

**SUJAY HAZRA**

**1NH17CS127**

**6B**

**Topic: “Virtual Classroom”**

**Sem 6**

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 PROBLEM DEFINITION**

It is necessary to have a medium to provide education in a convenient and flexible way, especially during difficult times such as a Global Pandemic. When traveling becomes risky, it is possible to stay connected and exchange information virtually, this not only eliminates the distance barriers but also provides a plethora of flexibility in time and location.

### **1.2 OBJECTIVES**

The aim of this project is to solve the problem by developing a GUI application that works as a virtual classroom, that can offer incomparable convenience and flexibility, this will ensure that learning doesn't stop during such difficult times.

### **1.3 EXPECTED OUTCOMES**

- This application enables teachers from institutions to create courses and have students enroll to them and access them just like a virtual classroom
- This environment allows two types of user Teacher/Student to access their unique dashboards and access the courses they have registered virtually.
- This application also maintains a record separately for each of the courses registered by each of the students in databases.
- This application provides a medium of communications between the students and teachers.

## 1.4 HARDWARE AND SOFTWARE REQUIREMENTS

- **Hardware Specifications:**

- Memory: 512MB
- Processor: 800MHz Intel Pentium III or equivalent.
- Speed: 2.50 GHz
- Disk Space: 750 MB of free disk space

- **Software Specifications:**

- Operating System: Windows 7+
- Developing Language used: Java, Structured query language
- Tools Used: NetBeans IDE, MySQL

## CHAPTER 2

### OBJECT ORIENTED CONCEPTS

Java is an object-oriented programming language. Object Oriented Programming refers to languages that use the programming constructs to represent and implement real world entities like inheritance, polymorphism, hiding, etc in programming. The main purpose of OOPS is to bind data and functions into a single simple unit known as a class.

The different concepts involved in OOPs are:

#### 2.1 CLASS

A class is an entity that determines how an object will behave. It consists of member data and member functions.

Syntax for creating class:

```
class ClassName  
  
{  
  
    //member data and member functions  
  
}
```

#### 2.2 OBJECT

An object is an OOPs component that is used to represent the properties and behaviours of a real-world entity.

Objects have a state and a behaviour. An object's state is stored in fields and behaviour is shown via functions or methods.

In software development, methods operate on the internal state of an object and the object-to-object communication is done via methods.

An object has three characteristics:

1. State: represents the data (value) of an object.
2. Behaviour: represents the behaviour (functionality) of an object such as deposit, withdrawal, etc.
3. Identity: An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

An object is created from a class. In Java, the new keyword is used to create new objects.

There are three steps when creating an object from a class:

1. Declaration: A variable declaration with a variable name with an object type.
2. Instantiation: The 'new' keyword is used to create the object.
3. Initialization: The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Syntax to create an object:

```
ClassName ObjectName = new ClassName();
```

Example of declaring an object of a class:

```
Class Project
```

```
{
```

```
    //methods and variables
```

```
}
```

```
Project obj1 = new Project();
```

## 2.3 INHERITANCE

Inheritance is a mechanism in which one class acquires all the properties and behaviours of another class.

The class acquiring properties is known as the child class or the sub class whereas the class whose properties are being acquired is known as the parent class or the super class.

The idea behind inheritance is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class while also being able to add new methods and fields in the current class without affecting the parent class.

Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

The syntax of Inheritance:

```
class SubclassName extends SuperclassName
{
    //methods and fields
}
```

Example of Inheritance:

```
class Test extends Project
{
    //methods and fields
}
```

The extends keyword indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

There are five types of inheritance in java: Single, Multilevel, Hierarchical and Hybrid.

Java does not support Multiple inheritance of classes in order to avoid ambiguity.

## Single Inheritance

When a class inherits another class, it is known as a single inheritance

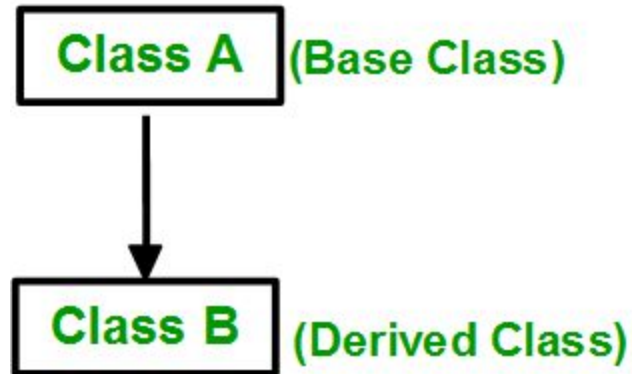


Fig 2.1 Single Inheritance

## Multilevel Inheritance

When there is a chain of inheritance, it is known as multilevel inheritance. That is a derived class becomes a base class.

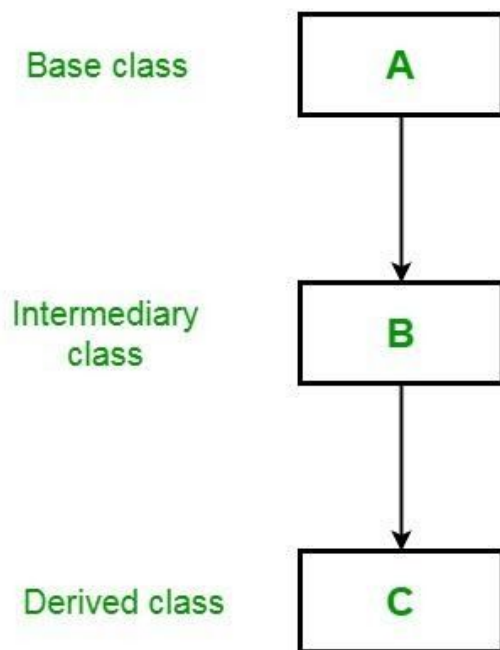


Fig 2.2 Multilevel Inheritance

## Hierarchical Inheritance

When a single class is inherited by multiple classes it is known as hierarchical inheritance.

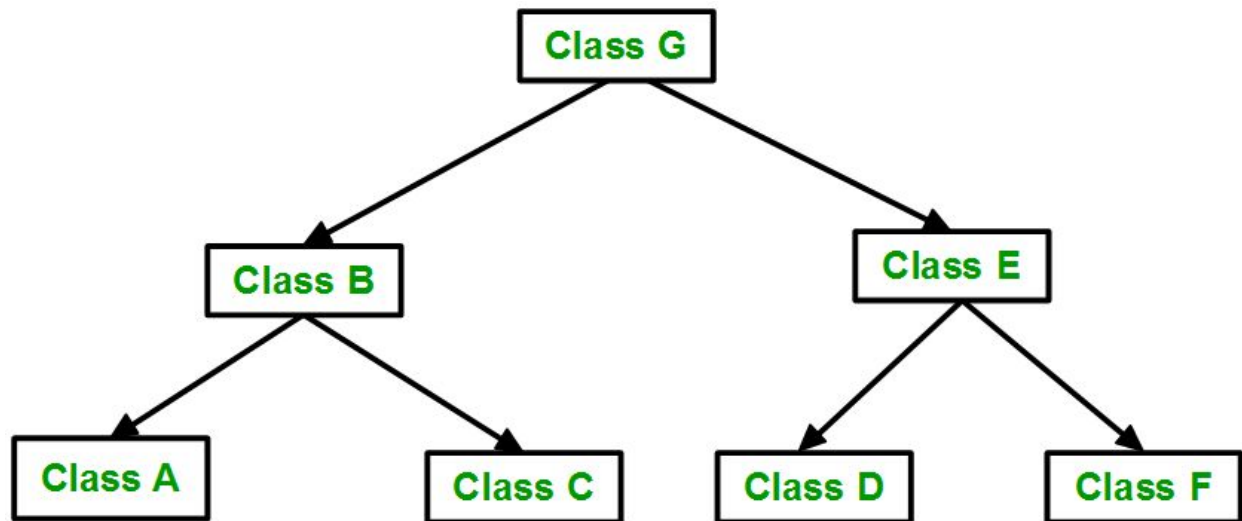


Fig 2.3 Hierarchical Inheritance

## Hybrid Inheritance

It is the combination of two or more types of Inheritance.

## 2.4 POLYMORPHISM

Polymorphism means one name in many forms. In Java, polymorphism can be achieved by method overloading and method overriding.

There are two types of polymorphism in java:

- Compile time polymorphism.
- Runtime polymorphism.



## **Compile Time Polymorphism**

Compile time polymorphism is nothing but method overloading in java. You can define various methods with the same name but different method arguments.

## **Runtime Polymorphism**

Runtime Polymorphism is nothing but a method overriding in java. If a subclass has the same method as base class then it is known as method overriding or in another word, if subclass provides specific implementation to any method which is present in its one of parent's classes then it is known as method overriding.

## **2.5 ABSTRACT CLASS**

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body)

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery. Abstraction lets you focus on what the object does instead of how it does it.

There are two ways to achieve abstraction in java:

1. Abstract class
2. Interface

## 2.6 MULTITHREADING

Multithreading in Java is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading rather than multiprocessing because threads use a shared memory area. They don't allocate separate memory areas so saves memory, and context-switching between the threads takes less time than process.

Threads can be created by using two mechanisms:

1. Extending the Thread class
2. Implementing the Runnable Interface

We create a class that extends the `java.lang.Thread` class. This class overrides the `run()` method available in the Thread class. A thread begins its life inside `run()` method. We create an object of our new class and call `start()` method to start the execution of a thread. `start()` invokes the `run()` method on the Thread object.

## 2.7 IO FUNCTIONS

Java performs I/O through Streams. A Stream is linked to a physical layer by java I/O system to make input and output operation in java. In general, a stream means continuous flow of data. Streams are a clean way to deal with input/output without having every part of your code understand the physical.

Java encapsulates Stream under the `java.io` package. Java defines two types of streams. They are,

1. Byte Stream: It provides a convenient means for handling input and output of byte.
2. Character Stream: It provides a convenient means for handling input and output of characters. Character stream uses Unicode and therefore can be internationalized.

Byte stream is defined by using two abstract classes at the top of hierarchy, they are `InputStream` and `OutputStream`.

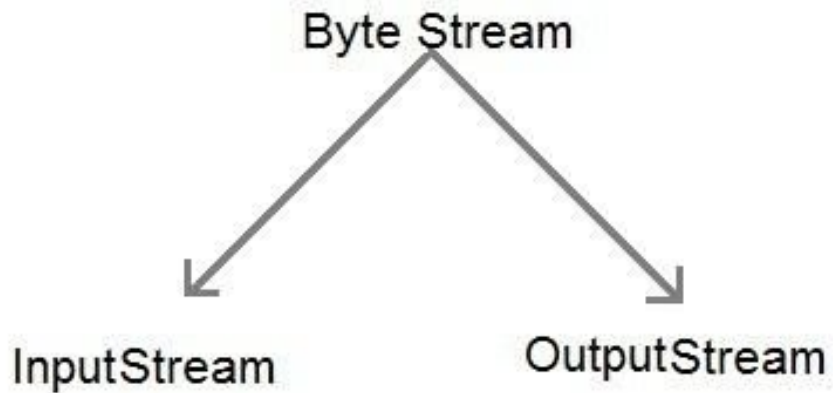


Fig 2.4 Byte Stream

These two abstract classes have several concrete classes that handle various devices such as disk files, network connection etc.

Character stream is also defined by using two abstract classes at the top of hierarchy, they are `Reader` and `Writer`.

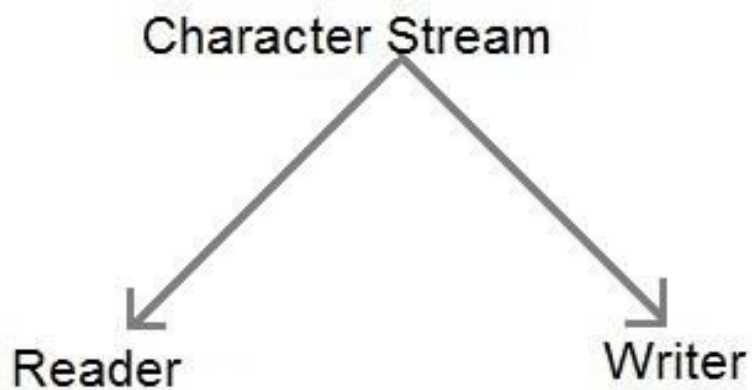


Fig 2.5 Character Stream

These two abstract classes have several concrete classes that handle unicode characters.

## **2.8 JAVA PACKAGES**

A package as the name suggests is a pack(group) of classes, interfaces and other packages. In java we use packages to organize our classes and interfaces.

We have two types of packages in Java:

1. User defined package: The package we create is called user-defined package.
2. Built-in package: The already defined package like `java.io.*`, `java.lang.*` etc are known as built-in packages.

## **2.9 EXCEPTION HANDLING**

Exception Handling in Java is a very interesting topic. Exception is an error event that can happen during the execution of a program and disrupts its normal flow. Java provides a robust and object-oriented way to handle exception scenarios, known as Java Exception Handling.

Java being an object-oriented programming language, whenever an error occurs while executing a statement, creates an exception object and then the normal flow of the program halts and JRE tries to find someone that can handle the raised exception. The exception object contains a lot of debugging information such as method hierarchy, line number where the exception occurred, type of exception etc. When the exception occurs in a method, the process of creating the exception object and handing it over to the runtime environment is called throwing the exception.

Java provides specific keywords for exception handling purposes:

Throw, Try, Catch, Finally

## CHAPTER 3

### DESIGN

#### 3.1 DESIGN GOALS

##### **Platform Independent**

The concept of Write-once-run-anywhere (known as the Platform independent) is one of the important key features of java language that makes java as the most powerful language. Not a single language is idle to this feature but java is closer to this feature. The programs written on one platform can run on any platform provided the platform must have the JVM.

##### **Simple**

There are various features that make java a simple language. Programs are easy to write and debug because java does not use pointers explicitly. It is much harder to write the java programs that can crash the system but that cannot be said about the other programming languages. Java provides the bug free system due to the strong memory management. It also has the automatic memory allocation and deallocation system.

##### **Object Oriented**

To be an Object-Oriented language, any language must follow at least the four characteristics.

- **Inheritance:** It is the process of creating the new classes and using the behaviour of the existing classes by extending them just to reuse the existing code and adding the additional features as needed.
- **Encapsulation:** It is the mechanism of combining the information and providing the abstraction.
- **Polymorphism:** As the name suggests one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the methods.

- **Dynamic binding:** Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

As the languages like Objective C, C++ fulfil the above four characteristics yet they are not fully object-oriented languages because they are structured as well as object-oriented languages. But in the case of java, it is a fully Object-Oriented language because the object is at the outermost level of data structure in java. No stand-alone methods, constants, and variables are there in java. Everything in java is object even the primitive data types can also be converted into objects by using the wrapper class.

### **Robust**

Java has the strong memory allocation and automatic garbage collection mechanism. It provides a powerful exception handling and type checking mechanism as compared to other programming languages. Compiler checks the program whether there is any error and the interpreter checks for any run time error and makes the system secure from crash. All of the above features make the java language robust.

### **Distributed**

The widely used protocols like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access to the files from any remote machine on the internet rather than writing codes on their local system.

### **Portable**

The feature Write-once-run-anywhere makes the Java language portable provided that the system must have an interpreter for the JVM. Java also has the standard data size irrespective of operating system or the processor. These features make java a portable language.

### **Dynamic**

While executing the java program the user can get the required files dynamically from a local drive or from a computer thousands of miles away from the user just by connecting with the Internet.

## **Secure**

Java does not use memory pointers explicitly. All the programs in java are run under an area known as the sand box. Security manager determines the accessibility options of a class like reading and writing a file to the local disk. Java uses the public key encryption system to allow the java applications to transmit over the internet in the secure encrypted form. The bytecode Verifier checks the classes after loading.

## **Performance**

Java uses native code usage, and a lightweight process called thread. In the beginning interpretation of bytecode resulted in slowing of performance but the advanced version of JVM uses the adaptive and just in time compilation technique that improves the performance.

## **Multithreaded**

As we all know several features of Java like Secure, Robust, Portable, dynamic etc; you will be more delighted to know another feature of Java which is Multithreaded.

Java is also a Multithreaded programming language. Multithreading means a single program having different threads executing independently at the same time. Multiple threads execute instructions according to the program code in a process or a program. Multithreading works the similar way as multiple processes run on one computer.

Multithreading programming is a very interesting concept in Java. In multithreaded programs not even a single thread disturbs the execution of other threads. Threads are obtained from the pool of available ready to run threads and they run on the system CPUs.

## **Interpreted**

We all know that Java is an interpreted language as well. With an interpreted language such as Java, programs run directly from the source code. The interpreter program reads the source

code and translates it on the fly into computations. Thus, Java as an interpreted language depends on an interpreter program.

The versatility of being platform independent makes Java to outshine from other languages. The source code to be written and distributed is platform independent.

Another advantage of Java as an interpreted language is its error debugging quality. Due to this any error occurring in the program gets traced. This is how it is different to work with Java.

### **Architecture Neutral**

The term architectural neutral seems to be weird, but yes Java is an architectural neutral language as well. The growing popularity of networks makes developers think distributed. In the world of network, it is essential that the applications must be able to migrate easily to different computer systems. Not only to computer systems but to a wide variety of hardware architecture and Operating system architectures as well.

The Java compiler does this by generating byte code instructions, to be easily interpreted on any machine and to be easily translated into native machine code on the fly. The compiler generates an architecture-neutral object file format to enable a Java application to execute anywhere on the network and then the compiled code is executed on many processors, given the presence of the Java runtime system. Hence Java was designed to support applications on the network. This feature of Java has thrived in the programming language.



## 3.2 ALGORITHM

### 1. Intro Page:

- 1.1. Select Either "Teacher" or "Student"
- 1.2. Exit

### 2. "Teacher" in Intro:

#### 2.1. Login(Enter Credentials)

##### 2.1.1. Credentials Matched: "Login Successful"

View Teacher Dashboard(3)

##### 2.1.2. Credentials Not Matched/Not exists: "Wrong Credential prompt and Re-Login"

#### 2.2. Register:

##### 2.2.1. Takes to Register page to fill a form and register the credentials used for logging in.

#### 2.3. Exit

##### 2.3.1. Takes back to 1.Intro Page

### 3. Teacher Dashboard:

#### 3.1. "My Courses" - This will take the teacher to a frame that will enable them to view their courses and manage them, view the students registered in these courses and also exchange information with students.

#### 3.2."Create Course"-This will take the teacher to a frame that will let them create their courses with various attributes.

### 4. "Student" in Intro page:

#### 4.1. Login(Enter Credentials)

##### 4.1.1. Credentials Matched: "Login Successful"

View Student Dashboard(3)

4.1.2. Credentials Not Matched/Not exists: "Wrong Credential prompt and Re-Login"

4.2. Register:

4.2.1. Takes to Register page to fill a form and register the credentials used for logging in.

4.3. Exit

4.3.1. Takes back to 1.Intro Page

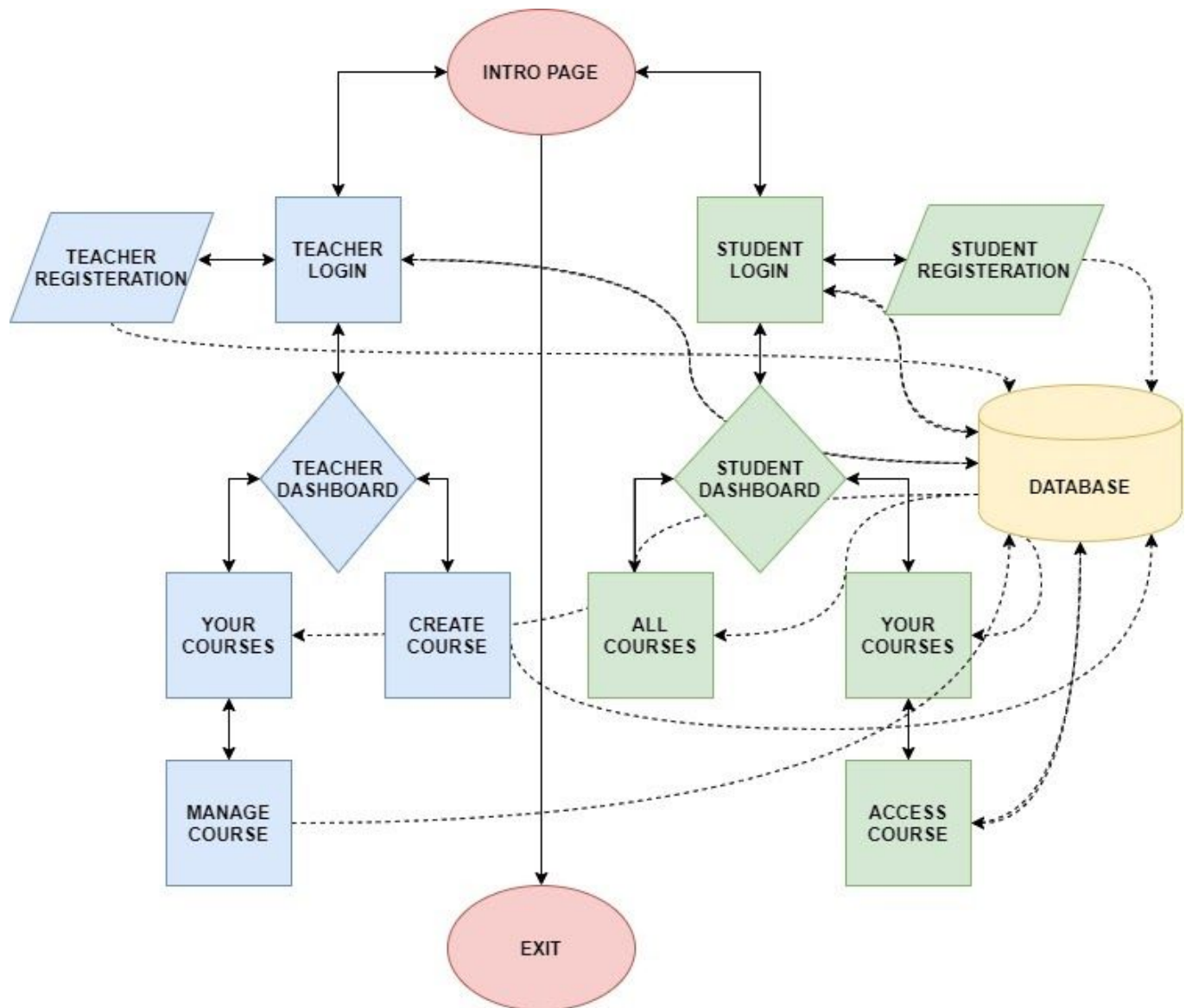
5. Student Dashboard:

5.1. "My Courses" - This will take the Student to a page that will enable them to view the courses they have enrolled and access the courses and also exchange information with Teachers.

5.2."All Courses"-This will take the Student to a page that will let them see all the courses

available along with their details and let them enroll them.

### 3.3 Flow Diagram



## CHAPTER 4

### IMPLEMENTATION

For this project the front end was implemented in Java language using NetBeans IDE and the Back end consists of Databases in the MySQL Server. A total of 13 JFrames and 9 tables were used to develop this application, lets see the backend of each of these JFrames:

#### JFRAMES:

##### 1) INTRO FRAME

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
TeacherLogin M=new TeacherLogin();  
    M.setVisible(true);  
    INTRO.this.setVisible(false);          // TODO add your handling code here:  
}  
  
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    StudentLogin M=new StudentLogin();  
    M.setVisible(true);  
    INTRO.this.setVisible(false);  
  
    // TODO add your handling code here:  
}  
  
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
System.exit(0);          // TODO add your handling code here:  
}
```

Fig 4.1 INTRO Page Implementation

The Intro page basically consists of three buttons to help the user to navigate to their preferred page, the user clicks the “Teacher” button if the user is a teacher or “Student” button if the user is a student. Based on the button clicked, the respective form is opened and the intro form is now closed. Clicking exit will exit the app.

## 2) TEACHER/STUDENT LOGIN FRAMES

```

        private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
String usn=tf.getText();
char[] password=tfp.getPassword();

int n=0;

        {
            {
                if(usn.isEmpty())
                    JOptionPane.showMessageDialog(this,"USN not entered");
                else if(password.length == 0)
                    JOptionPane.showMessageDialog(this,"Password not entered");
            }
        }
else
{ n=2;
    try
    {
        Class.forName("java.sql.DriverManager");
        Connection con=(Connection)
        DriverManager.getConnection("jdbc:mysql://localhost:3306/virtualclassroom","root","admin");
        java.sql.Statement stmt=con.createStatement();
        String query="select usn,password from student where usn='"+usn+"'";
        ResultSet rs=stmt.executeQuery(query);

        while(rs.next())
        {
            String uid=rs.getString("usn");
            String pas=rs.getString("password");

            if(usn.equals(uid) && pas.equals(new String(password)))
            {
                JOptionPane.showMessageDialog(this,"Login Successful, Click OK to continue !!!");

                n++;
                SDashboard SD=new SDashboard();
                SD.setVisible(true);
                StudentLogin.this.setVisible(false);
            }
            else if(!pas.equals(new String(password)))
            {n=4;
                JOptionPane.showMessageDialog(this,"Incorrect Password !!!");
                tfp.setText("");
            }
        }

    }

}

        catch (Exception e)

        catch (Exception e)
        {
            JOptionPane.showMessageDialog(this,e.getMessage());
        }
}

if (n==2)
{JOptionPane.showMessageDialog(this,"That is an unregistered USN, better GO REGISTER NOW !!!");
    tfp.setText("");
    tf.setText("");}

else if (n==3)
{
    try{
        Class.forName("java.sql.DriverManager");
        Connection con=(Connection)
        DriverManager.getConnection("jdbc:mysql://localhost:3306/virtualclassroom","root","admin");
        java.sql.Statement stmt=con.createStatement();
        String queryl="insert into slogin values('"+usn+"')";
        stmt.executeUpdate(queryl);
    }
    catch (Exception e)
    {
        JOptionPane.showMessageDialog(this,e.getMessage());
    }
}
}

```

Fig 4.2 Login Page Implementation

The above code is used to authorize a user into logging in to his/her account and retrieve all the previously stored info from the database, the authorization is done using a user name and a password which is to be entered during the creation of the account in the registration form, and this data upon successful log-in is saved in the login table for retrieval.

The above code ensures all the anomaly that could happen during a login is looked after and taken into account, if the user enters one of the credentials correctly the user is prompted for the incorrect credential, authorization is successful only when both the credentials are found in the database. Both Student login and Teacher login forms follow the same code.

### 3)TEACHER/STUDENT REGISTRATION FRAMES

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    int flag=0;  
    String password1=new String(tfp.getPassword());  
    String usn=tfuid.getText();  
    String password2=new String(tfpl.getPassword());  
    String fullname=tfnn.getText();  
    String school=tfs.getText();  
    String mobile=tfmn.getText();  
    if(usn.isEmpty())  
        JOptionPane.showMessageDialog(this,"USER ID NOT ENTERED");  
    else if(password1.isEmpty())  
        JOptionPane.showMessageDialog(this,"PASSWORD NOT ENTERED");  
    else if(password2.isEmpty())  
        JOptionPane.showMessageDialog(this,"PLEASE RE-ENTER PASSWORD");  
    else if(fullname.isEmpty())  
        JOptionPane.showMessageDialog(this,"NAME NOT ENTERED");  
    else if(school.isEmpty())  
        JOptionPane.showMessageDialog(this,"ORGANISATION/SCHOOL NOT ENTERED");  
    else if(mobile.isEmpty())  
        JOptionPane.showMessageDialog(this,"MOBILE NO. NOT ENTERED");  
    else if(!password1.equals(password2))  
    {  
        JOptionPane.showMessageDialog(this,"PASSWORDS NOT EQUAL,PLEASE RE-ENTER");  
        tfpl.setText("");  
        System.out.println(password1);  
        System.out.println(password2);  
    }  
}
```

```

else
{
    flag=2;
    try
    {
        Class.forName("java.sql.DriverManager");
        Connection
        con=(Connection)DriverManager.getConnection("jdbc:mysql://localhost:3306/virtualclassroom","root","admin");
        Statement stmt=con.createStatement();
        String query="Insert into student values ('"+usn+"', '"+fullname+"', '"+password1+"', '"+school+"', '"+mobile+"')";
        stmt.executeUpdate(query);

    }
    catch (Exception e)
    {
        flag++;
        JOptionPane.showMessageDialog(this,e.getMessage());
    }
}

if(flag==2)
{
    JOptionPane.showMessageDialog(this,"Congrats You Are Successfully Registered(^_^), \nNext Time Just Enter The USN & Password to Login To Your Account !!");
    StudentLogin M=new StudentLogin();
    M.setVisible(true);
    StudentRegister.this.setVisible(false);
}
}

```

Fig 4.3 Registration Implementation

The above code is used for creation of accounts, the same code is used for teachers as well as students, the info entered in the registration form is stored in a database in the form of separate tables “Student” & “Teacher”, the data here is then used for login purposes,etc.

#### 4)Student/Teacher Dashboard

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    SAllCourses M=new SAllCourses();
    M.setVisible(true);
    SDashboard.this.setVisible(false);
    // TODO add your handling code here:
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    SMyCourse M=new SMyCourse();
    M.setVisible(true);
    SDashboard.this.setVisible(false); // TODO add your handling code here:
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        Class.forName("java.sql.DriverManager");
        com.mysql.jdbc.Connection
        con=(com.mysql.jdbc.Connection)DriverManager.getConnection("jdbc:mysql://localhost:3306/virtualclassroom","root","admin");
        Statement stmt=con.createStatement();
        String query1="delete from slogin;";
        stmt.executeUpdate(query1);
        JOptionPane.showMessageDialog(this,"Logout Successful");
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(this,e.getMessage());
    }
    INTRO M=new INTRO();
    M.setVisible(true);
    SDashboard.this.setVisible(false);
    // TODO add your handling code here:
}
}

```

Fig 4.4 Dashboard Implementation



The Dashboard is again a frame used for navigation purpose, with a logout button, which upon clicking will remove the details of the user currently logged in from the login table.

## 5) Other JFrames and Important Codes

The other JFrames are navigatory frames, frames with forms (Create Course page), and forms displaying tables and maintaining input/output streams.

Let's look at the important codes of these forms as other codes are similar to the ones previously discussed.

### 5.1. Message Stream Custom Code in JText Area

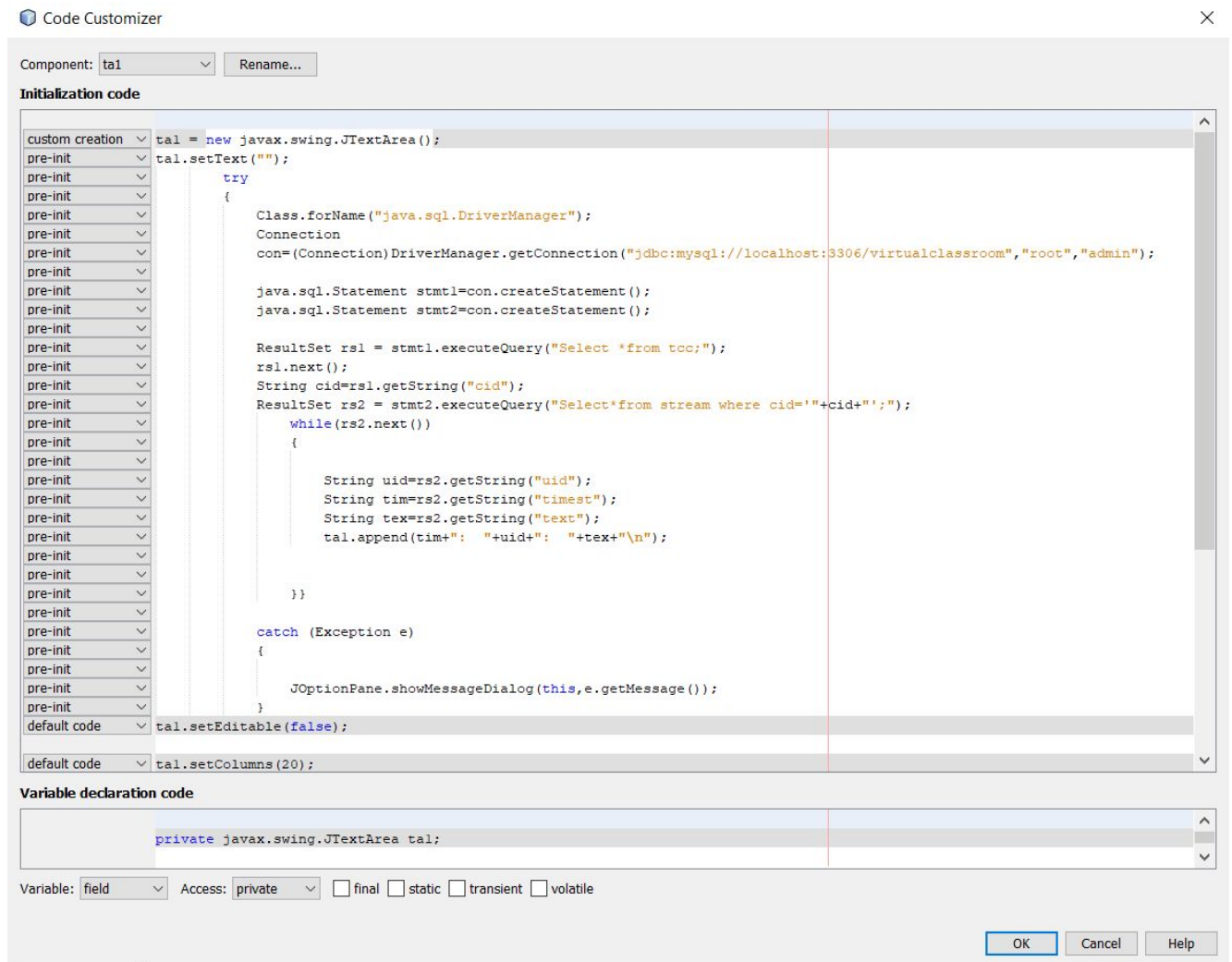


Fig 4.5 Displaying Messages in Text Area



The above code is used to fetch data from the stream table to show all the past conversations so far.

## 5.2. Enabling Tooltip Feature in Tables to view Texts that aren't fit into the cells

 Code Customizer

Component: jTable1

Rename...

Initialization code

custom creation

```
jTable1 = new javax.swing.JTable() {  
    //add tooltip to display the full cell text when not displayed  
    public String getToolTipText( MouseEvent e )  
    {  
        int row = rowAtPoint( e.getPoint() );  
        int column = columnAtPoint( e.getPoint() );  
  
        Object value = getValueAt( row, column );  
        return value == null ? null : value.toString();  
    }  
};
```

Fig 4.6 Displaying Messages in Text Area

This code is written in the customize code option for the JTable and is used to display the text in a particular cell where the mouse is hovered.

```

DefaultTableModel model=(DefaultTableModel)
jTable1.getModel();

int rows=model.getRowCount();
if (rows>0)
{
    for(int i=0;i<rows;i++)
        model.removeRow(0);
}

try
{
    Class.forName("java.sql.DriverManager");
    Connection
    con=(Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/virtualclassroom","root","admin");
    java.sql.Statement stmt=con.createStatement();
    java.sql.Statement stmt1=con.createStatement();
    java.sql.Statement stmt2=con.createStatement();
    String query="Select*from tcc;";
    ResultSet rs = stmt.executeQuery(query);
    rs.next();
    String cid=rs.getString("cid");
    String query1="Select usn from sc where cno='"+cid+"'";
    ResultSet rs1 = stmt1.executeQuery(query1);
    while(rs1.next())
    {
        String usn=rs1.getString("usn");
        String query2="Select*from student where usn='"+usn+"'";
        ResultSet rs2 = stmt2.executeQuery(query2);
        while(rs2.next())
        {
            String susn=rs2.getString("usn");
            String sname=rs2.getString("fullname");
            String scl=rs2.getString("school");
            String mob=rs2.getString("mobile");
            model.addRow(new Object[] {susn,sname,scl,mob});
        }
    }
}
catch (Exception e)
{
    JOptionPane.showMessageDialog(this,e.getMessage());
}

```

The above code is used in post-setColumnModel part of the customize code properties of the jTable, and is used to fetch particular data using queries from the table in the MySQL database.

## DATABASE:

```
mysql> show tables;
```

| Tables_in_virtualclassroom |
|----------------------------|
| sc                         |
| scc                        |
| slogin                     |
| stream                     |
| student                    |
| tcc                        |
| tcourse                    |
| teacher                    |
| tlogin                     |

```
9 rows in set (0.03 sec)
```

```
mysql> desc student;
```

| Field    | Type        | Null | Key | Default | Extra |
|----------|-------------|------|-----|---------|-------|
| usn      | varchar(10) | NO   | PRI | NULL    |       |
| fullname | varchar(30) | NO   |     | NULL    |       |
| password | varchar(30) | NO   |     | NULL    |       |
| school   | varchar(30) | NO   |     | NULL    |       |
| mobile   | bigint(10)  | YES  |     | NULL    |       |

```
5 rows in set (0.01 sec)
```

```
mysql> desc teacher;
```

| Field    | Type        | Null | Key | Default | Extra |
|----------|-------------|------|-----|---------|-------|
| usid     | varchar(10) | NO   | PRI | NULL    |       |
| fullname | varchar(30) | NO   |     | NULL    |       |
| password | varchar(30) | NO   |     | NULL    |       |
| school   | varchar(30) | NO   |     | NULL    |       |
| mobile   | bigint(10)  | YES  |     | NULL    |       |
| dept     | char(30)    | NO   |     | NULL    |       |

```
6 rows in set (0.01 sec)
```

```
mysql> desc tcourse;
```

| Field  | Type         | Null | Key | Default | Extra |
|--------|--------------|------|-----|---------|-------|
| c_no   | varchar(10)  | NO   | PRI | NULL    |       |
| c_name | varchar(30)  | NO   | MUL | NULL    |       |
| t_id   | varchar(10)  | NO   |     | NULL    |       |
| org    | varchar(30)  | NO   |     | NULL    |       |
| dept   | varchar(30)  | NO   |     | NULL    |       |
| c_info | varchar(100) | NO   |     | NULL    |       |

```
6 rows in set (0.01 sec)
```

```
mysql> desc sc;
```

| Field | Type        | Null | Key | Default | Extra |
|-------|-------------|------|-----|---------|-------|
| cno   | varchar(10) | NO   | PRI |         |       |
| usn   | varchar(10) | NO   | PRI |         |       |

```
2 rows in set (0.01 sec)
```

```
mysql> desc scc;
```

| Field | Type        | Null | Key | Default | Extra |
|-------|-------------|------|-----|---------|-------|
| cid   | varchar(10) | YES  |     | NULL    |       |

```
1 row in set (0.00 sec)
```

```
mysql> desc tcc;
```

| Field | Type        | Null | Key | Default | Extra |
|-------|-------------|------|-----|---------|-------|
| cid   | varchar(20) | YES  |     | NULL    |       |

```
mysql> desc slogin;
```

| Field | Type        | Null | Key | Default | Extra |
|-------|-------------|------|-----|---------|-------|
| susn  | varchar(10) | NO   | PRI | NULL    |       |

1 row in set (0.01 sec)

```
mysql> desc tlogin;
```

| Field | Type        | Null | Key | Default | Extra |
|-------|-------------|------|-----|---------|-------|
| tuid  | varchar(10) | NO   | PRI | NULL    |       |

1 row in set (0.01 sec)

```
mysql> desc stream;
```

| Field  | Type         | Null | Key | Default | Extra |
|--------|--------------|------|-----|---------|-------|
| cid    | varchar(20)  | YES  |     | NULL    |       |
| uid    | varchar(20)  | YES  |     | NULL    |       |
| timest | varchar(40)  | YES  |     | NULL    |       |
| text   | varchar(250) | YES  |     | NULL    |       |

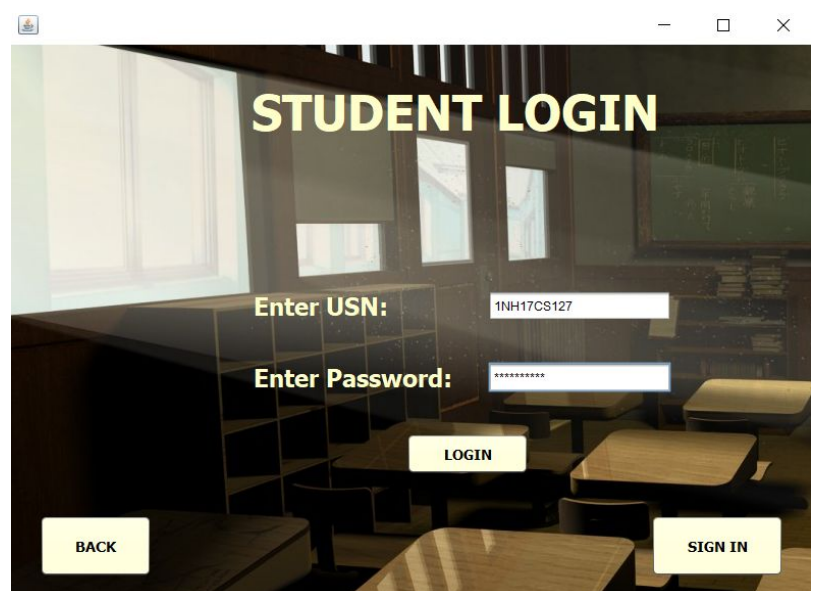
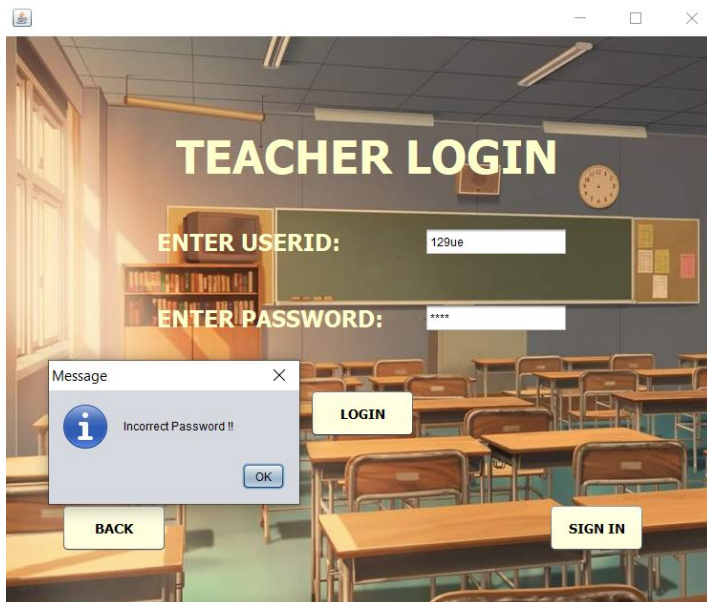
4 rows in set (0.01 sec)



## CHAPTER 5

## RESULTS

Below are the snapshots of the User Interface when we run the project:



# REGISTRATION

User ID :-

Full Name :-

Organisation/School :-

Mobile No. :-

Dept :-

Password :-

Confirm Password :-

**REGISTER**

Back to Log **EXIT**

Message  
i PASSWORDS NOT EQUAL, PLEASE RE-ENTER  
OK

# REGISTRATION

User ID :-

Full Name :-

Organisation/School :-

Mobile No. :-

Dept :-

Password :-

Confirm Password :-

**REGISTER**

Back to Login Page **EXIT**

Message  
i Congrats You Are Successfully Registered(^\_^),  
Next Time Just Enter The User ID & Password to Login To Your Account !!  
OK

## TEACHER DASHBOARD

**MY COURSES**

**CREATE NEW COURSE**

**LOG OUT**

## Student Dashboard

**MY COURSES**

**ALL COURSES**

**LOG OUT**

## TEACHER:

CREATE COURSE

COURSE NAME :-

Computer Networks

COURSE ID :-

CN1

Organisation/School :-

NHCE

Dept :-

CSE

COURSE INFO(provide a brief discussion of what the course is about):-

This Course gives a brief insight on the working of computer networks.

CREATE

Back

Upon creation of the course, visit "Manage Courses" in your dashboard to provide content to the course.

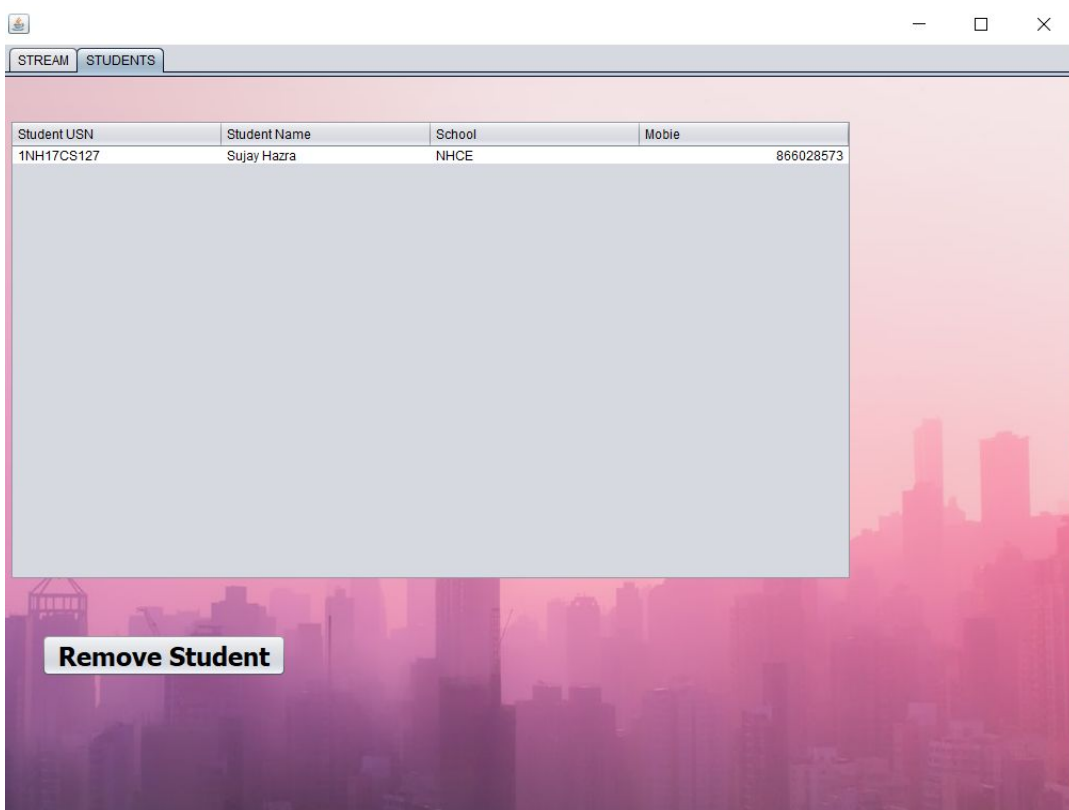
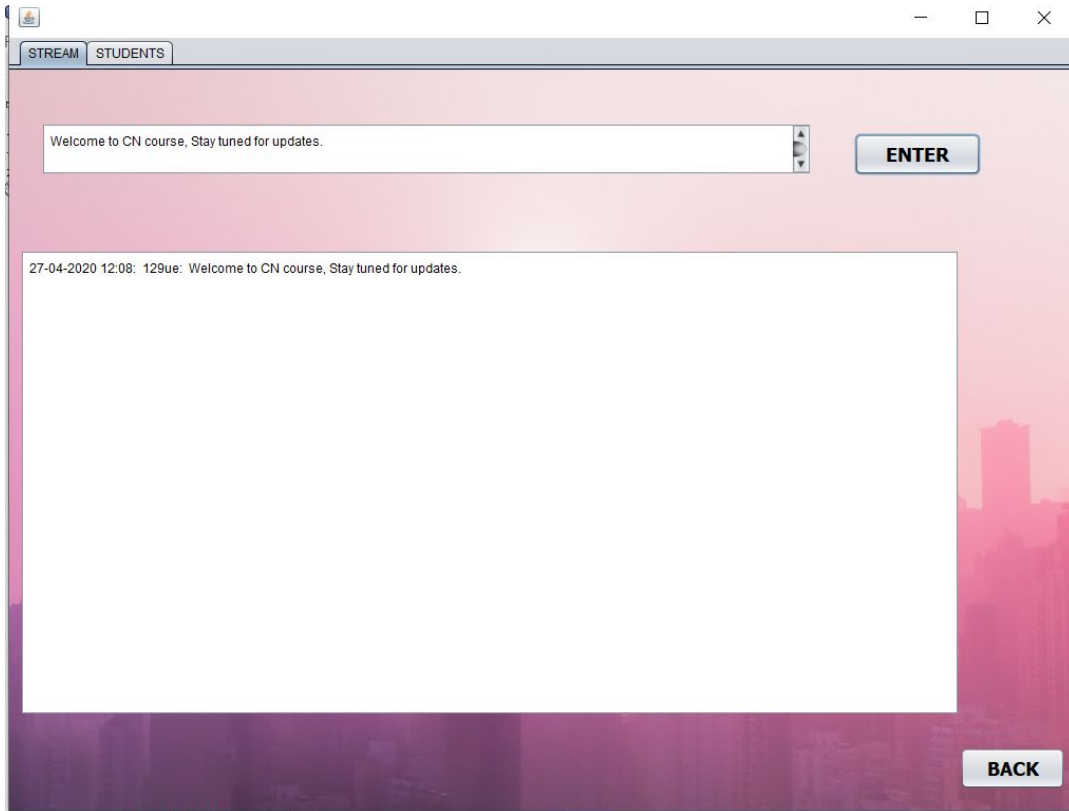
| Course_ID | Course_Name       | Org  | Dept |
|-----------|-------------------|------|------|
| CN1       | Computer Networks | NHCE | CSE  |

Manage Course

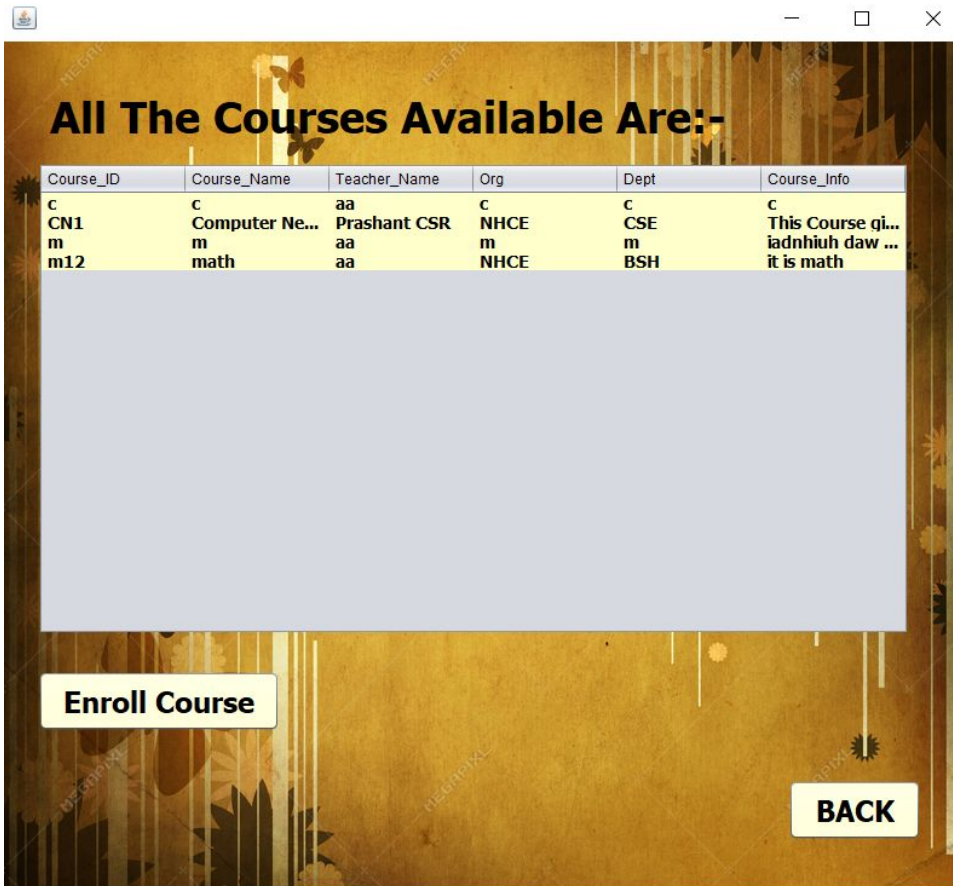
Delete Course

BACK





## STUDENT:

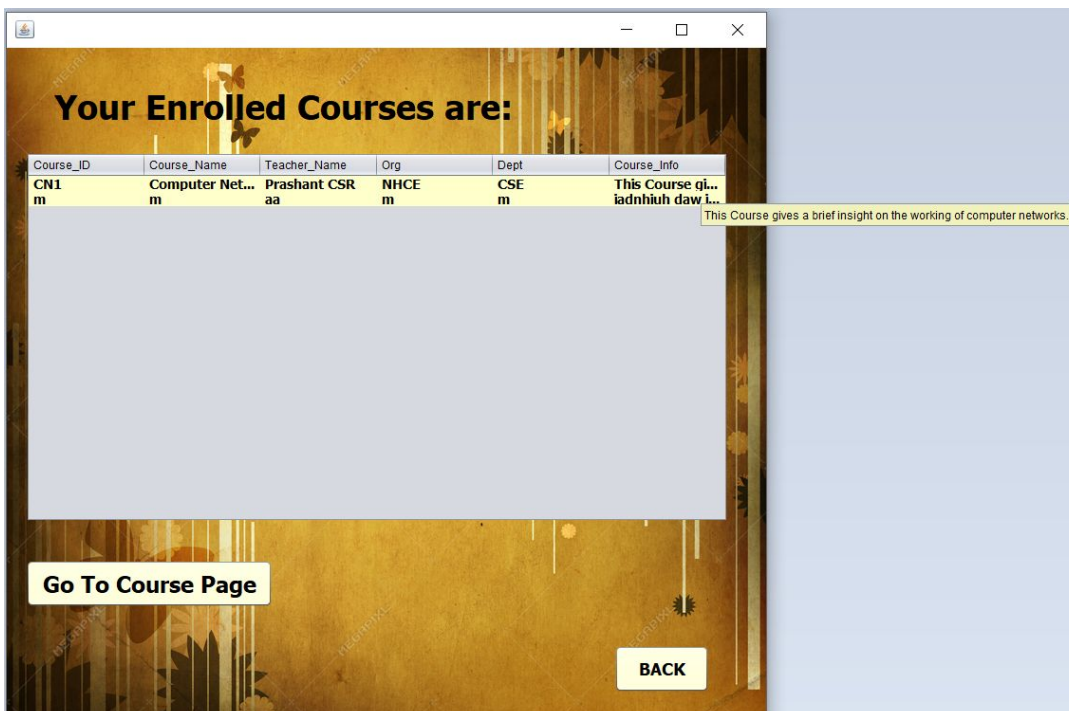


The screenshot shows a web application window with a title bar containing a small icon and standard window controls (minimize, maximize, close). The main content area has a decorative background with a warm, golden-brown color and abstract patterns. At the top, the text "All The Courses Available Are:-" is displayed in a bold, black font. Below this text is a table with six columns: Course\_ID, Course\_Name, Teacher\_Name, Org, Dept, and Course\_Info. The table contains three rows of data. Below the table is a large, empty rectangular area with a light blue background. At the bottom left, there is a button labeled "Enroll Course". At the bottom right, there is a button labeled "BACK".

| Course_ID | Course_Name         | Teacher_Name       | Org       | Dept     | Course_Info                       |
|-----------|---------------------|--------------------|-----------|----------|-----------------------------------|
| c<br>CN1  | c<br>Computer Ne... | aa<br>Prashant CSR | c<br>NHCE | c<br>CSE | c<br>This Course gi...            |
| m<br>m12  | m<br>math           | aa<br>aa           | m<br>NHCE | m<br>BSH | m<br>iadhuh daw ...<br>it is math |

Enroll Course

BACK



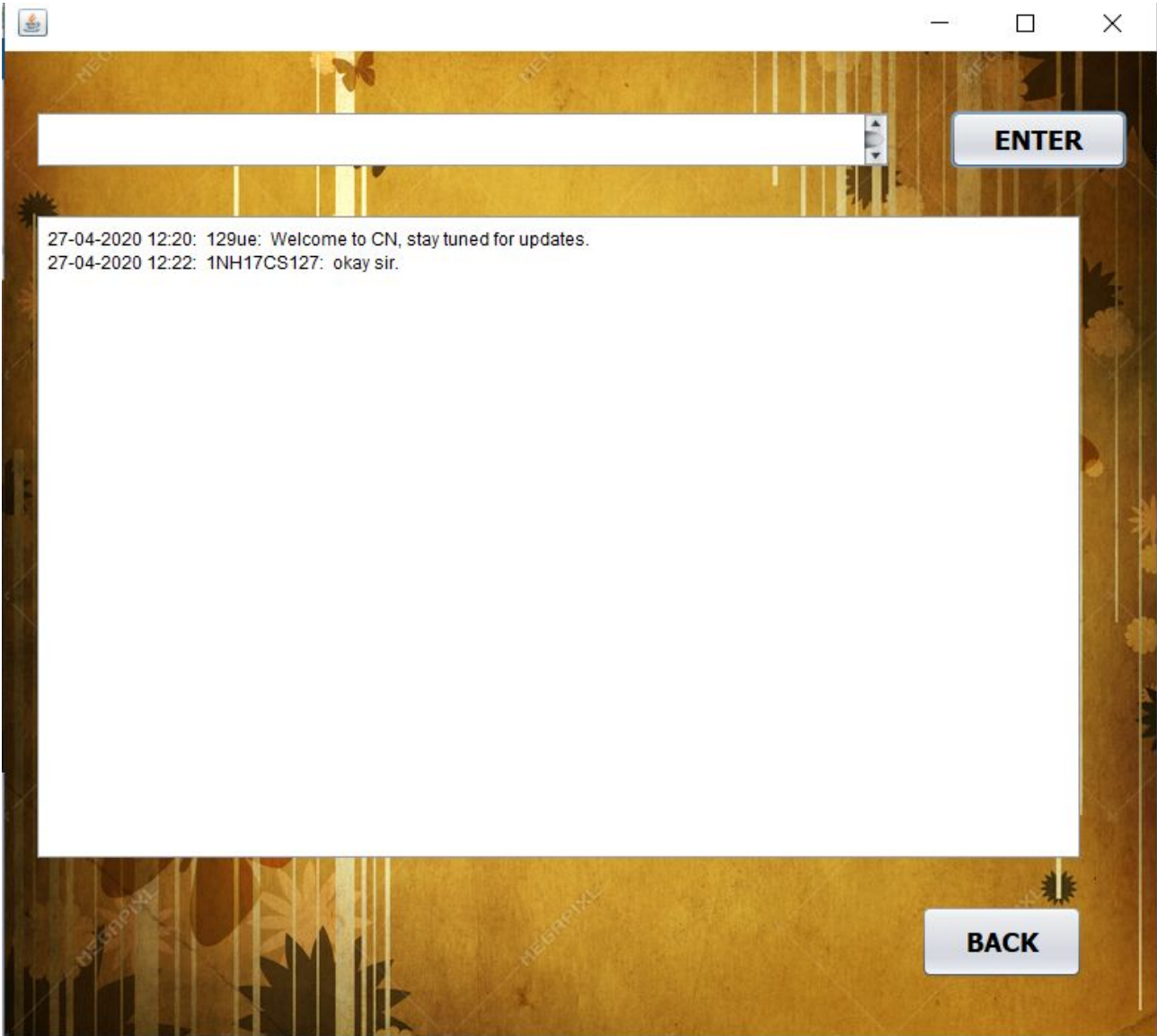
The screenshot shows a web application window with a title bar containing a small icon and standard window controls (minimize, maximize, close). The main content area has a decorative background with a warm, golden-brown color and abstract patterns. At the top, the text "Your Enrolled Courses are:" is displayed in a bold, black font. Below this text is a table with six columns: Course\_ID, Course\_Name, Teacher\_Name, Org, Dept, and Course\_Info. The table contains one row of data. Below the table is a large, empty rectangular area with a light blue background. At the bottom left, there is a button labeled "Go To Course Page". At the bottom right, there is a button labeled "BACK". A tooltip is visible over the "Course\_Info" cell of the table, displaying the text "This Course gives a brief insight on the working of computer networks."

| Course_ID | Course_Name          | Teacher_Name       | Org       | Dept     | Course_Info                          |
|-----------|----------------------|--------------------|-----------|----------|--------------------------------------|
| CN1<br>m  | Computer Net...<br>m | Prashant CSR<br>aa | NHCE<br>m | CSE<br>m | This Course gi...<br>iadhuh daw i... |

Go To Course Page

BACK

This Course gives a brief insight on the working of computer networks.



## CHAPTER 6

### CONCLUSION

This application has a UI that is really simple and has taken all the possible user inputs into consideration to avoid anomalies and discrepancies, the user has to ensure that correct userID and password is entered to login after registering themselves in the application, each user has to provide a unique user ID, all the courses are well maintained, and can also be removed by the teacher.

This application ensures learning doesn't stop in difficult times, and also can be a great tool to share resource links and tasks to the students, and save records with the help of a well maintained database.

The user also has tooltip texts for each of the buttons in case they need assistance on using the app.

## CHAPTER 7

### REFERENCES

1. <https://stackoverflow.com/questions>
2. <https://www.quora.com/>
3. <https://www.javatpoint.com/java-oops-concepts>
4. <https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/>