NAME: Ruman Ahmed Shaikh

USN: 1NH17CS112

PROJECT TITLE: AIHealth

1. INTRODUCTION

1.1 PROBLEM DEFINITION

Most of our health data are eating dust in hospital's record or databases. With the invention of advance data analytics tools like machine learning, we now have the ability of make sense of this data, find patters in it and even make prediction using it.

The patient can be given control of this data which will solve many privacy issues and machine learning can help them understand it in ways it has never been possible. This data can be used by doctors and researcher, with patient's permission of course to discover new patterns and drive medical research further into the future. Earlier a trained and experienced healthcare professional was needed to perform make sense of this data, but now with advance AI algorithms, we can capture their experience and knowledge to automate it.

DISCLAIMER AND CAUTION: This is not designed to replace any healthcare professional or supervise them. It is in its very early stages and therefore is prone to error. The system in the future can help the healthcare professionals reduce their strain. Just like any other technology AI is a tool and it depends on the user how it will be used, and like every other intelligent information processing system, it can and will make mistakes. This system is not a replacement but an example on how collaboration between humans and machines can make the world a better place.

India have one doctor for every 10,189 patients which is well below the recommended ratio of 1:1000 by World Health Organization. Diagnostic system like the one presented in this report can help reduce and save the very expensive time of the Indian doctors.

1.2. OBJECTIVES

The objectives of this project are as follows:

- Demonstrate the use of machine learning algorithm to predict certain diseases.
- Demonstrate an AI incorporated healthcare management system.
- Use graphical user interface and use object-oriented programming language implemented through java.
- Query from an SQL server for all kind of data.
- Use application program interface (API) to allow cross language software.

1.4. EXPECTED OUTCOMES

The expected outcomes of this project are as follows:

- To bring the power of AI in analyzing healthcare data to the tips of everyday person.
- A better and intelligent healthcare management system.
- Help patient better understand their health data ad get AI driven predictions from the system.
- Reduce the strain on the already thin stretched healthcare system by reducing doctor work of analysis.

1.5. HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

- Core i7 or above
- 8GB Random Access Memory
- Nvidia GTX 1050Ti or any other PASCAL architecture Graphical Processing Unit

Software Requirements:

- Windows 10 (Recommended)
- NetBeans for Easy GUI building (Recommended)
- Java with Swing framework
- MySQL Database server/client
- Python with sklearn, NumPy, etc.

2. OBJECT ORIENTED CONCEPTS

Object-oriented programming System

Object-oriented programming System (OOPs) is a programming paradigm based on the concept of “objects” that contain data and methods. The primary purpose of object- oriented programming is to increase the flexibility and maintainability of programs. Object oriented programming brings together data and its behaviour(methods) in a single location makes it easier to understand how a program works.

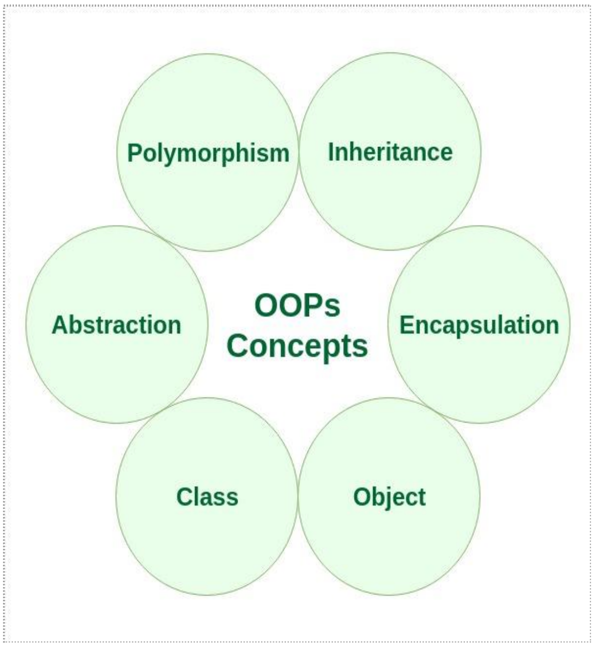
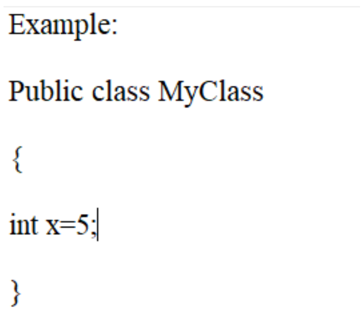


Fig 2.1 OOPs Concepts

2.1. CLASS

Class is the most fundamental and thus important concept of object-oriented programming paradigm. Classes are not actual runtime objects but are blueprints for what an object should contain and how it should behave.

In some way’s classes are a evolved version of structure in C programming language. Classes allow for more design consideration as it allows to incapsulate both data members (variables) and member methods (functions).



Example:

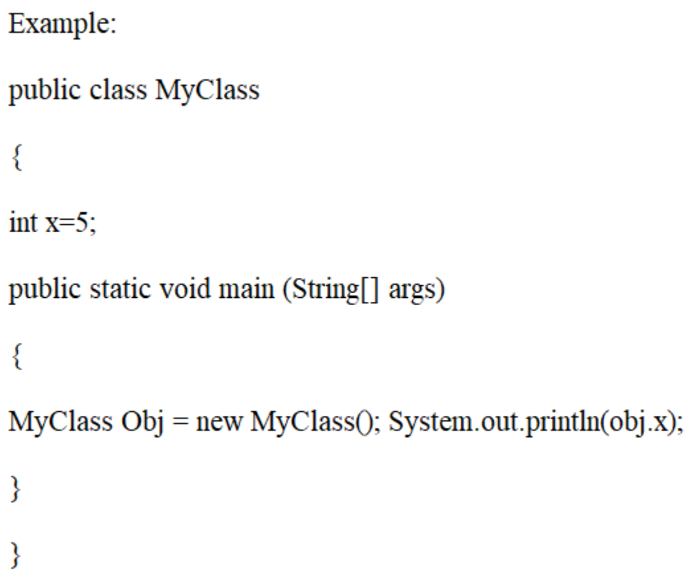
```
Public class MyClass

{

int x=5;

}
```

2.2 OBJECTS

Objects are the runtime instances of classes. These are ones that get created while the program is running. Their structure and behavior is defined by class.

In java and most other programming languages, an objects can access it data members and member methods by using the dot (.) operator.

For example; I am abject of class human while my dog would be an object of class dog.



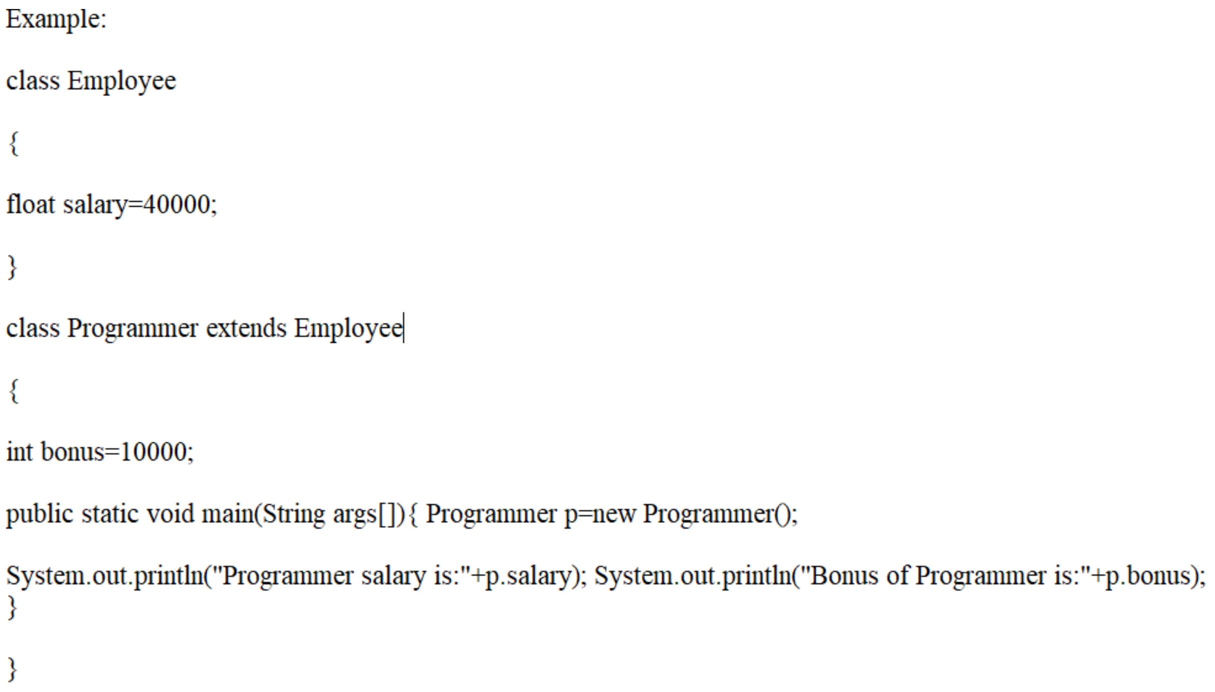
Example:

```
public class MyClass

{

int x=5;

public static void main (String[] args)

{

MyClass Obj = new MyClass(); System.out.println(obj.x);

}

}
```

2.3. INHERITANCE

Inheritance the principle by which a class inherits the properties of another class. The inheriting class is referred as subclass or child class while the inherited class from is referred as superclass or parent class.

For example; class doctor inherits from class person, thus doctor is the child or subclass and person is the parent class or superclass.



Example:

```
class Employee

{

float salary=40000;

}

class Programmer extends Employee

{

int bonus=10000;

public static void main(String args[]){ Programmer p=new Programmer();

System.out.println("Programmer salary is:"+p.salary); System.out.println("Bonus of Programmer is:"+p.bonus);
}

}
```

The different types of inheritance are:

- **Single Inheritance:** This type of inheritance allows a single subclass to inherit the properties of a parent class.

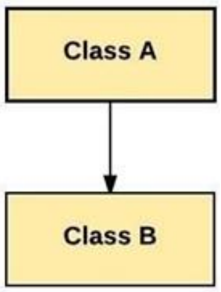


Fig 2.3.1 Single Inheritance

- **Multilevel Inheritance:** This type of inheritance allows a chain of inheritance where each subclass inherits the properties of the parent class and the grandparent class.

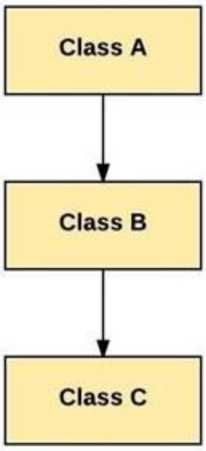


Fig 2.3.2 Multilevel Inheritance

- **Hierarchical Inheritance:** In this type of inheritance several subclasses inherit the properties of a single parent class.

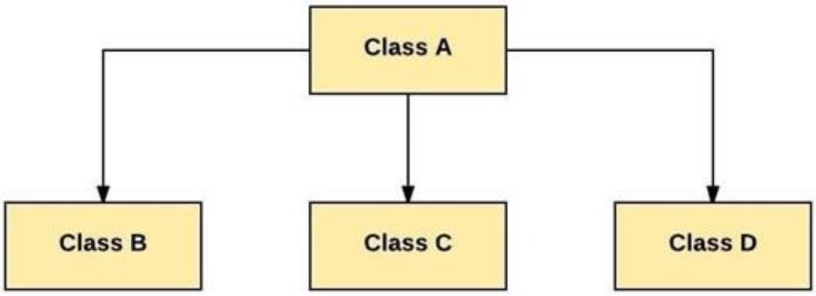
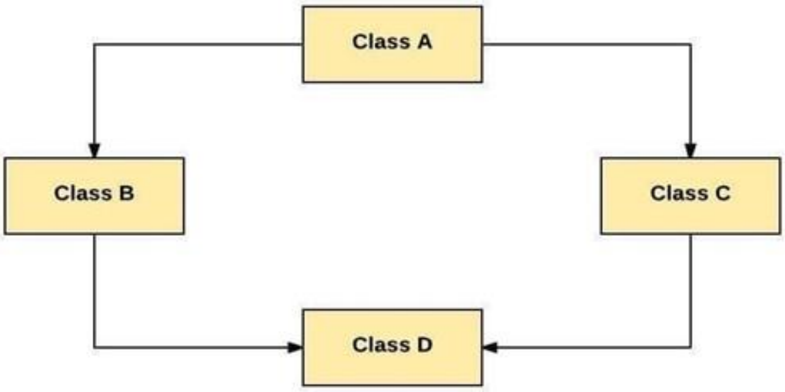


Fig 2.3.3 Hierarchical Inheritance

- **Hybrid inheritance:** This type of inheritance consists of different types of inheritances put together.



2.3.4 Hybrid Inheritance

2.3. POLYMORPHISM

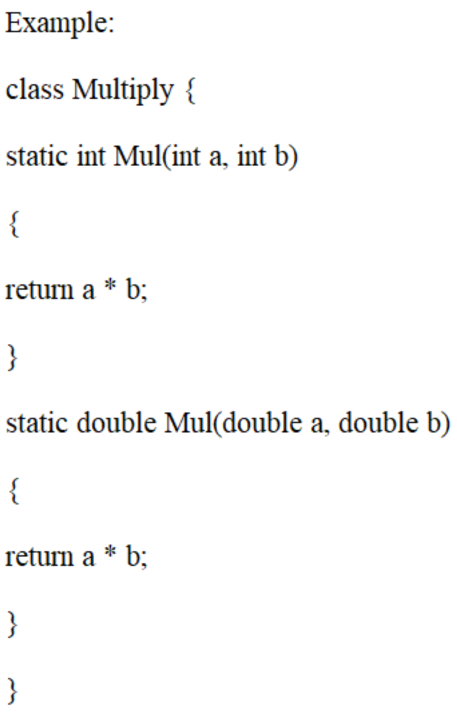
The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

There are two types of polymorphism:

- **Compile-time polymorphism:** The type of polymorphism which his achieved during operations such as method overloading is called compile-time polymorphism.

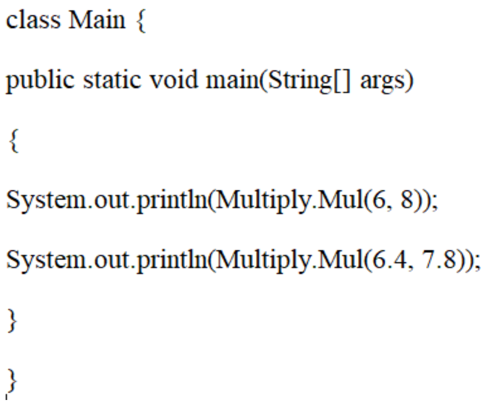
Method Overloading:

When there are multiple functions with same name but different parameters then these functions are said to be overloaded. Normally, functions can be overloaded either by change in number of arguments or by change in type of arguments.



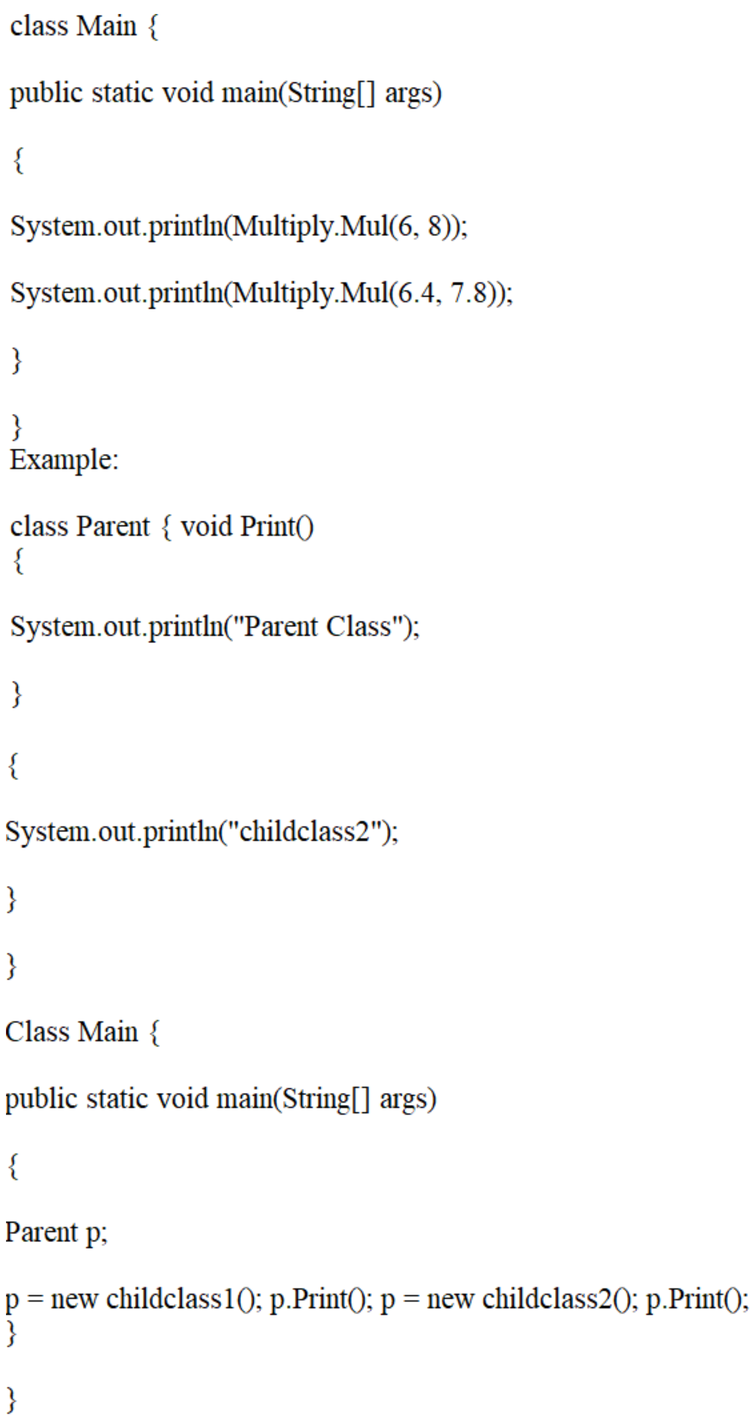
Example:

```
class Multiply {
static int Mul(int a, int b)
{
return a * b;
}
static double Mul(double a, double b)
{
return a * b;
}
}
```



```
class Main {
public static void main(String[] args)
{
System.out.println(Multiply.Mul(6, 8));
System.out.println(Multiply.Mul(6.4, 7.8));
}
}
```

- **Run-time polymorphism:** It is also known as Dynamic Method Dispatch. It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved during method overriding.

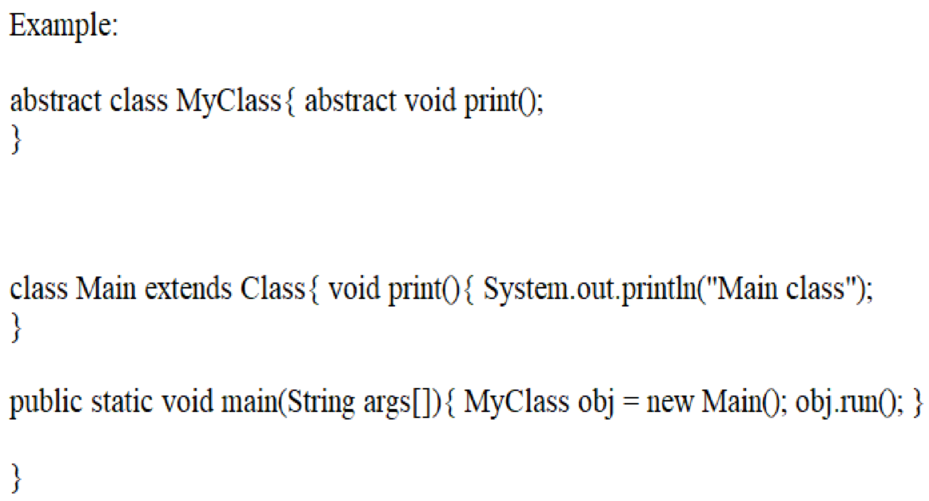


```
class Main {

public static void main(String[] args)

{

System.out.println(Multiply.Mul(6, 8));

System.out.println(Multiply.Mul(6.4, 7.8));

}

}
Example:

class Parent { void Print()
{

System.out.println("Parent Class");

}

{

System.out.println("childclass2");

}

}

Class Main {

public static void main(String[] args)

{

Parent p;

p = new childclass1(); p.Print(); p = new childclass2(); p.Print();
}

}
```

2.4 ABSTRACT CLASS

Abstraction is a process of hiding the implementation details and showing only functionality to the user. it shows only essential things to the user and hides the internal details. It can be achieved using two components: Abstract class and Interfaces.

Abstract class is a restricted class that cannot be used to create objects. In order to access an abstract class, it must be inherited from another class. It can have abstract and non- abstract methods. It needs to be extended. It cannot be instantiated.



Example:

```
abstract class MyClass{ abstract void print();
}



class Main extends Class{ void print(){ System.out.println("Main class");
}

public static void main(String args[]){ MyClass obj = new Main(); obj.run(); }

}
```

2.5 MULTITHREADING

A thread is a light-weight smallest part of a process that can run concurrently with the other parts(other threads) of the same process.

Java has a feature known as multithreading that allows concurrent execution of different parts of a program, this allows for maximum CPU utilization.

Two methods to implement multithreading are:

- By extending the Thread class
- By implementing the Runnable interface

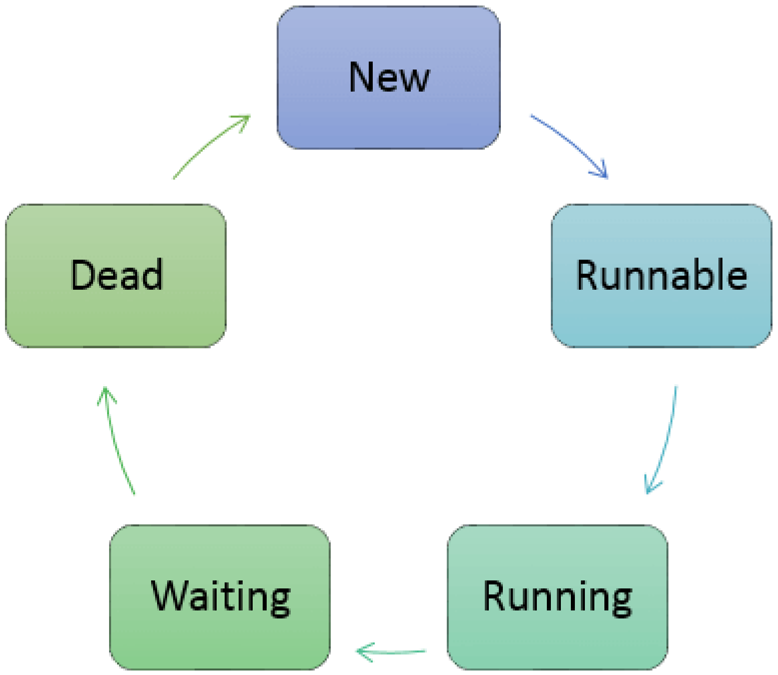


Fig 2.2 Thread cycle

2.6 I/O FUNCTIONS

Java has various I/O streams included with its I/O package the three main streams are as follows: -

System.in: System.in is used to read in data from any standard input device. Generally done using a java scanner.

System.out: System.out is used to show the result of a particular executed program on a standard output device. Some of the System.out functions: -

1. **print() : Prints data passed as an argument onto the console screen**
2. println() : Does the same function as print but moves cursor to the next line.

System.err: The standard error stream used to output the data that is thrown as an error by the program on a standard output device.

2.7 JAVA PACKAGES

A java package is a group of similar types of classes, interfaces and sub-packages. Package in java can be categorized in two forms, built-in package and user-defined package. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc. User-defned package is a package which is created by the user. It contains various methods, variables, etc. which have been added by the user for his purpose.

Example:

```
Package  mypack;
 public class Main
{
     public static void main(String[] args) {
   System.out.println("Welcome!");
}}
```

2.8 EXCEPTION HANDLING

An exception is an even that disrupts the flow of a program and is thrown at runtime.

The key words used to handle exceptions in java are as follows: -

1. try :This keyword is used to specify the code where an exception block is typed out.
2. catch :This keyword is used to handle the exceptions and precedes the try block.
3. throw :This keyword is used to throw an exception.
4. throws :It specifies that an exception may occur in a method.
5. finally :This keyword is used to execute a block irrespective of whether an exception is handled.

3. DESIGN

3.1. DESIGN GOALS

The software is going to use a client server architecture. The front-end of the system is written in java and the graphical user interface is to build using Swing framework in java. The system is using a database for storing and retrieving data that exits on a server which the client software connects for data driven transaction.

In order for data analysis, the system need to utilize python's very powerful machine learning and deep learning frameworks like sklearn and TensorFlow. This can be done on the client machine but it may be slow as it requires a substantial amount of compute. On top of that certain client system won't be able to run deep learning framework like TensorFlow.

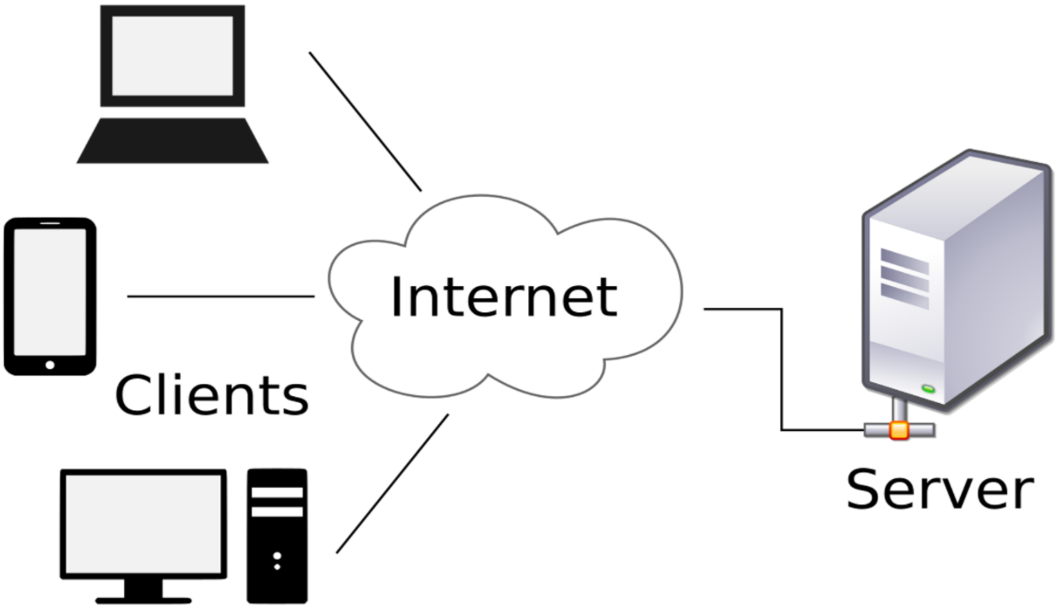
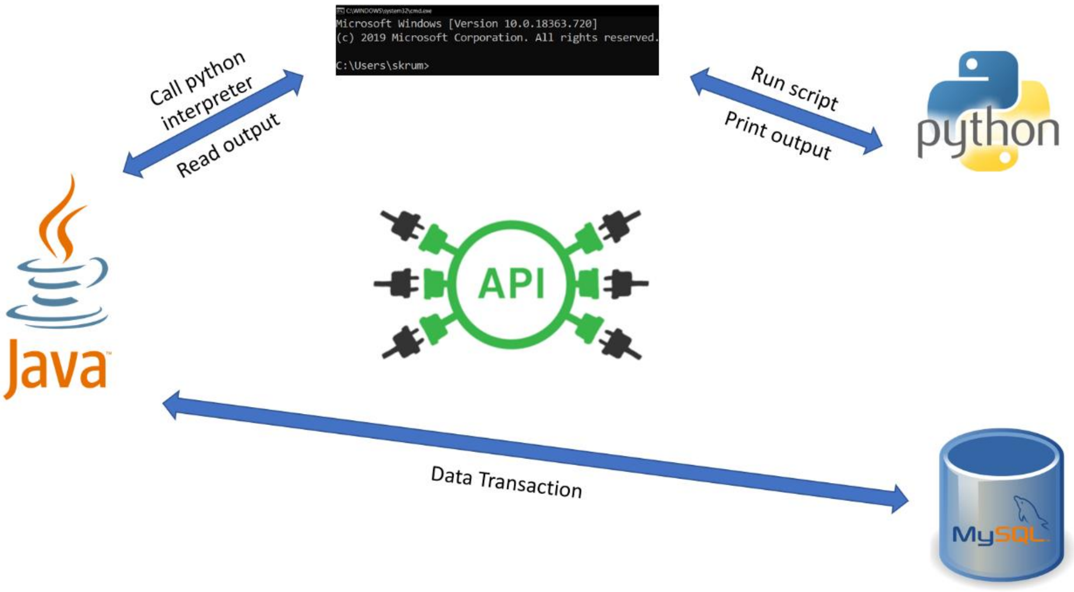


Fig 3.1.1

The above-mentioned design is implemented in one computer but the design of the software is such that, it will require very less effort to scale it to network level. This is achieved by using tailored API that talk between the java code and the python code and using MySQL software's local host feature.

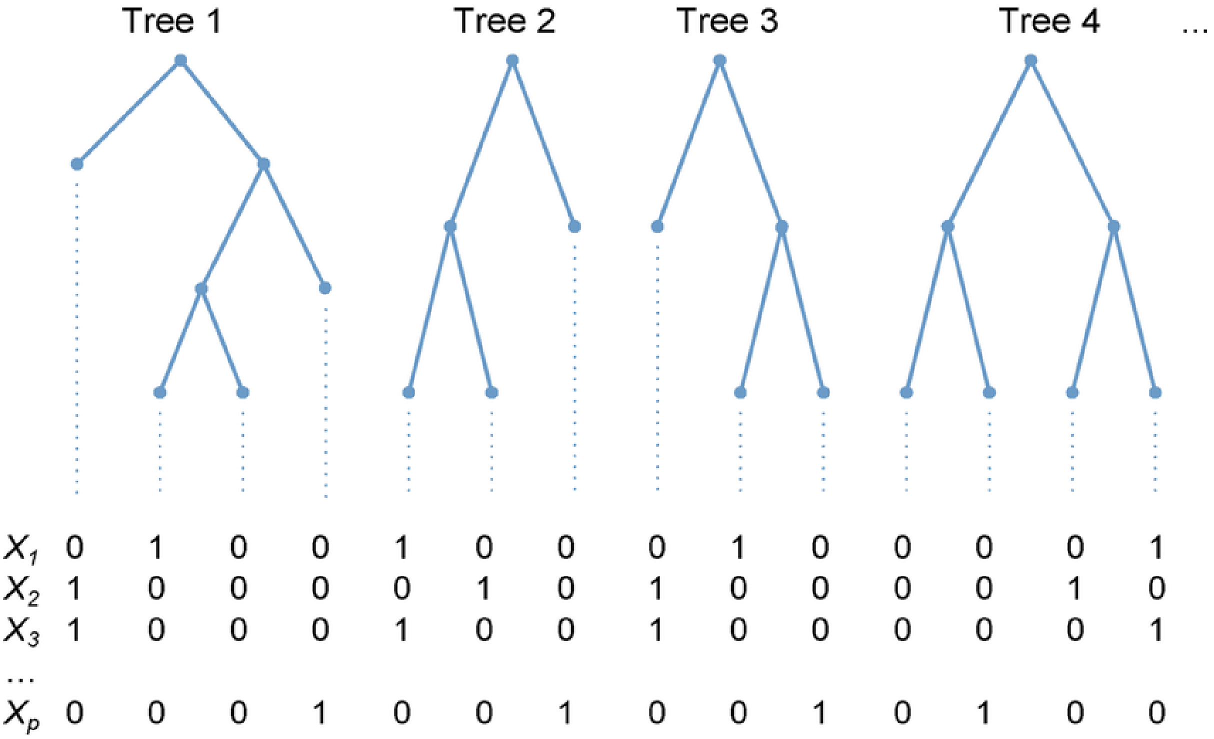


```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\skrum>
```

Fig 3.1.2

3.2. ALGORITHM/PSEUDOCODE

The algorithm used for analysis of the medical data to predict weather or not the patient have chronic kidney disease, is known as the random forest algorithm.



X_1	0	1	0	0	1	0	0	0	1	0	0	0	0	1
X_2	1	0	0	0	0	1	0	1	0	0	0	0	1	0
X_3	1	0	0	0	1	0	0	1	0	0	0	0	0	1
...														
X_p	0	0	0	1	0	0	1	0	0	1	0	1	0	0

Fig 3.2.1

The random forest classifier uses multiple decision trees to classify. Then it sums ups the output and the class label with the maximum votes is given as the final answer.

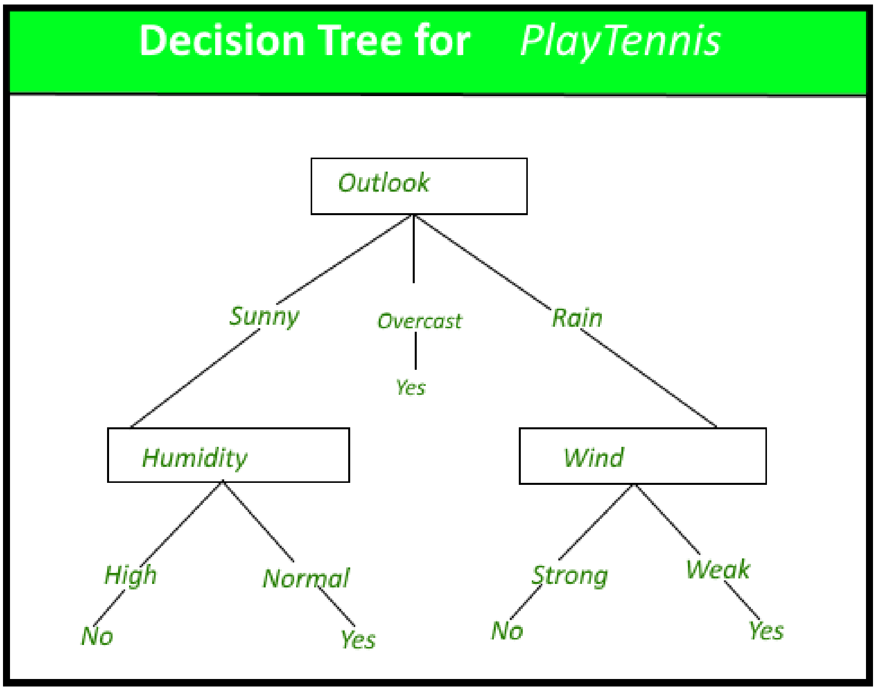


Fig 3.2.2

The above is an example of a single simple decision tree to help decide a system weather the condition is suitable for playing tennis or not. In case of healthcare data or any real world scenario for that matter, the tree is much for complex.

The decision tree algorithm repeatedly checks for every feature to see if classifying the data based on it will reduce entropy (degree of disorder). It performs this calculation for every feature. Once it settles on one feature to classify with, it does the same thing on the all the nodes recursively.

4. IMPLEMENATION

4.1. MODULE 1 Basic Design

The system is based on server client architecture. The client side is running the java code and the MySQL server and the python interpreter acts as the server. The first java file that is the Main.java file which on appropriate user event call the required java file. All this java files are contained in the same aihealth package, which allows any file to access any function or class with the public or protected access specifier.

4.2. MODULE 2 Swing Framework

The Swing java framework allows us to implements GUI in our java programs rather than relying on simple command line input. The Swing framework contains a number of useful GUI components, among which many of them has been used in this project.

Unlike AWT (Abstract Window Toolkit), which is platform dependent, swing is often called lightweight as they do not require a lot of allocation in OS's windowing toolkit.

Some of the important Swing class has been shown below:

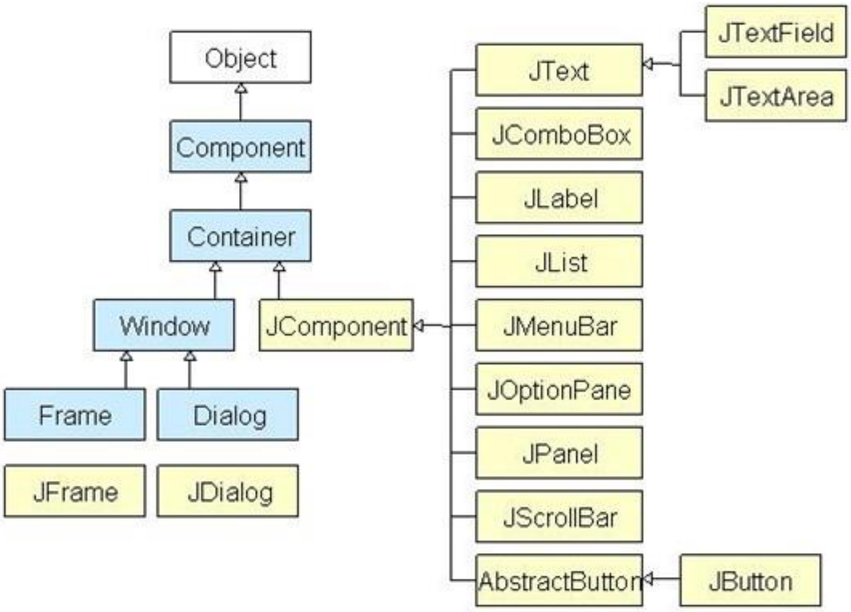


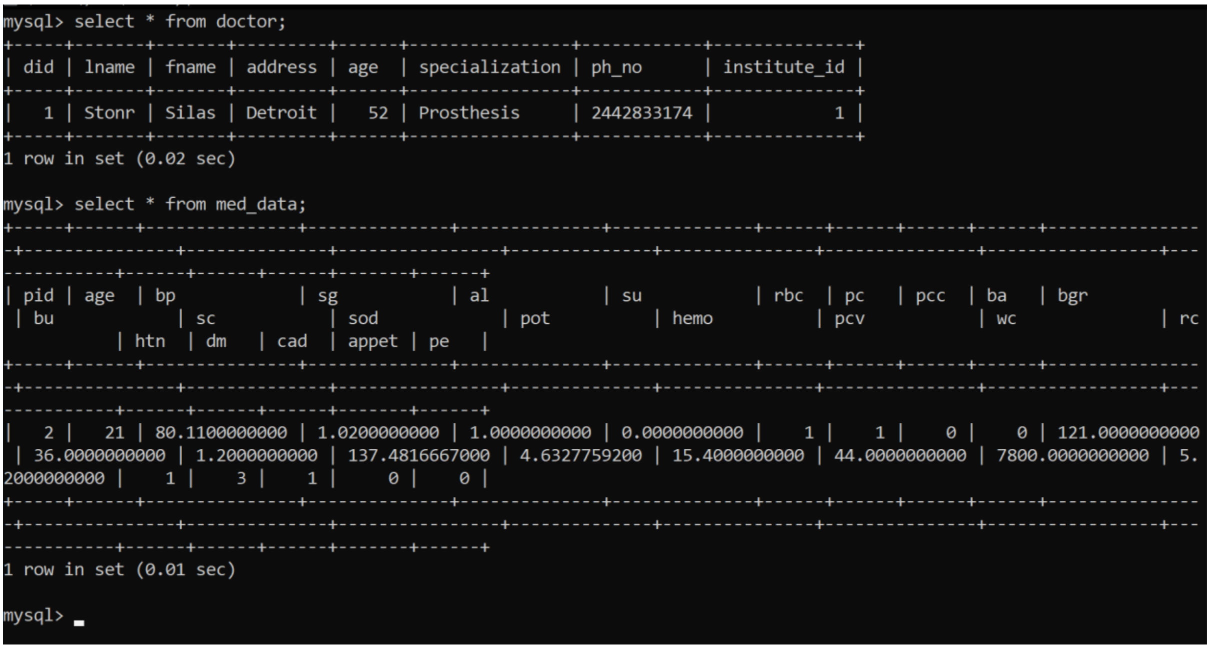
Fig 4.2.1 Java Swing Hierarchy

4.3. MODULE 3 Database

The database software used in this project is the MySQL software which allows the user to create a local server and client in the same system. The jconnector allows the java code to talk to the MySQL database, which means any CRUD queries from the java code can be run on the SQL client. A structured database is used to store the details such as login information, and sign-up information. MySQL is the database management system used for the implementation of these tables of data.

The login information consists of the user name and password fields which are retrieved during login to check user authentication.

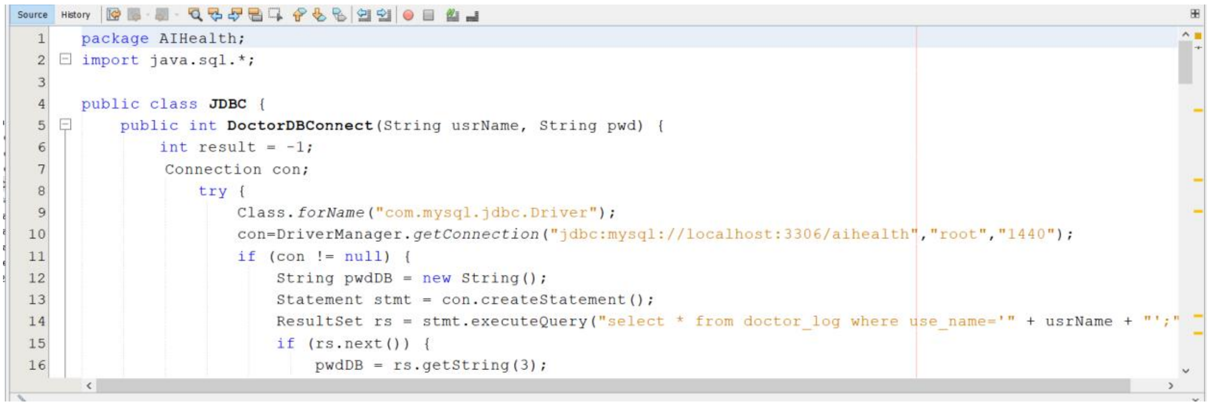
The signup information contains the username, password ,occupation, age and state which are retrieved in the database named database and therefore inside the database a table named account is created where all the login and sign-up information is stored.



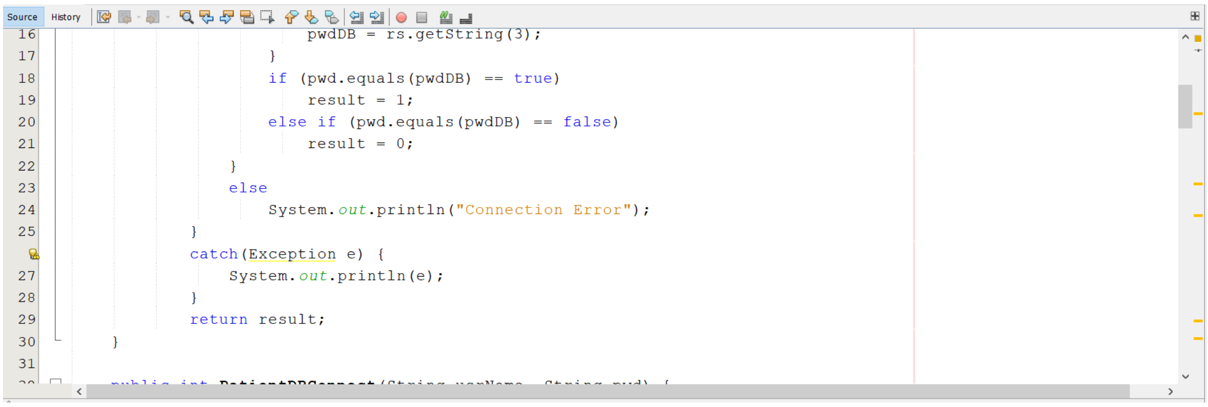
```
mysql> select * from doctor;
+-----+-------+-------+---------+------+----------------+------------+--------------+
| did | lname | fname | address | age  | specialization | ph_no      | institute_id |
+-----+-------+-------+---------+------+----------------+------------+--------------+
|   1 | Stonr | Silas | Detroit |   52 | Prosthesis     | 2442833174 |            1 |
+-----+-------+-------+---------+------+----------------+------------+--------------+
1 row in set (0.02 sec)

mysql> select * from med_data;
+-----+------+---------------+--------------+--------------+--------------+------+------+------+------+---------------
-+---------------+--------------+----------------+--------------+---------------+---------------+-----------------+---
-----------+------+------+------+-------+------+
| pid | age  | bp            | sg           | al           | su           | rbc  | pc   | pcc  | ba   | bgr
 | bu            | sc           | sod            | pot          | hemo          | pcv           | wc              | rc
           | htn  | dm   | cad  | appet | pe   |
+-----+------+---------------+--------------+--------------+--------------+------+------+------+------+---------------
-+---------------+--------------+----------------+--------------+---------------+---------------+-----------------+---
-----------+------+------+------+-------+------+
|   2 |   21 | 80.1100000000 | 1.0200000000 | 1.0000000000 | 0.0000000000 |    1 |    1 |    0 |    0 | 121.0000000000
 | 36.0000000000 | 1.2000000000 | 137.4816667000 | 4.6327759200 | 15.4000000000 | 44.0000000000 | 7800.0000000000 | 5.
2000000000 |    1 |    3 |    1 |     0 |    0 |
+-----+------+---------------+--------------+--------------+--------------+------+------+------+------+---------------
-+---------------+--------------+----------------+--------------+---------------+---------------+-----------------+---
-----------+------+------+------+-------+------+
1 row in set (0.01 sec)

mysql> _
```

Fig 4.3.1 Database Snapshot

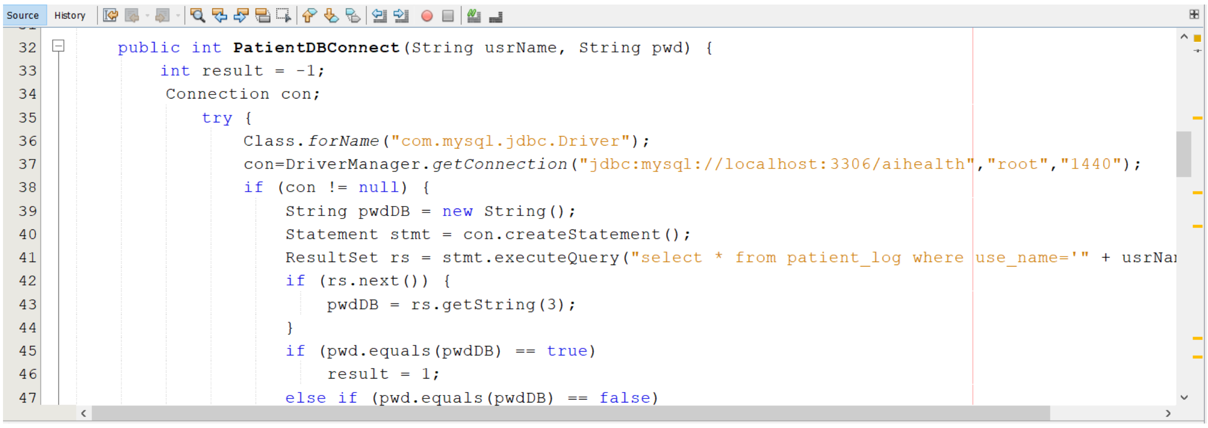
4.4. MODULE 4 Java Database Connection



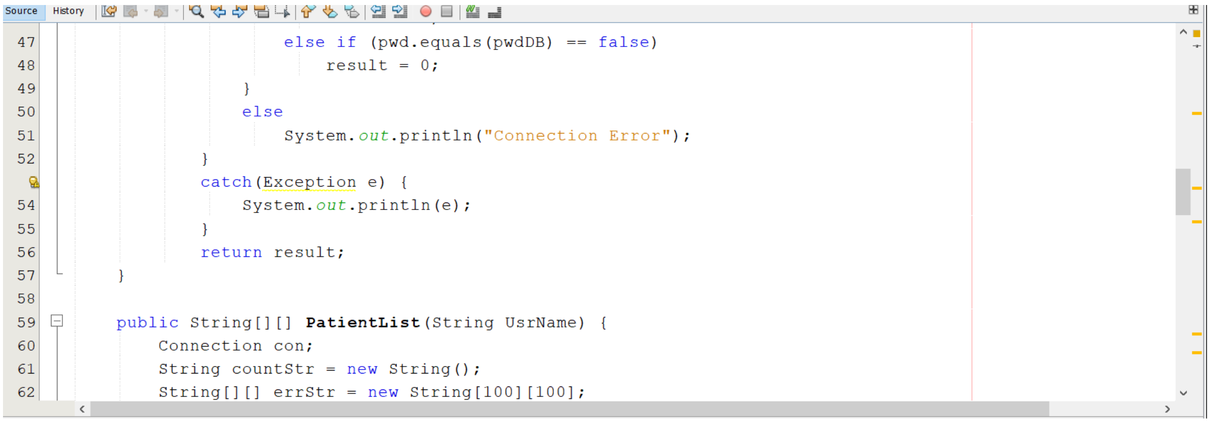
```
package AIHealth;
import java.sql.*;

public class JDBC {
    public int DoctorDBConnect(String usrName, String pwd) {
        int result = -1;
        Connection con;
            try {
                Class.forName("com.mysql.jdbc.Driver");
                con=DriverManager.getConnection("jdbc:mysql://localhost:3306/aihealth","root","1440");
                if (con != null) {
                    String pwdDB = new String();
                    Statement stmt = con.createStatement();
                    ResultSet rs = stmt.executeQuery("select * from doctor_log where use_name='" + usrName + "';"
                    if (rs.next()) {
                        pwdDB = rs.getString(3);
```



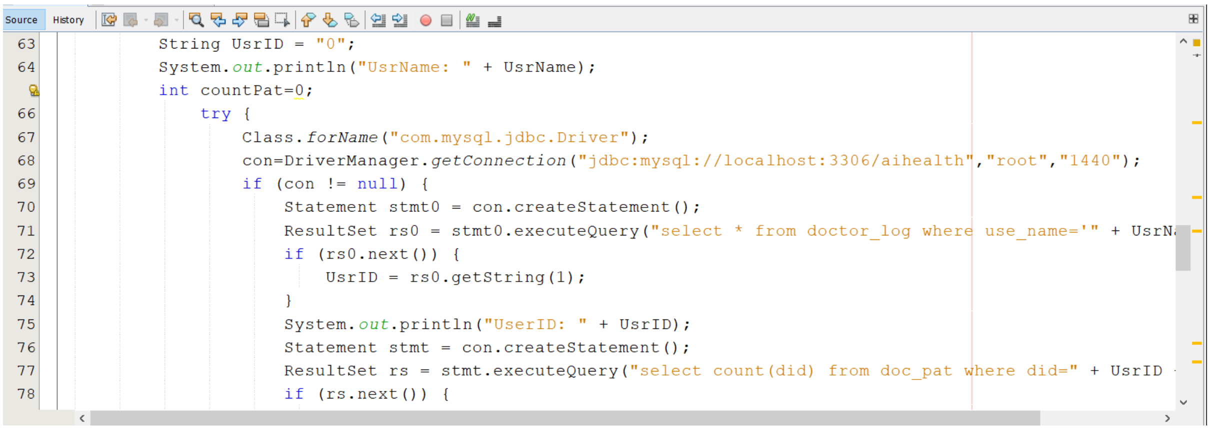
```
                        pwdDB = rs.getString(3);
                    }
                    if (pwd.equals(pwdDB) == true)
                        result = 1;
                    else if (pwd.equals(pwdDB) == false)
                        result = 0;
                }
                else
                    System.out.println("Connection Error");
            }
            catch(Exception e) {
                System.out.println(e);
            }
            return result;
    }

```



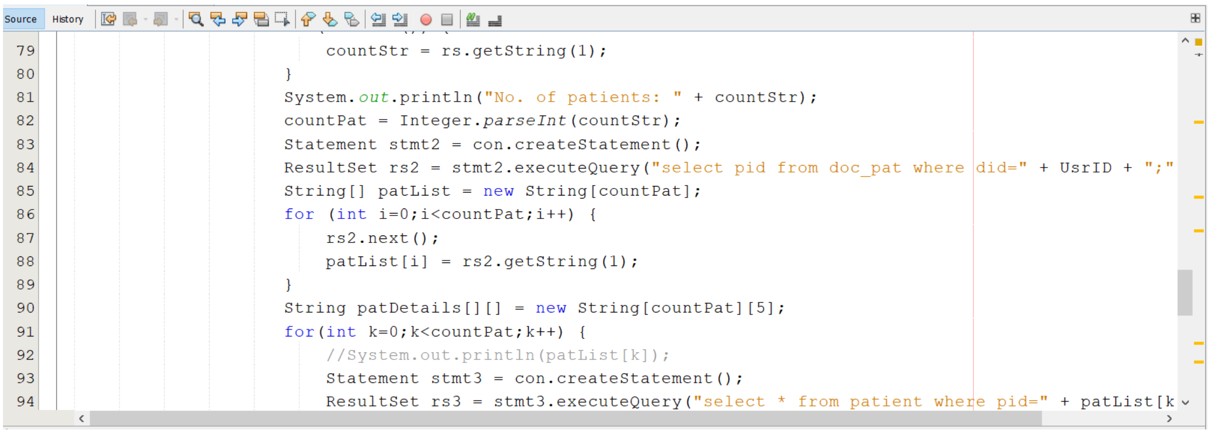
```
    public int PatientDBConnect(String usrName, String pwd) {
        int result = -1;
        Connection con;
            try {
                Class.forName("com.mysql.jdbc.Driver");
                con=DriverManager.getConnection("jdbc:mysql://localhost:3306/aihealth","root","1440");
                if (con != null) {
                    String pwdDB = new String();
                    Statement stmt = con.createStatement();
                    ResultSet rs = stmt.executeQuery("select * from patient_log where use_name='" + usrNa
                    if (rs.next()) {
                        pwdDB = rs.getString(3);
                    }
                    if (pwd.equals(pwdDB) == true)
                        result = 1;
                    else if (pwd.equals(pwdDB) == false)
```



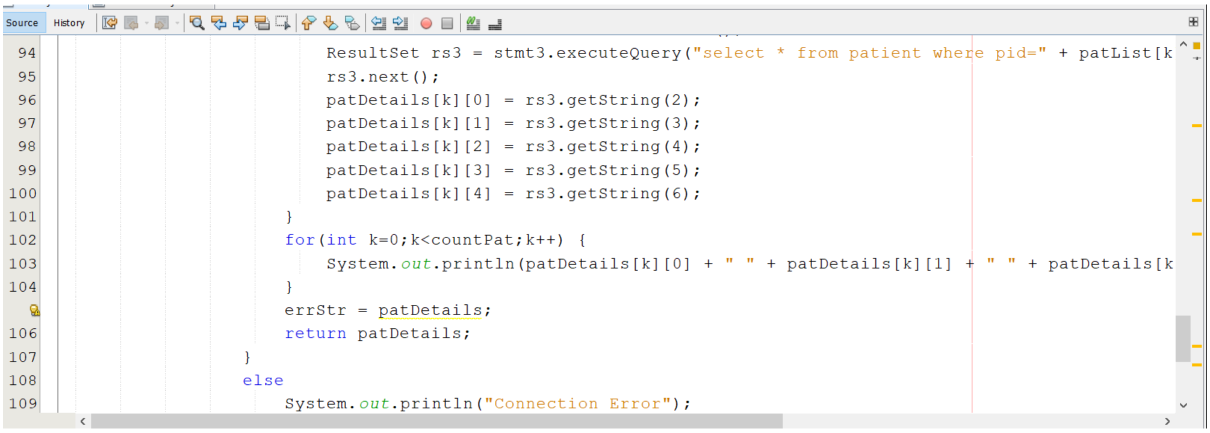
```
                    else if (pwd.equals(pwdDB) == false)
                        result = 0;
                }
                else
                    System.out.println("Connection Error");
            }
            catch(Exception e) {
                System.out.println(e);
            }
            return result;
    }

    public String[][] PatientList(String UsrName) {
        Connection con;
        String countStr = new String();
        String[][] errStr = new String[100][100];
```



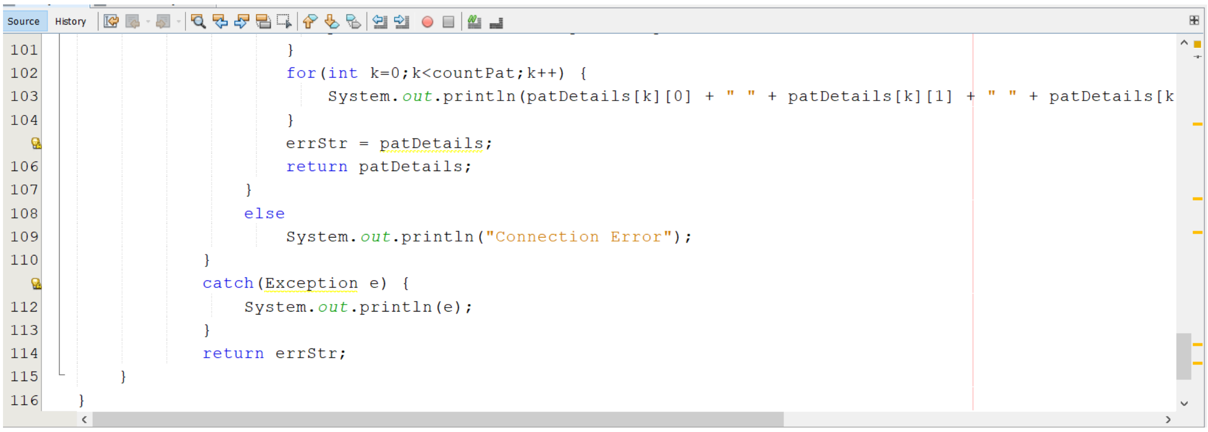
```
            String UsrID = "0";
            System.out.println("UsrName: " + UsrName);
            int countPat=0;
                try {
                    Class.forName("com.mysql.jdbc.Driver");
                    con=DriverManager.getConnection("jdbc:mysql://localhost:3306/aihealth","root","1440");
                    if (con != null) {
                        Statement stmt0 = con.createStatement();
                        ResultSet rs0 = stmt0.executeQuery("select * from doctor_log where use_name='" + UsrN
                        if (rs0.next()) {
                            UsrID = rs0.getString(1);
                        }
                        System.out.println("UserID: " + UsrID);
                        Statement stmt = con.createStatement();
                        ResultSet rs = stmt.executeQuery("select count(did) from doc_pat where did=" + UsrID
                        if (rs.next()) {
```



```
                            countStr = rs.getString(1);
                        }
                        System.out.println("No. of patients: " + countStr);
                        countPat = Integer.parseInt(countStr);
                        Statement stmt2 = con.createStatement();
                        ResultSet rs2 = stmt2.executeQuery("select pid from doc_pat where did=" + UsrID + ";"
                        String[] patList = new String[countPat];
                        for (int i=0;i<countPat;i++) {
                            rs2.next();
                            patList[i] = rs2.getString(1);
                        }
                        String patDetails[][] = new String[countPat][5];
                        for(int k=0;k<countPat;k++) {
                            //System.out.println(patList[k]);
                            Statement stmt3 = con.createStatement();
                            ResultSet rs3 = stmt3.executeQuery("select * from patient where pid=" + patList[k
```



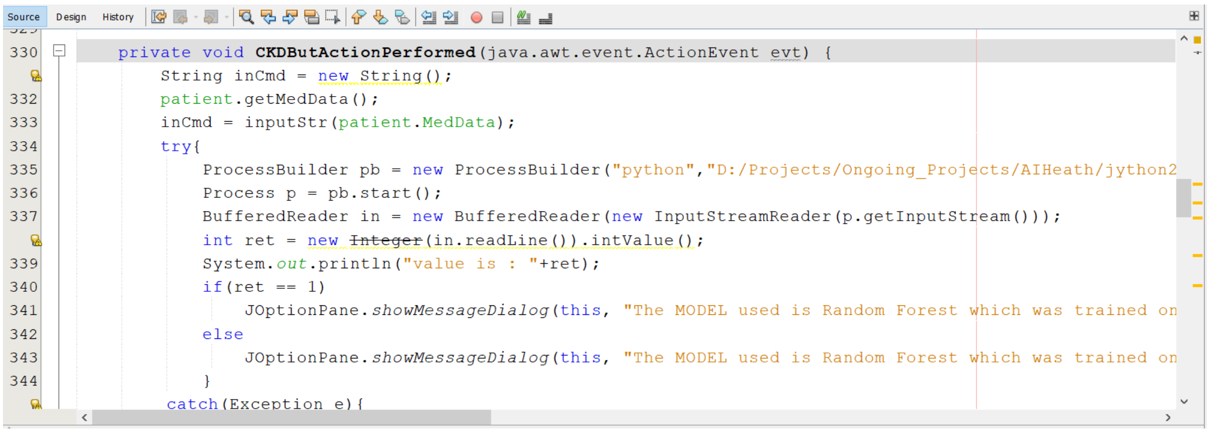
```
                            ResultSet rs3 = stmt3.executeQuery("select * from patient where pid=" + patList[k
                            rs3.next();
                            patDetails[k][0] = rs3.getString(2);
                            patDetails[k][1] = rs3.getString(3);
                            patDetails[k][2] = rs3.getString(4);
                            patDetails[k][3] = rs3.getString(5);
                            patDetails[k][4] = rs3.getString(6);
                        }
                        for(int k=0;k<countPat;k++) {
                            System.out.println(patDetails[k][0] + " " + patDetails[k][1] + " " + patDetails[k
                        }
                        errStr = patDetails;
                        return patDetails;
                    }
                    else
                        System.out.println("Connection Error");
```



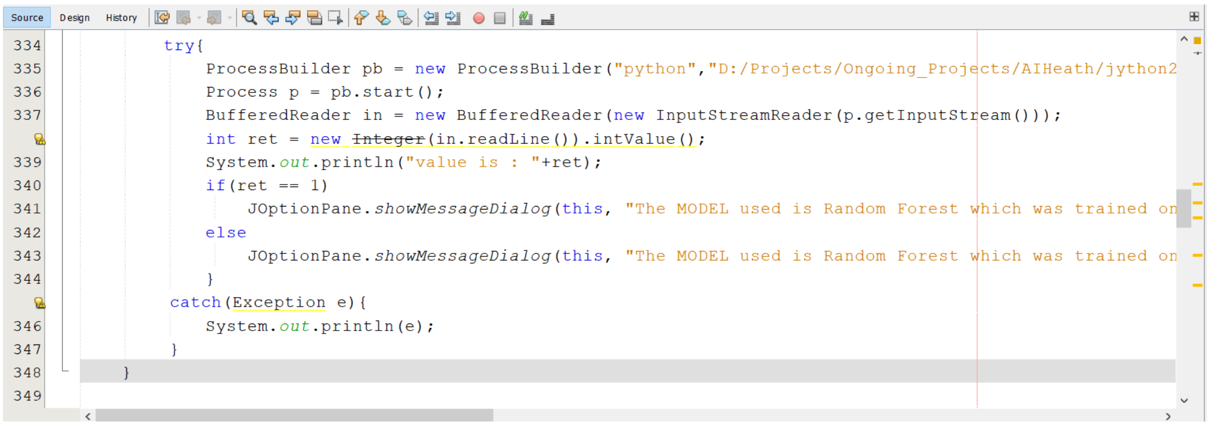
```
                        }
                        for(int k=0;k<countPat;k++) {
                            System.out.println(patDetails[k][0] + " " + patDetails[k][1] + " " + patDetails[k
                        }
                        errStr = patDetails;
                        return patDetails;
                    }
                    else
                        System.out.println("Connection Error");
                }
                catch(Exception e) {
                    System.out.println(e);
                }
                return errStr;
        }
    }
```

4.5. MODULE 5 Python API

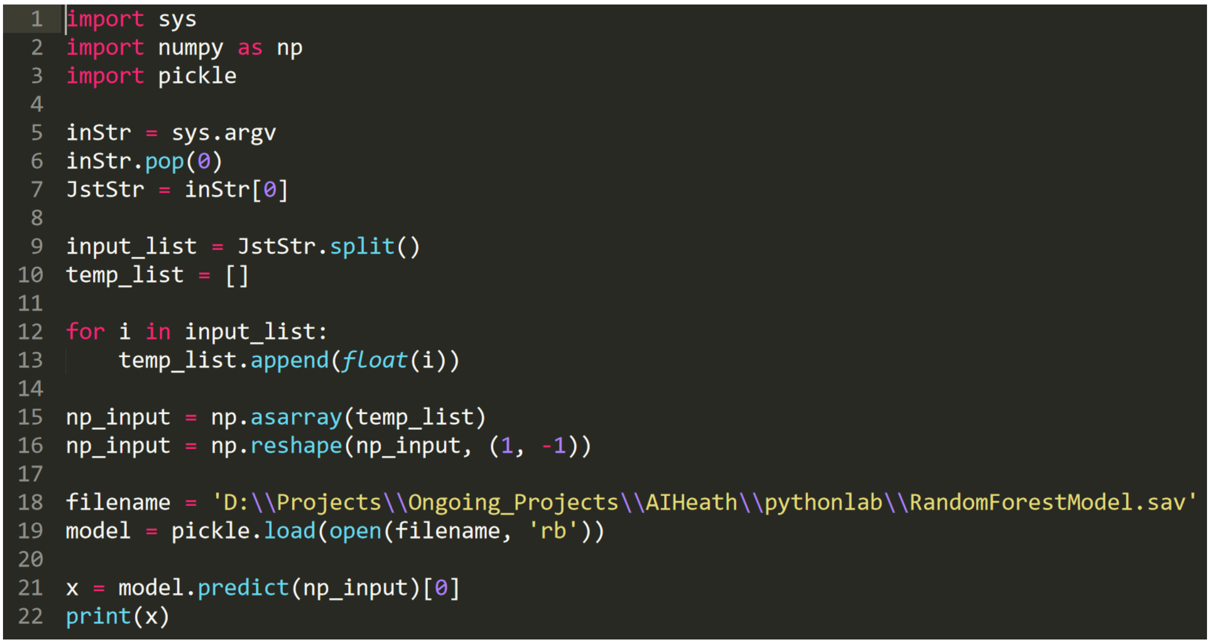
The machine learning library sklearn in not available in java, and implementing an entire or a part of an advance machine learning library is a project on it own. This make calling the python code and passing the user input from java code is much easier. This can be implemented in a number of ways, like creating a file which both code can access.



```
    private void CKDButActionPerformed(java.awt.event.ActionEvent evt) {
        String inCmd = new String();
        patient.getMedData();
        inCmd = inputStr(patient.MedData);
        try{
            ProcessBuilder pb = new ProcessBuilder("python","D:/Projects/Ongoing_Projects/AIHeath/jython2
            Process p = pb.start();
            BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()));
            int ret = new Integer(in.readLine()).intValue();
            System.out.println("value is : "+ret);
            if(ret == 1)
                JOptionPane.showMessageDialog(this, "The MODEL used is Random Forest which was trained on
            else
                JOptionPane.showMessageDialog(this, "The MODEL used is Random Forest which was trained on
            }
        catch(Exception e){
```



```
        try{
            ProcessBuilder pb = new ProcessBuilder("python","D:/Projects/Ongoing_Projects/AIHeath/jython2
            Process p = pb.start();
            BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()));
            int ret = new Integer(in.readLine()).intValue();
            System.out.println("value is : "+ret);
            if(ret == 1)
                JOptionPane.showMessageDialog(this, "The MODEL used is Random Forest which was trained on
            else
                JOptionPane.showMessageDialog(this, "The MODEL used is Random Forest which was trained on
            }
        catch(Exception e){
            System.out.println(e);
        }
    }
```

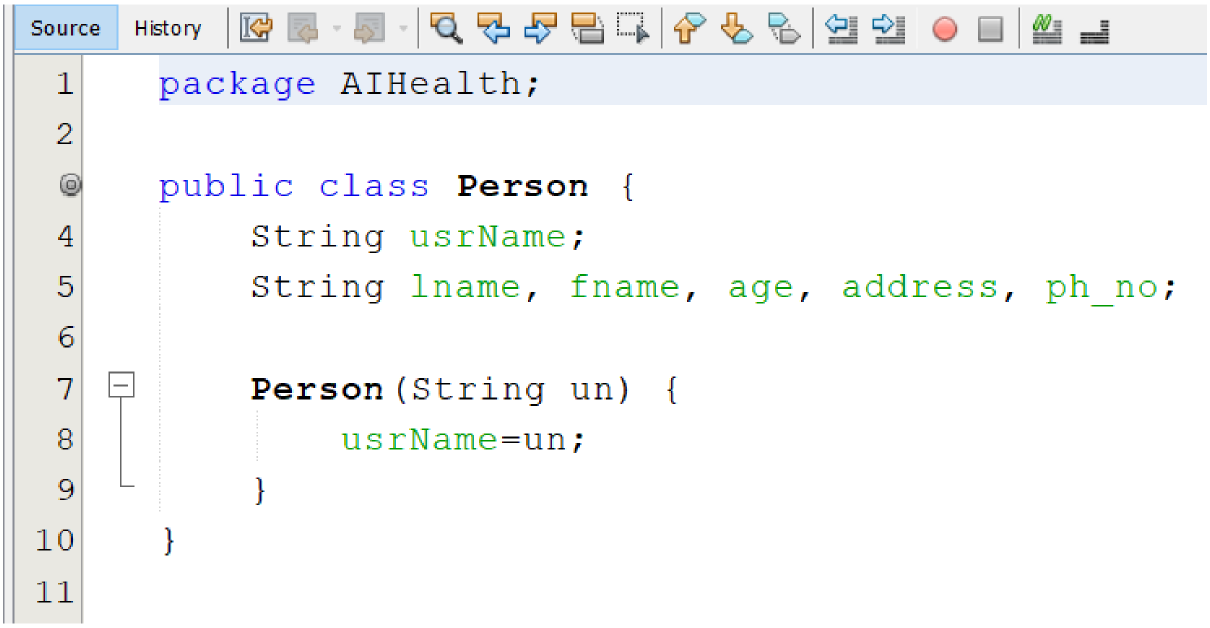


```
import sys
import numpy as np
import pickle

inStr = sys.argv
inStr.pop(0)
JstStr = inStr[0]

input_list = JstStr.split()
temp_list = []

for i in input_list:
    temp_list.append(float(i))

np_input = np.asarray(temp_list)
np_input = np.reshape(np_input, (1, -1))

filename = 'D:\\Projects\\Ongoing_Projects\\AIHeath\\pythonlab\\RandomForestModel.sav'
model = pickle.load(open(filename, 'rb'))

x = model.predict(np_input)[0]
print(x)
```

5. RESULTS

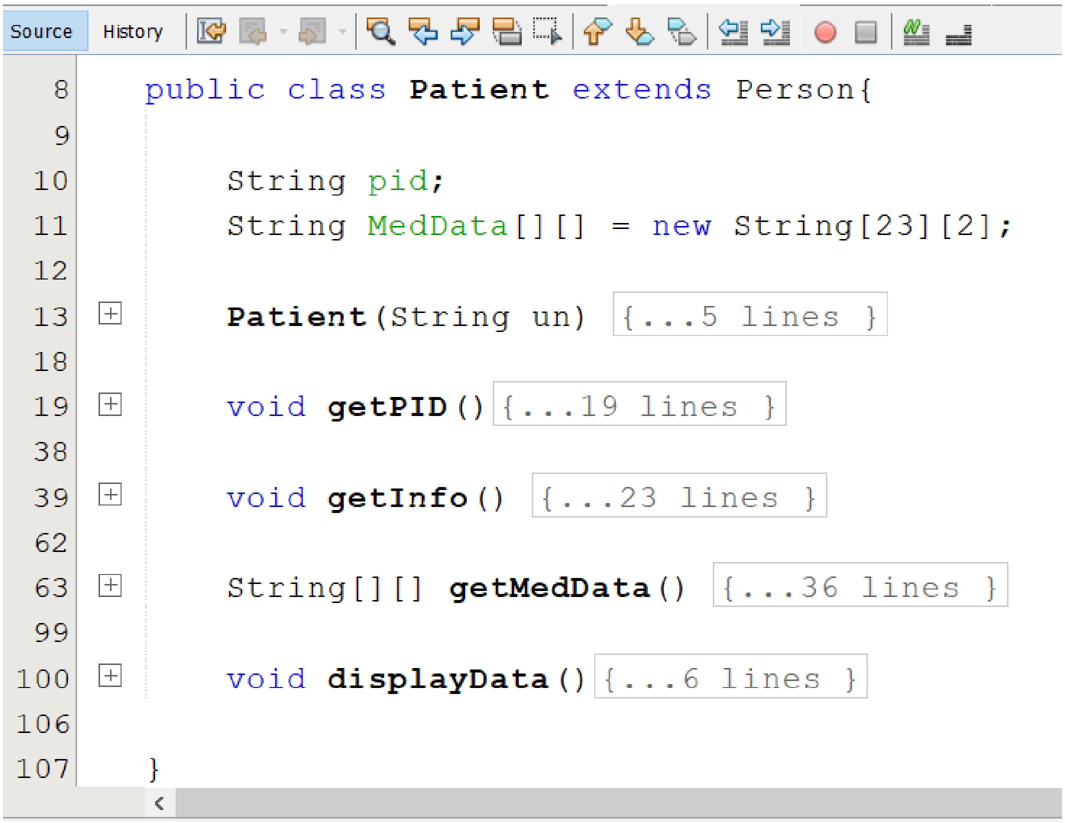
5.1. RESULT OF INHERITANCE

The snapshot of the person code is show below:

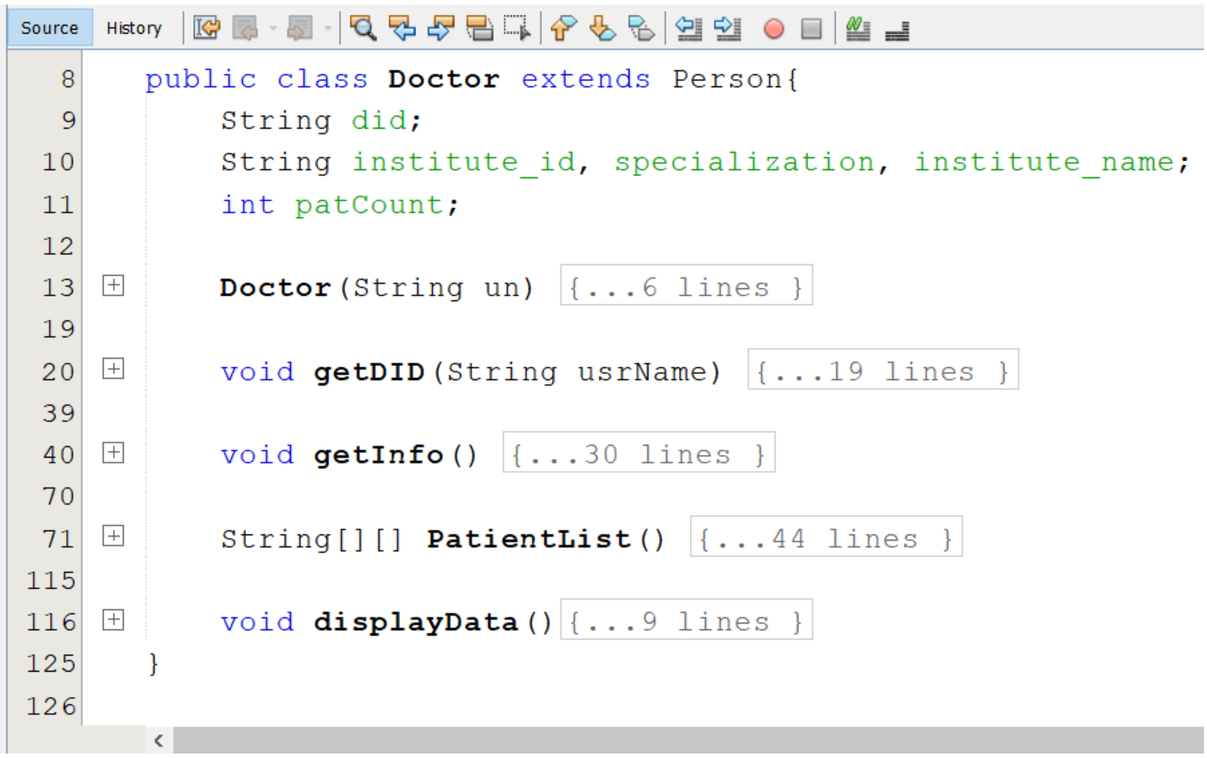


```java
package AIHealth;

public class Person {
    String usrName;
    String lname, fname, age, address, ph_no;

    Person(String un) {
        usrName=un;
    }
}

```

Both patient class and doctor class inherited from the person class as show below:



```java
public class Patient extends Person{

    String pid;
    String MedData[][] = new String[23][2];

    Patient(String un) {...5 lines }

    void getPID(){...19 lines }

    void getInfo() {...23 lines }

    String[][] getMedData() {...36 lines }

    void displayData(){...6 lines }

}
```



```
public class Doctor extends Person{
    String did;
    String institute_id, specialization, institute_name;
    int patCount;

    Doctor(String un) {...6 lines }

    void getDID(String usrName) {...19 lines }

    void getInfo() {...30 lines }

    String[][] PatientList() {...44 lines }

    void displayData() {...9 lines }
}
```

The function are abstracted away to keep the code snapshot clean.

5.2. RESULT VERIFICATION OF FRONTEND AND BACKEND

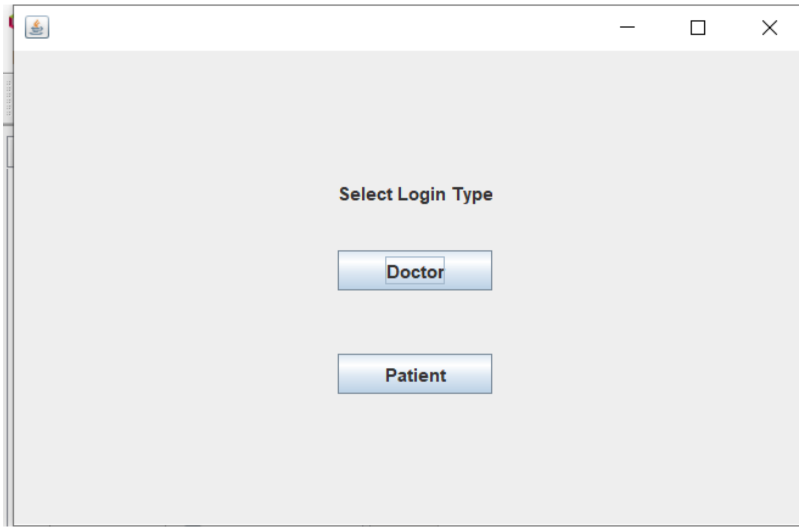
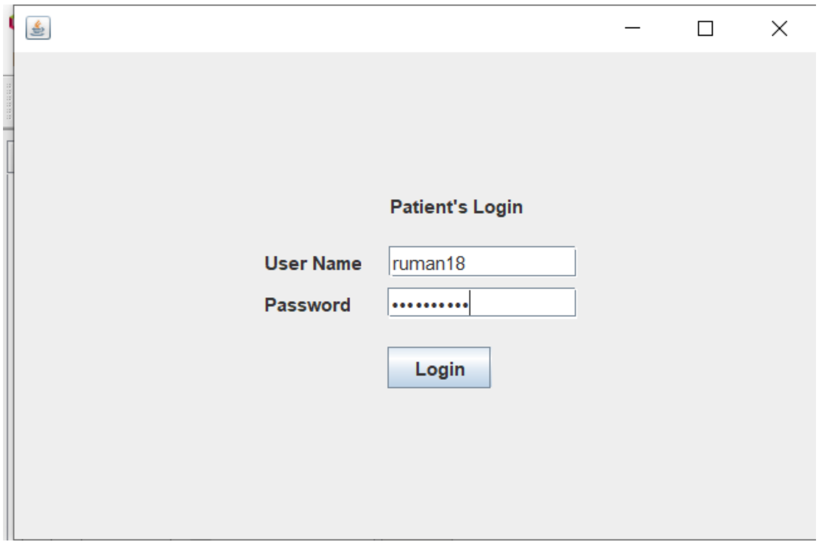


Fig   Login Type

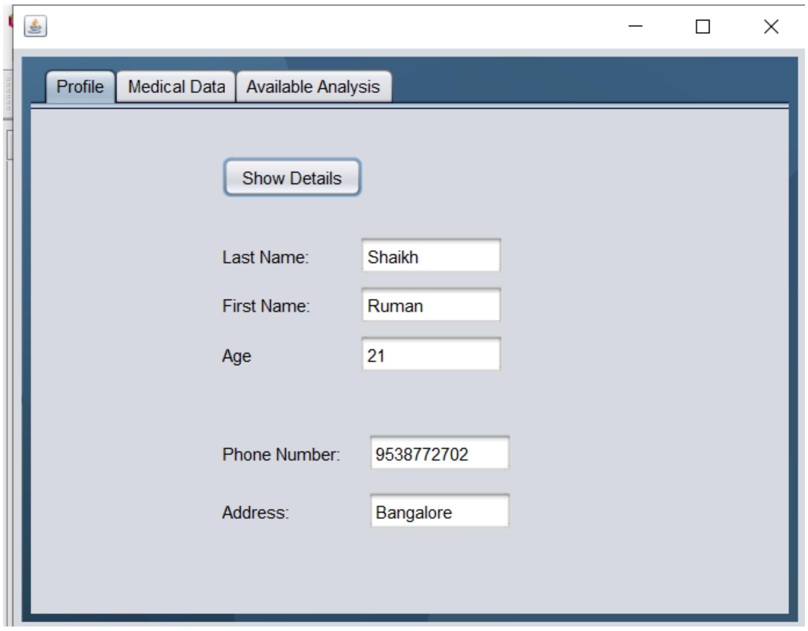


Patient's Login

User Name: ruman18

Password: ••••••••••

Fig   Login Window



Last Name: Shaikh

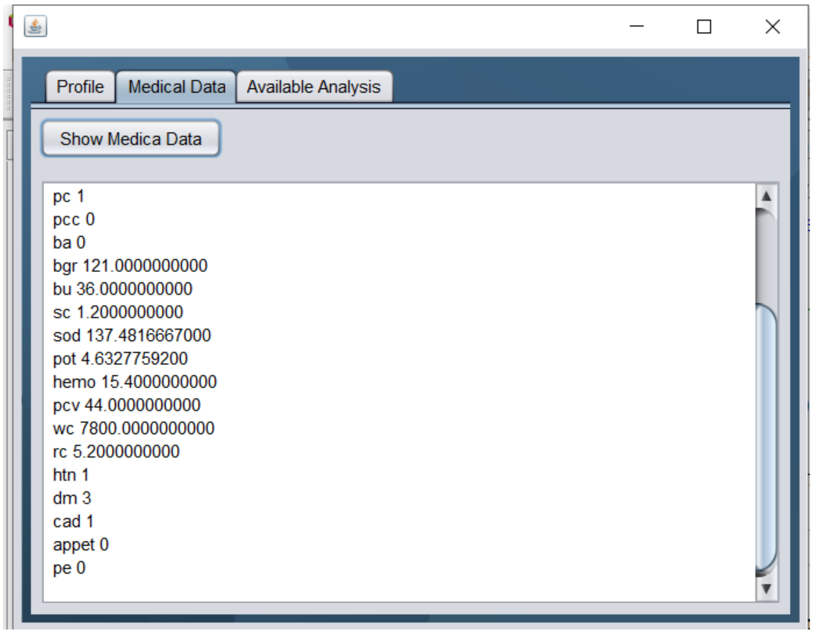
First Name: Ruman

Age: 21

Phone Number: 9538772702

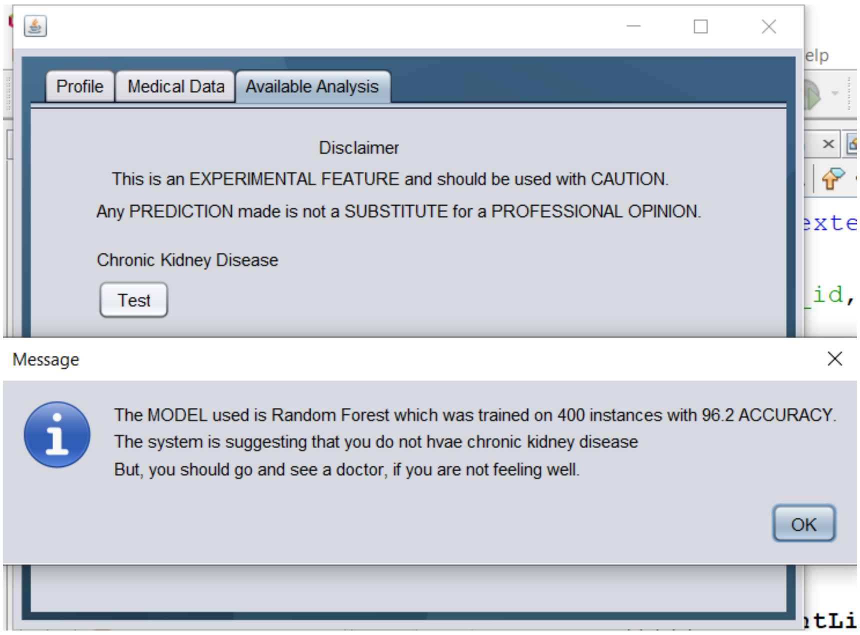
Address: Bangalore

Fig   General Data



```
pc 1
pcc 0
ba 0
bgr 121.0000000000
bu 36.0000000000
sc 1.2000000000
sod 137.4816667000
pot 4.6327759200
hemo 15.4000000000
pcv 44.0000000000
wc 7800.0000000000
rc 5.2000000000
htn 1
dm 3
cad 1
appet 0
pe 0
```

Fig Medical Data



Disclaimer

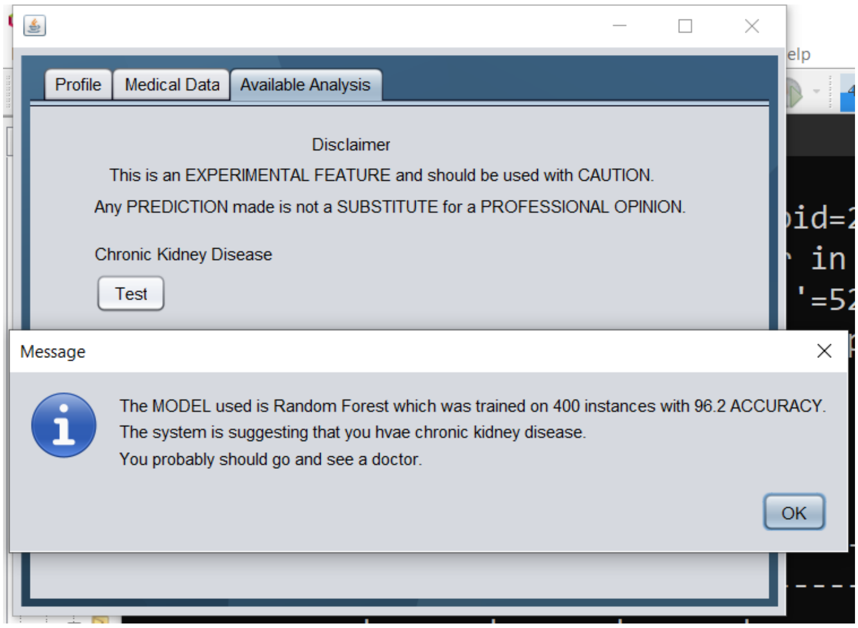
This is an EXPERIMENTAL FEATURE and should be used with CAUTION.

Any PREDICTION made is not a SUBSTITUTE for a PROFESSIONAL OPINION.

Chronic Kidney Disease

The MODEL used is Random Forest which was trained on 400 instances with 96.2 ACCURACY.
The system is suggesting that you do not hvae chronic kidney disease
But, you should go and see a doctor, if you are not feeling well.

Fig Prediction (Negative)



Disclaimer

This is an EXPERIMENTAL FEATURE and should be used with CAUTION.

Any PREDICTION made is not a SUBSTITUTE for a PROFESSIONAL OPINION.

Chronic Kidney Disease

The MODEL used is Random Forest which was trained on 400 instances with 96.2 ACCURACY.
The system is suggesting that you hvae chronic kidney disease.
You probably should go and see a doctor.

Fig Prediction (Positive)

6. CONCLUSION

This project satisfies the needs to manage the common people problems by making it user friendly. At the end of the project it's possible to conclude that the description of the background and context of the project and its relation to work is already done in the area. Made statement for the aims and objectives of the project. The description about the sources, scope and all of those is describe efficiently. Defined the problem on which the work of project is done. Understanding the domain that describes the operations being implemented in the program. Designed user interface and security issues related to the system. Finally the program is implemented and tested according to the test cases related to today's scenarios.

In a nutshell, it can be summarized that the future scope of the project circles around maintaining information regarding, giving more advance software for this project by including more facilities. Hosting the platform on online servers to make it accessible even to remote areas. Integration of multiple load balancers to distribute the loads of the system or by creating a master and slave database structure to reduce the overload of the database queries. Implement the backup mechanism for taking backup of codebase and database on regular basis on different servers.

7. REFERENCES

References for the projects are as follows:

- Random Forest Classifier Link: https://en.wikipedia.org/wiki/Random_forest
- SciKit Learn Documentation Link: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- ProcessBuilder or Runtime.getRuntime().exec Link: https://mkyong.com/java/how-to-execute-shell-command-from-java/
- Java Database Connectivity Link: https://en.wikipedia.org/wiki/Java_Database_Connectivity
- Swing (Java) Link 1: https://en.wikipedia.org/wiki/Swing_(Java)
- Swing (Java) Link 2: https://www.javatpoint.com/java-swing