb[6]=1;
      }// TODO add your handling code here:
}

    private void aSLItemStateChanged(java.awt.event.ItemEvent evt) {
       if (connector.cb[7]==1)
      {   connector.cb[7]=0;
      }
      else {
            connector.cb[7]=1;
      } // TODO add your handling code here:
}

    private void dWIItemStateChanged(java.awt.event.ItemEvent evt) {
        if (connector.cb[8]==1)
      {   connector.cb[8]=0;
            System.out.println(connector.cb[8]);
      }
      else {
            connector.cb[8]=1;
```

```
          System.out.println(connector.cb[8]);

      }// TODO add your handling code here:

}



    private void sWIItemStateChanged(java.awt.event.ItemEvent evt) {

        if (connector.cb[9]==1)

      {   connector.cb[9]=0;

      }

      else {

          connector.cb[9]=1;

      }// TODO add your handling code here:

}




    private void protonDensityActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

}



    private void fcMRIActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }





    private void SEARCHActionPerformed(java.awt.event.ActionEvent evt) {
    //connector = new DBStatement();// obj for SQL queries.
    try {

        connector.s1=this.jTextField1.getText();
        connector.s2=this.jTextField2.getText();
        connector.s3=this.jTextField3.getText();
        connector.s4=this.jTextField4.getText();

        result = connector.DBStatement1();
        //ResultSet excelRs = connector.excelRs;
        /*if(result != null) System.out.println("not null");

        while(result.next()) {
```

```
        System.out.println(result.getString("Last"));

    }*/




}

catch(Exception e) {

    //System.err.println("Result set cannot be executed, Since values from the tabs not selected :
(Exception from class NewJFrame)");

    System.err.println(e.getStackTrace());

    System.out.println("catch of newjframe");

}



finally {

    System.out.println("entereed finally");

    model = new SQL1(); // model is an obj for displaying table

    System.out.println("here");

    if(model ==  null) System.out.println("null");

    model.setResultSet(result);

    // JTable table = new JTable(model);
    rs = new ResultDisplay();



    rs.resultTable.setModel(model.modelForResult);

    rs.resultTable.setVisible(true);

    rs.resultTable.setToolTipText("loooop");

    tcm = rs.resultTable.getColumnModel();

    tc = tcm.getColumn(5);

    tc.setCellRenderer(new renderer());

    rs.setVisible(true);


}


// TODO add your handling code here:

}


private void T1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

    //System.out.println("t1");
```

```
}

private void ANDActionPerformed(java.awt.event.ActionEvent evt) {
this.actionPerformed(evt);

connector.c1=1;  //for AND

}

private void ORActionPerformed(java.awt.event.ActionEvent evt) {
this.actionPerformed(evt);

connector.c1=2; // for OR
}

private void jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt) {
    if(this.jCheckBox1.isSelected()) {
        connector.f1 = true;
    }
    else {
        connector.f1 = false;
    }
    System.out.println(connector.f1);
}


private void jCheckBox2ActionPerformed(java.awt.event.ActionEvent evt) {
    if(this.jCheckBox2.isSelected()) {
        connector.f2 = true;
    }
    else {
        connector.f2 = false;
    }
    System.out.println(connector.f2);// TODO add your handling code here:
}


private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
                  // TODO add your handling code here:
}
```

```
private void T2ActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:
}

private void dWIActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}

    // things to be done with Text Area JTextArea1

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
```

```
        public void run() {
            new NewJFrame().setVisible(true);
        }
    });
}


// Variables declaration - do not modify
private javax.swing.JRadioButton AND;
private javax.swing.JRadioButton OR;
private javax.swing.JButton SEARCH;
private javax.swing.JCheckBox T1;
private javax.swing.JCheckBox T2;
private javax.swing.JCheckBox aSL;
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.ButtonGroup buttonGroup2;
private javax.swing.JCheckBox dTI;
private javax.swing.JCheckBox dWI;
private javax.swing.JCheckBox fLAIR;
private javax.swing.JCheckBox fcMRI;

private javax.swing.JCheckBox jCheckBox1;

private javax.swing.JCheckBox jCheckBox2;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel10;

private javax.swing.JLabel jLabel11;

private javax.swing.JLabel jLabel12;

private javax.swing.JLabel jLabel13;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JLabel jLabel8;

private javax.swing.JLabel jLabel9;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanel3;
```

```
    private javax.swing.JPopupMenu jPopupMenu1;

    private javax.swing.JTabbedPane jTabbedPane1;

    private javax.swing.JTextField jTextField1;

    private javax.swing.JTextField jTextField2;

    private javax.swing.JTextField jTextField3;

    private javax.swing.JTextField jTextField4;

    private javax.swing.JCheckBox protonDensity;

    private javax.swing.JCheckBox sWI;

    // End of variables declaration



public String [] substrg;    //JFrame frame;

    public int counter;

    private SQL1 model; // obj for the table

    private JScrollPane resultPane;

    int radiobutton1, radiobutton2;

    private DBStatement connector = new DBStatement();

    public ResultDisplay rs;

    public TableColumnModel tcm;

    public TableColumn tc;
    public ResultSet result;

}
```

Class DBStatement

```
package guiprac;


import java.awt.*;

import java.awt.event.*;

import java.io.FileWriter;

import java.io.IOException;

import java.sql.*;

import java.util.*;

import javax.swing.*;




import javax.swing.table.TableModel;
```

```
/**

 *

 * @author AVMaster

 */

public class DBStatement{

    public int [] cb = new int [13];

    public ResultSet excelRs;

    public String link= new String("");

    public PreparedStatement stmt2;

    public int c1=1;

    public boolean f1,f2;

    public String s1 = new String(),s2 = new String(),s3 = new String(),s4 = new String();

    public boolean [] blankString = new boolean[4];

    public String[] stringArray = new String[4];

    DBStatement() {



        for (int j=0; j<13; j++)

            {
                cb[j]=0;

            }



    }



    public ResultSet DBStatement1() throws ClassNotFoundException {

        ResultSet result = null;

        StringBuffer contenu, contenu2;

        contenu = new StringBuffer();

        contenu2 = new StringBuffer();



        try {

            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

            String sourceURL = new String("jdbc:odbc:Test2");

            Connection DBConnection = DriverManager.getConnection(sourceURL);

            if(DBConnection.equals(null)) {

                System.out.println("program not working");

            }

           PreparedStatement stmt = DBConnection.prepareStatement("UPDATE adcp1 set
ProtonDensity =0 where ProtonDensity is null");
```

```
        stmt.executeUpdate();

        stmt = DBConnection.prepareStatement("UPDATE adcp1 set T1 =0 where T1 is null");

        stmt.executeUpdate();

        stmt = DBConnection.prepareStatement("UPDATE adcp1 set T2 =0 where T2 is null");

        stmt.executeUpdate();

        stmt = DBConnection.prepareStatement("UPDATE adcp1 set FLAIR =0 where FLAIR is
null");

        stmt.executeUpdate();

        stmt = DBConnection.prepareStatement("UPDATE adcp1 set DTI =0 where DTI is null");

        stmt.executeUpdate();

        stmt = DBConnection.prepareStatement("UPDATE adcp1 set fcMRI =0 where fcMRI is
null");

        stmt.executeUpdate();

        stmt = DBConnection.prepareStatement("UPDATE adcp1 set ASL =0 where ASL is null");

        stmt.executeUpdate();

        stmt = DBConnection.prepareStatement("UPDATE adcp1 set DWI =0 where DWI is null");

        stmt.executeUpdate();

        stmt = DBConnection.prepareStatement("UPDATE adcp1 set SWI =0 where SWI is null");

        stmt.executeUpdate();
        /*stmt = DBConnection.prepareStatement("UPDATE adcp1 set MPRAGE =0 where
MPRAGE is null");

        stmt.executeUpdate();

        stmt = DBConnection.prepareStatement("UPDATE adcp1 set Survey =0 where Survey is
null");

        stmt.executeUpdate();

        stmt = DBConnection.prepareStatement("UPDATE adcp1 set BOLD =0 where BOLD  is
null");

        stmt.executeUpdate();

    */



        if  ( c1==1)  // c1 for AND in Scan type

        {/*//stmt = DBConnection.prepareStatement("Select First,Last,ID,PID,ScanDate from
adcp1 where (ProtonDensity = 1 OR ProtonDensity = ?) and (T1 = 1 OR T1 = ? ) and (T2 = 1 OR
T2 = ?) and (FLAIR = 1 OR FLAIR = ?) and (DTI = 1 OR DTI = ?) and (fcMRI = 1 OR fcMRI = ?)
and (ASL = 1 OR ASL = ?) and (DWI = 1 OR DWI = ?) and (SWI = 1 OR SWI = ?) and (MPRAGE
= 1 OR MPRAGE = ?) and (Survey = 1 OR Survey = ?) and (BOLD = 1 OR BOLD = ?) and
(FieldStrength = ? OR FieldStrength = ?) and ((First like ? or Last like ? or Comments like ? or
Scandate like ? or ADCPID like ?) or (First like ? or Last like ? or Comments like ? or Scandate
like ? or ADCPID like ?) or (First like ? or Last like ? or Comments like ? or Scandate like ? or
ADCPID like ?) or (First like ? or Last like ? or Comments like ? or Scandate like ? or ADCPID like
?))");

        */
        stmt=DBConnection.prepareStatement("Select First,Last, ID,PID, ScanDate, hyperlink
from adcp1 where (ProtonDensity = 1 or ProtonDensity = ?) and (T1 = 1 or T1 = ?) and (T2 = 1 or
T2 = ?) and (FLAIR = 1 or FLAIR = ?) and (DTI = 1 or DTI = ?) and (fcMRI = 1 or fcMRI = ?) and
(ASL = 1 or ASL = ?) and (DWI = 1 or DWI = ?) and (SWI = 1 or SWI = ?) and (FieldStrength = ?
or FieldStrength = ?) and ((First like ? or Last like ? or Comments like ? or Scandate like ?) or
(First like ? or Last like ? or Comments like ? or Scandate like ?) or (First like ? or Last like ? or
Comments like ? or Scandate like ?) or (First like ? or Last like ? or Comments like ? or Scandate
like ?))");

        //stmt=DBConnection.prepareStatement("Select First,Last, ID, ScanDate from adcp1
where (ProtonDensity = 1 or ProtonDensity = 0) and ( T1 = 1 or T1 = 1) and (T2 = 1 or T2 = 1 )
and (FLAIR = 1 or FLAIR = 0) and (DTI = 1 or DTI = 0) and (fcMRI = 1 or fcMRI = 0) and (ASL = 1
or ASL = 0) and (DWI = 1 or DWI = 0) and (SWI = 1 or SWI = 0) and (FieldStrength =1 or
FieldStrength = 3) and ((First like "\*andr\*" or Last like "\*andr\*" or Comments like "\*andr\*" or
Scandate like "\*andr\*") or (First like "\*andr\*" or Last like "\*andr\*" or Comments like "\*andr\*"
or Scandate like "\*andr\*") or (First like "\*andr\*" or Last like "\*andr\*" or Comments like
"\*andr\*" or Scandate like "\*andr\*") or (First like "\*andr\*" or Last like "\*andr\*" or Comments
like "\*andr\*" or Scandate like "\*andr\*"))");

        //stmt = DBConnection.prepareStatement("select First, ID from adcp1 where Last like
?");

        //stmt = DBConnection.prepareStatement("select First, ID from adcp1 where Last like
?");

      }//stmt = DBConnection.prepareStatement(



      if (c1==2)   // c1 for OR in Scan type
```

```
            {

                //stmt = DBConnection.prepareStatement("Select First,Last, ID,PID,ScanDate from
adcp1 where (ProtonDensity = 1 and ProtonDensity = ?) or ( T1 = 1 and T1 = ? )or (T2 = 1 and
T2 = ? )or (FLAIR = 1 and FLAIR = ?) or (DTI = 1 and DTI = ?) or (fcMRI = 1 and fcMRI = ?) or
(ASL = 1 and ASL = ?) or (DWI = 1 and DWI = ?) or (SWI = 1 and SWI = ?) or (MPRAGE = 1 and
MPRAGE = ?) or (Survey = 1 and Survey = ?) or (BOLD = 1 and BOLD = ?) and (FieldStrength =
? OR FieldStrength = ?)and ((First like ? or Last like ? or Comments like ? or Scandate like ? or
ADCPID like ?)or (First like ? or Last like ? or Comments like ? or Scandate like ? or ADCPID like
?) or (First like ? or Last like ? or Comments like ? or Scandate like ? or ADCPID like ?) or (First
like ? or Last like ? or Comments like ? or Scandate like ? or ADCPID like ?))");

            stmt=DBConnection.prepareStatement("Select First,Last, ID, PID,ScanDate, hyperlink
from adcp1 where (ProtonDensity = 1 and ProtonDensity = ?) or ( T1 = 1 and T1 = ?) or (T2 = 1
and T2 = ?) or (FLAIR = 1 and FLAIR = ?) or (DTI = 1 and DTI = ?) or (fcMRI = 1 and fcMRI = ?)
or (ASL = 1 and ASL = ?) or (DWI = 1 and DWI = ?) or (SWI = 1 and SWI = ?) and (FieldStrength
= ? or FieldStrength = ?) and ((First like ? or Last like ? or Comments like ? or Scandate like ?) or
(First like ? or Last like ? or Comments like ? or Scandate like ?) or (First like ? or Last like ? or
Comments like ? or Scandate like ?) or (First like ? or Last like ? or Comments like ? or Scandate
like ?))");


            }


        for(int i =1; i<10; i++) {

            System.out.println(cb[i]);
```

```
    System.out.println("end");
}


stmt.setInt(1, cb[1]);
//System.out.println(cb[2]);
stmt.setInt(2, cb[2]);
//System.out.println(cb[3]);
stmt.setInt(3, cb[3]);


stmt.setInt(4, cb[4]);


stmt.setInt(5, cb[5]);
stmt.setInt(6, cb[6]);
stmt.setInt(7, cb[7]);
stmt.setInt(8, cb[8]);
stmt.setInt(9, cb[9]);
// stmt.setInt(10, cb[10]);
//stmt.setInt(11, cb[11]);
//stmt.setInt(12, cb[12]);
```

```
if(f1 == true && f2 == true) {

    stmt.setInt(10,3);

    stmt.setInt(11,1);

    System.out.println("1 or 3");

}

else if(f1 == true) {

    stmt.setInt(10,3);

    stmt.setInt(11,3);

    System.out.println("only 3");

}

else if(f2 == true) {

    stmt.setInt(10,1);

    stmt.setInt(11,1);

    System.out.println("only 1");

}

else {

    stmt.setInt(10,3);

    stmt.setInt(11,3);

    System.out.println("only 3 default");
```

```
}



stringArray[0] = s1;

stringArray[1] = s2;

stringArray[2] = s3;

stringArray[3] = s4;


int blankCount=0, trueWord = 0;


for(int i = 0; i<4; i++) {

    if(stringArray[i].equals("")) {

        blankString[i] = false;

        blankCount++;

    }

    else {

        blankString[i] = true;

        trueWord = i+1;

    }
```

```
}

if(trueWord == 0) {
     stringArray[0] = new String("%");
     System.out.println(stringArray[0]);
     stringArray[1] = new String("%");
     System.out.println(stringArray[1]);
     stringArray[2] = new String("%");
     System.out.println(stringArray[2]);
     stringArray[3] = new String("%");
     System.out.println(stringArray[3]);
}

else
{
    System.out.println("entered else");
    for(int i=0; i < 4; i++) {
         if(blankString[i] == false) {
             stringArray[i] = stringArray[trueWord-1];

         }

     }

    System.out.println("finished for");

    for(int i =0; i<4; i++) {

          stringArray[i] = (new String("%")).concat(stringArray[i]);

          stringArray[i] = stringArray[i].concat(new String("%"));

    }

}
```

```
System.out.println("loop starts");

for(int i=0;i<4;i++) {

    System.out.println(stringArray[i]);

}
```

```
stmt.setString(12, stringArray[0]);

stmt.setString(13, stringArray[0]);

stmt.setString(14, stringArray[0]);

stmt.setString(15, stringArray[0]);




stmt.setString(16, stringArray[1]);

stmt.setString(17, stringArray[1]);

stmt.setString(18, stringArray[1]);

stmt.setString(19, stringArray[1]);



stmt.setString(20, stringArray[2]);

stmt.setString(21, stringArray[2]);

stmt.setString(22, stringArray[2]);

stmt.setString(23, stringArray[2]);
stmt.setString(24, stringArray[3]);

stmt.setString(25, stringArray[3]);

stmt.setString(26, stringArray[3]);

stmt.setString(27, stringArray[3]);



//ParameterMetaData p = stmt.getParameterMetaData();

//stmt.setString(1, "%");

//stmt.setInt(1, 1);



result = stmt.executeQuery();

stmt2 = stmt;

excelRs = stmt2.executeQuery();



// Printing to Excel spreadsheet

ResultSetMetaData rsMeta = excelRs.getMetaData();

for(int i = 1; i<=rsMeta.getColumnCount(); i++){

   contenu2.append(rsMeta.getColumnLabel(i)+"\t");
```

```
    }



System.out.println("before while");

contenu2.append("\n");



int numCols = rsMeta.getColumnCount();



while(excelRs.next()){

    for(int j=1; j<=numCols; j++) {

       if(j==6) {

           link = excelRs.getString(j);

           char a = '#';

           char b = '\0';

           if(link != null) link = new String(link.replace(a, b));

           else link = new String("");



           link = new String("=HYPERLINK(\"").concat(link);

           link = link.concat("\")");
```

```
            System.out.println(link);

            contenu2.append(link + "\t");

        }



        else {

            contenu2.append(excelRs.getString(j) + "\t");

        }



 }

System.out.println("\n");

contenu2.append("\n");



 //now, just save the StringBuffer in a file

 FileWriter excelFile = new FileWriter("myResultSet.xls");

 //excelFile.write(new String(contenu));

 excelFile.write(new String(contenu2));

 excelFile.close();

 System.out.println("end");
 }



 excelRs.close();


 }


 catch(java.io.IOException e){
 //return result;
 }
 catch(SQLException e){


 }
 finally {
     System.out.println("finally");
    //return result;
 }
return result;
```

```
    }




}
```

Class SQL1

```
package guiprac;



import java.sql.*;

import java.util.*;

import javax.swing.table.AbstractTableModel;

//import java.sql.ResultSet;

import javax.swing.table.DefaultTableModel;
```

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author AVMaster
 */
public class SQL1 {


    private Vector colname= new Vector() ;
    DefaultTableModel modelForResult;
     Vector vector = new Vector();
    public ResultSetMetaData metadata;


    public void setResultSet(ResultSet result) {
         try {
            metadata = result.getMetaData();
            int col = metadata.getColumnCount();
            for (int i = 0; i < col; i++) {

                colname.add(metadata.getColumnLabel(i + 1));

            }

           //Get all rows



            vector.clear();  //empty vector to stare the data

            Vector rowdata; // stores one row

            while (result.next()) {

                rowdata =  new Vector();  // create array to hold data

                for (int j = 0; j < col; j++) {

                    rowdata.add(result.getString(j + 1));

                }



                vector.add(rowdata);   //store the row in the vector

            }

            //signal the table there is a new model data

            modelForResult = new DefaultTableModel(vector, colname);



        } catch (SQLException e) {
```

```
            System.err.println(e);
        }
        catch(Exception e){
            System.out.println(e.getMessage());
        }
    }

//
//    }
}
```

Class ResultDisplay

```
package guiprac;


import java.awt.Desktop;
import java.awt.Point;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.*;
```

```
/*

 * ResultDisplay.java

 *

 * Created on June 8, 2008, 3:53 PM

 */

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.net.URISyntaxException;

import java.sql.ResultSet;

import java.util.logging.Level;

import java.util.logging.Logger;




/**

 *

 * @author  AVMaster
 */

public class ResultDisplay extends javax.swing.JFrame {

    public Vector vect1;



    /** Creates new form ResultDisplay */

    public ResultDisplay() {

        initComponents();

        resultTable.addMouseListener(new MouseListener(){



            public void mouseClicked(MouseEvent e) {

                Point point= new Point(e.getX(),e.getY());





                int columnNumber=resultTable.columnAtPoint(point);

                if(columnNumber==5){

                int rowNumber=resultTable.rowAtPoint(point);

                String dataInCell= (String)resultTable.getValueAt(rowNumber, columnNumber);

                dataInCell= dataInCell.replace("#", "");

                dataInCell=dataInCell.replace("\\", "/");
                dataInCell=dataInCell.concat("/");

                        try {
                            Desktop.getDesktop().open(new File(dataInCell));

                    } catch (IOException ex) {
                        Logger.getLogger(ResultDisplay.class.getName()).log(Level.SEVERE, null, ex);
                    }

                }

             //   throw new UnsupportedOperationException("Not supported yet.");
            }

            public void mousePressed(MouseEvent e) {
            //    throw new UnsupportedOperationException("Not supported yet.");
            }

            public void mouseReleased(MouseEvent e) {
                  //     throw new UnsupportedOperationException("Not supported yet.");

                   }



                   public void mouseEntered(MouseEvent e) {

                    //   throw new UnsupportedOperationException("Not supported yet.");

                   }



                   public void mouseExited(MouseEvent e) {

                   //    throw new UnsupportedOperationException("Not supported yet.");

                   }



                }

                );

            }



            /** This method is called from within the constructor to

             * initialize the form.

             * WARNING: Do NOT modify this code. The content of this method is

             * always regenerated by the Form Editor.
             */
            @SuppressWarnings("unchecked")
            // <editor-fold defaultstate="collapsed" desc="Generated Code">
            private void initComponents() {

                jScrollPane1 = new javax.swing.JScrollPane();
                resultTable = new javax.swing.JTable();

                setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
                setMinimumSize(new java.awt.Dimension(800, 600));

                resultTable.setModel(new javax.swing.table.DefaultTableModel(
                   new Object [][] {
                       {null, null, null, null},
                       {null, null, null, null},
                       {null, null, null, null},
                       {null, null, null, null}
                   },
                   new String [] {
                       "Title 1", "Title 2", "Title 3", "Title 4"
                   }
                ));
                jScrollPane1.setViewportView(resultTable);

                getContentPane().add(jScrollPane1, java.awt.BorderLayout.CENTER);

                pack();
            }// </editor-fold>

            /**
            * @param args the command line arguments
             *
             *
            */
            /*public void excelFile ()
            {
                  try {
                      FileWriter excelFile = new FileWriter("myResultSet.xls");
```

```
            excelFile.write(new Vector(resultSet.vector,resultSet.colname));

            excelFile.close();

          }



      catch(Exception e) {

      e.printStackTrace();

      }

   }*/



   public static void main(String args[]) {

      java.awt.EventQueue.invokeLater(new Runnable() {

         public void run() {

            new ResultDisplay().setVisible(true);

         }

      });

   }



   // Variables declaration - do not modify
```

```
    private javax.swing.JScrollPane jScrollPane1;

    public javax.swing.JTable resultTable;

    // End of variables declaration



     public SQL1 resultSet;

     public DBStatement db;

}
```

Class Renderer

```
/*

 * To change this template, choose Tools | Templates

 * and open the template in the editor.

 */



package guiprac;

import java.awt.Component;

import javax.swing.*;

import javax.swing.table.TableCellRenderer;
```

```
/**
 *
 * @author AVMaster
 */
public class renderer extends JLabel implements TableCellRenderer  {
    public renderer() {
        super();
    }

    public Component getTableCellRendererComponent(JTable table, Object value, boolean
isSelected, boolean hasFocus, int
row, int column) {

        if(null==value){
            value=new String();
        }
         if (isSelected) {
           setForeground(table.getSelectionForeground());
           super.setBackground(table.getSelectionBackground());
         } else {
           setForeground(table.getForeground());

           setBackground(table.getBackground());

         }

         String urlString = (new String("<a href = \"")).concat(value.toString());

         urlString.concat("\">");

         urlString.concat(value.toString());

         urlString.concat("</a>");

         this.setText(urlString);

         return this;

       }



}
```

## REFERENCES

[1] http://en.wikipedia.org/wiki/Dicom

[2] http://www.mathworks.com/store/productIndexLink.do?s_cid=buynow_sv1

[3] http://www.java.com/en/

[4] http://en.wikipedia.org/wiki/Java

[5] http://en.wikipedia.org/wiki/Database

[6] http://support.microsoft.com/kb/110093

[7] http://msdn.microsoft.com/en-us/library/ms710252(VS.85).aspx

[8]Author Decker and Hirshfield, An Introduction to Programming using Java 2nd edition; QA76.73.J38D44; Brooks/Cole; CA; 1999

[9] http://en.wikipedia.org/wiki/Microsoft_.NET#Microsoft_.NET

[10] Ivor Horton, Title: *Ivor Horton's Beginning Java 2, JDK 5 Edition*. II. Title.QA76.73.J38H6758;

Wiley, the Wiley Publishing logo, Wrox, the Wrox logo, Programmer to P; 2004.

[11] http://www.mathworks.com/

[12] http://en.wikipedia.org/wiki/Unified_Modeling_Language

[13] http://en.wikipedia.org/wiki/.jar

[14] http://www.filesystems.org/~dquigley/cse219/index.php?it=netbeans&tt=jar&pf=y

## BIOGRAPHICAL INFORMATION

Ankit V. Master received his Bachelor of Engineering Degree, first class in Instrumentation engineering from the University of Mumbai, India. During his bachelors, he worked as an intern for one and a half year at the Indian Institute of Technology, Bombay (IITB), India in the Systems and Controls engineering department. After completing the bachelor's degree Ankit went ahead to pursue his Master's in Biomedical Engineering from the University of Texas at Arlington/ University of Texas Southwestern Medical Center, Dallas. During his Master's, Ankit pursued his thesis in developing two software tools for MRI image classification and image retrieval respectively, under the able guidance of Dr. Michael D. Devous Sr.