Java Projects

Submitted in partial fulfilment of the requirements for the degree of

Post Graduate Diploma in Information Technology

by

Vijayananda D Mohire

(Registration No.200508208)

Information Technology Department
Symbiosis Bhavan,
1065 B, Gokhale Cross
Road, Model Colony, Pune – 411016,
Maharashtra, India
(2007)

Java Projects

Table of Contents

Java Projects

Question 1

Create a class Computer that stores information about different types of Computers available with the dealer. The information to be stored about a single computer is,

- Company Name
- RAM size.
- Hard Disk Capacity.
- Processor Speed.
- Processor Make.
- Price .
- Quantity of the Computers.

This class contains following methods,

- Constructor method that assigns user input values to above mentioned variables.

- main ( ) method that creates array of 4 objects of Computer class and that takes input for all above details from the user and calls method to check validity of Quantity and Price values. If all values are valid then create the objects and display the details for all 4 toys.

- Method that checks validity of quantity and price. If the quantity is 0 or negative then method should throw user defined exception with appropriate message and come out of program. Similarly if the price is 0 or negative then it should throw user defined exception with appropriate message and come out of program.

- Method that displays all details about a single instance of Computer in the following Format.

Computer Name : IBM
RAM Size : 512 MB
Processor Make : Intel
Processor speed : 300
Quantity : 50
Total Price of Computers : 150,0000/-

Answer 1
Code:

----------Computer.java----------

```
package newcomputer;

import java.awt.Frame;
import javax.swing.BorderFactory;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import newcomputer.LabelledItemPanel;
import newcomputer.CustomerDialog;
import newcomputer.CustomerData;

/**
 * Define Computer class
 * @author Vijayananda D Mohire
 */
public class Computer  {

    /** Class Variables to hold the Computer Name
     */
    String Company_Name;
     /** Class Variables to hold the RAM Size
     */
    String RAM_size;
     /** Class Variables to hold the Hard Disk Capacity
     */
    String HardDisk_Capacity;
     /** Class Variables to hold the processor speed
     */
    String Processor_Speed;
     /** Class Variables to hold the Processor Make
     */
    String Processor_Make;
     /** Class Variables to hold the price
```

```
     */
    String Price;
     /** Class Variables to hold the Quantity of Computers
     */
    String Quantity_of_the_Computers;


    /** Creates a new instance of Computer */
    public Computer() {
    }

/**
     * Overloaded Constructor,inputs the Dialog data
     * @param ComputerData Holds Computer Data
     */

    public Computer(CustomerData ComputerData) {

       Company_Name = ComputerData.myCompanyName;
       RAM_size = ComputerData.myRamSize ;
       HardDisk_Capacity =ComputerData.myHardDiskCap;
        Processor_Speed =ComputerData.myProcessorSpeed;
        Processor_Make =ComputerData.myprocessorMake;
        Price = ComputerData.myPrice;
        Quantity_of_the_Computers =ComputerData.myQty;

    }
     /**
     * Main
     * @param args Input args
     */
     public static void main(String[] args)
    {

  Computer [] Comps = new Computer[3]; // array to hold data after validation
  int i=0;

  CustomerDialog dialog4 = new CustomerDialog();

// Obtain the required 4 types of Computer details, use of do while loop
```

```
 do
      {
      CustomerDialog dialog = new CustomerDialog();

      dialog.pack();
      // Present it to a User1
            dialog.show();

      // TODO : Check why Dialog is not coming out when cancel is clicked
      // Get the data or quit if user Cancel
      if(!dialog.hasUserCancelled())
      {
      CustomerData customerData = dialog.getCustomerData();

      try{
      // Validate for each Computer instance entry and show error message if any.

         boolean ValidFlag = dialog.isValidData();

if(ValidFlag)
 {

//Pass on the Dialog data to Constructor input, create Computer class for that
Computer instance

    Comps[i] = new Computer(customerData);
    i++;

 }

else {

Comps[0] = null;
Comps[1] = null;
Comps[2]= null;

System.exit(0);
}
      }
       catch(Exception err)
      {
```

```
    dialog.ShowErr(err);
       }
      }
       if(dialog.hasUserCancelled())
        break;

       }

  while(i < 3);

 if(i>=3)
  PrintData(Comps, dialog4);

//Release object memory for garbage collection.

Comps[0] = null;
Comps[1] = null;
Comps[2]= null;

System.exit(0);
   }


    /**
     * Prints the Computer details
     * @param Comps Array of Computers
     * @param dialog3 Pass the class variable
     */
   public static void PrintData(Computer [] Comps,CustomerDialog dialog3 )
   {

       dialog3.PrintData(Comps);

 }
}

------------------------------CustomerData.java------------------------------
package newcomputer;
/**
 * This class is a simple data structure for holding Customer data.
 * @author Vijayananda D Mohire
```

```
 */

public class CustomerData
{
  /** Class Variables to hold the Computer Name
    */
   public String myCompanyName;
    /** Class Variables to hold the RAM Size
    */
   public String myRamSize;
     /** Class Variables to hold the HArd Disk Capacity
    */
   public String myHardDiskCap;
     /** Class Variables to hold the processor speed
    */
   public String myProcessorSpeed;
   /** Class Variables to hold the Processor Make
    */
   public String myprocessorMake;
     /** Class Variables to hold the price
    */
   public String myPrice;
    /** Class Variables to hold the Quantity of Computers
    */
    public String myQty;

 }

-------------------------------CustomerDialog.java---------------------------------
package newcomputer;

/**
 *
 * @author Vijayananda.Mohire
 */
import java.awt.Frame;
import javax.swing.BorderFactory;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
```

```
import javax.swing.JTextArea;
import javax.swing.JTextField;
import newcomputer.LabelledItemPanel;
import javax.swing.*;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

/**
 * This class demonstrates the usage of the StandardDialog class.
 */
public class CustomerDialog extends StandardDialog {
// Constants

   /** Text Area Variables to hold the Company Name
    */
   private JTextArea myCompanyName = new JTextArea(3, 20);
    /** Text Field Variables to hold the RAMSize
    */
    private JTextField myRAMSize = new JTextField();
     /** Text Field Variables to hold the DiskCap acity
    */
   private JTextField myHardDiskCap = new JTextField();
     /** Text Field Variables to hold the ProcessorSpeed
    */
   private JTextField myProcessorSpeed = new JTextField();
   /** Text Field Variables to hold the ProcessorMake
    */
   private JTextField myProcessorMake = new JTextField();
   /** Text Field Variables to hold the Price
    */
   private JTextField myPrice = new JTextField();
   /** Text Field Variables to hold the Quantity
    */
   private JTextField myQuantity = new JTextField();
    /** LabelledItemPanel  Variables to hold the labels
    */
   private LabelledItemPanel myContentPane = new LabelledItemPanel();

    // Methods
```

```
    /**
     * This method is the default constructor.
     */
    public CustomerDialog() {
        init();
    }

    /**
     * This method initialises the components on the panel.
     */
    private void init() {
        setTitle("Computers available at Dealer Vijay ");

        myContentPane.setBorder(BorderFactory.createEtchedBorder());

            myContentPane.addItem("Company Name", new
JScrollPane(myCompanyName,
                JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
                JScrollPane.HORIZONTAL_SCROLLBAR_NEVER));

        myContentPane.addItem("RAM Size", myRAMSize);
        myContentPane.addItem("Hard Disk Capacity", myHardDiskCap);
        myContentPane.addItem("Processor Speed", myProcessorSpeed);
        myContentPane.addItem("Processor Make", myProcessorMake);
        myContentPane.addItem("Price", myPrice);
        myContentPane.addItem("Quantity", myQuantity);

setContentPane(myContentPane);
    }

    /**
     * This method gets the values of the panel entry fields.
     * @returns an object containing the Customer data
     * @return Returns Customer data
     */
    public CustomerData getCustomerData() {
        CustomerData customerData = new CustomerData();

       customerData.myCompanyName = this.myCompanyName.getText();
       customerData.myRamSize = myRAMSize.getText();
       customerData.myHardDiskCap = myHardDiskCap.getText();
```

```
    customerData.myProcessorSpeed = myProcessorSpeed.getText();
    customerData.myprocessorMake = myProcessorMake.getText();
    customerData.myPrice = myPrice.getText();
     customerData.myQty = myQuantity.getText();

    return customerData;
}

/**
 * This method sets the values of the panel entry fields.
 *
 * @param customerData The object containing the Customer data
 */
public void setCustomerData(CustomerData customerData) {
 this.myCompanyName.setText(customerData.myCompanyName);
  myRAMSize.setText(customerData.myRamSize);
    myHardDiskCap.setText(customerData.myHardDiskCap);
    myProcessorSpeed.setText(customerData.myProcessorSpeed);
    myProcessorMake.setText(customerData.myprocessorMake);
      myPrice.setText(customerData.myPrice);
       myQuantity.setText(customerData.myQty);

}

/**
 * This method checks that the data entered is valid.
 * To be valid, the following must be met:
 * <LI>Customer Code field is not blank
 * <LI>Name field is not blank
 * @return <code>true</code> if the data is valid, otherwise
 * <code>false</code>
 */
protected boolean isValidData() {

    int len =0;
   char ch ;

    if(myPrice.getText().equals(""))

    {
          JOptionPane.showMessageDialog(this,
```

```
            "Please enter Price",
             "Blank Price",
             JOptionPane.WARNING_MESSAGE);
              myRAMSize.requestFocus();
              return false;
     }

    len =   myPrice.getText().length();

    if(len == 0)
    {
       JOptionPane.showMessageDialog(this,
              "Price cannot be null",
              "Null price",
              JOptionPane.WARNING_MESSAGE);

        myRAMSize.requestFocus();
       return false;
    }

ch =  myPrice.getText().charAt(0);

 if(ch == '0')
    {
       JOptionPane.showMessageDialog(this,
              "Price cannot be zero",
              "Zero Price",
              JOptionPane.WARNING_MESSAGE);

        myRAMSize.requestFocus();
       return false;
    }

if(ch == '-')
    {
       JOptionPane.showMessageDialog(this,
              "Price cannot be negative",
              "Negative Price",
              JOptionPane.WARNING_MESSAGE);

        myRAMSize.requestFocus();
```

```
    return false;
}

 if(myQuantity.getText().equals("")) {
    JOptionPane.showMessageDialog(this,
            "Please enter Quantity",
            "Blank Quantity",
            JOptionPane.WARNING_MESSAGE);

    myQuantity.requestFocus();

    return false;
 }

len =   myQuantity.getText().length();
if(len == 0)
{
   JOptionPane.showMessageDialog(this,
            "Quantity cannot be empty",
            "Null Qty",
            JOptionPane.WARNING_MESSAGE);

    myRAMSize.requestFocus();
   return false;
}

ch =   myQuantity.getText().charAt(0);

 if(ch == '0')
{
   JOptionPane.showMessageDialog(this,
            "Quantity cannot be zero",
            "Zero Qty",
            JOptionPane.WARNING_MESSAGE);

    myRAMSize.requestFocus();
   return false;
}

if(ch == '-')
{
```

```
            JOptionPane.showMessageDialog(this,
                    "Quantity cannot be negative",
                    "Negative Qty",
                    JOptionPane.WARNING_MESSAGE);

             myRAMSize.requestFocus();
            return false;
          }
          return true;
        }

         /**
         * This method Prints each computer details.
         *
         * @param Comps The object containing the Customer data
         */
         protected void PrintData(Computer [] Comps) {
            for( int i=0;i< Comps.length;++i) // check for Upperbound
{
         StringBuilder br =  new StringBuilder();
        br.append("\n Company Name :");
        br.append(Comps[i].Company_Name);

        br.append("\n RAM :");
        br.append(Comps[i].RAM_size);

         br.append(" \n HDD:");
          br.append(Comps[i].HardDisk_Capacity);

          br.append("\n Speed :");
          br.append(Comps[i].Processor_Speed);

           br.append("\n Make:");
            br.append(Comps[i].Processor_Make);

          br.append("\n Price: ");
           br.append(Comps[i].Price);

             br.append("\n Qty :");
              br.append(Comps[i].Quantity_of_the_Computers);
```

```
        JOptionPane.showMessageDialog(this,br.toString(),"Computer Output",
            JOptionPane.INFORMATION_MESSAGE);
         }
    }
      /**
     * This method shows errors.
     *
     * @param err The variable containing the error details
     */
     public void ShowErr(Exception err)

     {
     JOptionPane.showMessageDialog(this,
                 err,
                 "System Error",
                 JOptionPane.WARNING_MESSAGE);
     }
}
```

----------LabelledItemPanel.java----------

```
package newcomputer;

/**
 *
 * @author Vijayananda.Mohire
 */
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;

import javax.swing.BorderFactory;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JPanel;

/**
 * This class provides a panel for laying out labelled elements neatly with
 * all the labels and elements aligned down the screen.
 *
 * @author Vijayananda.Mohire
 */
```

```
public class LabelledItemPanel extends JPanel
{
   /** The row to add the next labelled item to */
   private int myNextItemRow = 0;

   /**
    * This method is the default constructor.
    */
   public LabelledItemPanel()
   {
      init();
   }

   /**
    * This method initialises the panel and layout manager.
    */
   private void init()
   {
      setLayout(new GridBagLayout());

      // Create a blank label to use as a vertical fill so that the
      // label/item pairs are aligned to the top of the panel and are not
      // grouped in the centre if the parent component is taller than
      // the preferred size of the panel.

      GridBagConstraints constraints = new GridBagConstraints();
      constraints.gridx   = 0;
      constraints.gridy   = 99;
      constraints.insets  = new Insets(10, 0, 0, 0);
      constraints.weighty = 1.0;
      constraints.fill    = GridBagConstraints.VERTICAL;

      JLabel verticalFillLabel = new JLabel();

      add(verticalFillLabel, constraints);
   }

   /**
    * This method adds a labelled item to the panel. The item is added to
    * the row below the last item added.
    *
```

```
     * @param labelText The label text for the item.
     * @param item      The item to be added.
     */
    public void addItem(String labelText, JComponent item)
    {
        // Create the label and its constraints

        JLabel label = new JLabel(labelText);

        GridBagConstraints labelConstraints = new GridBagConstraints();

        labelConstraints.gridx   = 0;
        labelConstraints.gridy   = myNextItemRow;
        labelConstraints.insets  = new Insets(10, 10, 0, 0);
        labelConstraints.anchor  = GridBagConstraints.NORTHEAST;
        labelConstraints.fill    = GridBagConstraints.NONE;

        add(label, labelConstraints);

        // Add the component with its constraints

        GridBagConstraints itemConstraints = new GridBagConstraints();

        itemConstraints.gridx   = 1;
        itemConstraints.gridy   = myNextItemRow;
        itemConstraints.insets  = new Insets(10, 10, 0, 10);
        itemConstraints.weightx = 1.0;
        itemConstraints.anchor  = GridBagConstraints.WEST;
        itemConstraints.fill    = GridBagConstraints.HORIZONTAL;

        add(item, itemConstraints);

        myNextItemRow++;
    }
}

------------------------------StandardDialog.java------------------------------
package newcomputer;

import java.awt.BorderLayout;
import java.awt.Container;
```

```
import java.awt.Dialog;
import java.awt.Frame;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.AbstractAction;
import javax.swing.Action;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JPanel;

/**
 *This class provides a Custom Dailog for laying the controls
 * @author Vijayananda.Mohire
 */
public class StandardDialog extends JDialog
{
   // Constants

   /** The spacing between components in pixels */
   private static final int COMPONENT_SPACING = 10;

   // Attributes

   /** Flag indicating if the "Cancel" button was pressed to close dialog */
   private boolean myIsDialogCancelled = true;

   /** The content pane for holding user components */
   private Container myUserContentPane;

   // Methods

   /**
    * This method is the default constructor.
    */
   public StandardDialog()
   {
      init();
   }
```

```
/**
 * This method creates a StandardDialog with the given parent frame
 * and title.
 * *
 * @param parent The parent frame for the dialog.
 * @param title The title to display in the dialog.
 */
public StandardDialog(Frame parent, String title)
{
   super(parent, title);

   init();
}

/**
 * This method creates a StandardDialog with the given parent dialog
 * and title.
 * *
 * @param parent The parent dialog for the dialog.
 * @param title The title to display in the dialog.
 */
public StandardDialog(Dialog parent, String title)
{
   super(parent, title);

   init();
}

/**
 * This method sets up the default attributes of the dialog and
 * the content pane.
 */
private void init()
{
   setModal(true);
   setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);

   // Setup the internal content pane to hold the user content pane
   // and the standard button panel

   JPanel internalContentPane = new JPanel();
```

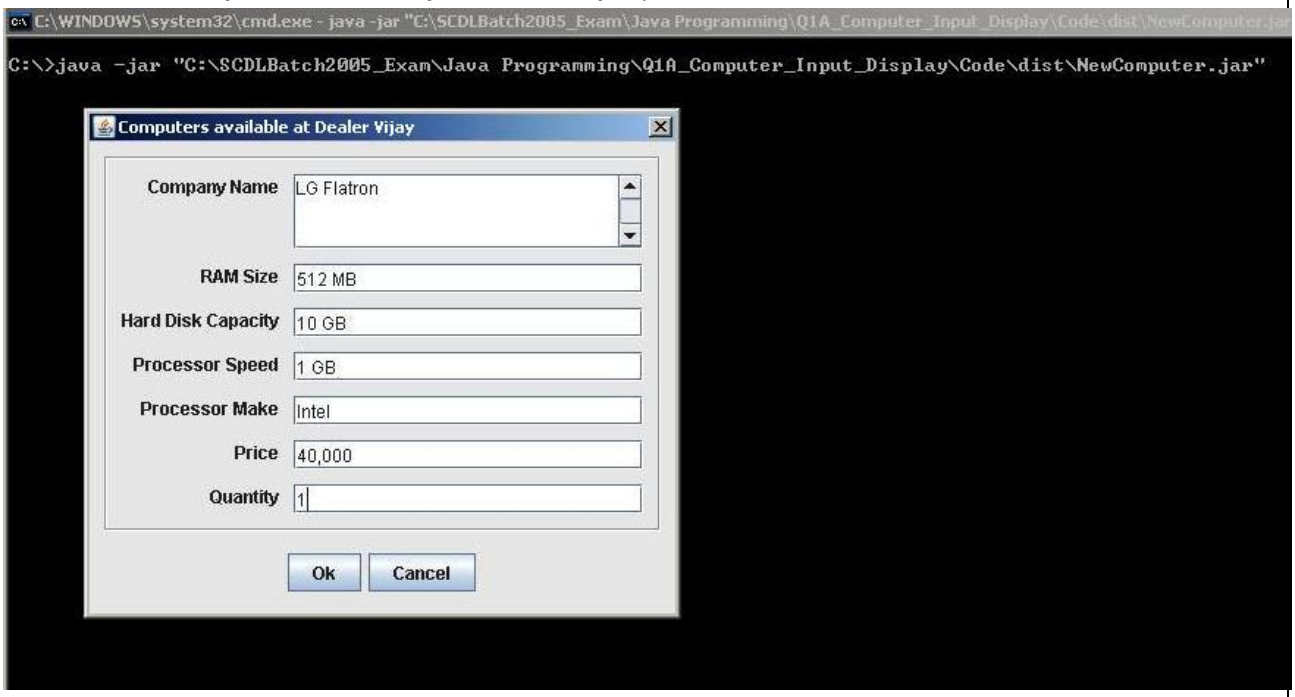
```
internalContentPane.setLayout(
   new BorderLayout(COMPONENT_SPACING, COMPONENT_SPACING));

internalContentPane.setBorder(
   BorderFactory.createEmptyBorder(COMPONENT_SPACING,
      COMPONENT_SPACING, COMPONENT_SPACING, COMPONENT_SPACING));

// Create the standard button panel with "Ok" and "Cancel"

Action okAction = new AbstractAction("Ok")
{
   public void actionPerformed(ActionEvent actionEvent)
   {
      if(isValidData())
      {
         myIsDialogCancelled = false;

         dispose();
      }
   }
};

Action cancelAction = new AbstractAction("Cancel")
{
   public void actionPerformed(ActionEvent actionEvent)
   {
      myIsDialogCancelled = true;

      dispose();
   }
};

JPanel buttonPanel = new JPanel();

buttonPanel.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 0));

buttonPanel.add(new JButton(okAction));
buttonPanel.add(new JButton(cancelAction));

internalContentPane.add(buttonPanel, BorderLayout.SOUTH);
```

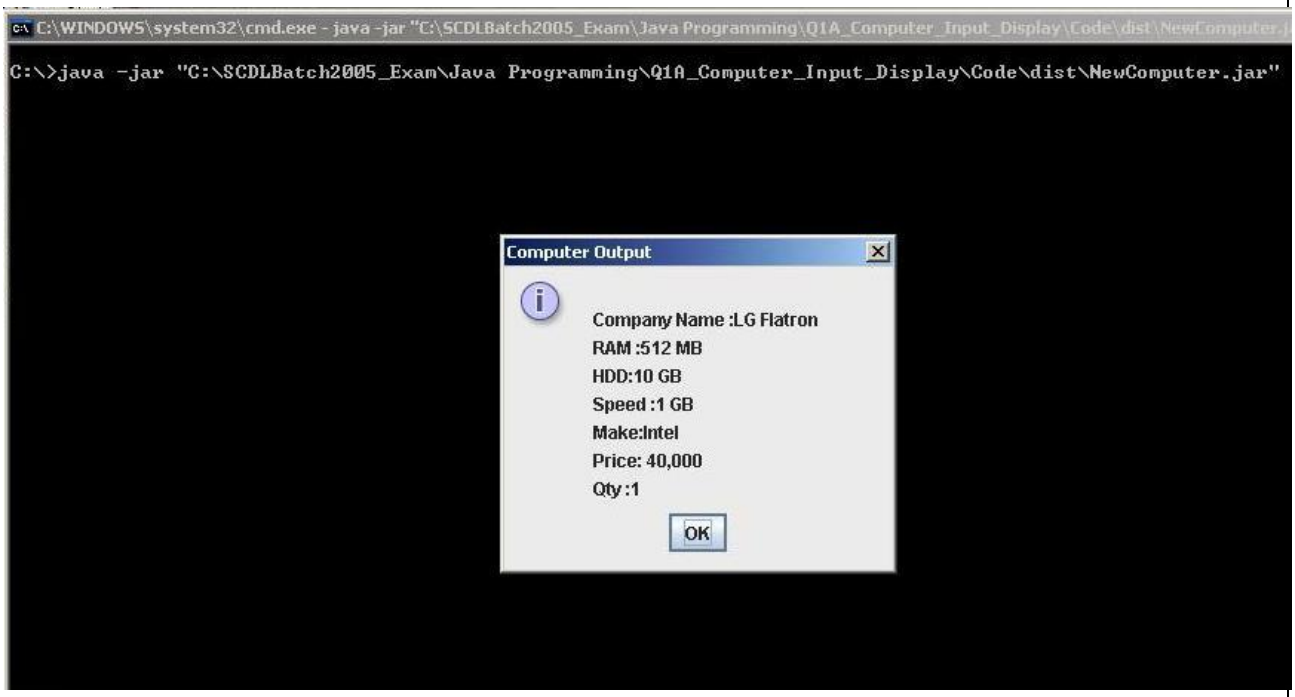
```
    // Initialise the user content pane with a JPanel

    setContentPane(new JPanel(new BorderLayout()));

    super.setContentPane(internalContentPane);

    // Finally, add a listener for the window close button.
    // Process this event the same as the "Cancel" button.

    WindowAdapter windowAdapter = new WindowAdapter()
    {
        public void windowClosing(WindowEvent windowEvent)
        {
            myIsDialogCancelled = true;

            dispose();
        }
    };

    addWindowListener(windowAdapter);
}

/**
 * This method gets the content pane for adding components.
 * Components should not be added directly to the dialog.
 * @return the content pane for the dialog.
 */
public Container getContentPane()
{
    return myUserContentPane;
}

/**
 * This method sets the content pane for adding components.
 * Components should not be added directly to the dialog.
 *
 * @param contentPane The content pane for the dialog.
 */
public void setContentPane(Container contentPane)
{
```

```
        myUserContentPane = contentPane;

        super.getContentPane().add(myUserContentPane, BorderLayout.CENTER);
    }

    /**
     * This method returns true if the User cancelled
     * the dialog otherwise false. The dialog is cancelled
     * if the "Cancel" button is pressed or the "Close" window button is
     * pressed, or the "Escape" key is pressed. In other words, if the
     * User has caused the dialog to close by any method other than by
     * pressing the "Ok" button, this method will return true.
     * @return Returns bool value
     */
    public boolean hasUserCancelled()
    {
        return myIsDialogCancelled;
    }

    /**
     * This method is used to validate the current dialog box.
     * @return a boolean indicating if the data is valid.
     *true> indicates that all of the fields were validated
     * correctly and false indicates the validation failed
     */
    protected boolean isValidData()
    {
        return true;
    }

}
```

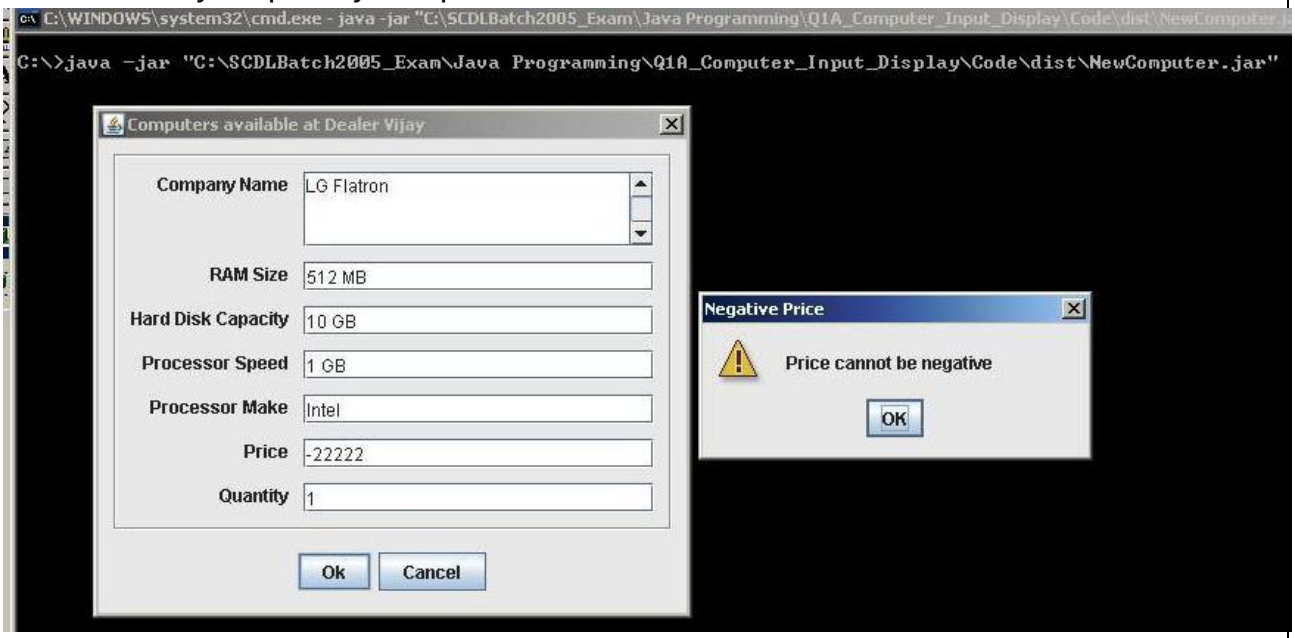
Results:

Create the Computer Class object and display the entered details





Checks validity of quantity and price



Evaluator's Comments if any:

Question 2

Write a Java program to create simple Calculator for 4 basic Math operations, Addition, Subtraction, Multiplication and Division.

The calculator should simulate the look of handheld calculator containing the following:
1. One textbox for displaying the input as well as output value.
2. Buttons for representing numbers from 0 to 9.
3. Buttons for representing mathematical operations .
Use suitable additional Swing components if required. Write proper code for making this calculator operational.

Students are expected to handle the related events like button click.

Answer 2

Code:

```
package calculator;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Window;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.KeyStroke;
/**
 *
 * @author Vijayananda.Mohire
 */
public class Calculator extends JFrame implements ActionListener{
        // Variables
        final int MAX_INPUT_LENGTH = 20;
        final int INPUT_MODE = 0;
        final int RESULT_MODE = 1;
        final int ERROR_MODE = 2;
        int displayMode;
```

```
        boolean clearOnNextDigit, percent;
        double lastNumber;
        String lastOperator;

        private JMenu jmenuFile, jmenuHelp;
        private JMenuItem jmenuitemExit, jmenuitemAbout;

        private JLabel jlbOutput;
        private JButton jbnButtons[];
        private JPanel jplMaster, jplBackSpace, jplControl;

        /*
         * Font(String name, int style, int size)
    Creates a new Font from the specified name, style and point size.
         */

        Font f12 = new Font("Times New Roman", 0, 12);
        Font f121 = new Font("Times New Roman", 1, 12);

        // Constructor
        public Calculator()
        {
                /* Set Up the JMenuBar.
                 * Have Provided All JMenu's with Mnemonics
                 * Have Provided some JMenuItem components with Keyboard
Accelerators
                 */

                jmenuFile = new JMenu("File");
                jmenuFile.setFont(f121);
                jmenuFile.setMnemonic(KeyEvent.VK_F);

                jmenuitemExit = new JMenuItem("Exit");
                jmenuitemExit.setFont(f12);
                jmenuitemExit.setAccelerator(KeyStroke.getKeyStroke(
KeyEvent.VK_X,ActionEvent.CTRL_MASK));
                jmenuFile.add(jmenuitemExit);

                jmenuHelp = new JMenu("Help");
                jmenuHelp.setFont(f121);
                jmenuHelp.setMnemonic(KeyEvent.VK_H);
```

```
        jmenuitemAbout = new JMenuItem("About Calculator");
        jmenuitemAbout.setFont(f12);
        jmenuHelp.add(jmenuitemAbout);

        JMenuBar mb = new JMenuBar();
        mb.add(jmenuFile);
        mb.add(jmenuHelp);
        setJMenuBar(mb);

        //Set frame layout manager

        setBackground(Color.gray);

        jplMaster = new JPanel();

        jlbOutput = new JLabel("0");
        jlbOutput.setHorizontalTextPosition(JLabel.RIGHT);
        jlbOutput.setBackground(Color.WHITE);
        jlbOutput.setOpaque(true);

        // Add components to frame
        getContentPane().add(jlbOutput, BorderLayout.NORTH);

        jbnButtons = new JButton[23];
//      GridLayout(int rows, int cols, int hgap, int vgap)

        JPanel jplButtons = new JPanel();       // container for Jbuttons

        // Create numeric Jbuttons
        for (int i=0; i<=9; i++)
        {
                // set each Jbutton label to the value of index
                jbnButtons[i] = new JButton(String.valueOf(i));
        }

        // Create operator Jbuttons
        jbnButtons[10] = new JButton("+/-");
        jbnButtons[11] = new JButton(".");
        jbnButtons[12] = new JButton("=");
        jbnButtons[13] = new JButton("/");
```

```
                jbnButtons[14] = new JButton("*");
                jbnButtons[15] = new JButton("-");
                jbnButtons[16] = new JButton("+");
                jbnButtons[17] = new JButton("sqrt");
                jbnButtons[18] = new JButton("1/x");
                jbnButtons[19] = new JButton("%");

                jplBackSpace = new JPanel();
                jplBackSpace.setLayout(new GridLayout(1, 1, 2, 2));

                jbnButtons[20] = new JButton("Backspace");
                jplBackSpace.add(jbnButtons[20]);

                jplControl = new JPanel();
                jplControl.setLayout(new GridLayout(1, 2, 2 ,2));

                jbnButtons[21] = new JButton(" CE ");
                jbnButtons[22] = new JButton("C");

                jplControl.add(jbnButtons[21]);
                jplControl.add(jbnButtons[22]);

//              Setting all Numbered JButton's to Blue. The rest to Red

for (int i=0; i<jbnButtons.length; i++)    {
                        jbnButtons[i].setFont(f12);

                        if (i<10)
                                jbnButtons[i].setForeground(Color.blue);

                        else
                                jbnButtons[i].setForeground(Color.red);
                }

                // Set panel layout manager for a 4 by 5 grid
                jplButtons.setLayout(new GridLayout(4, 5, 2, 2));

                //Add buttons to keypad panel starting at top left
                // First row
                for(int i=7; i<=9; i++)                 {
                        jplButtons.add(jbnButtons[i]);
```

```
}

// add button / and sqrt
jplButtons.add(jbnButtons[13]);
jplButtons.add(jbnButtons[17]);

// Second row
for(int i=4; i<=6; i++)
{
	jplButtons.add(jbnButtons[i]);
}

// add button * and x^2
jplButtons.add(jbnButtons[14]);
jplButtons.add(jbnButtons[18]);

// Third row
for( int i=1; i<=3; i++)
{
	jplButtons.add(jbnButtons[i]);
}

//adds button - and %
jplButtons.add(jbnButtons[15]);
jplButtons.add(jbnButtons[19]);

//Fourth Row
// add 0, +/-, ., +, and =
jplButtons.add(jbnButtons[0]);
jplButtons.add(jbnButtons[10]);
jplButtons.add(jbnButtons[11]);
jplButtons.add(jbnButtons[16]);
jplButtons.add(jbnButtons[12]);

jplMaster.setLayout(new BorderLayout());
jplMaster.add(jplBackSpace, BorderLayout.WEST);
jplMaster.add(jplControl, BorderLayout.EAST);
jplMaster.add(jplButtons, BorderLayout.SOUTH);

// Add components to frame
getContentPane().add(jplMaster, BorderLayout.SOUTH);
```

```
            requestFocus();

            //activate ActionListener
            for (int i=0; i<jbnButtons.length; i++){
                  jbnButtons[i].addActionListener(this);
            }

            jmenuitemAbout.addActionListener(this);
            jmenuitemExit.addActionListener(this);

            clearAll();

            //add WindowListener for closing frame and ending program
            addWindowListener(new WindowAdapter() {

                        public void windowClosed(WindowEvent e)
                        {
                              System.exit(0);
                        }
                  }
            );
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      }     //End of Contructor Calculator

      // Perform action
      public void actionPerformed(ActionEvent e){
            double result = 0;

            if(e.getSource() == jmenuitemAbout){
                  JDialog dlgAbout = new CustomABOUTDialog(this, "About Java
Swing Calculator", true);
                  dlgAbout.setVisible(true);
            }else if(e.getSource() == jmenuitemExit){
                  System.exit(0);
            }

            // Search for the button pressed until end of array or key found
            for (int i=0; i<jbnButtons.length; i++)
            {
                  if(e.getSource() == jbnButtons[i])
                  {
```

```
switch(i)
{
        case 0:
                addDigitToDisplay(i);
                break;

        case 1:
                addDigitToDisplay(i);
                break;

        case 2:
                addDigitToDisplay(i);
                break;

        case 3:
                addDigitToDisplay(i);
                break;

        case 4:
                addDigitToDisplay(i);
                break;

        case 5:
                addDigitToDisplay(i);
                break;

        case 6:
                addDigitToDisplay(i);
                break;

        case 7:
                addDigitToDisplay(i);
                break;

        case 8:
                addDigitToDisplay(i);
                break;

        case 9:
                addDigitToDisplay(i);
                break;
```

```
case 10:	// +/-
		processSignChange();
		break;

case 11:	// decimal point
		addDecimalPoint();
		break;

case 12:	// =
		processEquals();
		break;

case 13:	// divide
		processOperator("/");
		break;

case 14:	// *
		processOperator("*");
		break;

case 15:	// -
		processOperator("-");
		break;

case 16:	// +
		processOperator("+");
		break;

case 17:	// sqrt
		if (displayMode != ERROR_MODE)
		{
			try
			{
		if (getDisplayString().indexOf("-") == 0)
		displayError("Invalid input for function!");
		result = Math.sqrt(getNumberInDisplay());
				displayResult(result);
			}

			catch(Exception ex)
```

```
                                        {
                                                displayError("Invalid input for
function!");
                                                displayMode = ERROR_MODE;
                                        }
                                }
                                break;

                        case 18:        // 1/x
                                if (displayMode != ERROR_MODE){
                                        try
                                        {
                                        if (getNumberInDisplay() == 0)
                                        displayError("Cannot divide by zero!");
                                        result = 1 / getNumberInDisplay();
                                        displayResult(result);
                                        }

                                        catch(Exception ex){
                                        displayError("Cannot divide by zero!");
                                        displayMode = ERROR_MODE;
                                        }
                                }
                                break;

                        case 19:        // %
                                if (displayMode != ERROR_MODE){
                                        try     {
                                        result = getNumberInDisplay() / 100;
                                        displayResult(result);
                                        }

                                        catch(Exception ex){
                                displayError("Invalid input for function!");
                                displayMode = ERROR_MODE;
                                        }
                                }
                                break;

                        case 20:        // backspace
                                if (displayMode != ERROR_MODE){
```

```
setDisplayString(getDisplayString().substring(0,

getDisplayString().length() - 1));

                                        if (getDisplayString().length() < 1)
                                                setDisplayString("0");
                                }
                                break;

                        case 21:        // CE
                                clearExisting();
                                break;

                        case 22:        // C
                                clearAll();
                                break;
                        }
                }
        }
}

void setDisplayString(String s){
        jlbOutput.setText(s);
}

String getDisplayString (){
        return jlbOutput.getText();
}

void addDigitToDisplay(int digit){
        if (clearOnNextDigit)
                setDisplayString("");

        String inputString = getDisplayString();

        if (inputString.indexOf("0") == 0){
                inputString = inputString.substring(1);
        }

        if ((!inputString.equals("0") || digit > 0)  && inputString.length() <
```

```
MAX_INPUT_LENGTH){
                    setDisplayString(inputString + digit);
            }


            displayMode = INPUT_MODE;
            clearOnNextDigit = false;
        }

        void addDecimalPoint(){
            displayMode = INPUT_MODE;

            if (clearOnNextDigit)
                setDisplayString("");

            String inputString = getDisplayString();

            // If the input string already contains a decimal point, don't
            //  do anything to it.
            if (inputString.indexOf(".") < 0)
                setDisplayString(new String(inputString + "."));
        }

        void processSignChange(){
            if (displayMode == INPUT_MODE)
            {
                String input = getDisplayString();

                if (input.length() > 0 && !input.equals("0"))
                {
                    if (input.indexOf("-") == 0)
                        setDisplayString(input.substring(1));

                    else
                        setDisplayString("-" + input);
                }
            }

            else if (displayMode == RESULT_MODE)
            {
                double numberInDisplay = getNumberInDisplay();
```

```
                if (numberInDisplay != 0)
                        displayResult(-numberInDisplay);
        }
}

void clearAll()        {
        setDisplayString("0");
        lastOperator = "0";
        lastNumber = 0;
        displayMode = INPUT_MODE;
        clearOnNextDigit = true;
}

void clearExisting(){
        setDisplayString("0");
        clearOnNextDigit = true;
        displayMode = INPUT_MODE;
}

double getNumberInDisplay()      {
        String input = jlbOutput.getText();
        return Double.parseDouble(input);
}

void processOperator(String op) {
        if (displayMode != ERROR_MODE)
        {
                double numberInDisplay = getNumberInDisplay();

                if (!lastOperator.equals("0"))
                {
                        try
                        {
                                double result = processLastOperator();
                                displayResult(result);
                                lastNumber = result;
                        }

                        catch (DivideByZeroException e)
                        {
                        }
```

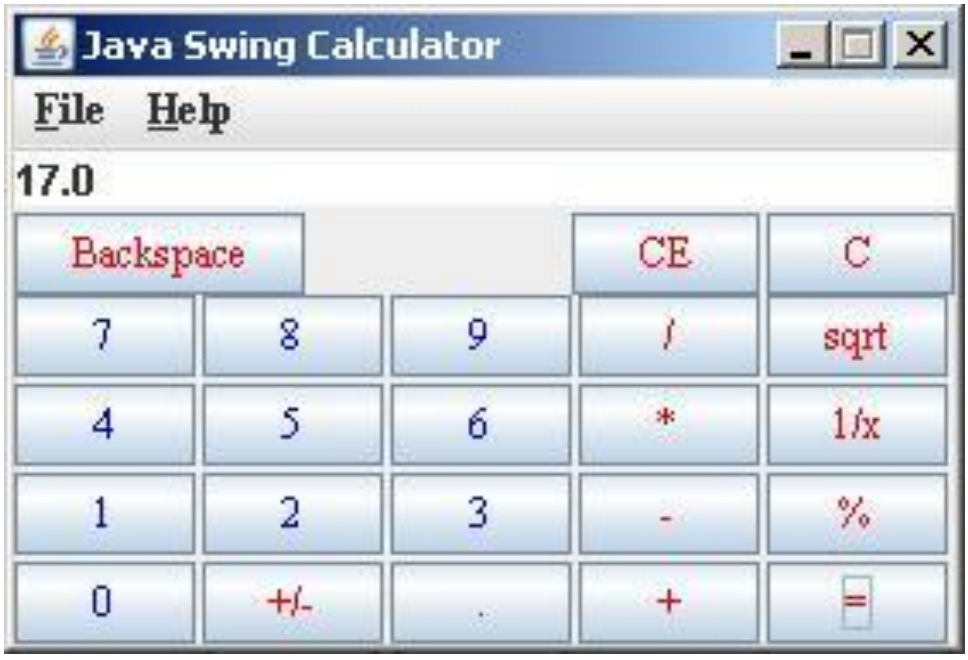
```
                }
        else
                {
                        lastNumber = numberInDisplay;
                }

                clearOnNextDigit = true;
                lastOperator = op;
        }
}

void processEquals(){
        double result = 0;

        if (displayMode != ERROR_MODE){
                try
                {
                        result = processLastOperator();
                        displayResult(result);
                }

                catch (DivideByZeroException e) {
                        displayError("Cannot divide by zero!");
                }

                lastOperator = "0";
        }
}

double processLastOperator() throws DivideByZeroException {
        double result = 0;
        double numberInDisplay = getNumberInDisplay();

        if (lastOperator.equals("/"))
        {
                if (numberInDisplay == 0)
                        throw (new DivideByZeroException());

                result = lastNumber / numberInDisplay;
        }
```

```
            if (lastOperator.equals("*"))
                result = lastNumber * numberInDisplay;

            if (lastOperator.equals("-"))
                result = lastNumber - numberInDisplay;

            if (lastOperator.equals("+"))
                result = lastNumber + numberInDisplay;

            return result;
        }

        void displayResult(double result){
            setDisplayString(Double.toString(result));
            lastNumber = result;
            displayMode = RESULT_MODE;
            clearOnNextDigit = true;
        }

        void displayError(String errorMessage){
            setDisplayString(errorMessage);
            lastNumber = 0;
            displayMode = ERROR_MODE;
            clearOnNextDigit = true;
        }

        public static void main(String args[]) {
            Calculator calci = new Calculator();
            Container contentPane = calci.getContentPane();
//          contentPane.setLayout(new BorderLayout());
            calci.setTitle("Java Swing Calculator");
            calci.setSize(241, 217);
            calci.pack();
            calci.setLocation(400, 250);
            calci.setVisible(true);
            calci.setResizable(false);
        }

}           //End of Swing Calculator Class.

class DivideByZeroException extends Exception{
```

```
        public DivideByZeroException()
        {
                super();
        }

        public DivideByZeroException(String s)
        {
                super(s);
        }
}

class CustomABOUTDialog extends JDialog implements ActionListener {
        JButton jbnOk;

        CustomABOUTDialog(JFrame parent, String title, boolean modal){
                super(parent, title, modal);
                setBackground(Color.black);

                JPanel p1 = new JPanel(new FlowLayout(FlowLayout.CENTER));

                StringBuffer text = new StringBuffer();
                text.append("Calculator Information\n\n");
                text.append("Developer:   Vijay Mohire\n");
                text.append("Version:      1.0");

                JTextArea jtAreaAbout = new JTextArea(5, 21);
                jtAreaAbout.setText(text.toString());
                jtAreaAbout.setFont(new Font("Times New Roman", 1, 13));
                jtAreaAbout.setEditable(false);

                p1.add(jtAreaAbout);
                p1.setBackground(Color.red);
                getContentPane().add(p1, BorderLayout.CENTER);

                JPanel p2 = new JPanel(new FlowLayout(FlowLayout.CENTER));
                jbnOk = new JButton(" OK ");
                jbnOk.addActionListener(this);

                p2.add(jbnOk);
                getContentPane().add(p2, BorderLayout.SOUTH);
```

```
            setLocation(408, 270);
            setResizable(false);

            addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e)
                        {
                                Window aboutDialog = e.getWindow();
                                aboutDialog.dispose();
                        }
                  }
            );

            pack();
      }

      public void actionPerformed(ActionEvent e)
      {
            if(e.getSource() == jbnOk) {
                  this.dispose();
            }
      }

}
---------------------------------
```

Results:


Java Swing Calculator
File
Help
17.0
Backspace
CE
C
7
8
9
/
sqrt
4
5
6
*
1/x
1
2
3
-
%
0
+/-
.
+
=