Desktop Application: ATM Interface Design

Enroll No.	- 101269
Name	- Raghav Gupta
Supervisor	- Dr. Pardeep Kumar

May -2014

Submitted in partial fulfillment of the Degree of
Bachelor of Technology.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKNAGHAT

Contents:

CERTIFICATE

This is to certify that the work titled "**Desktop Application: ATM Interface Design**" submitted by "**RAGHAV GUPTA**" in partial fulfillment for the award of degree of **Bachelor of Technology** of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor:	________________________
Name of Supervisor:	Dr. Pardeep Kumar
Designation:	Assistant Professor (Senior Grade)
Date	________________________

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my guide **Dr. PARDEEP KUMAR** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to staff members of **Jaypee University of Information Technology**, for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my project.

Lastly, I thank almighty, my parents, brother, sisters and friends for their constant encouragement without which this project would not be possible.

Introduction to Application Program:

An Application is a collection of certain function – embedded programs that execute in a serial fashion in order to achieve a primary goal of performing a user-defined task.

In the present world of Information Technology, an average computer programmer or a computer owner is surrounded by a hub of applications programs, many of which are basically meant for general functions, and as the requirement of a end –user continues to grow, more and more number of such applications will be designed and implemented to meet this challenge.

Whether we are considering any arithmetical query, transactional, textual; which require computational operation to be done; all such functions can be performed by using Application Software. As the market requirement of such applications is increasing on a day-to-day basis, more and improved quality of these softwares is expected to be seen.

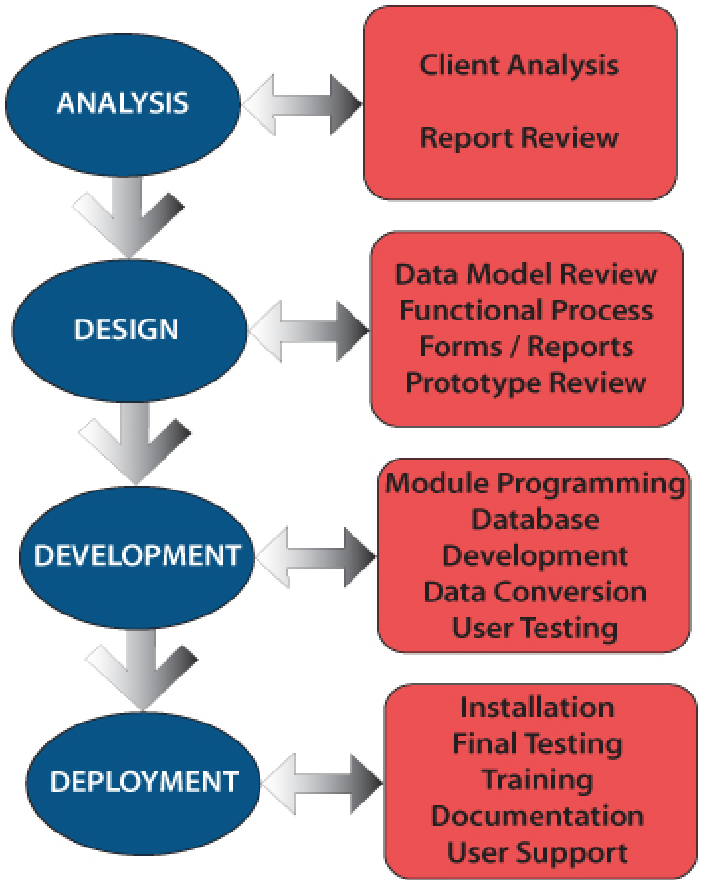
Access to any application software can be made through by executing such applications on a System or on a cloud platform (through the application of SaaS). Subsequently, the features of such application softwares can prove to be an aid in the process of selecting which type of programs or softwares to execute on a system or servers or purchase access from cloud service provider.

In this report, I will discuss about the tools and utilities for developing and practically implementing an application to run on OS. The sections that follow guide you through the process of application design, choosing a program language, and using a runtime environment.

Lifecycle of Application Development:

Lifecycle of an Application Development comprises of different phases/models which are required to be executed in pre-defined fashion in order to achieve the goal of developing and implementing Application Software.

The life cycle of an application to run on OS begins with the requirement gathering phase, continues through design and code development, and ends with testing before the application enters into a steady state in which improvements or maintenance changes are applied.



- **Analysis Phase:**

Life cycle of an Application commence from the Analysis Phase, where the tools (Software/Hardware) to develop an application software are to be acquired and set-up on a System.

The Analysis phase helps in determining the requirements, wheather functional or non-functional in behavior and no matter how these reuirements are acuired, whose operations are the important feature in development of the application.
In this phase, the problem statement of the project is made clear to the developer of the application. The outcome of this phase is a documentation that shows what is to be implemented in a clear and precise fashion.

It is also known as "**Requirement Gathering Phase**", represents the "what" phase.
The documentation at the end of this phase attain the requirements from the developer's perspective by defining goals and challenges that can be faced while designing. The phase is summarized in the table shown below.

Phase	**Deliverable**
Analysis	• **Requirements Document**
	• **Domain Ontology**
	❖ **Things**
	❖ **Actions**
	❖ **States**
	• **Typical Scenarios**
	• **Atypical Scenarios**

The Analysis Phase: What the system does ?

Traditionally, the document is written in English or another written language.
Outcome doesn't provide any implementation details but specifies the key requirements at a higher level of description.

The analysis team develops the requirement document, which talks about things and actions on things. This document should also include states, events, typical scenarios of usage, and atypical scenarios of usage.

Things:

The requirement document first of all defines the ontology of the system. Here the pieces and parts, constants, names, and their relationships to each other are specified.

Actions:

The document defines the actions to be taken after a particular event is occurred. Methods, functions, and procedures are all examples of actions.

States:

Most of the States are Domain Specific. This implies that our System may jump to any State, on the basis of the input given by user.
States can take forms like initial state, intermediate State, the final state, and can also lead to error states.

Typical Scenarios:

A **scenario** is a sequence of steps taken to accomplish a given goal. When the system is completed and the application is available, the customer should be able, in an easy and clearly specified manner, to accomplish all typical usage scenarios for the application.

The **typical scenarios** should represent a task to be accomplished by the end-user by providing domain of correct data type. A system with only one possible usage scenario will be easy to cover while a system with thousands of possible usage scenarios will be much harder to cover.

Atypical Scenarios:

An **atypical scenario** is something that needs to be accomplished within the system, but rarely. The actions have to be done correctly, but perhaps at lower efficiency. The customer should hope that an unexpected error condition is an atypical event.

- ## Design Phase

Phase	**Deliverable**
Design	• Architecture Design
	• Implementation Design
	• Critical Point Analysis
	• Performance Analysis
	• Test Plan

The Design Phase: What are the plans?

In the **design phase** the **architecture** is established. This phase starts with the requirement document delivered by the requirement phase and maps the requirements into an architecture. The **architecture** defines the components, their interfaces and behaviors. The deliverable design document is the architecture. The design document describes a plan to implement the requirements. This phase represents the “how” phase. Details on computer programming languages and environments, machines, packages, application architecture, distributed architecture layering, memory size, platform, algorithms, data structures, global type definitions, interfaces, and many other engineering details are established. The design may include the usage of existing components.

The architectural team can now expand upon the information established in the requirement document. Using the typical and atypical scenarios provided from the requirement document, performance trade-offs can be accomplished as well as complexity of implementation trade-offs.

In our approach, the team, given a complete requirement document, must also indicate **critical priorities** for the implementation team. A critical implementation priority leads to a task that has to be done right. If it fails, the product fails. If it succeeds, the product might succeed. At the very least, the confidence level of the team producing a successful product will increase. This will keep the implementation team focused. Exactly how this information is conveyed is a skill based on experience more than a science based on fundamental foundations.

Architecture:

The architecture defines the components, interfaces, and behaviors of the system.

The components are the building blocks for the system. These components give a blueprint of the system to be developed. The component captures the meaning of details from the requirement document.

The components are composed with other components using their interfaces. An interface forms a common boundary of two components. The interface is the architectural surface where independent components meet and communicate with each other. Over the interface, components interact and affect each other.

Implementation Plan:

The **implementation plan** establishes the schedule and needed resources. It defines implementation details including programming languages, platforms, programming environments, debuggers, and many more.

The implementation plan could be considered as part of the design, which is the subsequent phase of the development cycle. Implementation plan allows developer to choose a platform to implement the practical work. Thus it is very natural to include the implementation plan. Also, the trade-offs between alternative architectures can be influenced by differences in their implementation plans.

Critical Priority Analysis:

The **critical priority analysis** generates a list of **critical tasks**. It is absolutely necessary to successfully accomplish a critical task. The project will succeed or fail based on the outcome of these tasks. Some projects may have more than one critical task.

There are two major categories of critical tasks. One category of tasks are associated with the building of the system. These are the critical tasks that the teams must accomplish well. An example might be a high-quality implementation of a critical section of code in the system.

The other category of critical tasks is associated with the system itself. These are the critical tasks that the system, once built, must accomplish well. An example might be the successful flying of an airplane under automatic pilot.

Performance Analysis:

Once given the typical scenarios from the requirement document, the system can be designed to meet performance objectives. Different system architectures will yield different predicted performance characteristics for each typical scenario. Depending on the usage frequency of the scenarios in the system, each architecture will have benefits and drawbacks with advantages and disadvantages. The trade-offs are then weighted to establish the system architecture. Frequently a system is designed to give fast response to an action initiated by a human customer at the expense of having to do more complex systems work such as including indexes, cache management, and predictive pre-calculations.

Test Plan:

The **test plan** defines the testing necessary to establish quality for the system. If the system passes all tests in the test plan, then it is declared to be complete. If the system does pass all test then it is considered to be of high quality. The more complete the coverage of the system, the higher is the confidence in the system: hence the system's quality rises.

The test plan could be considered as part of the design, which is the position taken here, or it could be considered as the first accomplishment in the testing phase. One of the goals of the design phase, is to establish a plan to complete the system, thus it is very natural to include the test plan. Also the trade-offs between alternative architectures can be influenced by differences in their test plans.

• Implementation Phase:

Phase	Delivarable
Implementation	• **Code**
	• **Critical Error Removal**

The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging. The end deliverable is the product itself.

In the **implementation phase**, the team builds the end-product from the platform and the programming languages which is decided in implementation plan. As per the output of design phase and Analysis phase (requirement document), team must develop the end-product as pre-planned. Though any improvement in the phases prior to this is accepted. For example, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline.

Critical Error Removal:

There are three kinds of errors in a system, namely critical errors, non-critical errors, and unknown errors.

A **critical error** prevents the system from fully satisfying the usage scenarios. These errors have to be corrected before the system can be given to a customer or even before future development can progress.

A **non-critical error** is known but the presence of the error does not significantly affect the system's perceived quality. There may indeed be many known errors in the system. Usually these errors are listed in the release notes and have well established work around.

In fact, the system is likely to have many, yet-to-be-discovered errors. The effects of these errors are unknown. Some may turn out to be critical while some may be simply fixed by patches or fixed in the next release of the system.

- **The TESTING Phase:**

Phase	**Deliverable**
Testing	• Regression Testing
	• Internal Testing
	• Unit Testing
	• Application Testing
	• Stress Testing

Simply stated, quality is very important. Many companies have not learned that quality is important and deliver more claimed functionality but at a lower quality level. It is much easier to explain to a customer why there is a missing feature than to explain to a customer why the product lacks quality. A customer satisfied with the quality of a product will remain loyal and wait for new functionality in the next version. Quality is a distinguishing attribute of a system indicating the degree of excellence.

In many software engineering methodologies, the **testing phase** is a separate phase which is performed by a different team after the implementation is completed. There is merit in this approach; it is hard to see one's own mistakes, and a fresh eye can discover obvious errors much faster than the person who has read and re-read the material many times. Unfortunately, delegating testing to another team leads to a slack attitude regarding quality by the implementation team.

The testing technique is from the perspective of the system provider. Because it is nearly impossible to duplicate every possible customer's environment and because systems are released with yet-to-be-discovered errors, the customer plays an important, though reluctant, role in testing. As will be established later in the thesis, in the WaterSluice methodology this is accomplished in the alpha and beta release of the system.

Regression Test:

Quality is usually appraised by a collection of **regression tests** forming a suite of programs that test one or more features of the system.

A regression test is written and the results are generated. If the results are in error, then the offending bug is corrected. A valid regression test generates verified results. These verified results are called the **"gold standard".** This term is borrowed from financial markets where paper money issued by governments was backed by real gold.

Ideally, the validity of a test result is driven by the requirement document; in practice, the implementation team is responsible for validity interpretation.

Internal Testing:

Internal testing deals with low-level implementation. Here each function or component is tested. This testing is accomplished by the implementation teams. This focus is also called clear-box testing, or sometimes white-box testing, because all details are visible to the test. Internal limits are tested here.

Unit Testing:

Unit testing deals with testing a unit as a whole. This would test the interaction of many functions but confine the test within one unit. The exact scope of a unit is left to interpretation. Supporting test code, sometimes called**scaffolding**, may be necessary to support an individual test. This type of testing is driven by the architecture and implementation teams. This focus is also called black-box testing because only the details of the interface are visible to the test. Limits that are global to a unit are tested here.

Internal and unit testing can be automated with the help of coverage tools. A coverage tool analyzes the source code and generates a test that will execute every alternative thread of execution. It is still up to the programmer to combine these test into meaningful cases to validate the result of each thread of execution. Typically, the coverage tool is used in a slightly different way. First the coverage tool is used to augment the source by placing informational prints after each line of code. Then the testing suite is executed generating an audit trail. This audit trail is analyzed and reports the percent of the total system code executed during the test suite. If the coverage is high and the untested source lines are of low impact to the system's overall quality, then no more additional tests are required.

Application Testing:

Application testing deals with tests for the entire application. This is driven by the scenarios from the analysis team. Application limits and features are tested here.

The application must successfully execute all scenarios before it is ready for general customer availability. After all, the scenarios are a part of the requirement document and measure success. Application testing represents the bulk of the testing done by industry.

Unlike the internal and unit testing, which are programmed, these test are usually driven by scripts that run the system with a collection of parameters and collect results. In the past, these scripts may have been written by hand but in many modern systems this process can be automated.

Most current applications have graphical user interfaces (GUI). Testing a GUI to assure quality becomes a bit of a problem. Most, if not all, GUI systems have event loops. The GUI event loop contains signals for mouse, keyboard, window, and other related events. Associated with each event are the coordinates on the screen of the event. The screen coordinates can be related back to the GUI object and then the event can be serviced. Unfortunately, if some GUI object is positioned at a different location on the screen, then the coordinates change in the event loop. Logically the events at the new coordinates should be associated with the same GUI object. This logical association can be accomplished by giving unique names to all of the GUI objects and providing the unique names as additional information in the events in the event loop. The GUI application reads the next event off of the event loop, locates the GUI object, and services the event.

The events on the event loop are usually generated by human actions such as typing characters, clicking mouse buttons, and moving the cursor. A simple modification to the event loop can journal the events into a file. At a later time, this file could be used to regenerate the events, as if the human was present, and place them on the event loop. The GUI application will respond accordingly. A tester, using the GUI, now executes a scenario. A journal of the GUI event loop from the scenario is captured. At a later time the scenario can be repeated again and again in an automated fashion. The ability to repeat a test is key to automation and stress testing.

Stress Testing:

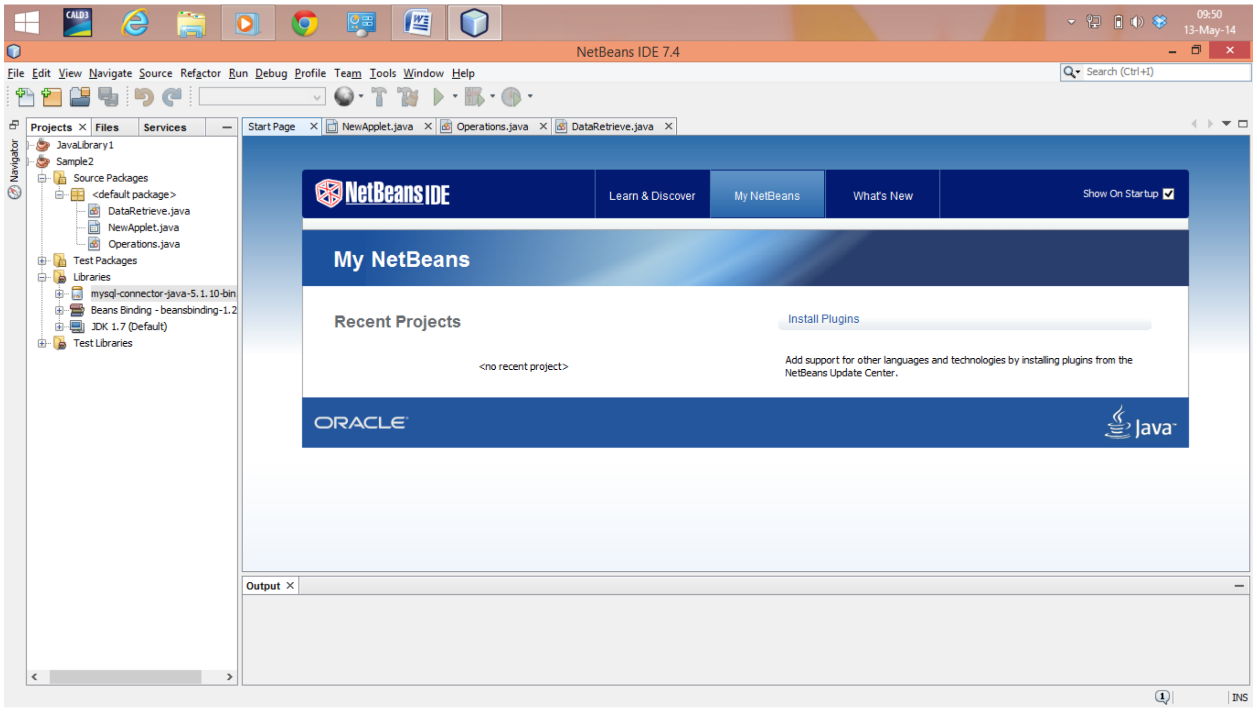
Stress testing deals with the quality of the application in the environment. The idea is to create an environment more demanding of the application than the application would experience under normal work loads. This is the hardest and most complex category of testing to accomplish and it requires a joint effort from all teams.

A test environment is established with many testing stations. At each station, a script is exercising the system. These scripts are usually based on the regression suite. More and more stations are added, all simultaneous hammering on the system, until the system breaks. The system is repaired and the stress test is repeated until a level of stress is reached that is higher than expected to be present at a customer site.

Race conditions and memory leaks are often found under stress testing. A race condition is a conflict between at least two tests. Each test works correctly when done in isolation. When the two tests are run in parallel, one or both of the tests fail. This is usually due to an incorrectly managed lock.

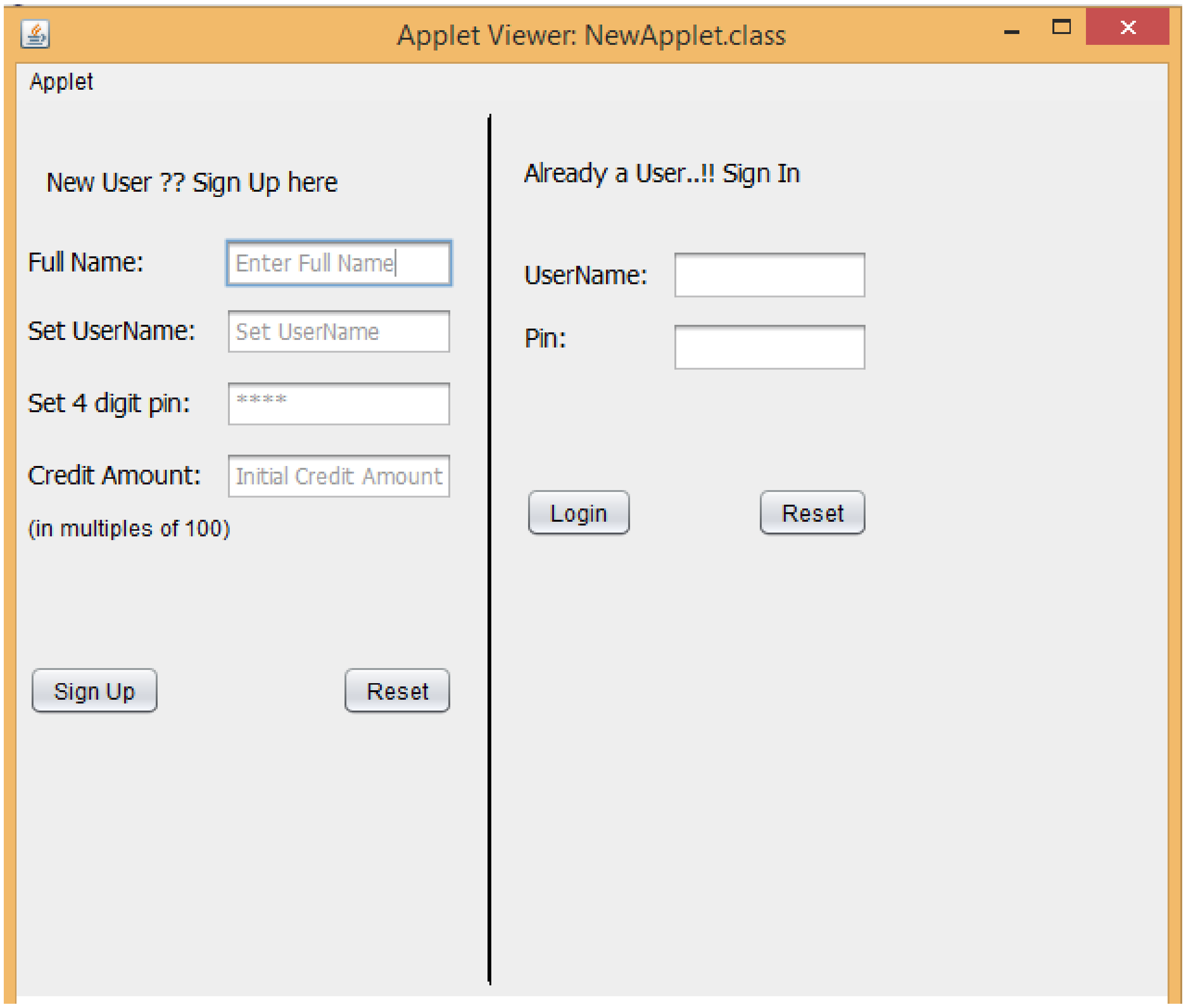
Practical Work

- **Requirements:**
  - **Functional Requirements:** Software used to design and implment this desktop application –
    - NetBeans 7.4
    - MySQL5.1.1 DataBase
    - mysql-connector-java-5.1.10-bin.jar



- **Design**
  - **Architecture Used:** Three-tier Architecture.
    - **Presentation Module:-**

      It consist of Graphical User Interface that the end-user will be facing after the completition of the project. It comprises of various swing components that act as a interface between system and end-user.



Applet Viewer: NewApplet.class

Applet

New User ?? Sign Up here

Full Name: Enter Full Name

Set UserName: Set UserName

Set 4 digit pin: ****

Credit Amount: Initial Credit Amount

(in multiples of 100)

Sign Up Reset

Already a User..!! Sign In

UserName:

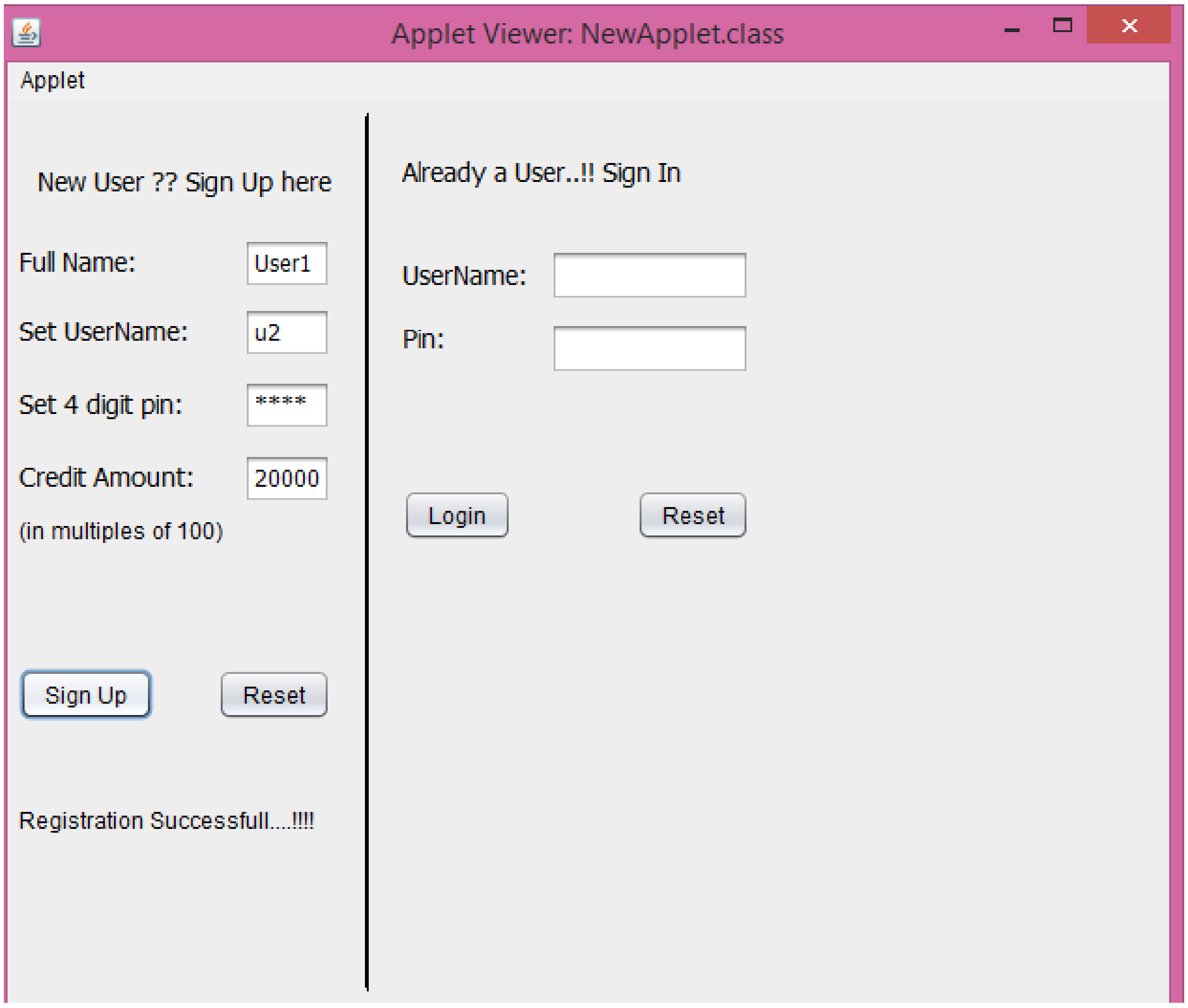
Pin:

Login Reset

Above figure shows the GUI of this project. It consist of swings component, namely, **JTextField, JPasswordField, JButton, JLabel.**

All the JButton components are registered with ActionListener Interface, i.e. they perform programmer defined functions when the command to execute is given.

On the LHS of the seperator shown in the GUI, there is a SignUp form, which performs Database insertQuery operation. On Successful registrstion of the user, result is shown beneath via JLabel.



New User ?? Sign Up here

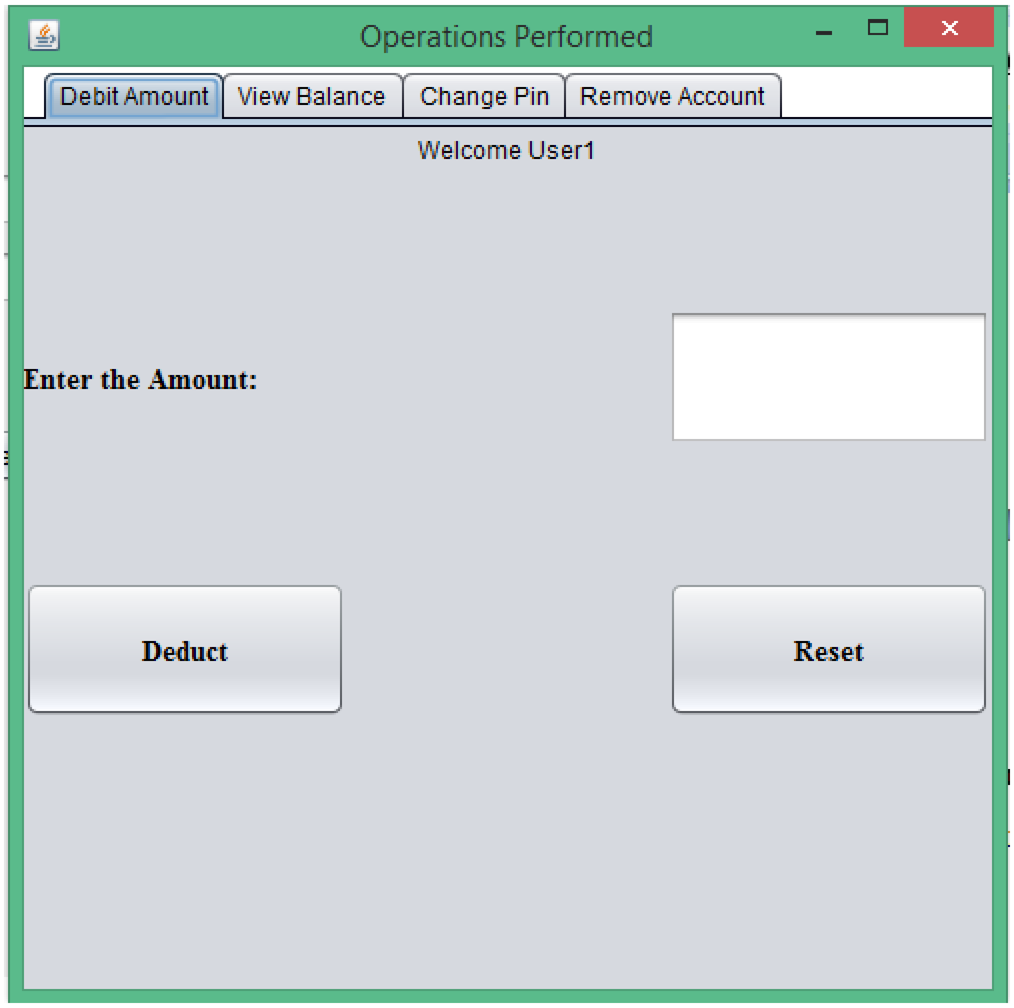
Full Name:	User1
Set UserName:	u2
Set 4 digit pin:	****
Credit Amount:	20000
(in multiples of 100)	

Registration Successfull....!!!!

Already a User..!! Sign In

UserName:	
Pin:	

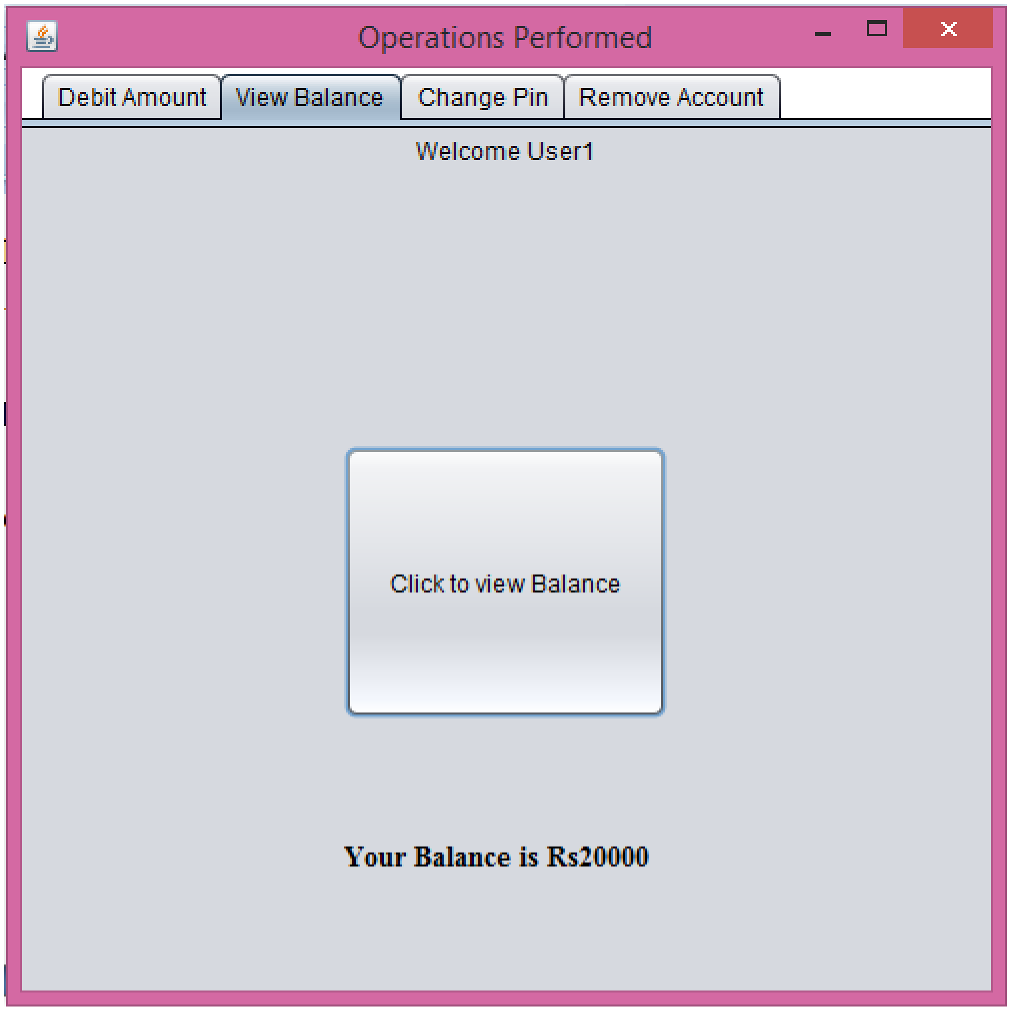
On the RHS of Seperator, GUI displays a LogIn form that ask for user credentials. On Successful LogIn of a user, there appears a new TabbedPane Window that shows various operations that a ATM Machine performs.



Initially, on the launch of the window, it displays “Debit Amount” TabbedPane which allows users to debit cash from their account.

This TabbedPane consist of Swings Component,namely, JButton, JTextField, JLabel;

JButton , registered with ActionListener Interface, performs functions embedded in it.



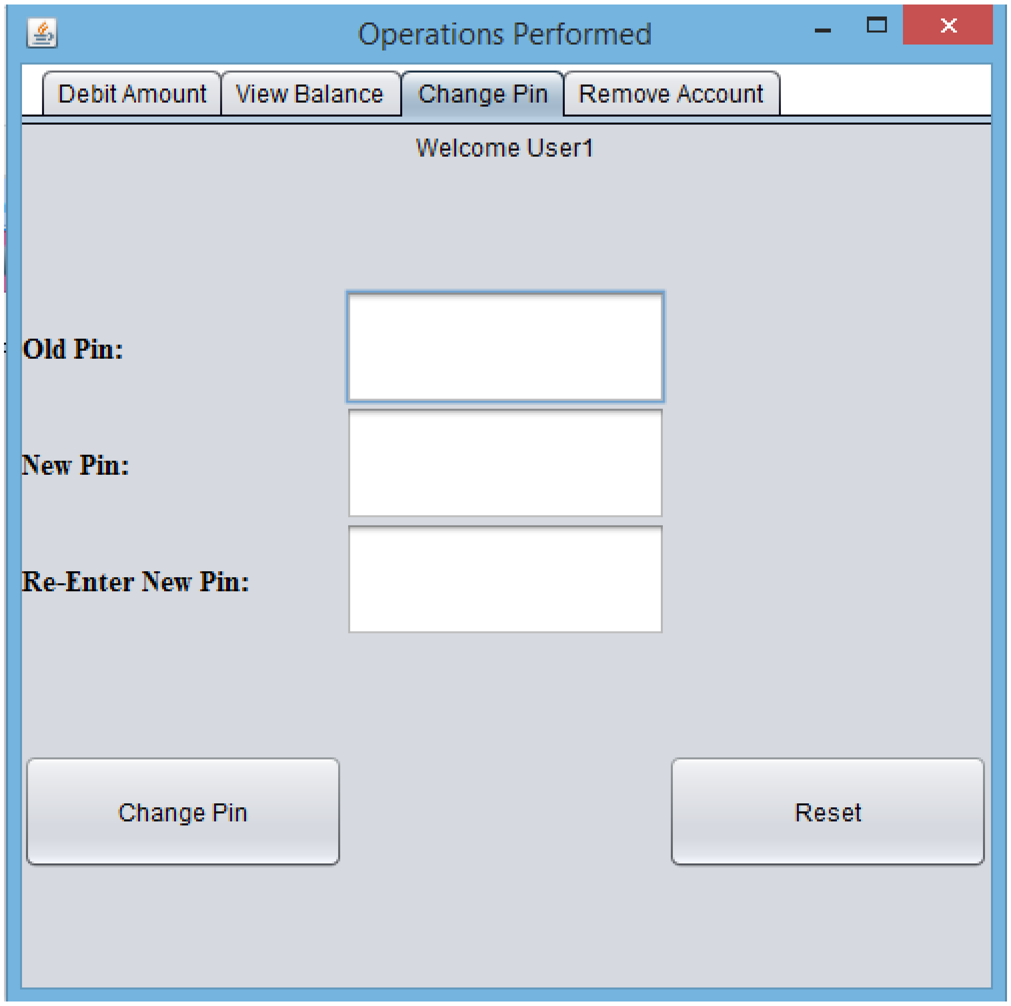
Welcome User1

Your Balance is Rs20000

Next is the "View Balance" TabbedPane that show the remaining balance in a User Account.

This TabbedPane also consists of JButton, that on clicking performs viewBalance() operation.

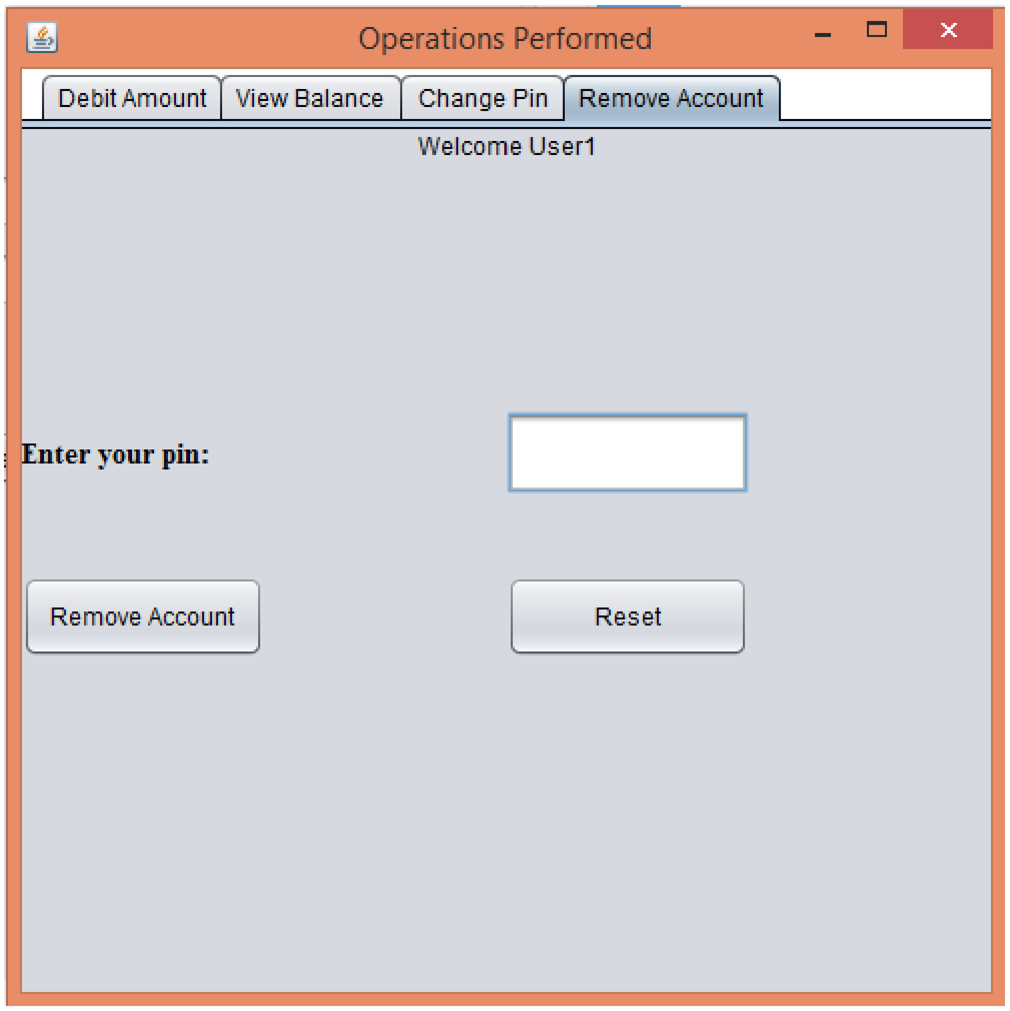
And also one JLabel, that displays the result of executing the viewBalance() function.



Change Pin TabbedPane offers its user to change the 4-digit pin that is used to successfully Login in their account.

This TabbedPane also consists of JButtons that perform functions that are assigned to them.

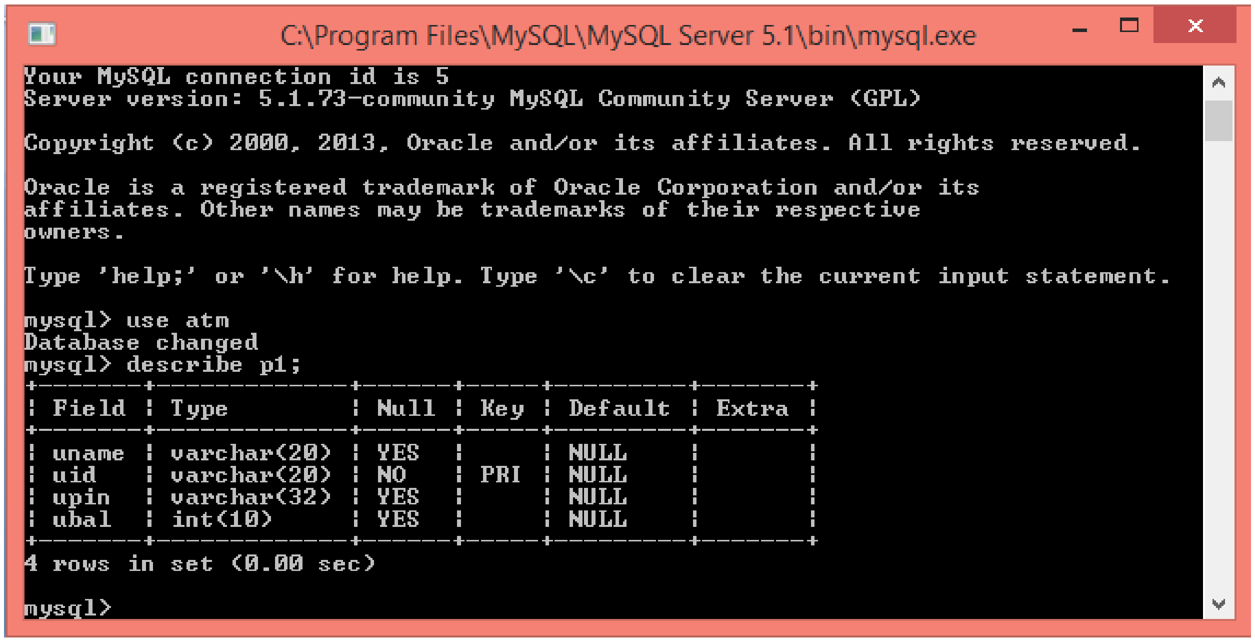
JButton "ChangePin" on clicking replaces the old value of pin with the new value of Pin in the database.



Remove Account Tabbedd Pane allow its user to delete their registered account from the database .

JButton “Remove Account” on clicking performs this operations. It checks the value in JTextfield (for pin), and if the pin matches with the value in the Textfield, Account will be deleted.

- **Data Module:** This module contains Database Connectivity with the Project. Here all the alterations are made that are requested by the User. It maintains a table that stores user data in various fields, maintaining Data Concurrency, Data Reliablity.



```
Your MySQL connection id is 5
Server version: 5.1.73-community MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use atm
Database changed
mysql> describe p1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| uname | varchar(20) | YES  |     | NULL    |       |
| uid   | varchar(20) | NO   | PRI | NULL    |       |
| upin  | varchar(32) | YES  |     | NULL    |       |
| ubal  | int(10)     | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql>
```

Here table named "p1" is maintaing all the transactions, alterations that are made by the user.

Fields used:

1. Uname – Stores the Name of User.
2. Uid – Stores the uid of the user. It is the primary key in this table, that uniquely identifies the user, that is why it can't be null.
3. Upin – Stores the Upin of the user in 32-bit HexaDecimal Format(encrypted).
4. Ubal – Stores the Initial Credit Amount that is set by user.

- **Logic Tier:** Initialization and memory allocation to variables, Deals with the actions performed when certain input is given or a particular action is taken.

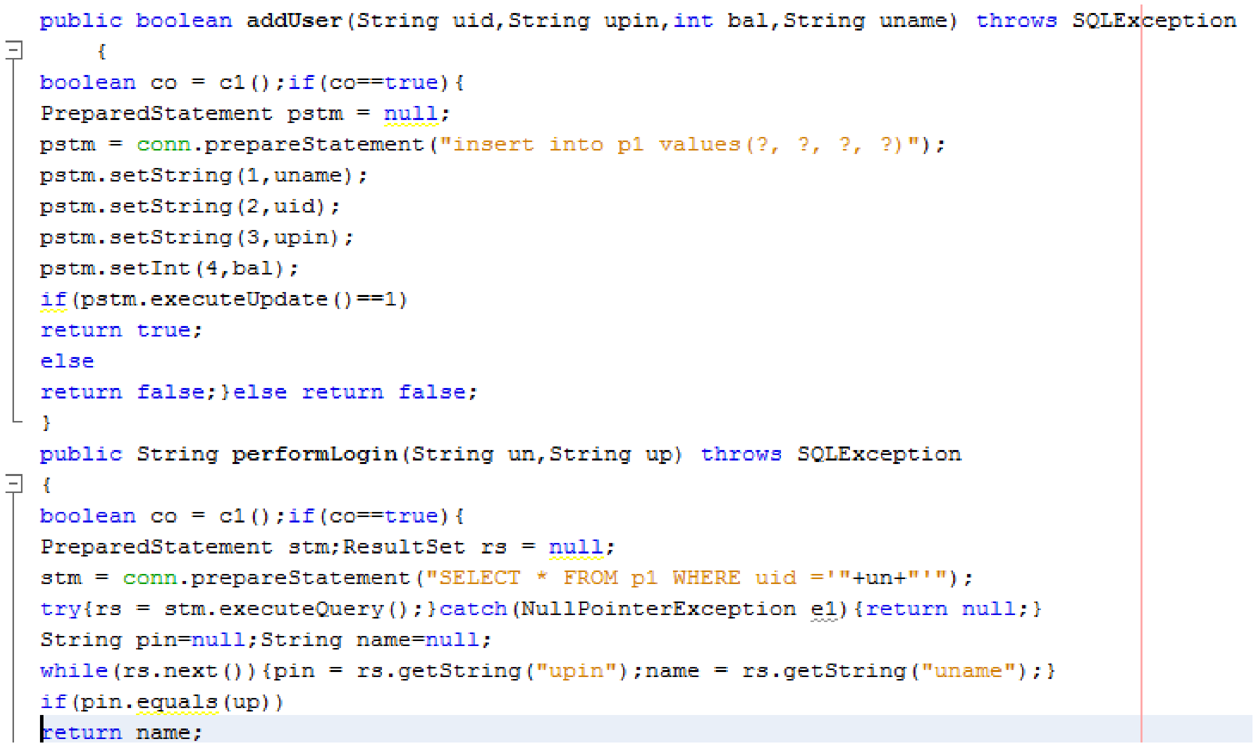
  For Instance:

  private void adduserActionPerformed(java.awt.event.ActionEvent evt);

  private void uloginActionPerformed(java.awt.event.ActionEvent evt) ;

  Functions with their Arguments:

  - addUser(uname,uid,upin,bal): Function Named addUser with arguments as specified is used to register any new User.
  - ulogin(uid,upin): Function Named ulogin with arguments as specified is used to perform Login Operation for a User.



```
public boolean addUser(String uid,String upin,int bal,String uname) throws SQLException
    {
boolean co = c1();if(co==true){
PreparedStatement pstm = null;
pstm = conn.prepareStatement("insert into p1 values(?, ?, ?, ?)");
pstm.setString(1,uname);
pstm.setString(2,uid);
pstm.setString(3,upin);
pstm.setInt(4,bal);
if(pstm.executeUpdate()==1)
return true;
else
return false;}else return false;
}
public String performLogin(String un,String up) throws SQLException
{
boolean co = c1();if(co==true){
PreparedStatement stm;ResultSet rs = null;
stm = conn.prepareStatement("SELECT * FROM p1 WHERE uid ='"+un+"'");
try{rs = stm.executeQuery();}catch(NullPointerException e1){return null;}
String pin=null;String name=null;
while(rs.next()){pin = rs.getString("upin");name = rs.getString("uname");}
if(pin.equals(up))
return name;
```

Code Files for the Project Implementation:

NewApplet.java

```
import java.awt.*;
import java.security.*;
import java.sql.*;
import javax.swing.text.*;
final class LengthRestrictedDocument extends PlainDocument
{
  private final int limit;
  public LengthRestrictedDocument(int limit){
    this.limit = limit;
}
@Override
public void insertString(int offs,String str,AttributeSet a)throws BadLocationException
  {
    if(str==null)
      return;
    if((getLength()+str.length())<=limit){
      super.insertString(offs,str,a);
    }
  }
}
class CryptWithMD5 {
   MessageDigest md;
public String cryptWithMD5(String pin){
    try {
        md = MessageDigest.getInstance("MD5");
        byte[] pinBytes = pin.getBytes();
        md.reset();
        byte[] digested = md.digest(pinBytes);
        StringBuffer sb = new StringBuffer();
        for(int i=0;i<digested.length;i++){
            sb.append(Integer.toHexString(0xff&digested[i]));
        }
        return sb.toString();
    } catch (Exception ex) {}
        return null;
   }
}
public class NewApplet extends javax.swing.JApplet {
    @Override
    public void init() {
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(NewApplet.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (InstantiationException ex) {
```

```
java.util.logging.Logger.getLogger(NewApplet.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(NewApplet.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(NewApplet.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        }
        //</editor-fold>
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jlabel1 = new javax.swing.JLabel();
        uname = new javax.swing.JTextField();
```

```
jLabel5 = new javax.swing.JLabel();
uid = new javax.swing.JTextField();
jLabel6 = new javax.swing.JLabel();
upin = new javax.swing.JPasswordField();
upin.setDocument(new LengthRestrictedDocument(4));
jLabel7 = new javax.swing.JLabel();
ubal = new javax.swing.JTextField();
adduser = new javax.swing.JButton();
jLabel2 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
un = new javax.swing.JTextField();
jLabel4 = new javax.swing.JLabel();
up = new javax.swing.JPasswordField();
ulogin = new javax.swing.JButton();
reset2 = new javax.swing.JButton();
userStatus = new javax.swing.JLabel();
reset1 = new javax.swing.JButton();
loginStatus = new javax.swing.JLabel();
signStatus = new javax.swing.JLabel();
jLabel8 = new javax.swing.JLabel();
jSeparator1 = new javax.swing.JSeparator();

jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
jLabel1.setText("New User ?? Sign Up here");

jlabel1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
jlabel1.setText("Full Name:");

uname.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
uname.setForeground(new java.awt.Color(153, 153, 153));
uname.setText("Enter Full Name");
uname.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        unameMouseClicked(evt);
    }
});

jLabel5.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
jLabel5.setText("Set UserName:");

uid.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
uid.setForeground(new java.awt.Color(153, 153, 153));
uid.setText("Set UserName");
uid.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        uidMouseClicked(evt);
    }
});

jLabel6.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
jLabel6.setText("Set 4 digit pin:");

upin.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
upin.setForeground(new java.awt.Color(153, 153, 153));
upin.setText("4pin");
upin.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        upinMouseClicked(evt);
    }
});

jLabel7.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
```

```
jLabel7.setText("Credit Amount:");

ubal.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
ubal.setForeground(new java.awt.Color(153, 153, 153));
ubal.setText("Initial Credit Amount");
ubal.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      ubalMouseClicked(evt);
   }
});

adduser.setText("Sign Up");
adduser.setAutoscrolls(true);
adduser.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      adduserActionPerformed(evt);
   }
});

jLabel2.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
jLabel2.setText("Already a User..!! Sign In");

jLabel3.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
jLabel3.setText("UserName:");

jLabel4.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
jLabel4.setText("Pin:");

up.setDocument(new LengthRestrictedDocument(4));
up.setMaximumSize(new java.awt.Dimension(2, 2));
```

```
        ulogin.setText("Login");
        ulogin.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                uloginActionPerformed(evt);
            }
        });

        reset2.setText("Reset");
        reset2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                reset2ActionPerformed(evt);
            }
        });

        reset1.setText("Reset");
        reset1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                reset1ActionPerformed(evt);
            }
        });

        loginStatus.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N

        jLabel8.setText("(in multiples of 100)");

        jSeparator1.setBackground(new java.awt.Color(0, 0, 0));
        jSeparator1.setForeground(new java.awt.Color(0, 0, 0));
        jSeparator1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 2,
true));

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(10, 10, 10)
                        .addComponent(jLabel1))
                    .addComponent(signStatus)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(jLabel7)
                            .addComponent(jLabel6)
                            .addComponent(jlabel1, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(adduser)
                            .addComponent(jLabel5, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

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

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(reset1)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                                .addComponent(uname, javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(ubal)
                        .addComponent(upin)
                        .addComponent(uid, javax.swing.GroupLayout.Alignment.LEADING))))
                  .addGroup(layout.createSequentialGroup()
                    .addGap(69, 69, 69)
                    .addComponent(userStatus))
                  .addComponent(jLabel8))
                .addGap(18, 18, 18)
                .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 2,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                  .addComponent(loginStatus)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(layout.createSequentialGroup()
                      .addComponent(ulogin)
                      .addGap(66, 66, 66)
                      .addComponent(reset2))
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel3)
                        .addComponent(jLabel4))

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

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(un, javax.swing.GroupLayout.DEFAULT_SIZE, 107,
Short.MAX_VALUE)
```

```
                        .addComponent(up, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
                .addComponent(jLabel2))
              .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
      layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                  .addGap(35, 35, 35)
                  .addComponent(jLabel1)
                  .addGap(22, 22, 22)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jlabel1)
                    .addComponent(uname, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGroup(layout.createSequentialGroup()
                  .addGap(30, 30, 30)
                  .addComponent(jLabel2)
                  .addGap(33, 33, 33)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(un, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3))))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(3, 3, 3)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel5)
                            .addComponent(uid, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel6)
                            .addComponent(upin, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

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

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel7)
                            .addComponent(ubal, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jLabel8)

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

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 69,
Short.MAX_VALUE)
                                .addComponent(userStatus)
                                .addGap(106, 106, 106))
                            .addGroup(layout.createSequentialGroup()
                                .addGap(66, 66, 66)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(adduser)
                                    .addComponent(reset1))
                                .addGap(45, 45, 45)
                                .addComponent(signStatus)
                                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(11, 11, 11)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel4)
                            .addComponent(up, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(61, 61, 61)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(ulogin)
                            .addComponent(reset2))
                        .addGap(53, 53, 53)
                        .addComponent(loginStatus)
                        .addContainerGap())))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jSeparator1)
                .addContainerGap())
        );
    }// </editor-fold>//GEN-END:initComponents
    private void reset2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_reset2ActionPerformed
        un.setText("");up.setText("");loginStatus.setText("");
    }//GEN-LAST:event_reset2ActionPerformed

    private void adduserActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_adduserActionPerformed
        boolean status = false;CryptWithMD5 c = null;
        DataRetrieve data = new DataRetrieve();String p11 = null;
        int amt1=0;String uid1 = null,pin = null,uname1= null;
        try
        {
        c = new CryptWithMD5();
        uname1 = uname.getText();
        upin.setSelectionStart(0);
        upin.setSelectionEnd(4);
        pin = upin.getSelectedText();
        uid1 = uid.getText();
        amt1 = Integer.parseInt(ubal.getText());}catch(Exception e) {}
        p11 = c.cryptWithMD5(pin);
        System.out.println(p11);
        int p1 = 0,p2=0;
        if(p2==0){try{p1 = Integer.parseInt(pin);}
        catch(Exception e){signStatus.setText("Pin should be 4-digit number");p2=-1;}}
        if(p2==0)
        {
            if(uname1.length()>=1&&uid1!=""&&amt1!=0&&amt1%100==0&&p1>=1000)
{try{status = data.addUser(uid1,p11,amt1,uname1);}catch(SQLException e){}}
if(status==true){
        signStatus.setText("Registration
Successfull....!!!!");System.out.println("Name:"+uname.getText());
```

```
        }
        else
        signStatus.setText("Fill in Exact details or Try with some other username");
        }
        else
        {
        signStatus.setText("Fill In Exact Details");
        }
    }//GEN-LAST:event_adduserActionPerformed
    private void unameMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_unameMouseClicked
        // TODO add your handling code here:
        if("Enter Full Name".equals(uname.getText())){
            uname.setText(null);
            uname.setForeground(Color.BLACK);
        }
    }//GEN-LAST:event_unameMouseClicked

    private void uidMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_uidMouseClicked
        // TODO add your handling code here:
        if("Set UserName".equals(uid.getText())){
            uid.setText("");uid.setForeground(Color.BLACK);}
    }//GEN-LAST:event_uidMouseClicked

    private void ubalMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_ubalMouseClicked
        // TODO add your handling code here:
        if("Initial Credit Amount".equals(ubal.getText())){
            ubal.setText("");ubal.setForeground(Color.BLACK);}
    }//GEN-LAST:event_ubalMouseClicked
    private void upinMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_upinMouseClicked
        // TODO add your handling code here:
    upin.setForeground(Color.BLACK);
    }//GEN-LAST:event_upinMouseClicked

    private void reset1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_reset1ActionPerformed
        // TODO add your handling code here:
        uname.setText("Enter Full Name");uid.setText("Set
UserName");upin.setText("4pin");ubal.setText("Initial Credit Amount");

uname.setForeground(Color.GRAY);uid.setForeground(Color.GRAY);upin.setForeground(Colo
r.GRAY);ubal.setForeground(Color.GRAY);
        signStatus.setText("");
    }//GEN-LAST:event_reset1ActionPerformed
    private void uloginActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_uloginActionPerformed
        String un1 = null;String up1 = null;String up11 = null;
        DataRetrieve dr = new DataRetrieve();CryptWithMD5 c;
        String status = null;c = new CryptWithMD5();
        try{un1 = un.getText();}catch(Exception e){}
        up.setSelectionStart(0);up.setSelectionEnd(4);up1 = up.getSelectedText();up11 =
c.cryptWithMD5(up1);
        try{status = dr.performLogin(un1,up11);}catch(SQLException e){}
        if(status==null)
        loginStatus.setText("Login Failed");
        else{
        Operations window1;
        window1 = new Operations("Operations Performed",status,un1);
        window1.setSize(500,500);
          window1.setVisible(true);
        }
    }
    private javax.swing.JButton adduser;
    private javax.swing.JLabel jLabel1;
    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.JSeparator jSeparator1;
    private javax.swing.JLabel jlabel1;
    private javax.swing.JLabel loginStatus;
    private javax.swing.JButton reset1;
    private javax.swing.JButton reset2;
    private javax.swing.JLabel signStatus;
    private javax.swing.JTextField ubal;
    private javax.swing.JTextField uid;
    private javax.swing.JButton ulogin;
    private javax.swing.JTextField un;
    private javax.swing.JTextField uname;
    private javax.swing.JPasswordField up;
    private javax.swing.JPasswordField upin;
    private javax.swing.JLabel userStatus;
}
```

- **Operations.java**

```
import javax.swing.*;
import java.sql.*;
import java.util.Date;
import java.awt.event.*;
import java.awt.*;
import java.security.MessageDigest;
public class Operations extends Frame
{

   Operations(String title,String user,String uid)
                    {
            super(title);
            JTabbedPane jtp = new JTabbedPane();
            jtp.addTab("Debit Amount", new DebitAmount(user,uid));
                  jtp.addTab("View Balance", new ViewBalance(user,uid));
                  jtp.addTab("Change Pin", new ChangePin(user,uid));
            jtp.addTab("Remove Account", new RemoveAccount(user,uid));
                  add(jtp);
                  }
}
class ChangePin extends JPanel implements ActionListener
         {
         JLabel l1,l2,l3;JButton b1,b2;JLabel l4 = new JLabel("");
         JPasswordField f1,f2,f3;String uid1;
                  public ChangePin(String user,String uid)
                  {
               uid1 = uid;
                JPanel rem = new JPanel(new GridLayout(7,3));
                setLayout(new BorderLayout());
                add(new Label("Welcome "+user,Label.CENTER),BorderLayout.NORTH);
                add(rem,BorderLayout.CENTER);b1 = new JButton("Change
Pin");b1.addActionListener((ActionListener) this);
                b2 = new JButton("Reset"); b2.addActionListener((ActionListener) this);
                f1 = new JPasswordField(10);f2 = new JPasswordField(10);f3 = new
JPasswordField(10);
f1.setDocument(new LengthRestrictedDocument(4));f2.setDocument(new
LengthRestrictedDocument(4));f3.setDocument(new LengthRestrictedDocument(4));
                l1 = new JLabel("Old Pin:");l2 = new JLabel("New Pin:"); l3 = new JLabel("Re-
Enter New Pin:");
                Font f = new
Font("TimesRoman",Font.BOLD,14);l1.setFont(f);l2.setFont(f);l3.setFont(f);
                f1.setFont(f);f2.setFont(f);f3.setFont(f);
                rem.add(new Label());rem.add(new Label());rem.add(new Label());
                rem.add(l1);rem.add(f1);rem.add(new
Label());rem.add(l2);rem.add(f2);rem.add(new Label());
```

```
                rem.add(l3);rem.add(f3);rem.add(new Label());
                rem.add(new Label());rem.add(new Label());rem.add(new Label());rem.add(b1);
                rem.add(new Label());rem.add(b2);rem.add(new Label());rem.add(l4);rem.add(new
Label());
                 }

   @Override
   public void actionPerformed(ActionEvent e) {
      if(e.getSource()==b2)
      {
      f1.setText("");f2.setText("");f3.setText("");l4.setText("");
      }
      if(e.getSource()==b1)
      {
         Encrypt e1 = new Encrypt();int i;
         boolean status = false,s =false;
         String oldPin,newPin,np1;
         f1.setSelectionStart(0);f1.setSelectionEnd(4);oldPin = f1.getSelectedText();
         f2.setSelectionStart(0);f2.setSelectionEnd(4);newPin = f2.getSelectedText();
         f3.setSelectionStart(0);f3.setSelectionEnd(4);np1 = f3.getSelectedText();
         String o1 = e1.cryptWithMD5(oldPin);
         DataRetrieve d = new DataRetrieve();i = Integer.parseInt(np1);
         try{status = d.changePin(o1,newPin,np1,uid1);}
         catch(SQLException ex){}
         if(status==true&&i>=0&&np1.length()==4)
         {
           String o2 = e1.cryptWithMD5(np1);
            try{s = d.cnfmCP(o2,uid1);}catch(Exception ex){}
            if(s==true)
               l4.setText("Successfully Changed Pin");
            else
               l4.setText("Connection Error");
         }
         else l4.setText("Insert Correct Details in Correct format");
      }
   }
         }
class RemoveAccount extends JPanel implements ActionListener
{
   JButton b1,b2;JLabel l1 = new JLabel();
   String u;JPasswordField f;
   RemoveAccount(String user,String uid)
   {
      u = uid;
      JLabel l2 = new JLabel("Welcome "+user,JLabel.CENTER);
      JPanel rem = new JPanel(new GridLayout(10,3));
```

```
        setLayout(new BorderLayout());
           add(l2,BorderLayout.NORTH);
        add(rem,BorderLayout.CENTER);
        Font f1 = new Font("TimesRoman",Font.BOLD,14);
        JLabel l = new JLabel("Enter your pin:");l.setFont(f1);
        f = new JPasswordField(10);f.setDocument(new LengthRestrictedDocument(4));
        b1 = new JButton("Remove Account");b1.addActionListener(this);
        b2 = new JButton("Reset");b2.addActionListener(this);
        rem.add(new Label());rem.add(new Label());rem.add(new Label());rem.add(new
Label());rem.add(new Label());rem.add(new Label());
        rem.add(new Label());rem.add(new Label());rem.add(new Label());rem.add(new
Label());rem.add(new Label());rem.add(new Label());
        rem.add(l);rem.add(new Label());rem.add(f);rem.add(new Label());rem.add(new
Label());rem.add(new Label());
        rem.add(new Label());rem.add(new Label());rem.add(b1);rem.add(new
Label());rem.add(b2);
        rem.add(new Label());rem.add(new Label());rem.add(l1);rem.add(new
Label());rem.add(new Label());rem.add(new Label());rem.add(new Label());rem.add(new
Label());
    }
 public void actionPerformed(ActionEvent ae)
    {
    if(ae.getSource()==b1)
    {
        boolean status = false;
        String pin;Encrypt p1 = new Encrypt();
        f.setSelectionStart(0);f.setSelectionEnd(4);pin = f.getSelectedText();
    DataRetrieve d = new DataRetrieve();
    pin = p1.cryptWithMD5(pin);Font f1 = new
Font("TimesRoman",Font.BOLD,16);l1.setFont(f1);
    try{status = d.removeAcc(u,pin);}catch(Exception e){};
    if(status==true){
        l1.setText("Account Deleted");
}
    else
        l1.setText("Wrong Pin Entered");
    }
    if(ae.getSource()==b2)
    {f.setText("");l1.setText("");}
    }
}
class DebitAmount extends JPanel implements ActionListener
{
    JLabel l1; JTextField f1 = new JTextField();JLabel l2 = new JLabel();
    JButton b1; String uid1=null;JButton b2 = new JButton("Reset");
    public DebitAmount(String user, String uid)
```

```
            {
            JPanel rem = new JPanel(new GridLayout(6,3));
            setLayout(new BorderLayout());uid1 = uid;
            add(new Label("Welcome "+user,Label.CENTER),BorderLayout.NORTH);
            add(rem,BorderLayout.CENTER);b1= new
JButton("Deduct");b1.addActionListener((ActionListener)this);
            l1 = new JLabel("Enter the Amount:");b2.addActionListener(this);
            Font f = new
Font("TimesRoman",Font.BOLD,14);l1.setFont(f);b1.setFont(f);b2.setFont(f);l2.setFont(f);
            rem.add(new JLabel());rem.add(new JLabel());rem.add(new
JLabel());rem.add(l1);rem.add(new JLabel());rem.add(f1);
            rem.add(new JLabel());rem.add(new JLabel());rem.add(new
JLabel());rem.add(b1);rem.add(new JLabel());rem.add(b2);rem.add(new JLabel());
            rem.add(new JLabel());rem.add(new JLabel());rem.add(new
JLabel());rem.add(l2);rem.add(new JLabel());
            }
   public void actionPerformed(ActionEvent ae)
   {
   if(ae.getSource()==b1)
   {
   Date d1 = new Date();
   int i = Integer.parseInt(f1.getText());
   if(i>0&&(i%100)==0){
      DataRetrieve d = new DataRetrieve();
   int j = 0;
   try{j = d.deductAmt(uid1,i);}catch(SQLException e){}
   if(j>=1000) l2.setText("Balance left\n Rs."+j);
   else if(j==0) l2.setText("Connection Error");
   else if(j==-1) l2.setText("Insufficient Amount");
   }
   else{ l2.setText("Only Numerical Data");}
   }
   if(ae.getSource()==b2)
   {f1.setText("");l2.setText("");}
   }
}
class ViewBalance extends JPanel implements ActionListener
            {
   JButton b1;JLabel l1;String uid1 = null;
                  public ViewBalance(String user,String uid)
                  {
                     JPanel rem = new JPanel(new GridLayout(3,3));
                     l1 = new JLabel();uid1 = uid;
                     setLayout(new BorderLayout());
                     add(new Label("Welcome "+user,Label.CENTER),BorderLayout.NORTH);
                     add(rem,BorderLayout.CENTER);
```

```
                        b1 = new JButton("Click to view
Balance");b1.addActionListener((ActionListener) this);
                        rem.add(new JLabel());rem.add(new JLabel());rem.add(new
JLabel());rem.add(new JLabel());
                        rem.add(b1);rem.add(new JLabel());rem.add(new
JLabel());rem.add(l1);rem.add(new JLabel());
                     }
              public void actionPerformed(ActionEvent ae)
              {
                 boolean stat = false;int i =0;
                        Font f = new Font("TimesRoman",Font.BOLD,14);l1.setFont(f);
                 if(ae.getSource()==b1)
                 {
                 DataRetrieve d = new DataRetrieve();
                 try{i = d.viewBalance(uid1);}catch(Exception ex){}
                 if(i>0)l1.setText("Your Balance is Rs"+i);
                 else l1.setText("Error");
                 }
              }
           }
class Encrypt {
  MessageDigest md;
public String cryptWithMD5(String pin){
   try {
      md = MessageDigest.getInstance("MD5");
      byte[] pinBytes = pin.getBytes();
      md.reset();
      byte[] digested = md.digest(pinBytes);
      StringBuffer sb = new StringBuffer();
      for(int i=0;i<digested.length;i++){
         sb.append(Integer.toHexString(0xff&digested[i]));
      }
      return sb.toString();
   } catch (Exception ex) {}
      return null;
  }
}
```

▪ **DataRetrieve.java**

```
import java.sql.* ;
public class DataRetrieve {
private Connection conn ;
private boolean createConnection()
   {
      try
      {
          Class.forName("com.mysql.jdbc.Driver");
          conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/atm","root","apple");
          return true ;
      }
      catch(Exception e){}
      return false ;
}
public boolean c1()
{
   if(conn==null)
      {
       if(!createConnection())
         {
             System.out.println("Connecton error");
               return false ;
         }
      }
   return true;
}
public boolean addUser(String uid,String upin,int bal,String uname) throws SQLException
{
boolean co = c1();if(co==true){
PreparedStatement pstm = null;
pstm = conn.prepareStatement("insert into p1 values(?, ?, ?, ?)");
pstm.setString(1,uname);
pstm.setString(2,uid);
pstm.setString(3,upin);
pstm.setInt(4,bal);
if(pstm.executeUpdate()==1)
return true;
else
return false;}else return false;
}
public String performLogin(String un,String up) throws SQLException
{
boolean co = c1();if(co==true){
PreparedStatement stm;ResultSet rs = null;
stm = conn.prepareStatement("SELECT * FROM p1 WHERE uid ='"+un+"'");
try{rs = stm.executeQuery();}catch(NullPointerException e1){return null;}
String pin=null;String name=null;
while(rs.next()){pin = rs.getString("upin");name = rs.getString("uname");}
if(pin.equals(up))
return name;
else
return null;
}else return null;}
public int deductAmt(String uid,int i) throws SQLException
{
boolean stat = c1();if(stat==true){
   PreparedStatement pst = null;int bal = 0;
   pst = conn.prepareStatement("Select * from p1 where uid = '"+uid+"'");
   ResultSet rs = pst.executeQuery();
   while(rs.next()){bal = rs.getInt("ubal");}
   if(bal>i){bal-=i;
   PreparedStatement pst1 = null;
   pst1 = conn.prepareStatement("Update p1 set ubal ="+bal+" where uid ='"+uid+"'");
   pst1.executeUpdate();
         return  bal;}
   else return -1;
}return 0;
}
public int viewBalance(String uid) throws SQLException
{
int bal = -1;boolean co = c1();if(co==true){
   PreparedStatement s;
   s = conn.prepareStatement("Select * from p1 where uid ='"+uid+"'");
   ResultSet rs = s.executeQuery();
    while(rs.next()){try{bal = rs.getInt("ubal");}catch(Exception ex){}}
return bal;}else return -1;
}
public boolean removeAcc(String uid,String pin) throws SQLException
{
if(conn==null)
      {
        if(!createConnection())
          {
              System.out.println("Connecton error");
                return false ;
          }
      }
String p1 = null;
PreparedStatement pst;
pst = conn.prepareStatement("Select * from p1 where uid ='"+uid+"'");
```

```
ResultSet rs = pst.executeQuery();
while(rs.next()){try{p1 = rs.getString("upin");}catch(Exception e){}}
if(p1.equals(pin)){
PreparedStatement pst1 = null;pst1 = conn.prepareStatement("Delete from p1 where uid =
'"+uid+"'");
int i = pst1.executeUpdate();
return true;}
else
return false;
}
public boolean changePin(String o,String n,String n1,String uid) throws SQLException
{
boolean co = c1();if(co==true){
   PreparedStatement pst;String o1 = null;
   pst = conn.prepareStatement("Select * from p1 where uid ='"+uid+"'");
   ResultSet rs = pst.executeQuery();
   while(rs.next()){try{o1 = rs.getString("upin");}catch(Exception e){}}
   if(o1.equals(o)&&n.equals(n1))
   return true;
   else return false;}else return false;
}
public boolean cnfmCP(String np,String uid) throws SQLException
{PreparedStatement pst;
pst = conn.prepareStatement("UPDATE p1 SET upin ='"+np+"' where uid ='"+uid+"'");
int i = pst.executeUpdate();
if(i==1)
return true;
else return false;}}
```

References:

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

[2] http://infolab.stanford.edu/

[3] http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp?topic=/com.ibm.zos.zappldev/zappldev_11.htm

[4] http://www.webopedia.com/TERM/A/Application_Lifecycle_Management.html

[5] Core Java Volume I Fundamentals (8 ed. September 2007 Prentice Hall) by Cay S. Horstmann and Gary Cornell

[6] Core Java 2- Volume I - Fundamentals, 5th Edition by Cay S. Horstmann and Gary Cornell