

## Criterion C: Development

### *Techniques used:*

#### Libraries Imported

```
//technique- importing packages
import java.sql.*;
import java.util.Date;
import java.text.*;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
```

SQL(Structured Query language) is used for the program to communicate with the MS-Access database used to hold the data.

util.Date, util.DateFormat, text.SimpleDateFormat and util.Calendar are used for collecting and formatting the dates of input.

awt and swing are used in the graphical components.

#### SQL commands used

```
addRow= "INSERT INTO "+tableName+" (month, year, investmentonraw) VALUES("+ month + " , " + year + " , "+ amnt + ")";
```

Figure 1 - adding rows

```
updateRow= "UPDATE "+tableName+" SET investmentonraw="+ (amnt+eamnt) +" WHERE month="+ month + " AND year="+year+"";
```

Figure 2 - Updating rows

```
String tableName= "RawandDesign";
ResultSet rs=stmt1.executeQuery("Select id,gonefordesign from "+tableName);
rs= stmt1.getResultSet();
```

Figure 3 - Reading fields from a table

```
String updateRow= "DELETE FROM "+tablename+" WHERE id="+ r +"";
stmt1.execute(updateRow);
```

Figure 4 - Deleting rows

#### Production of the GUI

The gui class handled almost all of the GUIs except for viewing the raw stock, and selecting the stock materials from a list.

#### Using JFrames and JPanels

```

class gui extends JFrame
{
    //Declaring all the panels used

    public JPanel panraw= new JPanel(); //the raw panel
    public JPanel pan= new JPanel(); //the design panel
    public JPanel pan3= new JPanel();
    public JPanel panaddin= new JPanel(); //the additional information panel
    public JPanel panred= new JPanel(); //the design to readymade panel
    public JPanel panared = new JPanel(); //the add new readymade panel
    public JPanel pansell = new JPanel(); // the sell item panel
    public JPanel pancal = new JPanel(); //the price calculator panel
    public JPanel panmin = new JPanel(); //the monthly investment panel
    public JPanel panyin = new JPanel(); //the yearly investment panel
    public JPanel panmpo = new JPanel(); //the monthly profit panel
    public JPanel panypro = new JPanel(); //the yearly profit panel
    public JPanel panrep = new JPanel(); //the report panel
}

```

The class gui inherited JFrame, while objects of JPanels were used for all the panels. I was sure that I wouldn't modify any of the default properties of JPanel while the same wasn't true for the main JFrame. I chose inheritance so that functions of JFrame could be overridden easily if necessary.

```

public class viewrawstock
{
    public static JFrame viewstockfr= new JFrame("Available Raw Stock");
    public static JPanel pante= new JPanel();
}

```

```

public class stockselection
{
    public static JFrame selectstockfr= new JFrame("Available Raw Stock");
    public static JPanel pante= new JPanel();
}

```

By the time I coded the viewrawstock and stockselection class, I knew that overriding functions wouldn't be necessary. Hence object creation seemed to suffice.

All panels are kept global so that they could be accessed from all functions in the class as well as from its objects.

### Centering JFrames

```

super("Meera's collection"); //transfers the call to the constructor of the JFrame class

Toolkit tk= Toolkit.getDefaultToolkit();
Dimension dim= tk.getScreenSize();
setSize(800,680); //resolution of the window- 800*600

//technique - centering the window
setLocation((dim.width- this.getWidth())/2, (dim.height- this.getHeight())/2);

```

This technique of implementing the Toolkit to find the screen size and then centring the JFrame was used throughout the program.

### Closing JFrames

```
//technique - Making the window responsive
addWindowListener(new WindowAdapter()
{
    //technique- closing the window completely without any delay when the close button is pressed
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
});
```

```
viewstockfr.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

To close the window, both the first and the second techniques was used. WindowListener was used where I was unsure that closing the window was the only task that required to be done when the cross button was clicked. In other places, setDefaultCloseOperation is used.

### Adding the menubar

```
public void addMenu()throws Exception
{
    JMenuBar mbar= new JMenuBar();
    JMenu mraw= new JMenu("Raw");
    JMenu mdes= new JMenu("Design");
    JMenu mready= new JMenu("Readymade");
    JMenu msell = new JMenu("Sell");
    JMenu minf= new JMenu("Information");
    JMenu madd= new JMenu("Additional");
    JMenuItem item1= new JMenuItem("Add new Raw Stock");
    JMenuItem item3= new JMenuItem("View Raw Stock");
    JMenuItem item2= new JMenuItem("Send raw to design");
    JMenuItem item4= new JMenuItem("Monthly Investment");
    JMenuItem item5= new JMenuItem("Yearly Investment");
    JMenuItem item6= new JMenuItem("Monthly Profit");
    JMenuItem item7= new JMenuItem("Yearly Profit");
    JMenuItem item10= new JMenuItem("Design to Readymade");
    JMenuItem item9= new JMenuItem("Add new Readvmade");
    JMenuItem item11= new JMenuItem("Add Additional Investment");
    JMenuItem item13 = new JMenuItem("Price Calculator");
```

```
mraw.add(item1);
mraw.add(item3);
mdes.add(item2);
minf.add(item4);
minf.add(item5);
minf.add(item6);
minf.add(item7);
mready.add(item10);
mready.add(item9);
madd.add(item11);
msell.add(item13);
mbar.add(mraw);
mbar.add(mdes);
mbar.add(mready);
mbar.add(msell);
```

The menubar, menus and menuitems are created and the menuitems added to the respective menus and the menus to the menubar.

### Refreshing panels

```
item1.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            // removing existing panels
            getContentPane().removeAll();

            //adding the new panel (panraw)
            getContentPane().add(panraw);
            getContentPane().validate();
            getContentPane().repaint();
        }catch(Exception r){}
    }
});
```

removeAll() is used to remove the current panel, add() to add the current panel, and validate() and repaint() to make the panel responsive to user interaction.

### Adding other swing components

```
pan.setLayout(null);
```

```
JLabel designlabel= new JLabel("Select design:");
```

```
JRadioButton batik= new JRadioButton("Batik");
batik.setBounds(60,100, 120, 20);
```

```
JScrollPane jsp= new JScrollPane(det);
```

```
final JComboBox selma= new JComboBox(tearry);
```

```
public JOptionPane msgdialog= new JOptionPane();
```

```
JButton i= new JButton("Select from stock list");
```

To position components in the JPanel manually, the layout is set to 'null' and things positioned with setBounds which allows you to enter the x and y co-ordinates of the starting point, and the length and breadth of the component desired. JLabels were used in conjunction to other components to label their functionality. Radiobuttons were used when the inputs were specific and pre-defined. JComboBox was also used for this purpose sometimes, although it was used mainly for adding dynamic selection lists.

```

ButtonGroup bgdes= new ButtonGroup();

bgdes.add(batik);
bgdes.add(embroidery);
bgdes.add(acid);
bgdes.add(aplick);
bgdes.add(hand);
bgdes.add(block);

```

Buttongroups were used where only one selection from a list of Radiobuttons was necessary.

#### Getting the present month and year and thereby updating the Financialchart table

```

public DateFormat curyear = new SimpleDateFormat("yyyy");
public DateFormat curmonth = new SimpleDateFormat("MM");
public Date date = new Date();

```

Figure 5 - Global Declarations

```

// updating the financialchart table
int cury= Integer.parseInt(curyear.format(date));
int curm =Integer.parseInt(curmonth.format(date));
iaadd.search(curm, cury, 'd', (Double.parseDouble(designprice.getText())*(desarr.length)));

```

Figure 6 - Calling the search method from the Investmentadd class

The search method in the Investmentadd class takes the current month, year, an investment or profit identifier, and the amount invested as parameters.

The investment or profit identifier tells the function which column to edit the information in. Therefore, investments on raw materials are denoted by an 'r'.

Similarly, investment on design – 'd'

Additional investment – 'a'

Profit – 'p'

It then checks whether the month and year are present in the table. If it is, then it updates it, or else, it creates a new row with the information.

```

public class Investmentadd
{
    public static String tableName = "Financialchart";

    public static void search(int month, int year, char r, double amnt) throws Exception
    {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
        Connection conn1= DriverManager.getConnection(dbUrl,"","");
        Statement stmt1= conn1.createStatement();

        ResultSet rs=stmt1.executeQuery("Select * from "+tableName);
        rs= stmt1.getResultSet();
        int ty, tm, flag = 0;
        double tam = 0;
        while((rs!=null) && (rs.next()))
        {
            ty = rs.getInt("year");
            tm = rs.getInt("month");

            if(r== 'r')
                tam = rs.getDouble("investmentonraw");
            else if(r == 'd')
                tam = rs.getDouble("investmentondesign");
            else if(r == 'a')
                tam = rs.getDouble("additionalinvestment");
            else if(r == 'p')
                tam = rs.getDouble("profit");

            if(ty == year && tm == month)
            {
                flag = 1;
                update(month, year, r, amnt, tam);
            }
        }
        if(flag == 0)
        {
            add(month, year, r, amnt);
        }
    }
}

```

Figure 7 - The search method from the Investmentadd class

```

public static void add(int month, int year, char r, double amnt)throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection connl= DriverManager.getConnection(dbUrl,"","");
    Statement stmtl= connl.createStatement();

    String addRow = "";

    if(r == 'r')
    {
        addRow= "INSERT INTO "+tableName+" (month, year, investmentonraw) VALUES("+ month + " , " + year + " ,"+ amnt +")";
    }
    if(r == 'd')
    {
        addRow= "INSERT INTO "+tableName+" (month, year, investmentondesign) VALUES("+ month + " , " + year + " ,"+ amnt +")";
    }
    if(r == 'a')
    {
        addRow= "INSERT INTO "+tableName+" (month, year, additionalinvestment) VALUES("+ month + " , " + year + " ,"+ amnt +")";
    }
    if(r == 'p')
    {
        addRow= "INSERT INTO "+tableName+" (month, year, profit) VALUES("+ month + " , " + year + " ,"+ amnt +")";
    }

    stmtl.execute(addRow);

    stmtl.close();
    connl.close();

    stmtl.close();
    connl.close();
}

```

Figure 8 - The add method in the Investmentadd class

```

public static void update(int month, int year, char r, double amnt, double eamnt)throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection connl= DriverManager.getConnection(dbUrl,"","");
    Statement stmtl= connl.createStatement();

    String updateRow = "";

    if(r == 'r')
    {
        updateRow= "UPDATE "+tableName+" SET investmentonraw="+ (amnt+eamnt) +" WHERE month="+ month + " AND year="+year+"";
    }
    if(r == 'd')
    {
        updateRow= "UPDATE "+tableName+" SET investmentondesign="+ (amnt+eamnt) +" WHERE month="+ month + " AND year="+year+"";
    }
    if(r == 'a')
    {
        updateRow= "UPDATE "+tableName+" SET additionalinvestment="+ (amnt+eamnt) +" WHERE month="+ month + " AND year="+year+"";
    }
    if(r == 'p')
    {
        updateRow= "UPDATE "+tableName+" SET profit="+ (amnt+eamnt) +" WHERE month="+ month + " AND year="+year+"";
    }

    //System.out.println(updateRow);
    stmtl.execute(updateRow);
}

```

Figure 9 - The update method in the Investmentadd class

### Creating new entries in the rawmaterial table

```
public void paneraw() throws Exception // Decorating the Raw panel
{
    // Setting up manual layout. The co-ordinates of every object will be placed manually.
    panraw.setLayout(null);

    // Using radiobuttons to create a predefined list of possible stock builds so that items can be segregated easily and grouped,
    // without having to type out a lot.
    // These predefined materials include Cotton, Taant, Silk, Tashor, Net, Jute, Patola and Chanderi

    JRadioButton cotton= new JRadioButton("Cotton");
    cotton.setBounds(60,100, 120, 20);

    JRadioButton taant= new JRadioButton("Taant");
    taant.setBounds(60,130, 120, 20);

    JRadioButton silk= new JRadioButton("Silk");
    silk.setBounds(60,160, 120, 20);

    JRadioButton tashor= new JRadioButton("Tashor");
    tashor.setBounds(60,190, 120, 20);

    JRadioButton net= new JRadioButton("Net");
    net.setBounds(60, 220, 120, 20);

    JRadioButton jute= new JRadioButton("Jute");
    jute. setBounds(60,250, 120, 20);

    JRadioButton patola= new JRadioButton("Patola");
    patola. setBounds(60,280, 120, 20);

    JRadioButton chanderi= new JRadioButton("Chanderi");
    chanderi. setBounds(60,310, 120, 20);
}
```

Figure 10 - Creating Radiobuttons of the possible material selections

```
ButtonGroup bgraw= new ButtonGroup();

bgraw. add(cotton);
bgraw. add(taant);
bgraw. add(silk);
bgraw. add(tashor);
bgraw. add(net);
bgraw. add(jute);
bgraw. add(patola);
bgraw. add(chanderi);

// Adding the radiobuttons to the panel(panraw)
panraw.add(cotton);
panraw.add(taant);
panraw.add(silk);
panraw.add(tashor);
panraw.add(net);
panraw.add(jute);
panraw.add(patola);
panraw.add(chanderi);
```

Figure 11 - Adding the Radiobuttons to a buttongroup

```
final JTextField cquan = new JTextField();
final JLabel cquanlabel= new JLabel("Enter cost per quantity : ");
final JTextField quan = new JTextField();
final JLabel quanlabel= new JLabel("Enter quantity : ");
```

Figure 12 - Declaring the visual elements in the panel



```

taant. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            materialtemp="taant";
        }catch(Exception r){}
    }
});

```

Figure 13 - Changing the value of materialtemp

materialtemp is a temporary variable to store the selection of the Radiobuttons. It is linked to ActionListeners of every Radiobutton in the panel, so it changes every time the selection is changed.

```

// Adding actionlistener to the button 'Get Identity nos.'
iden.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            // System.out.println(quan.getText());
            int r= Integer.parseInt(quan.getText());
            // System.out.println(r);
            for(int t1=0; t1<r; t1++)
            {
                long t= identity.read();
                //System.out.println(t);
                adding.idadd(t);
                adding.materialadd(materialtemp, t);
                double d= Double.parseDouble(cquan.getText());
                adding.rawcostpriceadd(d, t);
                identity.edit();
            }

            // updating the financialchart table

            int cury= Integer.parseInt(curyear.format(date));
            int curm =Integer.parseInt(curmonth.format(date));
            iaadd.search(curm, cury, 'r', (Double.parseDouble(cquan.getText())*Integer.parseInt(quan.getText())));

            // Refreshing the JTextFields

            quan.setText("");
            cquan.setText("");
        }catch(Exception r){}
    }
});

```

Figure 14 - Reading the present value of the identity number stored in Identity.txt, assigning it to a saree, passing it off to the ReadyMaterial table, and then editing Identity.txt

```

public long read()throws Exception
{
    BufferedReader stdin= new BufferedReader(new InputStreamReader(System.in));

    FileReader file2= new FileReader("Identity.txt");
    BufferedReader buffer2= new BufferedReader(file2);

    String record=buffer2.readLine();
    long nn= Integer.parseInt(record);

    buffer2.close();

    return(nn);
}

```

Figure 15 - The read function of the nextid class which is called in the previous diagram

```

public static String tablename= "RawandDesign";

public static void materialadd(String s, long r)throws Exception
{
    //System.out.println(tablename);
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection conn1= DriverManager.getConnection(dbUrl,"","");
    Statement stmt1= conn1.createStatement();

    String updateRow= "UPDATE "+tablename+" SET materialtype='"+ s +"' WHERE id="+ r +"";
    //System.out.println(updateRow);
    stmt1.execute(updateRow);

    stmt1.close();
    conn1.close();
}

public static void idadd(long z)throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection conn1= DriverManager.getConnection(dbUrl,"","");
    Statement stmt1= conn1.createStatement();

    String addRow= "INSERT INTO "+tablename+" (id) VALUES("+ z +")";
    stmt1.execute(addRow);

    stmt1.close();
    conn1.close();
}

```

```

public static void rawcostpriceadd(double m, long r) throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection conn1= DriverManager.getConnection(dbUrl,"","");
    Statement stmt1= conn1.createStatement();

    String updateRow= "UPDATE "+tablename+" SET rawcostprice='"+ m +"' WHERE id="+ r +"";
    stmt1.execute(updateRow);

    stmt1.close();
    conn1.close();
}

```

Figure 16 - The idadd, materialadd and rawcostpriceadd functions of the rawandesadd class being called

```

// Editing the text file that holds the next id number
public void edit() throws IOException
{
    String record;
    long nn;

    BufferedReader stdin= new BufferedReader(new InputStreamReader(System.in));

    FileWriter file1 = new FileWriter("temp.txt", false);
    BufferedWriter buffer1= new BufferedWriter(file1);
    PrintWriter outfile= new PrintWriter(buffer1);

    FileReader file2= new FileReader("Identity.txt");
    BufferedReader buffer2= new BufferedReader(file2);

    while ((record= buffer2.readLine())!=null)
    {
        nn=Long.parseLong(record);
        outfile.println(nn+1);
    }

    buffer2.close();
    outfile.close();

    File xyz= new File("Identity.txt");
    xyz.delete();
    File abc= new File("temp.txt");
    abc.renameTo(xyz);
}

```

Figure 17 - Editing Identity.txt

### Sample Input

We can see how the data changes at each step with the following example

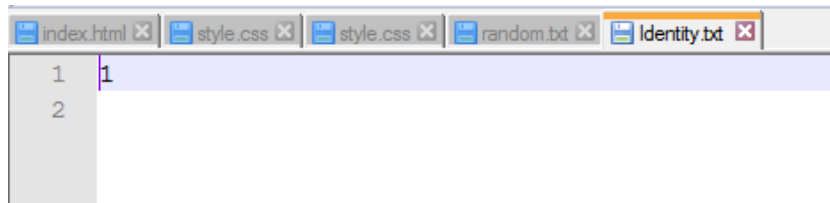


Figure 18 - Value stored in Identity.txt before running the program

id	materialtype	rawcostprice	gonefordesign	designtype	priceofdesign	nameofdesigner	contactnumberdesigner	detailsofdesigner
*								

Figure 19 - The RawandDesign table before running the program

Meera's collection

Raw Design Readymade Sell Information Additional

☐ Cotton  
☐ Taant  
☒ Silk  
☐ Tashor  
☐ Net  
☐ Jute  
☐ Patola  
☐ Chanderi

Enter quantity :

Enter cost per quantity :

Figure 20 - Sample Input

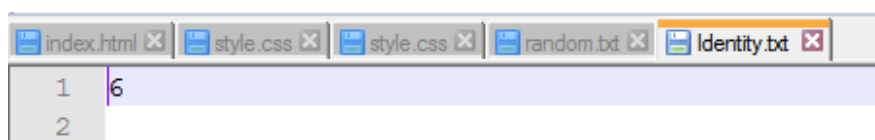


Figure 21 - Value of Identity.txt after the Input

id	materialtyp	rawcostprice	gonefordesi	designtype	priceofdesign	nameofdesi	contactnum	detailsofdesi
1	silk	300	0					
2	silk	300	0					
3	silk	300	0					
4	silk	300	0					
5	silk	300	0					
*								

Figure 22 - Changes in the RawandDesign table after the transaction

month	year	investment	investment	additionalin	profit
3	2014	1500			
*					

Figure 23 - Changes in the Financialchart table after the transaction

Deciphering the id nos. and updating design details to the rawmaterial table.

```

final JTextField quant = new JTextField();
JLabel quantlabel= new JLabel("Enter id nos. of sarees: ");

final JTextField named = new JTextField();
JLabel namedlabel= new JLabel("Enter name of designer: ");

final JTextField contn = new JTextField();
JLabel contnlabel= new JLabel("Enter contact no. of designer: ");

final JTextArea det = new JTextArea();
JLabel detlabel= new JLabel("Enter details of designer: ");

final JTextField designprice = new JTextField();
JLabel designpricelabel= new JLabel("Enter price per saree: ");

JScrollPane jsp= new JScrollPane(det);

// Adding a button that displays a list of available stock items and their ids that can be selected to send off.

JButton i= new JButton("Select from stock list");
JButton done = new JButton("OK");

```

Figure 24 - Components of the class

```

hand. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            designtemp="hand";
        }catch(Exception r){}
    }
});

```

Figure 25 - assigning a value to designtemp on selection of a Radiobutton

designtemp was a global variable meant to temporarily hold value of the selected radiobutton.

```

i. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            stc.displaypane();
            quant.setText(stc.returnst());
            System.out.println(stc.returnst());
        }catch(Exception r){}
    }
});

```

Figure 26 - Calls the displaypane method from the stockselection class when the button i is clicked

```

done. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            //creating the array with the number of ids in the input
            findid.createarray(quant.getText());
            long desarr[];

            //putting in the values in the array created
            desarr= findid.decipherids(quant.getText());

            //creating a for loop to run through the array and update records
            for(int i=0; i < desarr.length; i++)
            {
                //updating records
                //System.out.println(desarr[i]);
                adding.designtypeadd(designtemp,desarr[i]);
                adding.priceofdesignadd(Double.parseDouble(designprice.getText()), desarr[i]);
                adding.designernameadd(named.getText(), desarr[i]);
                adding.designercontactadd(contn.getText(), desarr[i]);
            }

            // updating the financialchart table
            int cury= Integer.parseInt(curyear.format(date));
            int curm =Integer.parseInt(curmonth.format(date));
            iaadd.search(curm, cury, 'd', (Double.parseDouble(designprice.getText())*(desarr.length)));

            //Refresghing the JTextFields and JTextAreas
            quant.setText("");
            designprice.setText("");
            named.setText("");
            contn.setText("");
            det.setText("");
        }catch(Exception r){}
    }
});

```

Figure 27 – Updating the records on the rawmaterial table

```
public class idsearch
{
    //creating the array that will hold the list of id nos.
    public long idlist[];
```

Figure 28 - Global Data members of the idsearch class being called

```
public void createarray(String ids)
{
    ids= ids+",";
    //System.out.println(ids);
    int d= ids.length();
    //creating the variable that will hold the size of the array temporarily;
    int temp=0;
    for(int i=0; i< d; i++)
    {
        char a= ids.charAt(i);
        if(a==' ' || a==',' || a==';' || a==' ')
            temp++;
    }

    idlist = new long[temp];
}
```

Figure 29 - The createarray function of the idsearch class

We first add a separator to the accepted string (no. of ids = no. of separators + 1). d becomes the length of the string entered, and temp becomes the number of ids.

```
//converting the String inputted into long values to be inserted into id slots.
public long[] decipherids (String ids)
{
    int cnt = 0;
    ids= ids+",";
    int d= ids.length();
    String temp = "";
    for(int i=0; i<d; i++)
    {
        char a= ids.charAt(i);
        if(a==' ' || a==',' || a==';' || a==' ')
        {
            idlist[cnt] = Long.parseLong(temp);
            //System.out.println(idlist[cnt]);
            temp = "";
            cnt++;
        }
        else
        {
            temp=temp+a;
        }
    }
    return idlist;
}
```

Figure 30 - The decipherids method from the idsearch class

```

public static void designypeadd(String s, long r) throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection connl= DriverManager.getConnection(dbUrl,"","");
    Statement stmtl= connl.createStatement();

    String updateRow= "UPDATE "+tablename+" SET designype='"+ s +"' WHERE id="+ r +"";
    stmtl.execute(updateRow);
    String updateRow2= "UPDATE "+tablename+" SET gonefordesign="+ 1 + " WHERE id="+ r +"";
    stmtl.execute(updateRow2);

    stmtl.close();
    connl.close();
}

public static void priceofdesignadd(double m, long r) throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection connl= DriverManager.getConnection(dbUrl,"","");
    Statement stmtl= connl.createStatement();

    String updateRow= "UPDATE "+tablename+" SET priceofdesign='"+ m +"' WHERE id="+ r +"";
    stmtl.execute(updateRow);

    stmtl.close();
    connl.close();
}

public static void designernameadd(String s, long r) throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection connl= DriverManager.getConnection(dbUrl,"","");
    Statement stmtl= connl.createStatement();

    String updateRow= "UPDATE "+tablename+" SET nameofdesigner='"+ s +"' WHERE id="+ r +"";
    stmtl.execute(updateRow);

    stmtl.close();
    connl.close();
}

public static void designercontactadd(String s, long r) throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection connl= DriverManager.getConnection(dbUrl,"","");
    Statement stmtl= connl.createStatement();

    String updateRow= "UPDATE "+tablename+" SET contactnumberdesigner='"+ s +"' WHERE id="+ r +"";
    stmtl.execute(updateRow);

    stmtl.close();
    connl.close();
}

```

Figure 31 - Updating the records in the ReadyMaterial table



### Sample Input

Financialchart									RawandDesign									ReadyMaterial								
id			materialtyp			rawcostprio			gonefordesi			designtype			priceofdesig			nameofdesi			contactnum			detailsofde		
1			silk			300			0																	
2			silk			300			0																	
3			silk			300			0																	
4			silk			300			0																	
5			silk			300			0																	
*																										

Figure 32 - RawandDesign table before the transaction

Financialchart		RawandDesign		ReadyMaterial	
month	year	investmentonraw	investmentondesign	additionalinvestment	profit
3		1500			
*					

Figure 33 - Financialchart table before the transaction

Meera's collection

Raw Design Readymade Sell Information Additional

☐ Batik  
☐ Embroidery  
☒ Acid Paint  
☐ Aplick  
☐ Hand Painting  
☐ Block

Enter id nos. of sarees:

Select from stock list

Enter price per saree:

Enter name of designer:

Enter contact no. of designer:

Enter details of designer:

OK

Figure 34 - Sample Input

id	materialtype	rawcostprice	gonefordesign	designtype	priceofdesign	nameofdesigner	contactnumberdesigner	detailsofdesigner
1	silk	300	-1	acid paint	200			
2	silk	300	0					
3	silk	300	-1	acid paint	200			
4	silk	300	0					
5	silk	300	0					
*								

Figure 35 - ReadyMaterial table after the transaction

month	year	investmentonraw	investmentondesign	additionalinvestment	profit
3		1500	400		
*					

Figure 36 - Financialchart table after the transaction

### Viewing Raw stock

```
public class viewrawstock
{
    public static JFrame viewstockfr= new JFrame("Available Raw Stock");
    public static JPanel pante= new JPanel();
    public static JTextArea textar= new JTextArea();
    public static JScrollPane js = new JScrollPane(textar);
    public static read readr= new read();

    public static JButton mc = new JButton("Cotton");
    public static JButton mtan = new JButton("Taant");
    public static JButton mt = new JButton("Tashor");
    public static JButton ms = new JButton("Silk");
    public static JButton mn = new JButton("Net");
    public static JButton mj = new JButton("Jute");
    public static JButton mp = new JButton("Patola");
    public static JButton mch = new JButton("Chanderi");
    public static JButton all = new JButton ("All");
}
```

Figure 37 - Global declarations in the viewrawstock class

Here, JButtons are used to fake the look of a JMenuBar. A JMenuBar was used at the beginning, but it did not respond properly to ActionListeners, and hence this was chosen as an alternative.

```
public static void displaypane()throws Exception
{
    Toolkit tk= Toolkit.getDefaultToolkit();
    Dimension dim= tk.getScreenSize();
    viewstockfr.setSize(735,600); //resolution of the window- 655*600
    pante.setLayout(null);

    //technique - centering the window
    viewstockfr.setLocation(((dim.width- viewstockfr.getWidth())/2)+40, (dim.height- viewstockfr.getHeight())/2);
    selectionbar();
    viewstockfr.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    viewstockfr.add(pante);
    viewstockfr.setVisible(true);
}
```

Figure 38 - The displaypane method in the viewrawstock class

```

public static void createtextbox(String mtype) throws Exception
{
    String darr[];

    js.setBounds(0,25, viewstockfr.getWidth(), (viewstockfr.getHeight()-18);
    textar.setVisible(true);
    pante.add(js);
    textar.setEditable(false);

    if(mtype.equals("all")== true)
    {
        textar.setText("\t" + "ID" + "\t" + "Material" + "\t" + "Price of Raw material");
        textar.append("\n");
        textar.append("\n");
    }
    else
    {
        textar.setText("\t" + "id nos." + "\t" + "Price of Raw material");
        textar.append("\n");
        textar.append("\n");
    }

    darr = readr.readrawforview(mtype);
    for (String x : darr)
    {
        textar.append(x);
        textar.append("\n");
    }
    System.out.println(darr[3]);
    selectionbar();
}

```

Figure 39 - The createtextbox method in the viewrawstock class

```

public static void selectionbar() throws Exception
{

```

The selectionbar method adds the graphical components to the JFrame and calls the createtextbox method with the appropriate input when any of the JButtons are clicked.

```

mc.setBorder(null);

```

As noted earlier, JButtons needed to fake the look of a JMenuBar, and hence their borders were removed, so that they looked like a continuous structure.

```

all.setBounds(0,0,80,25);

```

```

mtan.setBounds(80,0,80,25);

```

They are all placed side by side as demonstrated by the examples above.

```
mc. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            pante.removeAll();
            createtextbox("cotton");
            pante.validate();
            pante.repaint();
        } catch (Exception r) {}
    }
});
```

The panels are refreshed and changed at every click of the JButtons, similar to the main JFrame.

As noted above, from the selectionbar bar method, createtextbox is called, and from the createtextbox class, selectionbar is also called. This was done because whenever the JButtons were clicked and the screen refreshed, the call would go to createtextbox and because the JButtons are actually added to the screen at selectionbar, they would vanish. This was done to counteract that.

```
// Function to read all the details of a materialtype and return an array
public static String[] readrawforview(String rawtype) throws Exception
{
    // Creating an arraylist that is gonna hold the Strings read
    ArrayList<String> templist = new ArrayList<String>();

    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection connl= DriverManager.getConnection(dbUrl,"","");
    Statement stmtl= connl.createStatement();

    String tableName= "RawandDesign";
    ResultSet rs=stmtl.executeQuery("Select * from "+tableName);
    rs= stmtl.getResultSet();

    //flag variable that is gonna check if the saree is still in raw stock or has been sent off for design.
    int flag;

    //temp variable used for storing ids while the entry is being processed.
    Long templ;

    //temp variable used for storing cost while the entry is being processed.
    Double tempd;

    String temps="\t", mflag="";

    //will run until it reaches the last entry of the table
    while((rs!=null) && (rs.next()))
    {
        //Reading the gonefordesign and the materialtype fields of the database for a particular id
        flag = rs.getInt("gonefordesign");
        mflag = rs.getString("materialtype");

        //checking if its still in raw stock
        if(flag==0)
        {
            // A different formatting of the String returned is required if 'all' is chosen as the materialtype
            if(rawtype.equals("all")== true)
            {
                // cutting off chunks from their columns, converting them to strings and adding them to a temporary string variable
                templ = rs.getLong("id");
                temps= temps + templ.toString()+"\t";
                temps= temps + mflag +"\t";
                tempd= rs.getDouble("rawcostprice");
                temps= temps + tempd.toString()+"\t";

                // Adding the temp variable to the ArrayList
                templist.add(temps);

                // Refreshing the temporary variable
                temps="\t";
            }
        }
    }
}
```

```

else
{
    if(mflag.equals(rawtype)== true)
    {
        // cutting off chunks from their columns, converting them to strings and adding them to a string variable
        templ = rs.getLong("id");
        temps= temps+ templ.toString()+"\t";
        tempd= rs.getDouble("rawcostprice");
        temps= temps + tempd.toString()+ "\t";

        // Adding the temp variable to the ArrayList
        templist.add(temps);

        // Refreshing the temporary variable
        temps="\t";
    }
}
}

```

```

//Creating an array the size of the arraylist
arrrt= new String[templist.size()];

//copying the values of the arraylist to an array because its easier to handle
for(int i=0; i<templist.size(); i++)
{
    arrrt[i]= templist.get(i);
}

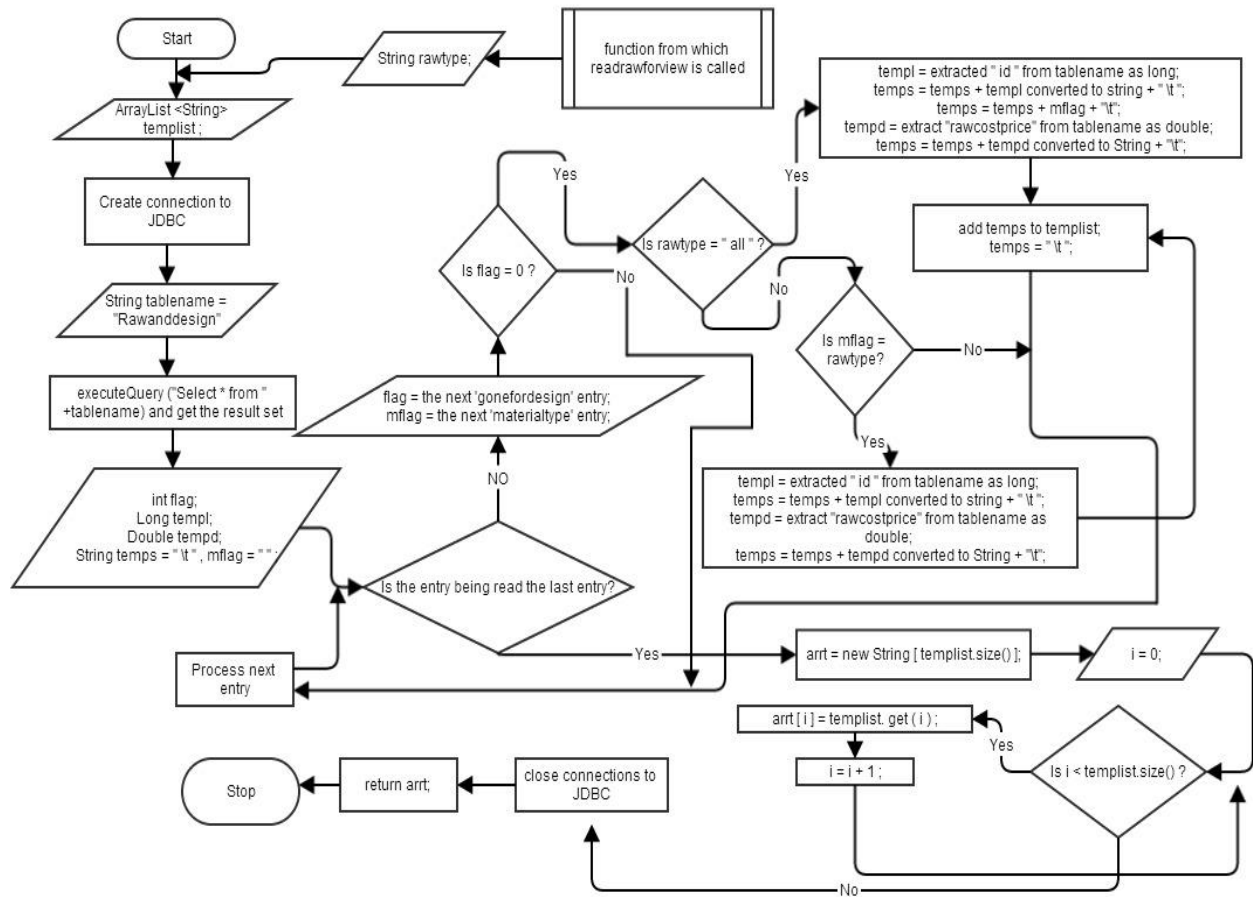
//closing the JDBC connections
stmt1.close();
conn1.close();

//Returning the array created
return arrrt;
}

```

Figure 40 - The readforview method in the read function that is called from the viewrawstock class

The flowchart that demonstrates the logic is shown below.



The effects of sample inputs are demonstrated along with the demonstrations of the other functions.

### Sending design to readymade

```
JLabel addc = new JLabel("enter additional costs");
JLabel idleb= new JLabel("enter id nos. of sarees");
final JTextField addcos= new JTextField("");
final JTextField idsar = new JTextField("");
JButton ver = new JButton("OK");
JButton can = new JButton("Cancel");
```

Figure 41 - Adding graphical components

```
can. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            addcos.setText("");
            idsar.setText("");
        }catch(Exception r){}
    }
});
```

Figure 42 - Refreshing the textfields if cancel is pressed

```
ver. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            long arr[];
            String tep = idsar.getText();
            findid.createarray(tep);
            arr= findid.decipherids(tep);
            int flag=0;
            for(long x: arr)
            {
                //System.out.println(x);
                int er= rr.check(x);
                if( er== 1)
                {
                    String t[]= new String[3];
                    t= rr.readfortableswap(x);
                    radding.idaddi(t[0], t[1], t[2], x);
                    deleting.entrydelete(x);

                    int cury= Integer.parseInt(curyear.format(date));
                    int curm =Integer.parseInt(curmonth.format(date));
                    isadd.search(curm, cury, 'a', Double.parseDouble(addcos.getText()));
                }
                else
                {
                    flag = 1;
                }
            }
            if(flag == 1)
            {
                msgdialog.showMessageDialog(null, "One or more of the ids you have entered does not exist or have not yet been sent off to design", "ERROR", msgdialog.PLAIN_MESSAGE);
            }
        }catch(Exception r){}
    }
});
```

Figure 43 - Deciphering ids accepted, deleting the entries from the RawandDesign table and putting them in the ReadyMaterial table

The id numbers are deciphered the same way as in the Send\_raw\_to\_design panel.

```

public static int check(long r) throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection connl= DriverManager.getConnection(dbUrl,"","");
    Statement stmtl= connl.createStatement();

    String tableName= "RawandDesign";
    ResultSet rs=stmtl.executeQuery("Select id,gonefordesign from "+tableName);
    rs= stmtl.getResultSet();

    int flag, tid, rv = 3;
    while((rs!=null) && (rs.next()))
    {
        flag = rs.getInt("gonefordesign");
        tid = rs.getInt("id");

        // Checking if the id of the entry currently being processed is the id recieved
        if(tid == r)
        {
            // Checking if its in raw stock or if has been passed off
            if (flag == 0)
                rv = 0;
            else
                rv = 1;
        }
    }

    // if the id doesn't exist at all
    if (rv == 3)
        rv = 2;

    return rv;
}

```

Figure 44 - The check method of the read class that checks if an id is valid (if it is present in the RawandDesign table) and if it has been sent off to design.

```

public static String[] readfortablesrap(long r) throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection connl= DriverManager.getConnection(dbUrl,"","");
    Statement stmtl= connl.createStatement();

    String tableName= "RawandDesign";
    ResultSet rs=stmtl.executeQuery("Select * from "+tableName);
    rs= stmtl.getResultSet();
    int tid;

    while((rs!=null) && (rs.next()))
    {
        tid = rs.getInt("id");

        if(tid == r)
        {
            ra[1] = rs.getString("designtype");
            ra[0] = rs.getString("materialtype");
            Double dd = rs.getDouble("rawcostprice")+rs.getDouble("priceofdesign");
            ra[2] = dd.toString();
        }
    }

    return ra;
}

```

Figure 45 - The readfortablesrap method from the read class that extracts the designtype, materialtype and totalcostprice from the RawandDesign table



```

public static String tablename2= "ReadyMaterial";

public static void idaddi(String m, String d, String cp, long z)throws Exception
{
    Double di= Double.parseDouble(cp);
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection conn1= DriverManager.getConnection(dbUrl,"","");
    Statement stmt1= conn1.createStatement();

    String addRow= "INSERT INTO "+tablename2+" (id) VALUES("+ z +")";
    stmt1.execute(addRow);

    String updateRow= "UPDATE "+tablename2+" SET design='"+ d +"' WHERE id="+ z +"";
    stmt1.execute(updateRow);

    String updateRow4= "UPDATE "+tablename2+" SET hassold='"+ 0 +"' WHERE id="+ z +"";
    stmt1.execute(updateRow4);

    String updateRow2= "UPDATE "+tablename2+" SET material='"+ m +"' WHERE id="+ z +"";
    stmt1.execute(updateRow2);

    String updateRow3= "UPDATE "+tablename2+" SET totcostprice='"+ di +"' WHERE id="+ z +"";
    stmt1.execute(updateRow3);

    stmt1.close();
    conn1.close();
}

```

Figure 46 - The idaddi method from the readyadd class that adds the materials to the Readymaterial table

```

public static void entrydelete(long r)throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection conn1= DriverManager.getConnection(dbUrl,"","");
    Statement stmt1= conn1.createStatement();

    String updateRow= "DELETE FROM "+tablename+" WHERE id="+ r +"";
    stmt1.execute(updateRow);

    stmt1.close();
    conn1.close();
}

```

Figure 47 – The entrydelete method from the rawanddesdel class that deletes the materials from the RawandDesign table

### Sample Input

id	materialtype	rawcostprice	gonefordesign	designtype	priceofdesign	nameofdesigner	contactnumberdesigner	detailsofdesigner
1	silk	300	-1	acid paint	200			
2	silk	300	0					
3	silk	300	-1	acid paint	200			
4	silk	300	0					
5	silk	300	0					
*								

Figure 48 - The RawandDesign table before the transaction

Financialchart										RawandDesign										ReadyMaterial									
id		hassold		material		design		additionalcc		totcostprice		estimatedpi		discount		finalsp		finalprofit											
*																													

Figure 49 - The ReadyMaterial table before the transaction

Meera's collection

Raw Design Readymade Sell Information Additional

enter id nos. of sarees

enter additional costs

OK Cancel

Figure 50 - Sample Input

Financialchart		RawandDesign		ReadyMaterial				
id	materialtype	rawcostprice	gonefordesign	designtype	priceofdesign	nameofdesigner	contactnumberdesigner	detailsofdesigner
2	silk	300	0					
4	silk	300	0					
5	silk	300	0					
*								

Figure 51 - RawandDesign table after the transaction

Financialchart										RawandDesign										ReadyMaterial									
id		hassold		material		design		additionalcc		totcostprice		estimatedpi		discount		finalsp		finalprofit											
1		0		silk		acid paint				500																			
3		0		silk		acid paint				500																			
*																													

Figure 52 - Readymaterial table after the transaction

### Adding new readymade material

```
panared.setLayout(null);
String tearry[] = {"Select from the menu", "jute", "cotton", "chanderi", "patola", "silk", "taant", "net"};
String destarry[] = {"Select from the menu", "batik", "acid paint", "embroidery", "aplick", "hand painting"};

final JComboBox selma= new JComboBox(tearry);
JLabel selmalab = new JLabel("Enter material of saree");
final JComboBox dest = new JComboBox(destarry);
JLabel destlab = new JLabel("Enter design of saree");
JLabel cplab = new JLabel("Enter Cost Price");
final JTextField cptext = new JTextField();

JButton can= new JButton("Cancel");
JButton ok= new JButton("Add");
```

Figure 53 - Setting up the graphical components

```
ok. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            String s= (selma.getSelectedItem()).toString();
            String ss= (dest.getSelectedItem()).toString();
            String sc = cptext.getText();
            if(s.equals("Select from the menu") == true)
            {
                msgdialog.showMessageDialog(null, "Material of saree not selected", "ERROR", msgdialog.PLAIN_MESSAGE);
                cptext.setText("");
            }
            else if(ss.equals("Select from the menu") == true)
            {
                msgdialog.showMessageDialog(null, "Design not selected", "ERROR", msgdialog.PLAIN_MESSAGE);
                cptext.setText("");
            }
            else
            {
                long tri = identity.read();
                radding.idaddi(s,ss,sc, tri);
                identity.edit();
                cptext.setText("");
            }
        }catch(Exception r){}
    }
});
```

Figure 54 - Showing error messages in case of wrong input or otherwise adding new entries to the ReadyMaterial class

The unique id numbers are generated the same way as when adding new raw material.

```
can. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            cptext.setText("");
        }catch(Exception r){}
    }
});
```

Figure 55 - Refreshing the JTextAreas after clicking on cancel

### Sample Input

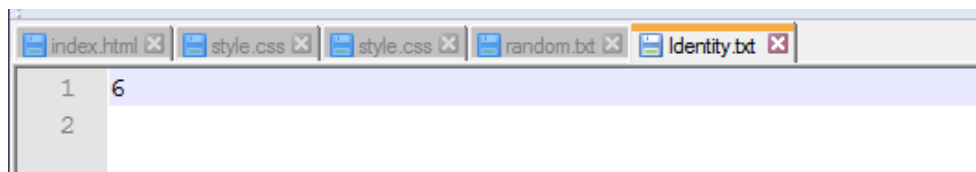


Figure 56 - Identity.txt before the transaction

id	hassold	material	design	additionalcc	totcostprice	estimatedpi	discount	finalsp	finalprofit
1	0	silk	acid paint		500				
3	0	silk	acid paint		500				
*									

Figure 57 - ReadyMaterial table before the transaction

Meera's collection

Raw Design Readymade Sell Information Additional

Enter material of saree

Enter design of saree

Enter Cost Price

Figure 58 - Sample Input

id	hassold	material	design	additionalcc	totcostprice	estimatedpi	discount	finalsp	finalprofit
1	0	silk	acid paint		500				
3	0	silk	acid paint		500				
6	0	patola	acid paint		700				
*									

Figure 59 - ReadyMaterial table after the transaction

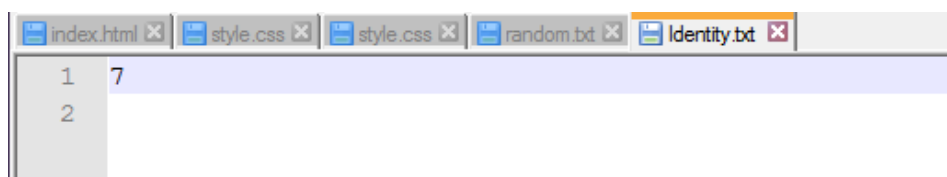


Figure 60 - Identity.txt after the transaction

## Selling items

```
public void panecal()throws Exception
{
    pancal.setLayout(null);

    String stra[] = {"percentage", "cash amount"};

    JLabel idlab = new JLabel("Enter id of saree to be calculated");
    final JTextField idfil = new JTextField();
    JLabel prolable = new JLabel("Enter profit percentage or profit amount");
    final JTextField pro = new JTextField();
    final JComboBox psel = new JComboBox(stra);
    JLabel dislabel = new JLabel("Enter discount percentage");
    final JTextField dis = new JTextField();
    JButton calcu = new JButton("Calculate");
    final JTextField pri = new JTextField();
    JLabel pril = new JLabel("Calculated selling price");
    JButton sell = new JButton("Sell Item");
}
```

Figure 61 - Setting up the graphical components (Source: Appendix – 1 - iii)

```
calcu. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            gui guio = new gui();
            String tx = (psel.getSelectedItem()).toString(); //JComboBox selection
            boolean fl;
            if (tx.equals("percentage") == true)
                fl = true;
            else
                fl=false;

            Double tmdo= Double.parseDouble(pro.getText()); // Converting the String entered as profit into double
            String idt= idfil.getText(); //Getting id no.

            if(idt.equals("") == false)
            {
                guio.temdc = rr.readpc(tmdo, Long.parseLong(idt)); //Sending profit and id no. to readpc in the read class

                if (guio.temdc!= -2.0) //Flag showing that id does exist
                {
                    if(fl == true)
                    {
                        guio.profitc = (tmdo/100)*temdc; // Calculating profit
                    }
                    else
                    {
                        guio.profitc = tmdo; // Calculating profit
                    }

                    guio.disntc = Double.parseDouble(dis.getText());
                    guio.spc = guio.temdc + guio.profitc - ((guio.disntc/100)*(guio.temdc+ guio.profitc));

                    pri.setText(Double.toString(guio.spc));
                }
                else
                {
                    msgdialog.showMessageDialog(null, "The id no. entered does not exist in the list. Please reselect", "ERROR", msgdialog.PLAIN_MESSAGE);
                    pro.setText("");
                    dis.setText("");
                    idfil.setText("");
                }
            }
            else
            {
                msgdialog.showMessageDialog(null, "You haven't entered an id. Please enter id", "ERROR", msgdialog.PLAIN_MESSAGE);
                pro.setText("");
                dis.setText("");
                idfil.setText("");
            }
        }catch(Exception r){}
    }
});
```

Figure 62 - Actions happening when the Calculate button is clicked

The flowchart for this process was given in the design section.

The variables temdc, profitc, disntc, spc are made public and static global variables so that any changes made from gui's objects are reflected on them. This is necessary because changing variables from inside an actionlistener isn't possible.

```
sell.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            gui guio = new gui();

            int cury= Integer.parseInt(curyear.format(date));
            int curm =Integer.parseInt(curmonth.format(date));
            iaadd.search(curm, cury, 'p', profitc);

            radding.selladdi(Long.parseLong(idfil.getText()),guio.profitc,guio.disntc, guio.spc, (guio.spc - guio.temdc));
        }
        catch(Exception r){}
    }
});
```

Figure 63 - Actions performed when the Sell Item button was clicked

```
public static void selladdi(long z, double esp, double dis, double selp, double fp)throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection conn1= DriverManager.getConnection(dbUrl,"","");
    Statement stmt1= conn1.createStatement();

    String updateRow= "UPDATE "+tablename2+" SET hassold='"+ 0 +"' WHERE id="+ z +"";
    stmt1.execute(updateRow);

    String updateRow2= "UPDATE "+tablename2+" SET estimatedprofit='"+ esp +"' WHERE id="+ z +"";
    stmt1.execute(updateRow2);

    String updateRow3= "UPDATE "+tablename2+" SET discount='"+ dis +"' WHERE id="+ z +"";
    stmt1.execute(updateRow3);

    String updateRow4= "UPDATE "+tablename2+" SET finalsp='"+ selp +"' WHERE id="+ z +"";
    stmt1.execute(updateRow4);

    String updateRow5= "UPDATE "+tablename2+" SET finalprofit='"+ fp + "' WHERE id="+ z +"";
    stmt1.execute(updateRow5);

    String updateRow6= "UPDATE "+tablename2+" SET hassold='"+ 1 + "' WHERE id="+ z +"";
    stmt1.execute(updateRow6);

    stmt1.close();
    conn1.close();
}
```

Figure 64 - The selladdi method from the readyadd class that is called

*Sample Input*

id	hassold	material	design	additionalcosts	totcostprice	estimatedprofit	discount	finalsp	finalprofit
1		0 silk	acid paint		500				
3		0 silk	acid paint		500				
6		0 patola	acid paint		700				
*									

Figure 65 - ReadyMaterial table before the transaction

month	year	investmentonraw	investmentondesign	additionalinvestment	profit
3		1500	400		
*					

Figure 66 - Financialchart table before the transaction

Meera's collection

Raw Design Readymade Sell Information Additional

Enter id of saree to be calculated

Enter profit percentage or profit amount percentage ▼

Enter discount percentage

Calculated selling price

Figure 67 - Sample Input

Meera's collection

Raw Design Readymade Sell Information Additional

Enter id of saree to be calculated

Enter profit percentage or profit amount percentage ▼

Enter discount percentage

Calculated selling price

Figure 68 - Output after pressing the Calculate button

id	hassold	material	design	additionalcosts	totcostprice	estimatedprofit	discount	finalsp	finalprofit
1	0	silk	acid paint		500				
3	0	silk	acid paint		500				
6	-1	patola	acid paint		700	140	5	798	98
*									

Figure 69 - Changes in the ReadyMaterial table after Sell Item button was clicked

month	year	investmentonraw	investmentondesign	additionalinvestment	profit
3		1500	1000		140
*					

Figure 70 - Changes in the Financialchart table after Sell Item was clicked



### Calculating monthly investment

```
public void panemin()throws Exception
{
    panmin.setLayout(null);

    String mnthlist[] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};
    String yrlist[] = ry.years();

    JLabel mnthlab = new JLabel("Select Month");
    JLabel yrlab = new JLabel("Select Year");

    JLabel ri = new JLabel("Investment on Raw materials");
    JLabel di = new JLabel ("Investment on design");
    JLabel ai = new JLabel ("Additional investment");
    JLabel ti = new JLabel ("Total investment");

    final JComboBox mnth = new JComboBox(mnthlist);
    final JComboBox yr = new JComboBox(yrlist);

    JButton si = new JButton ("Show Investment");

    final JTextField rit = new JTextField("");
    final JTextField dit = new JTextField("");
    final JTextField ait = new JTextField("");
    final JTextField tit = new JTextField("");
}
```

Figure 71 - Adding the graphical components

```
public static String[] years()throws Exception
{
    // Creating an arraylist that will hold the values of
    ArrayList<Integer> templist = new ArrayList<Integer>();

    String yearret[];

    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection connl= DriverManager.getConnection(dbUrl,"","");
    Statement stmtl= connl.createStatement();

    String tableName= "Financialchart";
    ResultSet rs=stmtl.executeQuery("Select year from "+tableName);
    rs= stmtl.getResultSet();

    int ty;
    while((rs!=null) && (rs.next()))
    {
        ty = rs.getInt("year");

        // Checking if the id of the entry currently being processed is the id recieved
        boolean flag = true;
        for(int x: templist)
        {
            if(x == ty)
            {
                flag = false;
            }
        }
        if(flag == true)
        {
            templist.add(ty);
        }
    }
}
```

```

yearret = new String[templist.size()];

for(int i=0; i<templist.size(); i++)
{
    yearret[i]= templist.get(i).toString();
}

return yearret;
}

```

Figure 72 - The years methods from the readyyears class that checks the years whose entries are present in the Financialchart table for the JComboBox

```

si. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            double tdarr[] = {0,0,0,0};
            String tm = mnth.getSelectedItem().toString();
            int tmi = 0;
            if(tm.equals("January")== true)
                tmi = 1;
            else if (tm.equals("February")== true)
                tmi = 2;
            else if (tm.equals("March")== true)
                tmi = 3;
            else if (tm.equals("April")== true)
                tmi = 4;
            else if (tm.equals("May")== true)
                tmi = 5;
            else if (tm.equals("June")== true)
                tmi = 6;
            else if (tm.equals("July")== true)
                tmi = 7;
            else if (tm.equals("August")== true)
                tmi = 8;
            else if (tm.equals("September")== true)
                tmi = 9;
            else if (tm.equals("October")== true)
                tmi = 10;
            else if (tm.equals("November")== true)
                tmi = 11;
            else if (tm.equals("December")== true)
                tmi = 12;

            tdarr = rr.readmnti(tmi,Integer.parseInt(yr.getSelectedItem().toString()));
            rit.setText(Double.toString(tdarr[0]));
            //System.out.println(tdarr[0]);
            dit.setText(Double.toString(tdarr[1]));
            ait.setText(Double.toString(tdarr[2]));
            tit.setText(Double.toString(tdarr[3]));
        }catch(Exception r){}
    }
}

```

Figure 73 - Converting the months into integers and displaying the investments when the Show Investment button is clicked

```

public static double[] readmti(int m, int y) throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection conn1= DriverManager.getConnection(dbUrl,"","");
    Statement stmt1= conn1.createStatement();

    String tableName= "Financialchart";
    ResultSet rs=stmt1.executeQuery("Select * from "+tableName);
    rs= stmt1.getResultSet();

    int mth, year;
    double rin, din, ain;
    double arr[] = new double[4];
    while((rs!=null) && (rs.next()))
    {
        mth = rs.getInt("month");
        year = rs.getInt("year");
        rin = rs.getDouble("investmentonraw");
        din = rs.getDouble("investmentondesign");
        ain = rs.getDouble("additionalinvestment");

        if((year == y) && (mth == m))
        {
            arr[0] = rin;
            arr[1] = din;
            arr[2] = ain;
            arr[3] = rin + din + ain;
        }
    }

    return arr;
}

```

Figure 74 - The readmti method from the read class that returns the investmentonraw, investmentondesign and additionalinvestment from the Financialchart table

### Sample Input

Financialchart	RawandDesign	ReadyMaterial				
month	year	investmentonraw	investmentondesign	additionalinvestment	profit	
3		1500	400			
*						

Figure 75- The Financialchart table before the transaction (Source : Appendix -1 - iv)

Meera's collection

Raw Design Readymade Sell Information Additional

Select Month

Select Year

Investment on Raw materials

Investment on design

Additional investment

Total investment

Figure 76 - Sample Input

Meera's collection

Raw Design Readymade Sell Information Additional

Select Month

Select Year

Investment on Raw materials

Investment on design

Additional investment

Total investment

Figure 77 - Output for the given input

### Calculating yearly investment

```
public void paneyin() throws Exception
{
    panyin.setLayout(null);

    String yrlist[] = ry.years();

    JLabel yrlab = new JLabel("Select Year");

    JLabel ri = new JLabel("Investment on Raw materials");
    JLabel di = new JLabel ("Investment on design");
    JLabel ai = new JLabel ("Additional investment");
    JLabel ti = new JLabel ("Total investment");

    final JComboBox yr = new JComboBox(yrlist);

    JButton si = new JButton ("Show Investment");

    final JTextField rit = new JTextField("");
    final JTextField dit = new JTextField("");
    final JTextField ait = new JTextField("");
    final JTextField tit = new JTextField("");
}
```

Figure 78 - Setting up the graphical components

```
si.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            double tdarr[];
            tdarr = rr.readYr(Integer.parseInt(yr.getSelectedItem().toString()));
            rit.setText(Double.toString(tdarr[0]));
            dit.setText(Double.toString(tdarr[1]));
            ait.setText(Double.toString(tdarr[2]));
            tit.setText(Double.toString(tdarr[3]));
        } catch (Exception r) {}
    }
});
```

Figure 79 - Displaying the investments

```

public static double[] readyri(int y) throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection conn1= DriverManager.getConnection(dbUrl,"","");
    Statement stmt1= conn1.createStatement();

    String tableName= "Financialchart";
    ResultSet rs=stmt1.executeQuery("Select * from "+tableName);
    rs= stmt1.getResultSet();

    int year;
    double rin, din, ain;
    double arr[] = {0, 0, 0, 0};

    while((rs!=null) && (rs.next()))
    {
        year = rs.getInt("year");
        rin = rs.getDouble("investmentonraw");
        din = rs.getDouble("investmentondesign");
        ain = rs.getDouble("additionalinvestment");

        if((year == y))
        {
            arr[0] = arr[0] + rin;
            arr[1] = arr[1] + din;
            arr[2] = arr[2] + ain;
            arr[3] = arr[1] + arr[2] + arr[3];
        }
    }

    return arr;
}

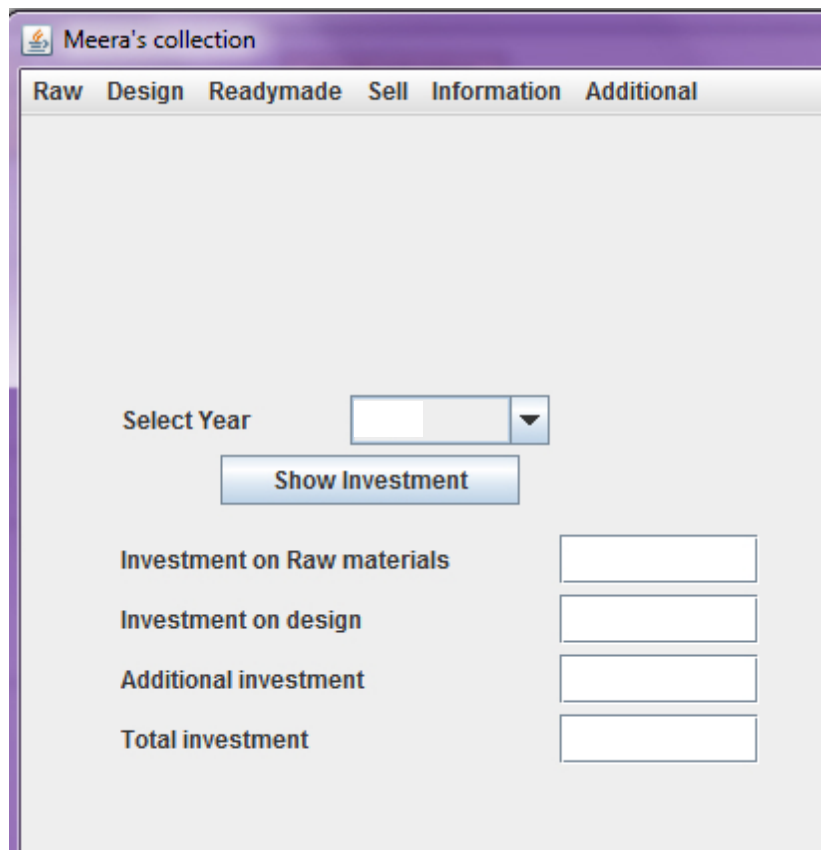
```

Figure 80 - The readyri function from the read class that reads the investmentonraw, investmentondesign and additionalinvestment

### Sample Input

Financialchart	RawandDesign	ReadyMaterial			
month	year	investmentonraw	investmentondesign	additionalinvestment	profit
	3	1500	400		
*					

Figure 56- The Financialchart table before the transaction



Meera's collection

Raw Design Readymade Sell Information Additional

Select Year

Show Investment

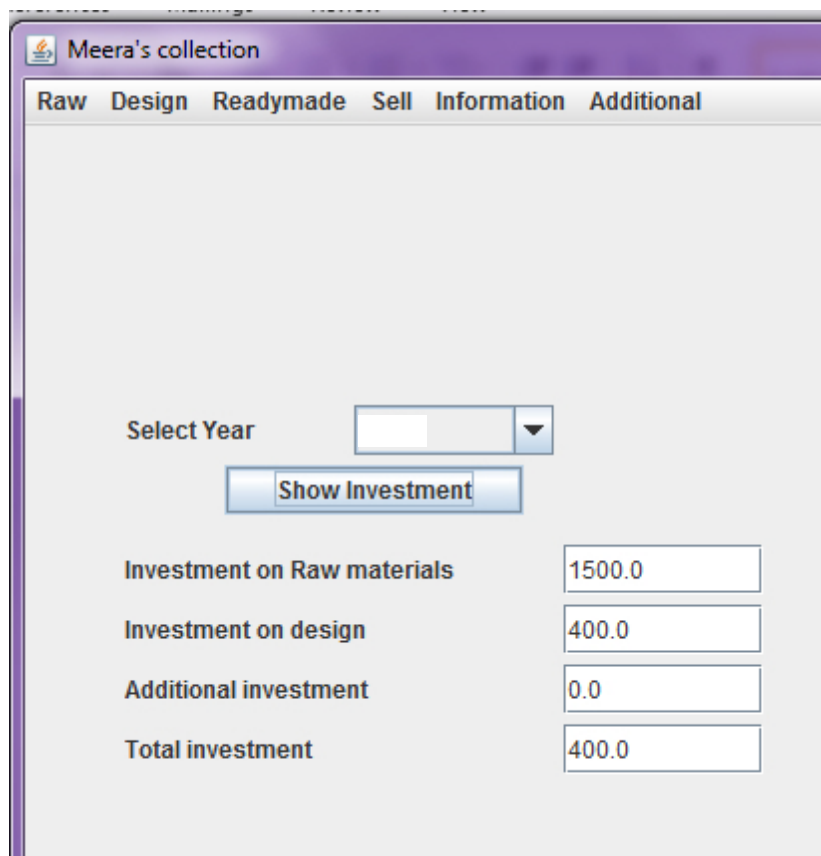
Investment on Raw materials

Investment on design

Additional investment

Total investment

Figure 81 - Sample Input



Meera's collection

Raw Design Readymade Sell Information Additional

Select Year

Show Investment

Investment on Raw materials

Investment on design

Additional investment

Total investment

Figure 82 - Output to the given Input

### Calculating monthly profit

```
public void panempro() throws Exception
{
    panmpro.setLayout(null);

    String mnthlist[] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "Octobor", "November", "December"};
    String yrlist[] = ry.years();

    JLabel mnthlab = new JLabel("Select Month");
    JLabel yrلاب = new JLabel("Select Year");

    JLabel p = new JLabel("Total profit");

    final JComboBox mnth = new JComboBox(mnthlist);
    final JComboBox yr = new JComboBox(yrlist);

    JButton si = new JButton ("Show Profit");

    final JTextField pt = new JTextField("");
```

Figure 83 - Adding the Graphical components

```
si. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            double td;
            String tm = mnth.getSelectedItem().toString();
            int tmi = 0;

            if(tm.equals("January")== true)
                tmi = 1;
            else if (tm.equals("February")== true)
                tmi = 2;
            else if (tm.equals("March")== true)
                tmi = 3;
            else if (tm.equals("April")== true)
                tmi = 4;
            else if (tm.equals("May")== true)
                tmi = 5;
            else if (tm.equals("June")== true)
                tmi = 6;
            else if (tm.equals("July")== true)
                tmi = 7;
            else if (tm.equals("August")== true)
                tmi = 8;
            else if (tm.equals("September")== true)
                tmi = 9;
            else if (tm.equals("Octobor")== true)
                tmi = 10;
            else if (tm.equals("November")== true)
                tmi = 11;
            else if (tm.equals("December")== true)
                tmi = 12;

            td = rr.readmpro(tmi,Integer.parseInt(yr.getSelectedItem().toString()));
            System.out.println(td);
            pt.setText(Double.toString(td));
        }catch(Exception r){}
    }
});
```

Figure 84 - Displaying the profit on screen



```

public static double readmpro(int m, int y) throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection conn1= DriverManager.getConnection(dbUrl,"","");
    Statement stmt1= conn1.createStatement();

    String tableName= "Financialchart";
    ResultSet rs=stmt1.executeQuery("Select * from "+tableName);
    rs= stmt1.getResultSet();

    int mth, year;
    double pro = 0, rpro = 0;
    while((rs!=null) && (rs.next()))
    {
        mth = rs.getInt("month");
        year = rs.getInt("year");
        pro = rs.getDouble("profit");

        if((year == y) && (mth == m))
        {
            rpro = pro;
        }
    }

    return rpro;
}

```

Figure 85 - The readmpro function from the read class that reads the profit from the Financialchart table

### Calculate yearly profit

```

public void paneypro() throws Exception
{
    panypro.setLayout(null);

    String yrlist[] = ry.years();

    JLabel yrlab = new JLabel("Select Year");

    JLabel p = new JLabel("Total profit");

    final JComboBox yr = new JComboBox(yrlist);

    JButton si = new JButton ("Show Profit");

    final JTextField pt = new JTextField("");
}

```

Figure 86 - Creating Graphical Components

```

si. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            double td;
            td = rr.readypro(Integer.parseInt(yr.getSelectedItem().toString()));
            pt.setText(Double.toString(td));
        } catch (Exception r) {}
    }
});

```

Figure 87 - Displaying the Profit

```

public static double readypro(int y) throws Exception
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String dbUrl="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=myDB.mdb";
    Connection conn1= DriverManager.getConnection(dbUrl,"","");
    Statement stmt1= conn1.createStatement();

    String tableName= "Financialchart";
    ResultSet rs=stmt1.executeQuery("Select * from "+tableName);
    rs= stmt1.getResultSet();

    int year;
    double pr = 0, tp;

    while((rs!=null) && (rs.next()))
    {
        year = rs.getInt("year");
        tp = rs.getDouble("profit");

        if((year == y))
        {
            pr = pr + tp;
        }
    }

    return pr;
}

```

Figure 88 - The readypro method from the read class that reads the yearly profit from the Financialchart table

### Entering additional Investment

```

public void paneaddin() throws Exception
{
    panaddin.setLayout(null);

    JTextField am= new JTextField();
    final JLabel aml= new JLabel("Amount of Investment");

    JTextField sr= new JTextField();
    JLabel srl= new JLabel("Source");

    JButton okie = new JButton("Okay");

    JTextArea addin= new JTextArea();
    JLabel addinl= new JLabel("Additional information");

    JScrollPane jsp= new JScrollPane(addin);

```

Figure 89 - Adding the Graphical Components (Source: Appendix – 1- iii)

```

okie. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            int cury= Integer.parseInt(curyear.format(date));
            int curm =Integer.parseInt(curmonth.format(date));
            iaadd.search(curm, cury, 'p', Double.parseDouble(aml.getText()));
        }catch(Exception r){}
    }
});

```

Figure 90 - Adding the additional Investment to the Financialchart table

### Selecting id numbers from stock list

A number of things needed to be kept in mind while designing the stockselection list, and yet at the end, it did not seem to work properly, for reasons I couldn't understand.

The first problem was to decide how to generate a dynamic list system that allows for multiple selections. I had decided to create a linked list and add JCheckBox-es for selection. However, extracting only the identity numbers from the selection became too much of a hassle.

JList seemed to be a pretty good alternative at the early phases when I wasn't aware of the problems. It allowed for multiple selections intrinsically, and extracting strings from selected components were easier.

```

stockselection

static JFrame selectstockfr= new JFrame("Available Raw Stock");
static JPanel pante= new JPanel();
static DefaultListModel model= new DefaultListModel();
static JList stocklist= new JList(model);
static JTextField fed = new JTextField("");

static read readr= new read();
static String rets, retval;

static JButton ok = new JButton("Okay");
static JButton mc = new JButton("Cotton");
static JButton mtan = new JButton("Taant");
static JButton mt = new JButton("Tashor");
static JButton ms = new JButton("Silk");
static JButton mn = new JButton("Net");
static JButton mj = new JButton("Jute");
static JButton mp = new JButton("Patola");
static JButton mch = new JButton("Chanderi");
static JButton all = new JButton ("All");

static void displaypane()throws Exception

{
    tk= Toolkit. getDefaultToolkit();
    Dimension dim= tk. getScreenSize();
    selectstockfr.setSize(735,600); //resolution of the window- 655*600
    selectstockfr.setLayout(null);

    //unique - centering the window
    selectstockfr.setLocation(((dim.width- selectstockfr.getWidth())/2)+40, (dim.height- selectstockfr.getHeight())/2);
    selectstockfr.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    selectstockfr.add(pante);

    selectstockfr.setVisible(true);
}

```

Figure 91 - Global declarations and displaying the JFrame

```

public static void createlist(String mtype)throws Exception

{
    String darr[];
    model.removeAllElements();
    stocklist.setBounds(0,68, selectstockfr.getWidth(), (selectstockfr.getHeight()- 30));
    ok.setBounds(selectstockfr.getWidth() - 100,30, 70, 25);
    fed.setBounds(10,30, 400, 25);
    pante.add(fed);
    stocklist.setVisible(true);
    pante.add(ok);
    pante.add(stocklist);

    darr = readr.readrawforview(mtype);
}

```

Figure 92 - Adding graphical components on the screen from the createlist method in the stockselection class

```

int i=0;
for(i=0; i< darr.length; i++)
{
    int j=0, count=0;
    char c;
    String tstr="";
    while( j < darr[i].length())
    {
        c= darr[i].charAt(j);
        if(c=='\t')
        {
            for (int x=0; x<=(25-count); x++)
            {
                tstr+=" ";
            }
            count=0;
        }
        else
        {
            count++;
            tstr+=c;
        }
        j++;
    }
    //System.out.println(tstr);
    darr[i]=tstr;
    model.addElement(darr[i]+"\\n");
}

```

```

stocklist.addListSelectionListener(new ListSelectionListener(){
    public void valueChanged( ListSelectionEvent e)
    {
        try {
            Object obj[]= stocklist.getSelectedValues();
            int a=0;
            String tm;
            rets = "";
            for(int i=0; i<obj.length; i++)
            {
                tm=(String) obj[i];
                char ch=' ';
                int flag = 0;
                for(int j=0; j< tm.length(); j++)
                {
                    ch=tm.charAt(j);
                    if(ch!='_')
                    {
                        flag = 1;
                        rets+=ch;
                    }
                    else if(ch == '_')
                    {
                        if (flag == 1)
                        {
                            rets+= ',';
                            break;
                        }
                    }
                }
            }
        }catch(Exception r){}
    }
}

```

```

ok. addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        try {
            fed.setText(rets);
            //selectstockfr.dispose();
        }catch(Exception r){}
    }
});

```

```

selectionbar();
}

```

The different portions of information are separated with ‘\_’ because I found out the JLists didn’t allow for spaces in between a string component, so I had to go for alternatives. Moreover, another disadvantage came from the fact that the columns couldn’t have heads because the heading in itself would be a selectable object. I decided to not take it in the final string if it was selected, but more problems came up. First of all, the string being built had to be imported by the class calling the function. Which was proving impossible from inside an actionlistener. So I used a textfield to show the string there and then indirectly extract it.

It seemed to be working, except that the function returned the string before the button was pressed and returned nothing when it was. I tried numerous methods, but nothing seemed to work.

Word Count : 1068 word (not including headings, captions, screenshots and flowcharts)