BANKING SYSTEM

A MINI-PROJECT BY:

Karneesh D	**230701141**
Lokeshwaraprasad	**230701167**

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project **“BANKING SYSTEM”** is the
bonafide work of **“KARNEESH D(230701141)” and “LOKESHWARAPRASAD (230701167)”** who carried out the project work under my supervision.

Submitted for the practical examination held on ____________________

Signature
Ms.Dharshini B S
Assistant Professor
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

BANKING SYSTEM, sometimes known as a project for day to day life completed as a part of undergraduate or postgraduate degree. Typically, a banking system allows students present their application for the banks to update their day to day work in systems.

The project helps the people to create an account, deposit money, withdraw money, delete the existing account and search an account in the bank.

Our project aims to streamline BANKING SYSTEM process and provide support to bankers. Bankers will do their everyday work with this application so that they manage large number of customers.

The BANKING SYSTEM will be used as a part of banking in this new era. The banking process will be simplified as easy as possible, so that the customers will enjoy to use this application.

TABLE OF CONTENTS

INTRODUCTION

1.0 INTRODUCTION

The project helps the bankers to do their banking work and manage the customers easily in day to day life. They can easily manage all the customers and functioning the banking process without any difficulty.

1.1 IMPLEMENTATION

The **BANKING SYSTEM** project discussed here is implemented using the concepts of **JAVA SWINGS** and **MYSQL**.

1.2 SCOPE OF THE PROJECT

The project is designed in a way where bankers will have to create an account for customers and do all the banking work very fast and accurately. Thus saving them time and giving a sense of professionalism.

1.3 FEATURE OF THE PROJECT

In everyday life, the banking system is very important for all the sectors in our society. If searching an account, deposit or withdraw money from the bank takes more time then we lose the customers very easily. So the banking system helps the bankers to do their work very fast and very proper manner.

SYSTEM SPECIFICATIONS

2.0 HARDWARE SPECIFICATIONS:

PROCESSOR	:	Intel i10
MEMORY SIZE	:	GB(Minimum)
HARD DISK	:	100 GB of free space

2.1 SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE	:	Java, MySQL
FRONT-END	:	Java
BACK-END	:	MySQL
OPERATING SYSTEM	:	Windows 10 or other updated version

SAMPLE CODE

3.0 BANKING SYSTEM

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.util.*;
import java.text.*;
import java.io.*;
import java.awt.PrintJob.*;
import javax.swing.plaf.metal.*;

public class BankSystem extends JFrame implements ActionListener, ItemListener {
	private JDesktopPane desktop = new JDesktopPane ();
	private JMenuBar bar;
	private JMenu mnuFile, mnuEdit, mnuView, mnuOpt, mnuWin, mnuHelp;
	private JMenuItem addNew, printRec, end;
	private JMenuItem  deposit, withdraw, delRec, search, searchName;
	private JMenuItem oneByOne, allCustomer;
	private JMenuItem change, style, theme;
	private JMenuItem close, closeAll;
	private JMenuItem content, keyHelp, about;
	private JPopupMenu popMenu = new JPopupMenu ();
	private JMenuItem open, report, dep, with, del, find, all;
	private JToolBar toolBar;
	private JButton btnNew, btnDep, btnWith, btnRec, btnDel, btnSrch, btnHelp, btnKey;
	private JPanel statusBar = new JPanel ();
	private JLabel welcome;
	private JLabel author;
	private String strings[] = {"1. Metal", "2. Motif", "3. Windows"};
	private UIManager.LookAndFeelInfo looks[]= UIManager.getInstalledLookAndFeels ();
	private ButtonGroup group = new ButtonGroup ();
	private JRadioButtonMenuItem radio[] = new JRadioButtonMenuItem[strings.length];
	private java.util.Date currDate = new java.util.Date ();
	private SimpleDateFormat sdf = new SimpleDateFormat ("dd MMMM yyyy",
	Locale.getDefault());
	private String d = sdf.format (currDate);
	private int count = 0;
	private int rows = 0;
	private int total = 0;
	private String records[][] = new String [500][6];
	private FileInputStream fis;
	private DataInputStream dis;
	public BankSystem () {
```

```
 super ("BankSystem [Pvt] Limited.");
 UIManager.addPropertyChangeListener (new UISwitchListener
((JComponent)getRootPane()));
 bar = new JMenuBar ();
 setIconImage (getToolkit().getImage ("Images/Bank.gif"));
 setSize (700, 550);          setJMenuBar (bar);
 addWindowListener (new WindowAdapter () {
          public void windowClosing (WindowEvent we) {
                   quitApp ();
          }
 }                          );
 setLocation((Toolkit.getDefaultToolkit().getScreenSize().width  - getWidth()) / 2,
 (Toolkit.getDefaultToolkit().getScreenSize().height - getHeight()) / 2);
 mnuFile = new JMenu ("File");
 mnuFile.setMnemonic ((int)'F');
 mnuEdit = new JMenu ("Edit");
 mnuEdit.setMnemonic ((int)'E');
 mnuView = new JMenu ("View");
 mnuView.setMnemonic ((int)'V');
 mnuOpt = new JMenu ("Options");
 mnuOpt.setMnemonic ((int)'O');
 mnuWin = new JMenu ("Window");
 mnuWin.setMnemonic ((int)'W');
 mnuHelp = new JMenu ("Help");
 mnuHelp.setMnemonic ((int)'H');
 addNew = new JMenuItem ("Open New Account", new ImageIcon
("Images/Open.gif"));
 addNew.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_N,
Event.CTRL_MASK));
 addNew.setMnemonic ((int)'N');
 addNew.addActionListener (this);
 printRec = new JMenuItem ("Print Customer Balance", new ImageIcon
("Images/New.gif"));
 printRec.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_R,
Event.CTRL_MASK));
 printRec.setMnemonic ((int)'R');
 printRec.addActionListener (this);
 end = new JMenuItem ("Quit BankSystem ?", new ImageIcon ("Images/export.gif"));
 end.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_Q, Event.CTRL_MASK));
 end.setMnemonic ((int)'Q');
 end.addActionListener (this);
 deposit = new JMenuItem ("Deposit Money");
 deposit.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_T,
Event.CTRL_MASK));
 deposit.setMnemonic ((int)'T');
 deposit.addActionListener (this);
 withdraw = new JMenuItem ("Withdraw Money");
 withdraw.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_W,
Event.CTRL_MASK));
 withdraw.setMnemonic ((int)'W');
 withdraw.addActionListener (this);
 delRec = new JMenuItem ("Delete Customer", new ImageIcon ("Images/Delete.gif"));
 delRec.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_D,
Event.CTRL_MASK));
 delRec.setMnemonic ((int)'D');
 delRec.addActionListener (this);
 search = new JMenuItem ("Search By No.", new ImageIcon ("Images/find.gif"));
 search.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_S,
Event.CTRL_MASK));
 search.setMnemonic ((int)'S');
 search.addActionListener (this);
 searchName = new JMenuItem ("Search By Name");
 searchName.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_M,
Event.CTRL_MASK));
 searchName.setMnemonic ((int)'M');
 searchName.addActionListener (this);
 oneByOne = new JMenuItem ("View One By One");
 oneByOne.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_O,
Event.CTRL_MASK));
 oneByOne.setMnemonic ((int)'O');
 oneByOne.addActionListener (this);
 allCustomer = new JMenuItem ("View All Customer", new ImageIcon
("Images/refresh.gif"));
 allCustomer.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_A,
Event.CTRL_MASK));
 allCustomer.setMnemonic ((int)'A');
 allCustomer.addActionListener (this);
 change = new JMenuItem ("Change Background Color");
 change.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_B,
Event.CTRL_MASK));
 change.setMnemonic ((int)'B');
 change.addActionListener (this);
 style = new JMenu ("Change Layout Style");
 style.setMnemonic ((int)'L');
 for( int i = 0; i < radio.length ; i++ ) {            //Creating the subMenus of Style
Menu.
       radio[i] = new JRadioButtonMenuItem (strings[i]); //Build an Array of Layouts to
Apply.
       radio[i].addItemListener (this);                //Setting their Actions.
       group.add (radio[i]);                                   //Making them Grouped.
       style.add (radio[i]);                                   //Adding to Style
MenuOption.
 }
 MetalTheme[] themes = { new DefaultMetalTheme(), new GreenTheme(), new
AquaTheme(),
                              new SandTheme(), new SolidTheme(), new MilkyTheme(), new
GrayTheme() };
 theme = new MetalThemeMenu ("Apply Theme", themes);                  //Putting the
Themes in ThemeMenu.
 theme.setMnemonic ((int)'M');
 close = new JMenuItem ("Close Active Window");
 close.setMnemonic ((int)'C');
 close.addActionListener (this);
 closeAll = new JMenuItem ("Close All Windows...");
 closeAll.setMnemonic ((int)'A');
 closeAll.addActionListener (this);
 content = new JMenuItem ("Help Contents", new ImageIcon ("Images/paste.gif"));
 content.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_H,
Event.CTRL_MASK));
 content.setMnemonic ((int)'H');
 content.addActionListener (this);
 keyHelp = new JMenuItem ("Help on Shortcuts...");
 keyHelp.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_K,
Event.CTRL_MASK));
 keyHelp.setMnemonic ((int)'K');
 keyHelp.addActionListener (this);
 about = new JMenuItem ("About BankSystem", new ImageIcon ("Images/Save.gif"));
 about.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_C,
Event.CTRL_MASK));
 about.setMnemonic ((int)'C');    about.addActionListener (this);
 mnuFile.add (addNew);            mnuFile.addSeparator ();
 mnuFile.add (printRec);          mnuFile.addSeparator ();
 mnuFile.add (end);               mnuEdit.add (deposit);
 mnuEdit.add (withdraw);          mnuEdit.addSeparator ();
 mnuEdit.add (delRec);            mnuEdit.addSeparator ();
 mnuEdit.add (search);            mnuEdit.add (searchName);
 mnuView.add (oneByOne);          mnuView.addSeparator ();
 mnuView.add (allCustomer);       mnuOpt.add (change);
 mnuOpt.addSeparator ();          mnuOpt.add (style);        mnuOpt.addSeparator ();
 mnuOpt.add (theme);              mnuWin.add (close);
 mnuWin.add (closeAll);           mnuHelp.add (content);
 mnuHelp.addSeparator ();         mnuHelp.add (keyHelp);
 mnuHelp.addSeparator ();         mnuHelp.add (about);
 bar.add (mnuFile);               bar.add (mnuEdit);
 bar.add (mnuView);               bar.add (mnuOpt);
 bar.add (mnuWin);                bar.add (mnuHelp);
 open = new JMenuItem ("Open New Account", new ImageIcon ("Images/Open.gif"));
 open.addActionListener (this);
 report = new JMenuItem ("Print BankSystem Report", new ImageIcon
("Images/New.gif"));
```

```
report.addActionListener (this);
dep = new JMenuItem ("Deposit Money");
dep.addActionListener (this);
with = new JMenuItem ("Withdraw Money");
with.addActionListener (this);
del = new JMenuItem ("Delete Customer", new ImageIcon ("Images/Delete.gif"));
del.addActionListener (this);
find = new JMenuItem ("Search Customer", new ImageIcon ("Images/find.gif"));
find.addActionListener (this);
all = new JMenuItem ("View All Customer", new ImageIcon ("Images/refresh.gif"));
all.addActionListener (this);
popMenu.add (open);             popMenu.add (report);
popMenu.add (dep);              popMenu.add (with);
popMenu.add (del);              popMenu.add (find);
popMenu.add (all);
addMouseListener (new MouseAdapter () {
        public void mousePressed (MouseEvent me) { checkMouseTrigger (me); }
        public void mouseReleased (MouseEvent me) { checkMouseTrigger (me); }
        private void checkMouseTrigger (MouseEvent me) {
                if (me.isPopupTrigger ())
                        popMenu.show (me.getComponent (), me.getX (), me.getY ());
        }       }       );

btnNew = new JButton (new ImageIcon ("Images/NotePad.gif"));
btnNew.setToolTipText ("Create New Account");
btnNew.addActionListener (this);
btnDep = new JButton (new ImageIcon ("Images/ImationDisk.gif"));
btnDep.setToolTipText ("Deposit Money");
btnDep.addActionListener (this);
btnWith = new JButton (new ImageIcon ("Images/SuperDisk.gif"));
btnWith.setToolTipText ("Withdraw Money");
btnWith.addActionListener (this);
btnRec = new JButton (new ImageIcon ("Images/Paproll.gif"));
btnRec.setToolTipText ("Print Customer Balance");
btnRec.addActionListener (this);
btnDel = new JButton (new ImageIcon ("Images/Toaster.gif"));
btnDel.setToolTipText ("Delete Customer");
btnDel.addActionListener (this);
btnSrch = new JButton (new ImageIcon ("Images/Search.gif"));
btnSrch.setToolTipText ("Search Customer");
btnSrch.addActionListener (this);
btnHelp = new JButton (new ImageIcon ("Images/Help.gif"));
btnHelp.setToolTipText ("Help on Bank System");
btnHelp.addActionListener (this);
btnKey = new JButton (new ImageIcon ("Images/Keys.gif"));
btnKey.setToolTipText ("Shortcut Keys of BankSystem");
btnKey.addActionListener (this);
```

```
 toolBar = new JToolBar ();       toolBar.add (btnNew);
 toolBar.addSeparator ();         toolBar.add (btnDep);
 toolBar.add (btnWith);           toolBar.addSeparator ();
 toolBar.add (btnRec);            toolBar.addSeparator ();
 toolBar.add (btnDel);            toolBar.addSeparator ();
 toolBar.add (btnSrch);           toolBar.addSeparator ();
 toolBar.add (btnHelp);           toolBar.add (btnKey);
 author = new JLabel (" " + "BankSystem [Pvt] Limited.", Label.LEFT);
 author.setForeground (Color.black);
 author.setToolTipText ("Program's Title");
 welcome = new JLabel ("Welcome Today is " + d + " ", JLabel.RIGHT);
 welcome.setForeground (Color.black);
 welcome.setToolTipText ("Welcoming the User & System Current Date");
 statusBar.setLayout (new BorderLayout());
 statusBar.add (author, BorderLayout.WEST);
 statusBar.add (welcome, BorderLayout.EAST);
 desktop.putClientProperty ("JDesktopPane.dragMode", "outline");
 getContentPane().add (toolBar, BorderLayout.NORTH);
 getContentPane().add (desktop, BorderLayout.CENTER);
 getContentPane().add (statusBar, BorderLayout.SOUTH);
 setVisible (true);    }
public void actionPerformed (ActionEvent ae) {
Object obj = ae.getSource();
if (obj == addNew || obj == open || obj == btnNew) {
 boolean b = openChildWindow ("Create New Account");
        if (b == false) {
               NewAccount newAcc = new NewAccount ();
               desktop.add (newAcc);
               newAcc.show ();
        }              }
 else if (obj == printRec || obj == btnRec || obj == report) {
        getAccountNo ();
 }
 else if (obj == end) {
        quitApp ();
 }
 else if (obj == deposit || obj == dep || obj == btnDep) {

        boolean b = openChildWindow ("Deposit Money");
        if (b == false) {
               DepositMoney depMon = new DepositMoney ();
               desktop.add (depMon);
               depMon.show ();
        }              }
 else if (obj == withdraw || obj == with || obj == btnWith) {
        boolean b = openChildWindow ("Withdraw Money");
        if (b == false) {
```

```
				WithdrawMoney withMon = new WithdrawMoney ();
				desktop.add (withMon);		withMon.show ();
			}
		}
		else if (obj == delRec || obj == del || obj == btnDel) {

			boolean b = openChildWindow ("Delete Account Holder");
			if (b == false) {
				DeleteCustomer delCus = new DeleteCustomer ();
				desktop.add (delCus);	delCus.show ();
			}
		}
		else if (obj == search || obj == find || obj == btnSrch) {

			boolean b = openChildWindow ("Search Customer [By No.]");
			if (b == false) {
				FindAccount fndAcc = new FindAccount ();
				desktop.add (fndAcc);	fndAcc.show ();
			}
		}
		else if (obj == searchName) {

			boolean b = openChildWindow ("Search Customer [By Name]");
			if (b == false) {
				FindName fndNm = new FindName ();
				desktop.add (fndNm);	fndNm.show ();
			}
		}
		else if (obj == oneByOne) {

			boolean b = openChildWindow ("View Account Holders");
			if (b == false) {
				ViewOne vwOne = new ViewOne ();
				desktop.add (vwOne);	vwOne.show ();
			}
		}
		else if (obj == allCustomer || obj == all) {

			boolean b = openChildWindow ("View All Account Holders");
			if (b == false) {
				ViewCustomer viewCus = new ViewCustomer ();
				desktop.add (viewCus);	viewCus.show ();
			}
		}
		else if (obj == change) {

			Color cl = new Color (153, 153, 204);
		cl = JColorChooser.showDialog (this, "Choose Background Color", cl);
		if (cl == null) { }
		else {
			desktop.setBackground (cl);	desktop.repaint ();
		}	}
	else if (obj == close) {

		try {
			desktop.getSelectedFrame().setClosed(true);
		}
		catch (Exception CloseExc) { }
	}
	else if (obj == closeAll) {

		JInternalFrame Frames[] = desktop.getAllFrames (); //Getting all Open Frames.
		for(int getFrameLoop = 0; getFrameLoop < Frames.length; getFrameLoop++) {
			try {
				Frames[getFrameLoop].setClosed (true); //Close the frame.
			}
			catch (Exception CloseExc) { } //if we can't close it then we have a
problem.
		}
	}
	else if (obj == content || obj == btnHelp) {

		boolean b = openChildWindow ("BankSystem Help");
		if (b == false) {
			BankHelp hlpBank = new BankHelp ("BankSystem Help",
"Help/Bank.htm");
			desktop.add (hlpBank);
			hlpBank.show ();
		}
	}
	else if (obj == keyHelp || obj == btnKey) {

		boolean b = openChildWindow ("BankSystem Keys");
		if (b == false) {
			BankHelp hlpKey = new BankHelp ("BankSystem Keys",
"Help/Keys.htm");
			desktop.add (hlpKey);
			hlpKey.show ();
		}
	}
	else if (obj == about) {

		String msg = "BankSystem [Pvt] Limited.\n\n" + "Created & Designed By:\n" +
			"Muhammad Wasif Javed\n\n" + "E-mail me:\n
```

```
wasi_javed@hotmail.com";
		JOptionPane.showMessageDialog (this, msg, "About BankSystem",
JOptionPane.PLAIN_MESSAGE);

  }	}
public void itemStateChanged (ItemEvent e) {

  for( int i = 0; i < radio.length; i++ )
		if(radio[i].isSelected()) {
			changeLookAndFeel (i);
		}
}
private void quitApp () {
  try {
		//Show a Confirmation Dialog.
		int reply = JOptionPane.showConfirmDialog (this,
					"Are you really want to exit\nFrom BankSystem?",
					"BankSystem - Exit", JOptionPane.YES_NO_OPTION,
JOptionPane.PLAIN_MESSAGE);
		//Check the User Selection.
		if (reply == JOptionPane.YES_OPTION) {
			setVisible (false);	//Hide the Frame.
			dispose();		//Free the System Resources.
			System.out.println ("Thanks for Using BankSystem\nAuthor - Muhammad
Wasif Javed");
			System.exit (0);	//Close the Application.
		}
		else if (reply == JOptionPane.NO_OPTION) {
			setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
		}	}

  catch (Exception e) {}

}

  public void changeLookAndFeel (int val) {

  try {
		UIManager.setLookAndFeel (looks[val].getClassName());
		SwingUtilities.updateComponentTreeUI (this);
  }
  catch (Exception e) { }

}
private boolean openChildWindow (String title) {
  JInternalFrame[] childs = desktop.getAllFrames ();
  for (int i = 0; i < childs.length; i++) {
```

```
		if (childs[i].getTitle().equalsIgnoreCase (title)) {
			childs[i].show ();
			return true;
		}	}
 return false;		}
void getAccountNo () {
 String printing;
 rows = 0;
 boolean b = populateArray ();
 if (b == false) { }
 else {
	try {
		printing = JOptionPane.showInputDialog (this, "Enter Account No. to
Print Customer Balance.\n" +
		"(Tip: Account No. Contains only Digits)", "BankSystem - PrintRecord",
JOptionPane.PLAIN_MESSAGE);
		if (printing == null) { }
		if (printing.equals ("")) {
			JOptionPane.showMessageDialog (this, "Provide Account No. to
Print.",
					"BankSystem - EmptyField",
JOptionPane.PLAIN_MESSAGE);
			getAccountNo ();
		}
		else {
			findRec (printing);
		}
	}
	catch (Exception e) { }
 }
}
boolean populateArray () {
 boolean b = false;
 try {
	fis = new FileInputStream ("Bank.dat");
	dis = new DataInputStream (fis);
	//Loop to Populate the Array.
	while (true) {
		for (int i = 0; i < 6; i++) {
			records[rows][i] = dis.readUTF ();
		}
		rows++;
	}
 }
 catch (Exception ex) {
	total = rows;
	if (total == 0) {
```

```
                JOptionPane.showMessageDialog (null, "Records File is Empty.\nEnter
Records First to Display.",
                        "BankSystem - EmptyFile", JOptionPane.PLAIN_MESSAGE);
                b = false;
        }
        else {
                b = true;
                try {
                        dis.close();
                        fis.close();
                }
                catch (Exception exp) { }
        }
 }
 return b;

}

//Function use to Find Record by Matching the Contents of Records Array with
InputBox.

void findRec (String rec) {

 boolean found = false;
 for (int x = 0; x < total; x++) {
        if (records[x][0].equals (rec)) {
                found = true;
                printRecord (makeRecordPrint (x));
                break;
        }
 }
 if (found == false) {
        JOptionPane.showMessageDialog (this, "Account No. " + rec + " doesn't Exist.",
                        "BankSystem - WrongNo", JOptionPane.PLAIN_MESSAGE);
        getAccountNo ();
 }

}

//Function use to make Current Record ready for Print.

String makeRecordPrint (int rec) {

 String data;
 String data0 = "          BankSystem [Pvt] Limited.          \n";       //Page Title.
 String data1 = "          Customer Balance Report.          \n\n";      //Page Header.
 String data2 = "  Account No.:      " + records[rec][0] + "\n";
```

```
  String data3 = "  Customer Name:     " + records[rec][1] + "\n";
  String data4 = "  Last Transaction:  " + records[rec][2] + ", " + records[rec][3] + ", " +
records[rec][4] + "\n";
  String data5 = "  Current Balance:   " + records[rec][5] + "\n\n";
  String data6 = "          Copyright   2003 Muhammad Wasif Javed.\n";   //Page Footer.
  String sep0 = " ----------------------------------------------------------\n";
  String sep1 = " ----------------------------------------------------------\n";
  String sep2 = " ----------------------------------------------------------\n";
  String sep3 = " ----------------------------------------------------------\n";
  String sep4 = " ----------------------------------------------------------\n\n";

  data = data0 + sep0 + data1 + data2 + sep1 + data3 + sep2 + data4 + sep3 + data5 +
sep4 + data6;
  return data;

}

//Function use to Print the Current Record.

void printRecord (String rec) {

  StringReader sr = new StringReader (rec);
  LineNumberReader lnr = new LineNumberReader (sr);
  Font typeface = new Font ("Times New Roman", Font.PLAIN, 12);
  Properties p = new Properties ();
  PrintJob pJob = getToolkit().getPrintJob (this, "Print Customer Balance Report", p);

  if (pJob != null) {
        Graphics gr = pJob.getGraphics ();
        if (gr != null) {
              FontMetrics fm = gr.getFontMetrics (typeface);
              int margin = 20;
              int pageHeight = pJob.getPageDimension().height - margin;
              int fontHeight = fm.getHeight();
                    int fontDescent = fm.getDescent();
              int curHeight = margin;
              String nextLine;
              gr.setFont (typeface);

              try {
                    do {
                          nextLine = lnr.readLine ();
                          if (nextLine != null) {
                                if ((curHeight + fontHeight) > pageHeight) {//New
Page.
                                      gr.dispose();
                                      gr = pJob.getGraphics ();
```

```
                                                        curHeight = margin;
                                                }
                                                curHeight += fontHeight;
                                                if (gr != null) {
                                                        gr.setFont (typeface);
                                                        gr.drawString (nextLine, margin, curHeight -
        fontDescent);
                                                }
                                        }
                                }
                                while (nextLine != null);
                        }
                        catch (EOFException eof) { }
                        catch (Throwable t) { }
                }
                gr.dispose();
         }
         if (pJob != null)
                pJob.end ();
        }       }
```

3.1 CREATE NEW ACCOUNT

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class NewAccount extends JInternalFrame implements ActionListener {
        private JPanel jpInfo = new JPanel();
        private JLabel lbNo, lbName, lbDate, lbDeposit;
        private JTextField txtNo, txtName, txtDeposit;
        private JComboBox cboMonth, cboDay, cboYear;
        private JButton btnSave, btnCancel;
        private int count = 0;        private int rows = 0;
        private         int total = 0;
        private String records[][] = new String [500][6];
        private String saves[][] = new String [500][6];
        private FileInputStream fis;
        private DataInputStream dis;
        NewAccount () {
                super ("Create New Account", false, true, false, true);
                setSize (335, 235);
                jpInfo.setBounds (0, 0, 500, 115);
                jpInfo.setLayout (null);
                lbNo = new JLabel ("Account No:");
                lbNo.setForeground (Color.black);
                lbNo.setBounds (15, 20, 80, 25);
```

```
		lbName = new JLabel ("Person Name:");
		  lbName.setForeground (Color.black);
		lbName.setBounds (15, 55, 80, 25);
		  lbDate = new JLabel ("Deposit Date:");
		  lbDate.setForeground (Color.black);
		  lbDate.setBounds (15, 90, 80, 25);
		  lbDeposit = new JLabel ("Dep. Amount:");
		  lbDeposit.setForeground (Color.black);
		  lbDeposit.setBounds (15, 125, 80, 25);
		  txtNo = new JTextField ();
		  txtNo.setHorizontalAlignment (JTextField.RIGHT);
		  txtNo.setBounds (105, 20, 205, 25);
		  txtName = new JTextField ();
		  txtName.setBounds (105, 55, 205, 25);
		  txtDeposit = new JTextField ();
		  txtDeposit.setHorizontalAlignment (JTextField.RIGHT);
		  txtDeposit.setBounds (105, 125, 205, 25);
		  txtNo.addKeyListener (new KeyAdapter() {
			public void keyTyped (KeyEvent ke) {
				char c = ke.getKeyChar ();
				if (!((Character.isDigit (c) || (c ==
KeyEvent.VK_BACK_SPACE)))) {
					getToolkit().beep ();
					ke.consume ();
				}
			}	}
		  );
		  txtDeposit.addKeyListener (new KeyAdapter() {
			public void keyTyped (KeyEvent ke) {
				char c = ke.getKeyChar ();
				if (!((Character.isDigit (c) || (c ==
KeyEvent.VK_BACK_SPACE)))) {
					getToolkit().beep ();
					ke.consume ();
				}
			}	}
		  );
		  String Months[] = {"January", "February", "March", "April", "May", "June",
			"July", "August", "September", "October", "November", "December"};
		  cboMonth = new JComboBox (Months);
		  cboDay = new JComboBox ();
		  cboYear = new JComboBox ();
		  for (int i = 1; i <= 31; i++) {
			String days = "" + i;
			cboDay.addItem (days);
		  }
		  for (int i = 2000; i <= 2024; i++) {
```

```
                        String years = "" + i;
                        cboYear.addItem (years);
                }
                cboMonth.setBounds (105, 90, 92, 25);
                cboDay.setBounds (202, 90, 43, 25);
                cboYear.setBounds (250, 90, 60, 25);
                btnSave = new JButton ("Save");
                btnSave.setBounds (20, 165, 120, 25);
                btnSave.addActionListener (this);
                btnCancel = new JButton ("Cancel");
                btnCancel.setBounds (185, 165, 120, 25);
                btnCancel.addActionListener (this);
                jpInfo.add (lbNo);
                jpInfo.add (txtNo);
                jpInfo.add (lbName);
                jpInfo.add (txtName);
                jpInfo.add (lbDate);
                jpInfo.add (cboMonth);
                jpInfo.add (cboDay);
                jpInfo.add (cboYear);
                jpInfo.add (lbDeposit);
                jpInfo.add (txtDeposit);
                jpInfo.add (btnSave);
                jpInfo.add (btnCancel);
                getContentPane().add (jpInfo);
                setVisible (true);
        }
        public void actionPerformed (ActionEvent ae) {
                Object obj = ae.getSource();
                if (obj == btnSave) {
                        if (txtNo.getText().equals("")) {
                                JOptionPane.showMessageDialog (this, "Please! Provide Id of
Customer.",
                                                "BankSystem - EmptyField",
JOptionPane.PLAIN_MESSAGE);
                                txtNo.requestFocus();
                        }
                        else if (txtName.getText().equals("")) {
                                JOptionPane.showMessageDialog (this, "Please! Provide Name of
Customer.",
                                                "BankSystem - EmptyField",
JOptionPane.PLAIN_MESSAGE);
                                txtName.requestFocus ();
                        }
                        else if (txtDeposit.getText().equals("")) {
                                JOptionPane.showMessageDialog (this, "Please! Provide Deposit
Amount.",
                                                "BankSystem - EmptyField",
JOptionPane.PLAIN_MESSAGE);
                                txtDeposit.requestFocus ();
                        }
                        else {
                                populateArray ();   //Load All Existing Records in Memory.
                                findRec ();         //Finding if Account No. Already Exist or Not.
                        }
                }
                if (obj == btnCancel) {
                        txtClear ();
                        setVisible (false);
                        dispose();
                }       }
        void populateArray () {
                try {
                        fis = new FileInputStream ("Bank.dat");
                        dis = new DataInputStream (fis);
                        //Loop to Populate the Array.
                        while (true) {
                                for (int i = 0; i < 6; i++) {
                                        records[rows][i] = dis.readUTF ();
                                }
                                rows++;
                        }
                }
                catch (Exception ex) {
                        total = rows;
                        if (total == 0) { }
                        else {
                                try {
                                        dis.close();
                                        fis.close();
                                }
                                catch (Exception exp) { }
                        }
                }

        }
        void findRec () {

                boolean found = false;
                for (int x = 0; x < total; x++) {
                        if (records[x][0].equals (txtNo.getText())) {
                                found = true;
                                JOptionPane.showMessageDialog (this, "Account No. " +
txtNo.getText () + " is Already Exist.",
                                                "BankSystem - WrongNo",
JOptionPane.PLAIN_MESSAGE);
                        txtClear ();
                        break;
                    }
                }
                if (found == false) {
                    saveArray ();
                }

        }

        //Function use to add new Element to Array.
        void saveArray () {

                saves[count][0] = txtNo.getText ();
                saves[count][1] = txtName.getText ();
                saves[count][2] = "" + cboMonth.getSelectedItem ();
                saves[count][3] = "" + cboDay.getSelectedItem ();
                saves[count][4] = "" + cboYear.getSelectedItem ();
                saves[count][5] = txtDeposit.getText ();
                saveFile ();  //Save This Array to File.
                count++;

        }
        void saveFile () {
                try {
                        FileOutputStream fos = new FileOutputStream ("Bank.dat", true);
                        DataOutputStream dos = new DataOutputStream (fos);
                        dos.writeUTF (saves[count][0]);
                        dos.writeUTF (saves[count][1]);
                        dos.writeUTF (saves[count][2]);
                        dos.writeUTF (saves[count][3]);
                        dos.writeUTF (saves[count][4]);
                        dos.writeUTF (saves[count][5]);
                        JOptionPane.showMessageDialog (this, "The Record has been Saved
Successfully",
                                                "BankSystem - Record Saved",
JOptionPane.PLAIN_MESSAGE);
                        txtClear ();
                        dos.close();
                        fos.close();
                }
                catch (IOException ioe) {
                        JOptionPane.showMessageDialog (this, "There are Some Problem with
File",
                                                "BankSystem - Problem",
JOptionPane.PLAIN_MESSAGE);
                }

        }

        //Function use to Clear all TextFields of Window.
        void txtClear () {

                txtNo.setText ("");
                txtName.setText ("");
                txtDeposit.setText ("");
                txtNo.requestFocus ();

        }       }
```

3.2 DELETE CUSTOMERS

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class DeleteCustomer extends JInternalFrame implements ActionListener {

        private JPanel jpDel = new JPanel();
        private JLabel lbNo, lbName, lbDate, lbBal;
        private JTextField txtNo, txtName, txtDate, txtBal;
        private JButton btnDel, btnCancel;
        private int recCount = 0;
        private int rows = 0;
        private         int total = 0;
        private String records[][] = new String [500][6];
        private FileInputStream fis;
        private DataInputStream dis;
        DeleteCustomer () {
                super ("Delete Account Holder", false, true, false, true);
                setSize (350, 235);
                jpDel.setLayout (null);
                lbNo = new JLabel ("Account No:");
                lbNo.setForeground (Color.black);
                lbNo.setBounds (15, 20, 80, 25);
               lbName = new JLabel ("Person Name:");
                lbName.setForeground (Color.black);
               lbName.setBounds (15, 55, 90, 25);
                lbDate = new JLabel ("Last Transaction:");
                lbDate.setForeground (Color.black);
                lbDate.setBounds (15, 90, 100, 25);
                lbBal = new JLabel ("Balance:");
                lbBal.setForeground (Color.black);
                lbBal.setBounds (15, 125, 80, 25);
                txtNo = new JTextField ();
                txtNo.setHorizontalAlignment (JTextField.RIGHT);
                txtNo.setBounds (125, 20, 200, 25);
                txtName = new JTextField ();
                txtName.setEnabled (false);
                txtName.setBounds (125, 55, 200, 25);
                txtDate = new JTextField ();
                txtDate.setEnabled (false);
                txtDate.setBounds (125, 90, 200, 25);
                txtBal = new JTextField ();
                txtBal.setEnabled (false);
                txtBal.setHorizontalAlignment (JTextField.RIGHT);
                txtBal.setBounds (125, 125, 200, 25);
                btnDel = new JButton ("Delete");
                btnDel.setBounds (20, 165, 120, 25);
                btnDel.addActionListener (this);
                btnCancel = new JButton ("Cancel");
                btnCancel.setBounds (200, 165, 120, 25);
                btnCancel.addActionListener (this);
                jpDel.add (lbNo);         jpDel.add (txtNo);
                jpDel.add (lbName);       jpDel.add (txtName);
                jpDel.add (lbDate);       jpDel.add (txtDate);
                jpDel.add (lbBal);        jpDel.add (txtBal);
                jpDel.add (btnDel);       jpDel.add (btnCancel);
                txtNo.addKeyListener (new KeyAdapter() {
                        public void keyTyped (KeyEvent ke) {
                                char c = ke.getKeyChar ();
                        if (!((Character.isDigit (c) || (c == KeyEvent.VK_BACK_SPACE)))) {
                                        getToolkit().beep ();
                                        ke.consume ();
                                }
                        }       }       );
                        txtNo.addFocusListener (new FocusListener () {
                        public void focusGained (FocusEvent e) { }
                        public void focusLost (FocusEvent fe) {
                                if (txtNo.getText().equals ("")) { }
                                else {
                                        rows = 0;
                                        populateArray ();   findRec ();
                                }
                        }       }       );
                getContentPane().add (jpDel);
                populateArray ();   //Load All Existing Records in Memory.
                setVisible (true); }
```

```
                public void actionPerformed (ActionEvent ae) {
                Object obj = ae.getSource();
                if (obj == btnDel) {
                if (txtNo.getText().equals("")) {
                JOptionPane.showMessageDialog (this, "Please! Provide Id of Customer.",
                                        "BankSystem - EmptyField",
JOptionPane.PLAIN_MESSAGE);
                                txtNo.requestFocus();
                        }
                        else {
                                deletion ();   //Confirm Deletion of Current Record.
                        }
                }
                if (obj == btnCancel) {
                        txtClear ();
                        setVisible (false);
                        dispose();
                }       }
        void populateArray () {
                try {
                        fis = new FileInputStream ("Bank.dat");
                        dis = new DataInputStream (fis);
                        //Loop to Populate the Array.
                        while (true) {
                                for (int i = 0; i < 6; i++) {
                                        records[rows][i] = dis.readUTF ();
                                }
                                rows++;
                        }
                }
                catch (Exception ex) {
                        total = rows;
                        if (total == 0) {
                                JOptionPane.showMessageDialog (null, "Records File is
Empty.\nEnter Records First to Display.",
                                        "BankSystem - EmptyFile",
JOptionPane.PLAIN_MESSAGE);
                                btnEnable ();
                        }
                        else {
                                try {
                                        dis.close();
                                        fis.close();
                                }
                                catch (Exception exp) { }
                        }
                }       }
```

```
	void findRec () {
		boolean found = false;
		for (int x = 0; x < total; x++) {
			if (records[x][0].equals (txtNo.getText())) {
				found = true;
				showRec (x);
				break;
			}
		}
		if (found == false) {
			String str = txtNo.getText ();
			txtClear ();
			JOptionPane.showMessageDialog (this, "Account No. " + str + " doesn't
Exist.",
					"BankSystem - WrongNo",
JOptionPane.PLAIN_MESSAGE);
		}

	}

	//Function which display Record from Array onto the Form.
	void showRec (int intRec) {

		txtNo.setText (records[intRec][0]);
		txtName.setText (records[intRec][1]);
		txtDate.setText (records[intRec][2] + ", " + records[intRec][3] + ", " +
records[intRec][4]);
		txtBal.setText (records[intRec][5]);
		recCount = intRec;

	}

	//Confirming the Deletion Decision made By User of Program.
	void deletion () {

		try {
			//Show a Confirmation Dialog.
			int reply = JOptionPane.showConfirmDialog (this,
					"Are you Sure you want to Delete\n" + txtName.getText () + "
Record From BankSystem?",
					"Bank System - Delete", JOptionPane.YES_NO_OPTION,
JOptionPane.PLAIN_MESSAGE);
			//Check the User Selection.
			if (reply == JOptionPane.YES_OPTION) {
				delRec ();	//Delete the Selected Contents of Array.
			}
			else if (reply == JOptionPane.NO_OPTION) { }
		}

		catch (Exception e) {}

	}

	//Function use to Delete an Element from the Array.
	void delRec () {

		try {
			if (records != null) {
				for(int i = recCount; i < total; i++) {
					for (int r = 0; r < 6; r++) {
						records[i][r] = records[i+1][r];
						if (records[i][r] == null) break;
					}
				}
				total = total - 1;
				deleteFile ();
			}
		}
		catch (ArrayIndexOutOfBoundsException ex) { }

	}

	//Function use to Save Records to File After Deleting the Record of User Choice.
	void deleteFile () {

		try {
			FileOutputStream fos = new FileOutputStream ("Bank.dat");
			DataOutputStream dos = new DataOutputStream (fos);
			if (records != null) {
				for (int i = 0; i < total; i++) {
					for (int r = 0; r < 6; r++) {
						dos.writeUTF (records[i][r]);
						if (records[i][r] == null) break;
					}
				}
				JOptionPane.showMessageDialog (this, "Record has been Deleted
Successfuly.",
					"BankSystem - Record Deleted",
JOptionPane.PLAIN_MESSAGE);
				txtClear ();
			}
			else { }
			dos.close();
			fos.close();
```

```
                }
                catch (IOException ioe) {
                        JOptionPane.showMessageDialog (this, "There are Some Problem with
File",
                                                "BankSystem - Problem",
JOptionPane.PLAIN_MESSAGE);
                }

        }

        //Function use to Clear all TextFields of Window.
        void txtClear () {

                txtNo.setText ("");
                txtName.setText ("");
                txtDate.setText ("");
                txtBal.setText ("");
                txtNo.requestFocus ();

        }

        //Function use to Lock Controls of Window.
        void btnEnable () {

                txtNo.setEnabled (false);
                btnDel.setEnabled (false);

        }

}
```

3.3 DEPOSIT MONEY

```
  import java.awt.*;
  import java.awt.event.*;
  import javax.swing.*;
  import java.io.*;
  public class DepositMoney extends JInternalFrame implements ActionListener {
       private JPanel jpDep = new JPanel();
       private JLabel lbNo, lbName, lbDate, lbDeposit;
       private JTextField txtNo, txtName, txtDeposit;
       private JComboBox cboMonth, cboDay, cboYear;
       private JButton btnSave, btnCancel;
       private int recCount = 0;
       private int rows = 0;
       private int total = 0;
       private int curr;
       private int deposit;
```

```
private String records[][] = new String [500][6];
private FileInputStream fis;
private DataInputStream dis;
DepositMoney () {
  super ("Deposit Money", false, true, false, true);
  setSize (335, 235);
  jpDep.setLayout (null);
  lbNo = new JLabel ("Account No:");
  lbNo.setForeground (Color.black);
  lbNo.setBounds (15, 20, 80, 25);
  lbName = new JLabel ("Person Name:");
  lbName.setForeground (Color.black);
  lbName.setBounds (15, 55, 80, 25);
  lbDate = new JLabel ("Deposit Date:");
  lbDate.setForeground (Color.black);
  lbDate.setBounds (15, 90, 80, 25);
  lbDeposit = new JLabel ("Dep. Amount:");
  lbDeposit.setForeground (Color.black);
  lbDeposit.setBounds (15, 125, 80, 25);
  txtNo = new JTextField ();
  txtNo.setHorizontalAlignment (JTextField.RIGHT);
  txtNo.setBounds (105, 20, 205, 25);
  txtName = new JTextField ();
  txtName.setEnabled (false);
  txtName.setBounds (105, 55, 205, 25);
  txtDeposit = new JTextField ();
  txtDeposit.setHorizontalAlignment (JTextField.RIGHT);
  txtDeposit.setBounds (105, 125, 205, 25);
  String Months[] = {"January", "February", "March", "April", "May", "June",
          "July", "August", "September", "October", "November", "December"};
  cboMonth = new JComboBox (Months);
  cboDay = new JComboBox ();
  cboYear = new JComboBox ();
  for (int i = 1; i <= 31; i++) {
          String days = "" + i;
          cboDay.addItem (days);
  }
  for (int i = 2000; i <= 2024; i++) {
          String years = "" + i;
          cboYear.addItem (years);
  }
  cboMonth.setBounds (105, 90, 92, 25);
  cboDay.setBounds (202, 90, 43, 25);
  cboYear.setBounds (250, 90, 60, 25);
  btnSave = new JButton ("Save");
  btnSave.setBounds (20, 165, 120, 25);
  btnSave.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (185, 165, 120, 25);
btnCancel.addActionListener (this);
txtNo.addKeyListener (new KeyAdapter() {
        public void keyTyped (KeyEvent ke) {
                char c = ke.getKeyChar ();
                if (!((Character.isDigit (c) || (c == KeyEvent.VK_BACK_SPACE)))) {
                        getToolkit().beep ();
                        ke.consume ();
                }
        }
}
);
txtDeposit.addKeyListener (new KeyAdapter() {
        public void keyTyped (KeyEvent ke) {
                char c = ke.getKeyChar ();
                if (!((Character.isDigit (c) || (c == KeyEvent.VK_BACK_SPACE)))) {
                        getToolkit().beep ();
                        ke.consume ();
                }
        }
}
);
//Checking the Accunt No. Provided By User on Lost Focus of the TextBox.
txtNo.addFocusListener (new FocusListener () {
        public void focusGained (FocusEvent e) { }
        public void focusLost (FocusEvent fe) {
                if (txtNo.getText().equals ("")) { }
                else {
                        rows = 0;
                        populateArray ();   //Load All Existing Records in Memory.
                        findRec ();         //Finding if Account No. Already Exist or Not.
                }
        }
}
);

//Adding the All the Controls to Panel.
jpDep.add (lbNo);          jpDep.add (txtNo);
jpDep.add (lbName);        jpDep.add (txtName);
jpDep.add (lbDate);        jpDep.add (cboMonth);
jpDep.add (cboDay);        jpDep.add (cboYear);
jpDep.add (lbDeposit);     jpDep.add (txtDeposit);
jpDep.add (btnSave);       jpDep.add (btnCancel);
getContentPane().add (jpDep);
populateArray ();   setVisible (true);          }
```

```
public void actionPerformed (ActionEvent ae) {
 Object obj = ae.getSource();
 if (obj == btnSave) {
         if (txtNo.getText().equals("")) {
                 JOptionPane.showMessageDialog (this, "Please! Provide Id of Customer.",
                                 "BankSystem - EmptyField",
JOptionPane.PLAIN_MESSAGE);
                 txtNo.requestFocus();
         }
         else if (txtDeposit.getText().equals("")) {
         JOptionPane.showMessageDialog (this, "Please! Provide Deposit Amount.",
                                 "BankSystem - EmptyField",
JOptionPane.PLAIN_MESSAGE);
                 txtDeposit.requestFocus ();
         }
         else {
                 editRec ();   //Update the Contents of Array.
         }
 }
 if (obj == btnCancel) {
         txtClear ();   setVisible (false);   dispose();
 }
}
void populateArray () {
 try {
         fis = new FileInputStream ("Bank.dat");
         dis = new DataInputStream (fis);
         //Loop to Populate the Array.
         while (true) {
                 for (int i = 0; i < 6; i++) {
                         records[rows][i] = dis.readUTF ();
                 }
                 rows++;
         }
 }
 catch (Exception ex) {
         total = rows;
         if (total == 0) {
                 JOptionPane.showMessageDialog (null, "Records File is Empty.\nEnter
Records First to Display.",
                                         "BankSystem - EmptyFile",
JOptionPane.PLAIN_MESSAGE);
                 btnEnable ();
         }
         else {
                 try {
                         dis.close();
```

```
                        fis.close();
                }
                catch (Exception exp) { }
        }
 }      }

void findRec () {
 boolean found = false;
 for (int x = 0; x < total; x++) {
        if (records[x][0].equals (txtNo.getText())) {
                found = true;
                showRec (x);
                break;
        }
 }
 if (found == false) {
        String str = txtNo.getText ();
        txtClear ();
        JOptionPane.showMessageDialog (this, "Account No. " + str + " doesn't Exist.",
                                "BankSystem - WrongNo",
JOptionPane.PLAIN_MESSAGE);
 }      }
public void showRec (int intRec) {
 txtNo.setText (records[intRec][0]);
 txtName.setText (records[intRec][1]);
 curr = Integer.parseInt (records[intRec][5]);
 recCount = intRec;
}
void txtClear () {
 txtNo.setText ("");
 txtName.setText ("");
 txtDeposit.setText ("");
 txtNo.requestFocus ();
}
public void editRec () {
 deposit = Integer.parseInt (txtDeposit.getText ());
 records[recCount][0] = txtNo.getText ();
 records[recCount][1] = txtName.getText ();
 records[recCount][2] = "" + cboMonth.getSelectedItem ();
 records[recCount][3] = "" + cboDay.getSelectedItem ();
 records[recCount][4] = "" + cboYear.getSelectedItem ();
 records[recCount][5] = "" + (curr + deposit);
 editFile ();   //Save This Array to File.

}
```

```
	public void editFile () {
	  try {
		FileOutputStream fos = new FileOutputStream ("Bank.dat");
		DataOutputStream dos = new DataOutputStream (fos);
		if (records != null) {
			for (int i = 0; i < total; i++) {
				for (int c = 0; c < 6; c++) {
					dos.writeUTF (records[i][c]);
					if (records[i][c] == null) break;
				}
			}
			JOptionPane.showMessageDialog (this, "The File is Updated 
	Successfully",
					"BankSystem - Record Saved", 
	JOptionPane.PLAIN_MESSAGE);
			txtClear ();
			dos.close();
			fos.close();
		}
	  }
	  catch (IOException ioe) {
		JOptionPane.showMessageDialog (this, "There are Some Problem with File",
					"BankSystem - Problem", JOptionPane.PLAIN_MESSAGE);
	  }	}

	void btnEnable () {
	  txtNo.setEnabled (false);
	  cboMonth.setEnabled (false);
	  cboDay.setEnabled (false);
	  cboYear.setEnabled (false);
	  txtDeposit.setEnabled (false);
	  btnSave.setEnabled (false);

	}}
```

3.4 FINDING AN ACCOUNT

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
public class FindAccount extends JInternalFrame implements ActionListener {
	  private JPanel jpFind = new JPanel();
	  private JLabel lbNo, lbName, lbDate, lbBal;
	  private JTextField txtNo, txtName, txtDate, txtBal;
	  private JButton btnFind, btnCancel;
	  private int count = 0;
	  private int rows = 0;
private         int total = 0;
private String records[][] = new String [500][6];
private FileInputStream fis;
private DataInputStream dis;
FindAccount () {
        super ("Search Customer [By No.]", false, true, false, true);
        setSize (350, 235);
        jpFind.setLayout (null);
        lbNo = new JLabel ("Account No:");
        lbNo.setForeground (Color.black);
        lbNo.setBounds (15, 20, 80, 25);
      lbName = new JLabel ("Person Name:");
        lbName.setForeground (Color.black);
      lbName.setBounds (15, 55, 80, 25);
        lbDate = new JLabel ("Last Transaction:");
        lbDate.setForeground (Color.black);
        lbDate.setBounds (15, 90, 100, 25);
        lbBal = new JLabel ("Balance:");
        lbBal.setForeground (Color.black);
        lbBal.setBounds (15, 125, 80, 25);
        txtNo = new JTextField ();
        txtNo.setHorizontalAlignment (JTextField.RIGHT);
        txtNo.setBounds (125, 20, 200, 25);
        txtName = new JTextField ();
        txtName.setEnabled (false);
        txtName.setBounds (125, 55, 200, 25);
        txtDate = new JTextField ();
        txtDate.setEnabled (false);
        txtDate.setBounds (125, 90, 200, 25);
        txtBal = new JTextField ();
        txtBal.setHorizontalAlignment (JTextField.RIGHT);
        txtBal.setEnabled (false);
        txtBal.setBounds (125, 125, 200, 25);
        txtNo.addKeyListener (new KeyAdapter() {
                public void keyTyped (KeyEvent ke) {
                        char c = ke.getKeyChar ();
        if (!((Character.isDigit (c) || (c == KeyEvent.VK_BACK_SPACE)))) {
                                getToolkit().beep ();
                                ke.consume ();
                        }
                }
        }
        );
        btnFind = new JButton ("Search");
        btnFind.setBounds (20, 165, 120, 25);
        btnFind.addActionListener (this);
        btnCancel = new JButton ("Cancel");
```

```
                btnCancel.setBounds (200, 165, 120, 25);
                btnCancel.addActionListener (this);
                jpFind.add (lbNo);
                jpFind.add (txtNo);
                jpFind.add (lbName);
                jpFind.add (txtName);
                jpFind.add (lbDate);
                jpFind.add (txtDate);
                jpFind.add (lbBal);
                jpFind.add (txtBal);
                jpFind.add (btnFind);
                jpFind.add (btnCancel);
                getContentPane().add (jpFind);
                populateArray ();
                setVisible (true);
        }
        public void actionPerformed (ActionEvent ae) {
                Object obj = ae.getSource();
                if (obj == btnFind) {
                        if (txtNo.getText().equals("")) {
                                JOptionPane.showMessageDialog (this, "Please! Provide Id of
Customer to Search.",
                "BankSystem - EmptyField", JOptionPane.PLAIN_MESSAGE);
                                txtNo.requestFocus();
                        }
                        else {
                                rows = 0;
                                populateArray ();   //Load All Existing Records in Memory.
                                findRec ();         //Finding if Account No. Exist or Not.
                        }
                }
                if (obj == btnCancel) {
                        txtClear ();
                        setVisible (false);
                        dispose();
                }       }
        void populateArray () {
                try {
                        fis = new FileInputStream ("Bank.dat");
                        dis = new DataInputStream (fis);
                        //Loop to Populate the Array.
                        while (true) {
                                for (int i = 0; i < 6; i++) {
                                        records[rows][i] = dis.readUTF ();
                                }
                                rows++;
                        }
                }
                catch (Exception ex) {
                        total = rows;
                        if (total == 0) {
                                JOptionPane.showMessageDialog (null, "Records File is
Empty.\nEnter Records First to Display.",
                "BankSystem - EmptyFile", JOptionPane.PLAIN_MESSAGE);
                                btnEnable ();
                        }
                        else {
                                try {
                                        dis.close();
                                        fis.close();
                                }
                                catch (Exception exp) { }
                        }
                }

        }
        void findRec () {
                boolean found = false;
                for (int x = 0; x < total; x++) {
                        if (records[x][0].equals (txtNo.getText())) {
                                found = true;
                                showRec (x);
                                break;
                        }
                }
                if (found == false) {
                        JOptionPane.showMessageDialog (this, "Account No. " + txtNo.getText ()
+ " doesn't Exist.",
                                                        "BankSystem - WrongNo",
JOptionPane.PLAIN_MESSAGE);
                        txtClear ();
                }

        }

        //Function which display Record from Array onto the Form.
        public void showRec (int intRec) {

                txtNo.setText (records[intRec][0]);
                txtName.setText (records[intRec][1]);
                txtDate.setText (records[intRec][2] + ", " + records[intRec][3] + ", " +
records[intRec][4]);
                txtBal.setText (records[intRec][5]);
```

```
	}

	//Function use to Clear all TextFields of Window.
	void txtClear () {

		txtNo.setText ("");		txtName.setText ("");
		txtDate.setText ("");		txtBal.setText ("");
		txtNo.requestFocus ();
	}
	void btnEnable () {
		txtNo.setEnabled (false);
		btnFind.setEnabled (false);
	}
}
```

3.5 WITHDRAW MONEY

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
public class WithdrawMoney extends JInternalFrame implements ActionListener {
	private JPanel jpWith = new JPanel();
	private JLabel lbNo, lbName, lbDate, lbWithdraw;
	private JTextField txtNo, txtName, txtWithdraw;
	private JComboBox cboMonth, cboDay, cboYear;
	private JButton btnSave, btnCancel;
	private int recCount = 0;
	private int rows = 0;
	private	int total = 0;
	private	int curr;
	private	int withdraw;
	private String records[][] = new String [500][6];
	private FileInputStream fis;
	private DataInputStream dis;
	WithdrawMoney () {
		super ("Withdraw Money", false, true, false, true);
		setSize (335, 235);
		jpWith.setLayout (null);
		lbNo = new JLabel ("Account No:");
		lbNo.setForeground (Color.black);
		lbNo.setBounds (15, 20, 80, 25);
	       lbName = new JLabel ("Person Name:");
		lbName.setForeground (Color.black);
	       lbName.setBounds (15, 55, 80, 25);
		lbDate = new JLabel ("With. Date:");
		lbDate.setForeground (Color.black);
		lbDate.setBounds (15, 90, 80, 25);
		lbWithdraw = new JLabel ("With. Amount:");
		lbWithdraw.setForeground (Color.black);
		lbWithdraw.setBounds (15, 125, 80, 25);
		txtNo = new JTextField ();
		txtNo.setHorizontalAlignment (JTextField.RIGHT);
		//Checking the Accunt No. Provided By User on Lost Focus of the TextBox.
		txtNo.addFocusListener (new FocusListener () {
			public void focusGained (FocusEvent e) { }
			public void focusLost (FocusEvent fe) {
				if (txtNo.getText().equals ("")) { }
				else {
					rows = 0;
					populateArray ();	//Load All Existing Records in Memory.
					findRec ();		//Finding if Account No. Already Exist or
Not.
				}
			}
		}
		);
		txtNo.setBounds (105, 20, 205, 25);

		txtName = new JTextField ();
		txtName.setEnabled (false);
		txtName.setBounds (105, 55, 205, 25);
		txtWithdraw = new JTextField ();
		txtWithdraw.setHorizontalAlignment (JTextField.RIGHT);
		txtWithdraw.setBounds (105, 125, 205, 25);
		txtNo.addKeyListener (new KeyAdapter() {
			public void keyTyped (KeyEvent ke) {
				char c = ke.getKeyChar ();
		if (!((Character.isDigit (c) || (c == KeyEvent.VK_BACK_SPACE)))) {
					getToolkit().beep ();
					ke.consume ();
				}
			}
		}
		);
		txtWithdraw.addKeyListener (new KeyAdapter() {
			public void keyTyped (KeyEvent ke) {
				char c = ke.getKeyChar ();
				if (!((Character.isDigit (c) || (c ==
KeyEvent.VK_BACK_SPACE)))) {
					getToolkit().beep ();
					ke.consume ();
				}
			}
		}
		);
		String Months[] = {"January", "February", "March", "April", "May", "June",
			"July", "August", "September", "October", "November", "December"};
		cboMonth = new JComboBox (Months);
		cboDay = new JComboBox ();
		cboYear = new JComboBox ();
		for (int i = 1; i <= 31; i++) {
			String days = "" + i;
			cboDay.addItem (days);
		}
		for (int i = 1900; i <= 2024; i++) {
			String years = "" + i;
			cboYear.addItem (years);
		}
		cboMonth.setBounds (105, 90, 92, 25);
		cboDay.setBounds (202, 90, 43, 25);
		cboYear.setBounds (250, 90, 60, 25);
		btnSave = new JButton ("Save");
		btnSave.setBounds (20, 165, 120, 25);
		btnSave.addActionListener (this);
		btnCancel = new JButton ("Cancel");
		btnCancel.setBounds (185, 165, 120, 25);
		btnCancel.addActionListener (this);
		jpWith.add (lbNo);
		jpWith.add (txtNo);
		jpWith.add (lbName);
		jpWith.add (txtName);
		jpWith.add (lbDate);
		jpWith.add (cboMonth);
		jpWith.add (cboDay);
		jpWith.add (cboYear);
		jpWith.add (lbWithdraw);
		jpWith.add (txtWithdraw);
		jpWith.add (btnSave);
		jpWith.add (btnCancel);
		getContentPane().add (jpWith);
		populateArray ();   //Load All Existing Records in Memory.
		//In the End Showing the New Account Window.
		setVisible (true);

	}
	public void actionPerformed (ActionEvent ae) {
		Object obj = ae.getSource();
		if (obj == btnSave) {
			if (txtNo.getText().equals("")) {
				JOptionPane.showMessageDialog (this, "Please! Provide Id of
Customer.",
                                        "BankSystem - EmptyField",
JOptionPane.PLAIN_MESSAGE);
                              txtNo.requestFocus();
                        }
                        else if (txtWithdraw.getText().equals("")) {
                              JOptionPane.showMessageDialog (this, "Please! Provide Withdraw
Amount.",
                                        "BankSystem - EmptyField",
JOptionPane.PLAIN_MESSAGE);
                              txtWithdraw.requestFocus ();
                        }
                        else {
                              withdraw = Integer.parseInt (txtWithdraw.getText ());
                              if (curr == 0) {
                                    JOptionPane.showMessageDialog (this, txtName.getText () +
" doesn't have any Amount in Balance.",
                                              "BankSystem - EmptyAccount",
JOptionPane.PLAIN_MESSAGE);
                                    txtClear ();
                              }
                              else if (withdraw > curr) {
                                    JOptionPane.showMessageDialog (this, "Withdraw Amount
can't greater than Actual Balance.",
                                              "BankSystem - Large Amount",
JOptionPane.PLAIN_MESSAGE);
                                    txtWithdraw.setText ("");
                                    txtWithdraw.requestFocus ();
                              }
                              else {
                                    editRec ();   //Update the Contents of Array.
                              }
                        }
                  }
                  if (obj == btnCancel) {
                        txtClear ();
                        setVisible (false);
                        dispose();
                  }
            }
            void populateArray () {
                  try {
                        fis = new FileInputStream ("Bank.dat");
                        dis = new DataInputStream (fis);
                        //Loop to Populate the Array.
                        while (true) {
                              for (int i = 0; i < 6; i++) {
                                    records[rows][i] = dis.readUTF ();
```

```
                                }
                                rows++;
                        }
                }
                catch (Exception ex) {
                        total = rows;
                        if (total == 0) {
                                JOptionPane.showMessageDialog (null, "Records File is
Empty.\nEnter Records First to Display.",
                                                        "BankSystem - EmptyFile",
JOptionPane.PLAIN_MESSAGE);
                                btnEnable ();
                        }
                        else {
                                try {
                                        dis.close();
                                        fis.close();
                                }
                                catch (Exception exp) { }
                        }
                }
        }
        void findRec () {

                boolean found = false;
                for (int x = 0; x < total; x++) {
                        if (records[x][0].equals (txtNo.getText())) {
                                found = true;
                                showRec (x);
                                break;
                        }
                }
                if (found == false) {
                        String str = txtNo.getText ();
                        txtClear ();
                        JOptionPane.showMessageDialog (this, "Account No. " + str + " doesn't
Exist.",
                        "BankSystem - WrongNo", JOptionPane.PLAIN_MESSAGE);
                }

        }
        //Function which display Record from Array onto the Form.
        public void showRec (int intRec) {

                txtNo.setText (records[intRec][0]);
                txtName.setText (records[intRec][1]);
                curr = Integer.parseInt (records[intRec][5]);
```

```
		recCount = intRec;

	}
		void txtClear () {
		txtNo.setText ("");
		txtName.setText ("");
		txtWithdraw.setText ("");
		txtNo.requestFocus ();

	}


		public void editRec () {
		records[recCount][0] = txtNo.getText ();
		records[recCount][1] = txtName.getText ();
		records[recCount][2] = "" + cboMonth.getSelectedItem ();
		records[recCount][3] = "" + cboDay.getSelectedItem ();
		records[recCount][4] = "" + cboYear.getSelectedItem ();
		records[recCount][5] = "" + (curr - withdraw);
		editFile ();	//Save This Array to File.

	}
	public void editFile () {
		try {
			FileOutputStream fos = new FileOutputStream ("Bank.dat");
			DataOutputStream dos = new DataOutputStream (fos);
			if (records != null) {
				for (int i = 0; i < total; i++) {
					for (int c = 0; c < 6; c++) {
						dos.writeUTF (records[i][c]);
						if (records[i][c] == null) break;
					}
				}
				JOptionPane.showMessageDialog (this, "The File is Updated
Successfully",
						"BankSystem - Record Saved",
JOptionPane.PLAIN_MESSAGE);
				txtClear ();
				dos.close();  fos.close();
			}
		}
		catch (IOException ioe) {
			JOptionPane.showMessageDialog (this, "There are Some Problem with
File",
					"BankSystem - Problem", JOptionPane.PLAIN_MESSAGE);
		}
	}
```

```
		void btnEnable () {
		txtNo.setEnabled (false);
		cboMonth.setEnabled (false);
		cboDay.setEnabled (false);
		cboYear.setEnabled (false);
		txtWithdraw.setEnabled (false);
		btnSave.setEnabled (false);

	}
}
```

3.6 VIEW ALL CUSTOMERS

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.table.DefaultTableModel;
public class ViewCustomer extends JInternalFrame {
	private JPanel jpShow = new JPanel ();
	private DefaultTableModel dtmCustomer;
	private JTable tbCustomer;
	private JScrollPane jspTable;
	private int row = 0;
	private int total = 0;
	private String rowData[][];
	private FileInputStream fis;
	private DataInputStream dis;
	ViewCustomer () {
		super ("View All Account Holders", false, true, false, true);
		setSize (475, 280);
		jpShow.setLayout (null);
		populateArray ();
		tbCustomer = makeTable ();
		jspTable = new JScrollPane (tbCustomer);
		jspTable.setBounds (20, 20, 425, 200);
		jpShow.add (jspTable);
		getContentPane().add (jpShow);
		setVisible (true);
	}
	void populateArray () {
		String rows[][] = new String [500][6];
		try {
			fis = new FileInputStream ("Bank.dat");
			dis = new DataInputStream (fis);

			while (true) {
				for (int i = 0; i < 6; i++) {
					rows[row][i] = dis.readUTF ();
				}
				row++;
			}
		}
		catch (Exception ex) {
			total = row;
			rowData = new String [total][4];
			if (total == 0) {
				JOptionPane.showMessageDialog (null, "Records File is
Empty.\nEnter Records to Display.",
		"BankSystem - EmptyFile", JOptionPane.PLAIN_MESSAGE);
			}
			else {
				for (int i = 0; i < total; i++) {
					rowData[i][0] = rows[i][0];
					rowData[i][1] = rows[i][1];
					rowData[i][2] = rows[i][2] + ", " + rows[i][3] + ", " +
rows[i][4];
					rowData[i][3] = rows[i][5];
				}
				try {
					dis.close();
					fis.close();
				}
				catch (Exception exp) { }
			}
		}

	}

	//Function to Create the Table and Add Data to Show.
	private JTable makeTable () {

		//String Type Array use to Give Table Column Names.
		String cols[] = {"Account No.", "Customer Name", "Opening Date", "Bank
Balance"};

		dtmCustomer  = new DefaultTableModel (rowData, cols);
		tbCustomer = new JTable (dtmCustomer) {
			public boolean isCellEditable (int iRow, int iCol) {
				return false; //Disable All Columns of Table.
			}
		};
		(tbCustomer.getColumnModel().getColumn(0)).setPreferredWidth (180);
		(tbCustomer.getColumnModel().getColumn(1)).setPreferredWidth (275);
		(tbCustomer.getColumnModel().getColumn(2)).setPreferredWidth (275);
		(tbCustomer.getColumnModel().getColumn(3)).setPreferredWidth (200);
		tbCustomer.setRowHeight (20);
		tbCustomer.setSelectionMode (ListSelectionModel.SINGLE_SELECTION);
		return tbCustomer;

	}
}
```

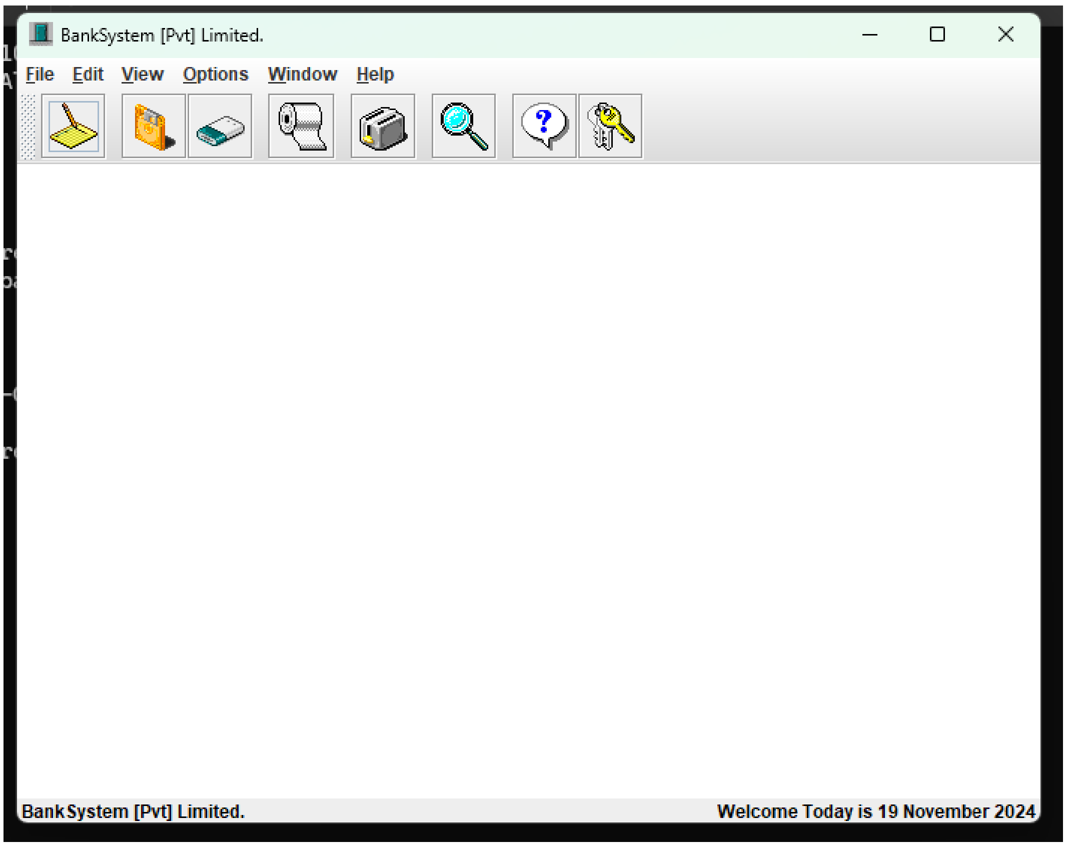
3.7 VIEW PARTICULAR CUSTOMER

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class ViewOne extends JInternalFrame implements ActionListener {
	private JPanel jpRec = new JPanel();
	private JLabel lbNo, lbName, lbDate, lbBal;
	private JTextField txtNo, txtName, txtDate, txtBal, txtRec;
	private JButton btnFirst, btnBack, btnNext, btnLast;

	private int recCount = 0;
	private int rows = 0;	private	int total = 0;
	private String records[][] = new String [500][6];
	private FileInputStream fis;	private DataInputStream dis;
	ViewOne () {
		super ("View Account Holders", false, true, false, true);
		setSize (350, 235);
		jpRec.setLayout (null);
		lbNo = new JLabel ("Account No:");
		lbNo.setForeground (Color.black);
		lbNo.setBounds (15, 20, 80, 25);
		lbName = new JLabel ("Person Name:");
		lbName.setForeground (Color.black);
		lbName.setBounds (15, 55, 80, 25);
		lbDate = new JLabel ("Last Transaction:");
		lbDate.setForeground (Color.black);
		lbDate.setBounds (15, 90, 100, 25);
		lbBal = new JLabel ("Balance:");
		lbBal.setForeground (Color.black);
		lbBal.setBounds (15, 125, 80, 25);
		txtNo = new JTextField ();
		txtNo.setHorizontalAlignment (JTextField.RIGHT);
		txtNo.setEnabled (false);
		txtNo.setBounds (125, 20, 200, 25);
		txtName = new JTextField ();
		txtName.setEnabled (false);
		txtName.setBounds (125, 55, 200, 25);
		txtDate = new JTextField ();
		txtDate.setEnabled (false);
		txtDate.setBounds (125, 90, 200, 25);
		txtBal = new JTextField ();
		txtBal.setHorizontalAlignment (JTextField.RIGHT);
		txtBal.setEnabled (false);
		txtBal.setBounds (125, 125, 200, 25);
		btnFirst = new JButton ("<<");
		btnFirst.setBounds (15, 165, 50, 25);
		btnFirst.addActionListener (this);
		btnBack = new JButton ("<");
		btnBack.setBounds (65, 165, 50, 25);
		btnBack.addActionListener (this);
		btnNext = new JButton (">");
		btnNext.setBounds (225, 165, 50, 25);
		btnNext.addActionListener (this);
		btnLast = new JButton (">>");
		btnLast.setBounds (275, 165, 50, 25);
		btnLast.addActionListener (this);
		txtRec = new JTextField ();
		txtRec.setEnabled (false);
		txtRec.setHorizontalAlignment (JTextField.CENTER);
		txtRec.setBounds (115, 165, 109, 25);
		jpRec.add (lbNo);  jpRec.add (txtNo); jpRec.add (lbName);
		jpRec.add (txtName); jpRec.add (lbDate); jpRec.add (txtDate);
		jpRec.add (lbBal);		jpRec.add (txtBal);
		jpRec.add (btnFirst);	jpRec.add (btnBack);
		jpRec.add (btnNext);	jpRec.add (btnLast);
		jpRec.add (txtRec);
		getContentPane().add (jpRec);
		populateArray ();		showRec (0);
		setVisible (true);
	}
	public void actionPerformed (ActionEvent ae) {
		Object obj = ae.getSource();
		if (obj == btnFirst) {
			recCount = 0;
			showRec (recCount);
		}
		else if (obj == btnBack) {
			recCount = recCount - 1;
			if (recCount < 0) {
				recCount = 0;
				showRec (recCount);
				JOptionPane.showMessageDialog (this, "You are on First Record.",
							"BankSystem - 1st Record",
JOptionPane.PLAIN_MESSAGE);
                        }
                        else {
                                showRec (recCount);
                        }
                }
                else if (obj == btnNext) {
                        recCount = recCount + 1;
                        if (recCount == total) {
                                recCount = total - 1;
                                showRec (recCount);
                                JOptionPane.showMessageDialog (this, "You are on Last Record.",
                                                "BankSystem - End of Records",
JOptionPane.PLAIN_MESSAGE);
                        }
                        else {
                                showRec (recCount);
                        }
                }
                else if (obj == btnLast) {
                        recCount = total - 1;
                        showRec (recCount);
                }
        }
        void populateArray () {

                try {
                        fis = new FileInputStream ("Bank.dat");
                        dis = new DataInputStream (fis);
                        //Loop to Populate the Array.
                        while (true) {
                                for (int i = 0; i < 6; i++) {
                                        records[rows][i] = dis.readUTF ();
                                }
                                rows++;
                        }
                }
                catch (Exception ex) {
                        total = rows;
                        if (total == 0) {
                                JOptionPane.showMessageDialog (null, "Records File is
Empty.\nEnter Records First to Display.",
                                                        "BankSystem - EmptyFile",
JOptionPane.PLAIN_MESSAGE);
                                btnEnable ();
                        }
                        else {
```

```
				try {
					dis.close();
					fis.close();
				}
				catch (Exception exp) { }
			}
		}

	}

	//Function which display Record from Array onto the Form.
	public void showRec (int intRec) {

		txtNo.setText (records[intRec][0]);
		txtName.setText (records[intRec][1]);
		txtDate.setText (records[intRec][2] + ", " + records[intRec][3] + ", " +
records[intRec][4]);
		txtBal.setText (records[intRec][5]);
		if (total == 0) {
			txtRec.setText (intRec + "/" + total); //Showing Record No. and Total
Records.
			txtDate.setText ("");
		}
		else {
			txtRec.setText ((intRec + 1) + "/" + total); //Showing Record No. and Total
Records.
		}

	}

	//Function use to Lock all Buttons of Window.
	void btnEnable () {

		btnFirst.setEnabled (false);
		btnBack.setEnabled (false);
		btnNext.setEnabled (false);
		btnLast.setEnabled (false);

	}

}
```

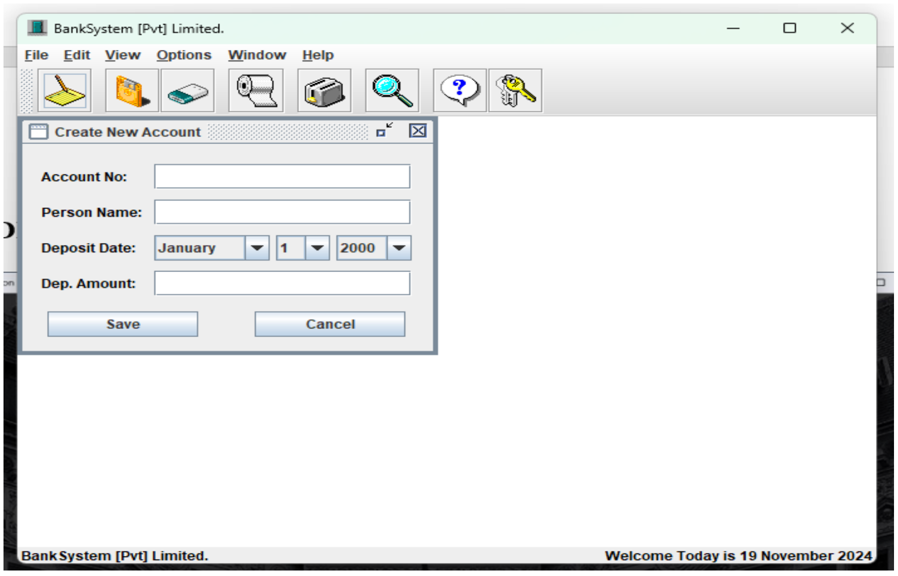
SNAPSHOTS

4.1 HOME PAGE



BankSystem [Pvt] Limited. Welcome Today is 19 November 2024

4.2 CREATE NEW ACCOUNT PAGE



Account No:

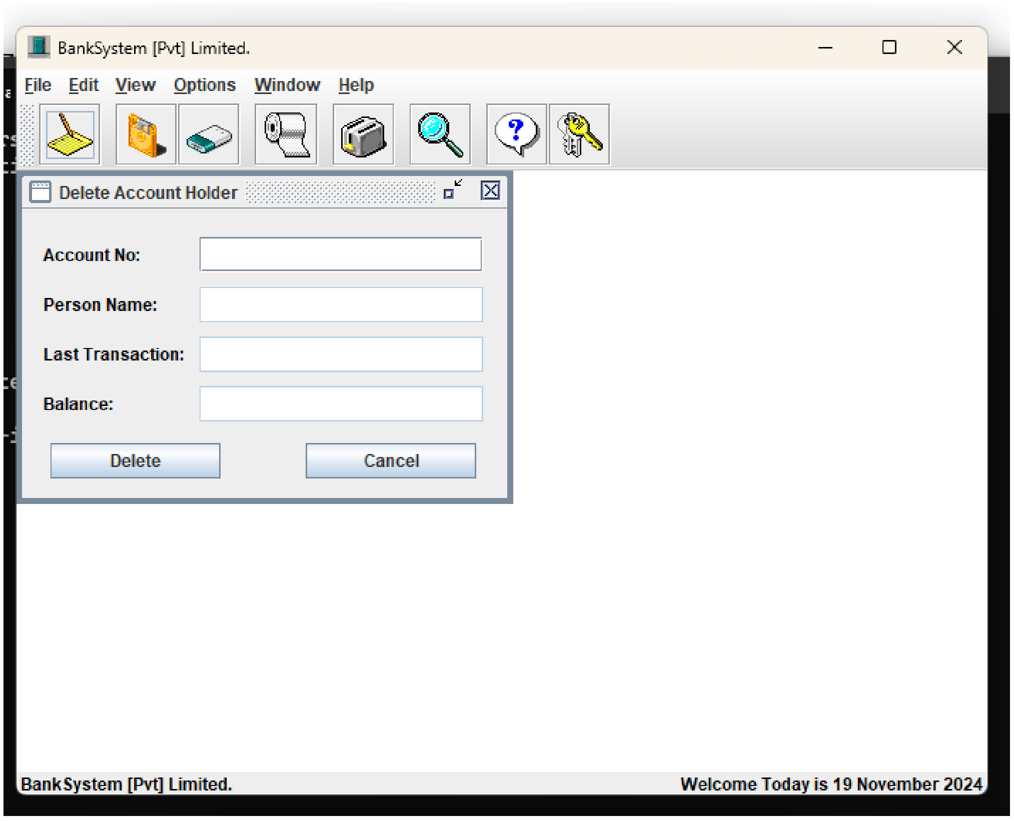
Person Name:

Deposit Date: January 1 2000

Dep. Amount:

BankSystem [Pvt] Limited. Welcome Today is 19 November 2024

4.3 DELETING AN ACCOUNT PAGE



Account No:

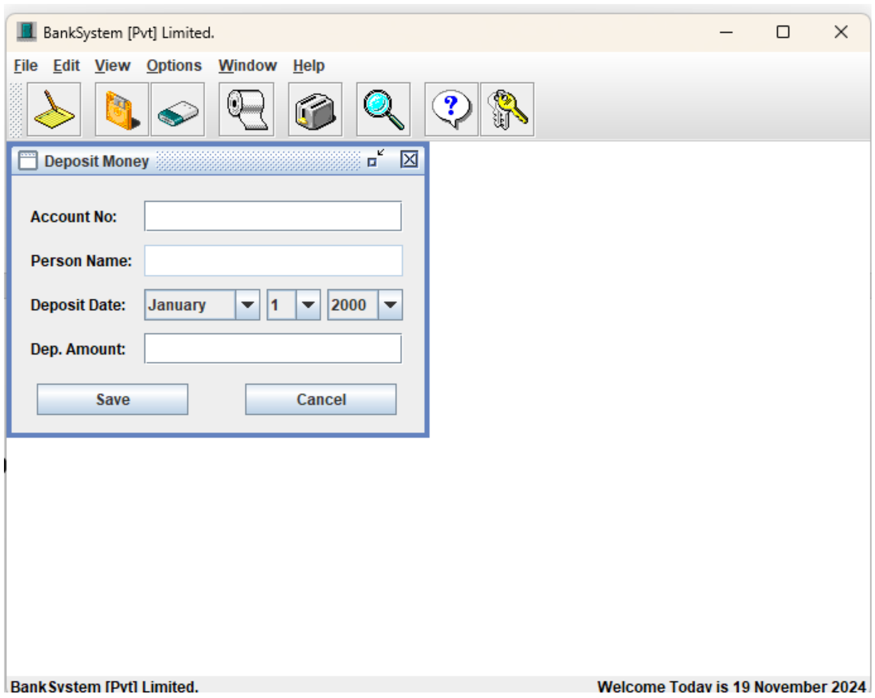
Person Name:

Last Transaction:

Balance:

BankSystem [Pvt] Limited. Welcome Today is 19 November 2024

4.4 DEPOSIT MONEY PAGE



Account No:

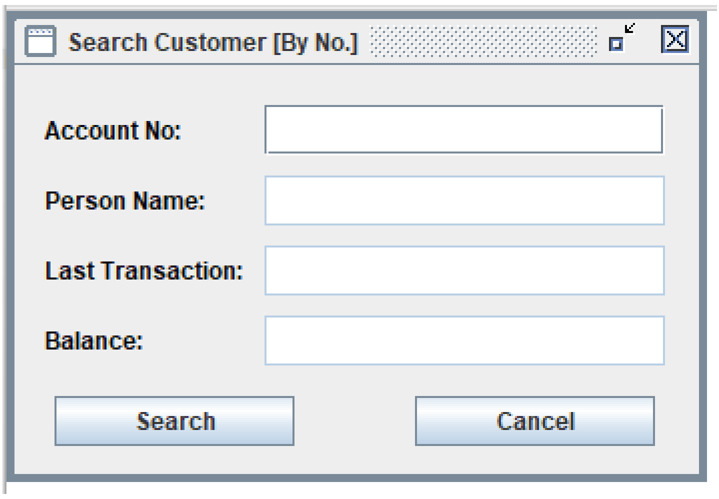
Person Name:

Deposit Date: January 1 2000

Dep. Amount:

BankSystem [Pvt] Limited. Welcome Today is 19 November 2024

4.5 FIND AN ACCOUNT PAGE

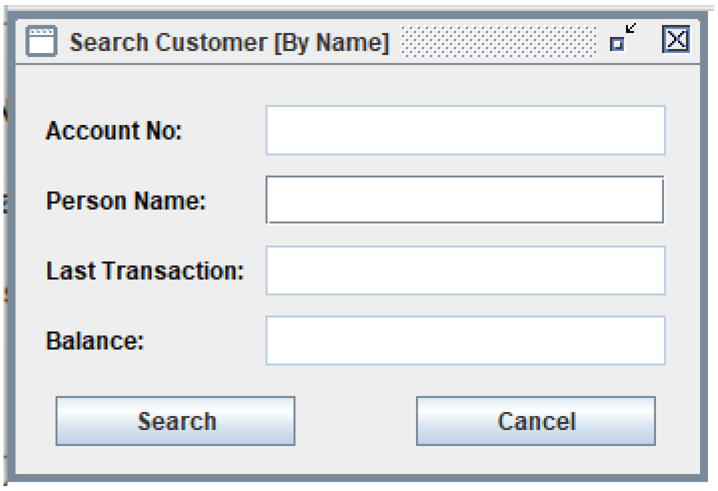

Search Customer [By No.]

Account No:

Person Name:

Last Transaction:

Balance:


Search Customer [By Name]

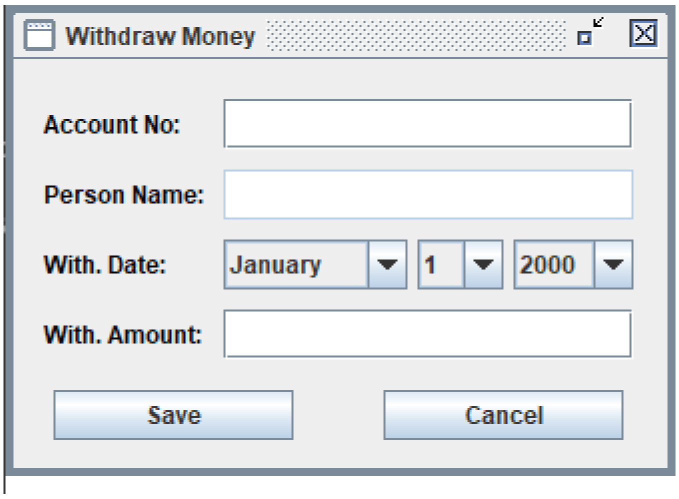
Account No:

Person Name:

Last Transaction:

Balance:

4.6 WITHDRAW AN ACCOUNT PAGE


Withdraw Money

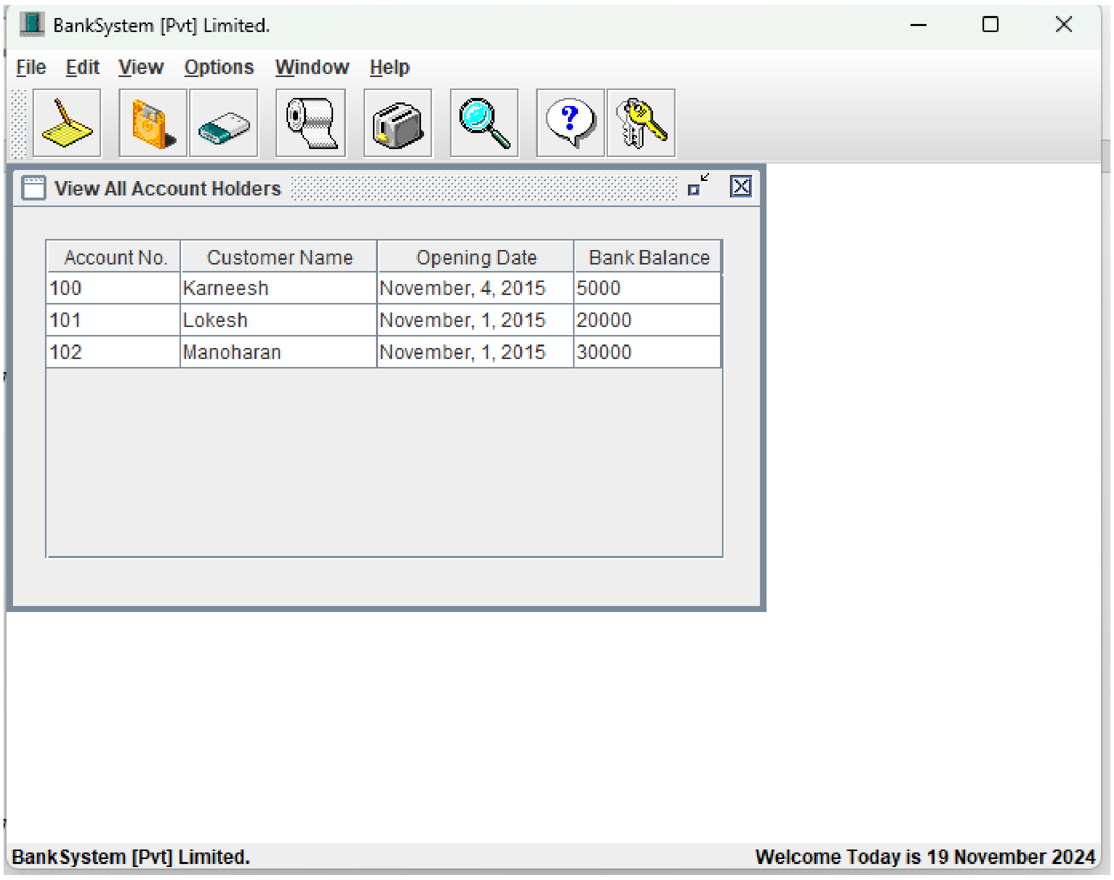
Account No:

Person Name:

With. Date: January 1 2000

With. Amount:

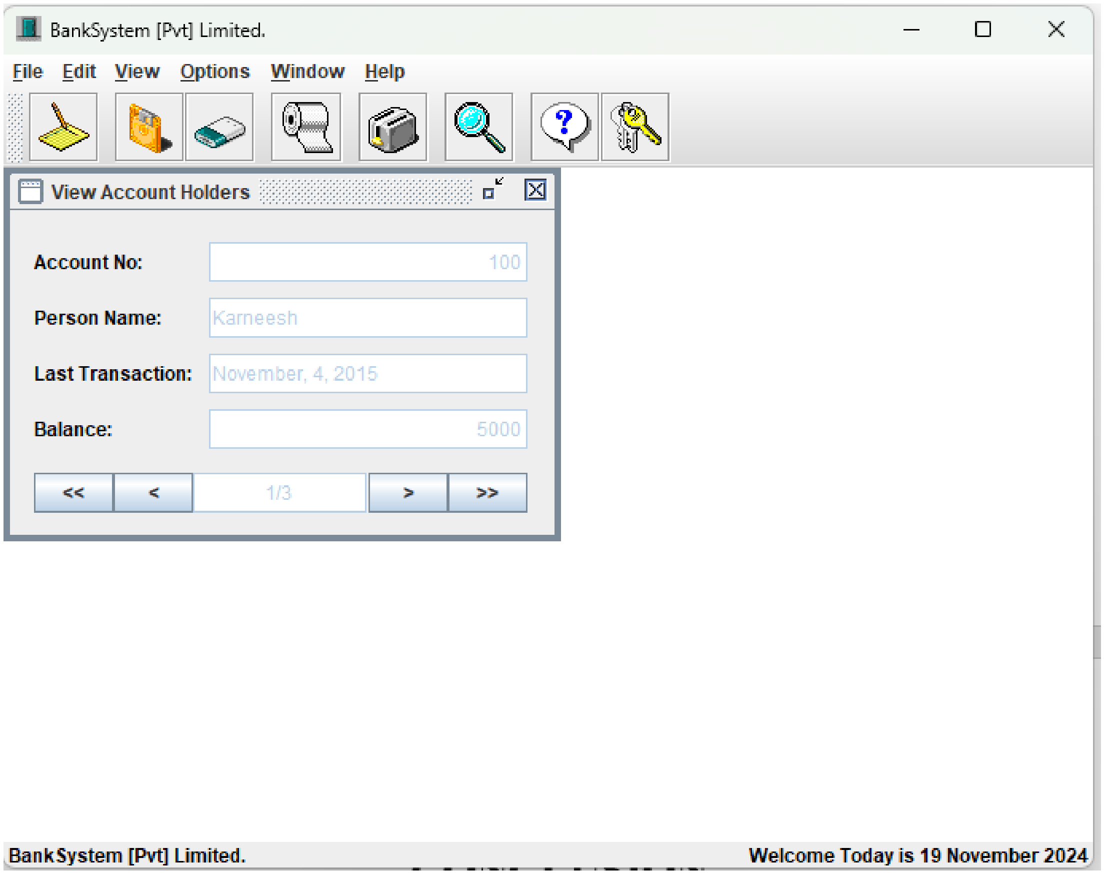
4.7 VIEW ALL CUSTOMER PAGE



Account No.	Customer Name	Opening Date	Bank Balance
100	Karneesh	November, 4, 2015	5000
101	Lokesh	November, 1, 2015	20000
102	Manoharan	November, 1, 2015	30000

BankSystem [Pvt] Limited. Welcome Today is 19 November 2024

4.8 VIEW PARTICULAR CUSTOMER PAGE



Account No: 100

Person Name: Karneesh

Last Transaction: November, 4, 2015

Balance: 5000

BankSystem [Pvt] Limited. Welcome Today is 19 November 2024

CONCLUSION

With the help of our project, bankers will be able to do their banking work without any cumbersome and save their precious time as well as they can manage many customers at the same time without any difficulty and it makes a good relationship between customers and bankers.

REFERENCES

1. **https://www.javatutorial.com**

2. https://www.geeksforgeeks.org

3. https://www.w3schools.com/sql/

4. https:// www.programmiz.com

5. https:// www.oracle.com

6. SQL | Codecademy