```
/*PRO 2.8 Write a java program that demonstrates the concept of
HashMap constructors and its all methods.*/


import java.util.HashMap;

import java.util.Map;


public class HashMapDemo {


public static void main(String[] args)

{

        // Demonstrate constructors

        Map<Integer, String> hashMap1 = new HashMap<Integer,
String>(); // Default constructor

        Map<Integer, String> hashMap2 = new HashMap<Integer,
String>(16); // Constructor with initial capacity

        Map<Integer, String> hashMap3 = new HashMap<Integer,
String>(hashMap1); // Constructor with another map


        // Demonstrate methods Adding elements

        hashMap1.put(1, "One");

        hashMap1.put(2, "Two");

        hashMap1.put(3, "Three");


        // Accessing elements

       System.out.println("Value at key 1: " + hashMap1.get(1));


        // Checking if key exists

      System.out.println("Contains key 2: " +
      ashMap1.containsKey(2));
```

```
        // Checking if value exists
        System.out.println("Contains value 'Three': " +
hashMap1.containsValue("Three"));

        // Removing elements
        hashMap1.remove(3);

        // Iterating over keys
        System.out.println("Keys in hashMap1:");
        for (Integer key : hashMap1.keySet()) {
            System.out.println(key);
        }

        // Iterating over values
        System.out.println("Values in hashMap1:");
        for (String value : hashMap1.values()) {
            System.out.println(value);
        }

        // Iterating over entries
        System.out.println("Entries in hashMap1:");
        for (Map.Entry<Integer, String> entry :
hashMap1.entrySet()) {
            System.out.println(entry.getKey() + " -> " +
entry.getValue());
        }

        // Size of the map
        System.out.println("Size of hashMap1: " +
hashMap1.size());

        // Clearing the map
        hashMap1.clear();
        System.out.println("Size of hashMap1 after clear(): " +
hashMap1.size());
    }
}
/*PRO 2.9 Write a program that demonstrates the concept of
ArrayList constructors and its all methods.*/

import java.util.ArrayList;
import java.util.List;

public class ArrayListDemo2 {

    public static void main(String[] args) {
        // Constructors
        List<String> arrayList = new ArrayList<String>();
       // Default constructor Methods Adding elements
        arrayList.add("One");
        arrayList.add("Two");
        arrayList.add("Three");

        // Accessing elements
        System.out.println("Element at index 1: " +
arrayList.get(1));
```

```
        // Checking if an element exists
        System.out.println("Contains 'Two': " +
arrayList.contains("Two"));

        // Removing elements
        arrayList.remove("Two");

        // Iterating over elements
        System.out.println("Elements:");
        for (String element : arrayList) {
            System.out.println(element);
        }

        // Size of the list
        System.out.println("Size: " + arrayList.size());

        // Clearing the list
        arrayList.clear();
        System.out.println("Size after clear(): " +
arrayList.size());
    }
}
/*PRO 3.1 Write a java program that demonstrates the concept of
multithreading.*/

public class MultithreadingDemo {
    public static void main(String[] args) {
        // Thread 1
        Thread thread1 = new Thread(new Runnable() {
```

```
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println("Thread 1: " + i);
            try {
               Thread.sleep(1000); // Sleep for 1 second
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
});

// Thread 2
Thread thread2 = new Thread(new Runnable() {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println("Thread 2: " + i);
            try {
               Thread.sleep(1000); // Sleep for 1 second
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
});

// Start the threads
```

```
        thread1.start();
        thread2.start();
    }
}
/*PRO 3.2 Write a java program which shows the use of wait() and
notify() methods.*/

public class WaitNotifyExample1{
    public static void main(String[] args) {
        SharedObject sharedObject = new SharedObject();

 Thread producerThread = new Thread(new Producer(sharedObject));
 Thread consumerThread = new Thread(new Consumer(sharedObject));

        producerThread.start();
        consumerThread.start();
    }
}
class SharedObject {
    private int value;
    private boolean produced = false;

    synchronized void produce(int value) {
        while (produced) {
        try {
          wait(); // Wait until the consumer consumes the value
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        this.value = value;
        System.out.println("Producer produced: " + value);
        produced = true;
        notify(); // Notify the consumer that value is produced
    }

    synchronized int consume() {
        while (!produced) {
         try {
           wait(); // Wait until the producer produces the value
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println("Consumer consumed: " + value);
        produced = false;
        notify(); // Notify the producer that value is consumed
        return value;
    }
}

class Producer implements Runnable {
    private SharedObject sharedObject;

    Producer(SharedObject sharedObject) {
        this.sharedObject = sharedObject;
    }
    public void run() {
        for (int i = 1; i <= 5; i++) {
            sharedObject.produce(i);
            try {
               Thread.sleep(1000); // Simulating production time
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class Consumer implements Runnable {
    private SharedObject sharedObject;

    Consumer(SharedObject sharedObject) {
        this.sharedObject = sharedObject;
    }
    public void run() {
        for (int i = 0; i < 5; i++) {
            sharedObject.consume();
            try {
              Thread.sleep(1500); // Simulating consumption time
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
/*PRO 4.1 Write a program which lists all files in a specific
directory with its name, length, and last modification date. The
directory name should be passed as command line argument.*/


import java.io.File;
import java.text.SimpleDateFormat;

public class DirectoryListing {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("Usage: java DirectoryListing
<directory_path>");
            System.exit(1);
        }

        String directoryPath = args[0];
        File directory = new File(directoryPath);

        if (!directory.exists() || !directory.isDirectory()) {
            System.err.println("Invalid directory path
provided.");
            System.exit(1);
        }

        File[] files = directory.listFiles();

        if (files != null) {
            System.out.println("Listing files in directory: " +
directory.getAbsolutePath());
            System.out.printf("%-50s %-15s %-30s\n", "File
Name", "Length (bytes)", "Last Modified Date");
           System.out.println(String.format("%-100s",
"").replace(' ', '-'));
            SimpleDateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

            for (File file : files) {
                if (file.isFile()) {
                    String fileName = file.getName();
                    long fileLength = file.length();
                    String lastModified =
dateFormat.format(file.lastModified());
                    System.out.printf("%-50s %-15d %-30s\n",
fileName, fileLength, lastModified);
                }
            }
        } else {
            System.err.println("Failed to list files in the
directory.");
        }
    }
}
```

OUTPUT:

```
  Javac DirectoryListing.java
  Java DirectoryListing d:\java\Project
```

```
/*PRO 4.2 Write a program that copies all the content from one
file to another.*/


import java.io.*;
class CopyChar
{
public static void main(String s[])
{
try
{
FileReader fr = new FileReader("city.txt");
FileWriter fw = new FileWriter("cityCopy.txt");
int x;

while((x=fr.read())!=-1)
{
fw.write(x);
}
System.out.println("Data cpoy to the file successfully");
fw.close();
fr.close();
}
catch(Exception e)
{
System.out.println("Exception : " + e);
}
}
}
```

```
/*PRO 4.3 Write a program to compare two files. The filename
must be passed as a command line argument.
  Provide proper error messages and perform appropriate
exception handling where ever required*/


import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;


public class FileComparator
{
    public static void main(String[] args)
 {
        if (args.length != 2)
 {
            System.err.println("Usage: java FileComparator
file1.txt file2.txt");
            System.exit(1);
        }

        String file1 = args[0];
        String file2 = args[1];

        File file1 = new File(file1);
        File file2 = new File(file2);

        try {(FileInputStream fis1 = new FileInputStream(file1);
             FileInputStream fis2 = new FileInputStream(file2))
            {
           int byte1, byte2;
           long byteCount = 0;

           while ((byte1 = fis1.read()) != -1 && (byte2 =
fis2.read()) != -1)
      {
               byteCount++;
               if (byte1 != byte2)
     {
      System.out.printf("Files differ at byte %d\n", byteCount);
      System.out.println("Files are not identical.");
       return;
       }
       }
     }
           if (fis1.read() != fis2.read()) {
          System.out.println("Files have different lengths.");
           } else {
               System.out.println("Files are identical.");
            }

        } catch (FileNotFoundException e) {
       System.err.println("File not found: " + e.getMessage());
        } catch (IOException e) {
   System.err.println("Error reading files: " + e.getMessage());
        }
    }
```

```
}
/* PRO 4.4 Write program that creates simple java bean which
will give the appropriate message such as
      Good Morning, Good Noon or Good Night to the user based on
the hours of the day.*/


import java.util.Calendar;


 class Greetings {
    public static String getGreetingMessage() {
        int hourOfDay =
Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
        if (hourOfDay >= 0 && hourOfDay < 12) {
            return "Good Morning";
        } else if (hourOfDay >= 12 && hourOfDay < 18) {
            return "Good Afternoon";
        } else {
            return "Good Evening";
        }
    }
}
public class Main4 {
    public static void main(String[] args) {
        System.out.println(Greetings.getGreetingMessage());
    }
}
/* PRO 5.1 Create a frame with three text Fields and two buttons
add and subtract. User will enter numeric
```

```
  values in the Text Fields. When add button is pressed, the
addition of the two values should be
  displayed in the third Text Field. Same the Subtract button
should perform the subtraction operation*/

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

public class CalculatorFrame extends JFrame {
    private JTextField textField1, textField2, resultField;

    public CalculatorFrame() {
        setTitle("Calculator");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create text fields
        textField1 = new JTextField(10);
        textField2 = new JTextField(10);
        resultField = new JTextField(10);
        resultField.setEditable(false); // Make result field
read-only

        // Create buttons
        JButton addButton = new JButton("Add");
        addButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
         double num1 = Double.parseDouble(textField1.getText());
        double num2 = Double.parseDouble(textField2.getText());
                    double result = num1 + num2;
                    resultField.setText(String.valueOf(result));
                } catch (NumberFormatException ex) {
                    resultField.setText("Invalid Input");
                }
            }
        });

        JButton subtractButton = new JButton("Subtract");
        subtractButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
        double num1 = Double.parseDouble(textField1.getText());
        double num2 = Double.parseDouble(textField2.getText());
                    double result = num1 - num2;
                    resultField.setText(String.valueOf(result));
                } catch (NumberFormatException ex) {
                    resultField.setText("Invalid Input");
                }
            }
        });

        // Create panel for buttons
        JPanel buttonPanel = new JPanel();
        buttonPanel.add(addButton);
        buttonPanel.add(subtractButton);
```

```
        // Create main panel
        JPanel mainPanel = new JPanel();
        mainPanel.setLayout(new GridLayout(4, 1));
        mainPanel.add(textField1);
        mainPanel.add(textField2);
        mainPanel.add(resultField);
        mainPanel.add(buttonPanel);

        // Add main panel to frame
        getContentPane().add(mainPanel);

        pack();
        setLocationRelativeTo(null); // Center the frame
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new CalculatorFrame().setVisible(true);
            }
        });
    }
}
/*PRO 5.2 Make a frame that contains three scroll-bars. The
scroll-bars adjust the Red, Green and Blue components of the
frame color.*/


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

public class ColorAdjustmentFrame extends JFrame {
    private JScrollBar redScrollBar, greenScrollBar,
blueScrollBar;
    private JPanel colorPanel;

    public ColorAdjustmentFrame() {
        setTitle("Color Adjustment");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create scroll bars
         redScrollBar = new JScrollBar(JScrollBar.HORIZONTAL,
         0, 1, 0, 256);
        greenScrollBar = new JScrollBar(JScrollBar.HORIZONTAL,
0, 1, 0, 256);
        blueScrollBar = new JScrollBar(JScrollBar.HORIZONTAL, 0,
1, 0, 256);

        // Create panel to display color
        colorPanel = new JPanel();
        colorPanel.setPreferredSize(new Dimension(200, 200));
        updateColor();

        // Add scroll bar listeners
        redScrollBar.addAdjustmentListener(new
ScrollBarListener());
        greenScrollBar.addAdjustmentListener(new
ScrollBarListener());
```

```
        blueScrollBar.addAdjustmentListener(new
ScrollBarListener());

        // Create main panel
        JPanel mainPanel = new JPanel();
        mainPanel.setLayout(new GridLayout(4, 1));
        mainPanel.add(new JLabel("Red"));
        mainPanel.add(redScrollBar);
        mainPanel.add(new JLabel("Green"));
        mainPanel.add(greenScrollBar);
        mainPanel.add(new JLabel("Blue"));
        mainPanel.add(blueScrollBar);
        mainPanel.add(new JLabel("Color Preview"));
        mainPanel.add(colorPanel);

        // Add main panel to frame
        getContentPane().add(mainPanel);

        pack();
        setLocationRelativeTo(null); // Center the frame
    }

    private void updateColor() {
        int red = redScrollBar.getValue();
        int green = greenScrollBar.getValue();
        int blue = blueScrollBar.getValue();
        Color color = new Color(red, green, blue);
        colorPanel.setBackground(color);
    }

    private class ScrollBarListener implements
AdjustmentListener {
        public void adjustmentValueChanged(AdjustmentEvent e) {
            updateColor();
        }
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new ColorAdjustmentFrame().setVisible(true);
            }
        });
    }
}
/*PRO 5.3 Write a program that creates a scrolling list with
several choices and informs you about selection of items using a
label.*/
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;

public class ScrollingListExample extends JFrame {
    public JList list;
    public JLabel label;

    public ScrollingListExample() {
        setTitle("Scrolling List Example");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 200);

        // Creating choices
        String[] choices = {"Choice 1", "Choice 2", "Choice 3",
"Choice 4", "Choice 5"};

        // Creating JList with choices
        list = new JList(choices);
list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        list.addListSelectionListener(new
ListSelectionListener() {
            public void valueChanged(ListSelectionEvent e) {
                if (!e.getValueIsAdjusting()) {
                    // Update label with the selected item
                    label.setText("Selected: " +
list.getSelectedValue());
                }
            }
        });

        // Creating label to display selection
        label = new JLabel("Selected: ");
        label.setHorizontalAlignment(SwingConstants.CENTER);

        // Adding list and label to the frame
        getContentPane().setLayout(new BorderLayout());
        getContentPane().add(new JScrollPane(list),
BorderLayout.CENTER);
```

```
        getContentPane().add(label, BorderLayout.SOUTH);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new ScrollingListExample().setVisible(true);
            }
        });
    }
}

/*PRO 5.4 Write a program to display the focus status of
components in the label.*/
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class FocusStatusExample extends JFrame {
    private JLabel focusLabel;

    public FocusStatusExample() {
        setTitle("Focus Status Example");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 150);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3, 1));
        // Text field
        JTextField textField = new JTextField();
        textField.addFocusListener(new FocusListener() {
            public void focusGained(FocusEvent e) {
                focusLabel.setText("Focus gained on
TextField.");
            }

            public void focusLost(FocusEvent e) {
                focusLabel.setText("Focus lost from
TextField.");
            }
        });
        panel.add(textField);

        // Button
        JButton button = new JButton("Click Me");
        button.addFocusListener(new FocusListener() {
            public void focusGained(FocusEvent e) {
                focusLabel.setText("Focus gained on Button.");
            }

            public void focusLost(FocusEvent e) {
                focusLabel.setText("Focus lost from Button.");
            }
        });
        panel.add(button);
        // Label to display focus status
        focusLabel = new JLabel("No focus on any component.");
        panel.add(focusLabel);

        getContentPane().add(panel);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new FocusStatusExample().setVisible(true);
            }
        });
    }
}
/*PRO 5.5 Make font size and font type List Boxes and give that
effect in your Label or Text Box.*/

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

public class FontSelectorExample extends JFrame {
    private JLabel textLabel;
    private JComboBox fontSizeComboBox;
    private JComboBox fontTypeComboBox;
    public FontSelectorExample() {
        setTitle("Font Selector Example");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 200);
        setLayout(new FlowLayout());

        // Create label
        textLabel = new JLabel("Sample Text");
        add(textLabel);

        // Create font size list box
        String[] fontSizeOptions = {"8", "10", "12", "14", "16",
"18", "20", "22", "24"};
        fontSizeComboBox = new JComboBox(fontSizeOptions);
        fontSizeComboBox.addActionListener(new ActionListener()
{
            public void actionPerformed(ActionEvent e) {
                updateLabelFont();
            }
        });
        add(new JLabel("Font Size: "));
        add(fontSizeComboBox);

        // Create font type list box
        String[] fontTypeOptions = {"Arial", "Times New Roman",
"Courier New"};
        fontTypeComboBox = new JComboBox(fontTypeOptions);
        fontTypeComboBox.addActionListener(new ActionListener()
{
```

```
            public void actionPerformed(ActionEvent e) {
                updateLabelFont();
            }
        });
        add(new JLabel("Font Type: "));
        add(fontTypeComboBox);
    }

    private void updateLabelFont() {
        String selectedFontSize = (String)
fontSizeComboBox.getSelectedItem();
        String selectedFontType = (String)
fontTypeComboBox.getSelectedItem();
        int fontSize = Integer.parseInt(selectedFontSize);
        Font font = new Font(selectedFontType, Font.PLAIN,
fontSize);
        textLabel.setFont(font);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new FontSelectorExample().setVisible(true);
            }
        });
    }
}
/* PRO 5.6 Create a user entry form for student data. Users will
enter roll no, name, department and semester in the form. Use a
```

```
Radio Button for the department.When the user clicks on the
Insert button all the values should be displayed in the Text
Area.*/

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

public class StudentEntryForm extends JFrame {
    private JTextField rollNoField;
    private JTextField nameField;
    private JRadioButton cseRadio;
    private JRadioButton eceRadio;
    private JRadioButton mechRadio;
    private JTextField semesterField;
    private JTextArea displayArea;

    public StudentEntryForm() {
        setTitle("Student Entry Form");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 300);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(6, 2));

        // Roll No
        panel.add(new JLabel("Roll No:"));
        rollNoField = new JTextField();
```

```
panel.add(rollNoField);

// Name
panel.add(new JLabel("Name:"));
nameField = new JTextField();
panel.add(nameField);

// Department
panel.add(new JLabel("Department:"));
ButtonGroup deptGroup = new ButtonGroup();
cseRadio = new JRadioButton("CSE");
eceRadio = new JRadioButton("ECE");
mechRadio = new JRadioButton("Mechanical");
deptGroup.add(cseRadio);
deptGroup.add(eceRadio);
deptGroup.add(mechRadio);
JPanel deptPanel = new JPanel(new GridLayout(1, 3));
deptPanel.add(cseRadio);
deptPanel.add(eceRadio);
deptPanel.add(mechRadio);
panel.add(deptPanel);

// Semester
panel.add(new JLabel("Semester:"));
semesterField = new JTextField();
panel.add(semesterField);

// Insert button
```

```
        JButton insertButton = new JButton("Insert");
        insertButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                displayData();
            }
        });
        panel.add(insertButton);

        // Display area
        displayArea = new JTextArea();
        JScrollPane scrollPane = new JScrollPane(displayArea);
        panel.add(scrollPane);

        getContentPane().add(panel);
    }
    private void displayData() {
        String rollNo = rollNoField.getText();
        String name = nameField.getText();
        String department = "";
        if (cseRadio.isSelected()) {
            department = "CSE";
        } else if (eceRadio.isSelected()) {
            department = "ECE";
        } else if (mechRadio.isSelected()) {
            department = "Mechanical";
        }
        String semester = semesterField.getText();
        String data = "Roll No: " + rollNo + "\n"
                    + "Name: " + name + "\n"
                    + "Department: " + department + "\n"
                    + "Semester: " + semester + "\n\n";

        displayArea.append(data);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new StudentEntryForm().setVisible(true);
            }
        });
    }
}

/*PRO 5.7 Write a program to draw line, rectangle, oval and text
using graphics methods.*/
import javax.swing.*;
import java.awt.*;

public class DrawingExample extends JPanel {

    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Draw a line
        g.drawLine(50, 50, 200, 50);

        // Draw a rectangle
        g.drawRect(50, 100, 150, 80);

        // Draw a filled rectangle
        g.setColor(Color.blue);
        g.fillRect(250, 100, 150, 80);

        // Draw an oval
        g.setColor(Color.red);
        g.drawOval(50, 200, 150, 100);

        // Draw a filled oval
        g.setColor(Color.green);
        g.fillOval(250, 200, 150, 100);

        // Draw text
        g.setColor(Color.black);
        g.setFont(new Font("Arial", Font.BOLD, 20));
        g.drawString("Hello, Java Graphics!", 100, 350);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Drawing Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 500);
        frame.add(new DrawingExample());
        frame.setVisible(true);
    }
}

//PRO 5.8 Write a program to create the Menu within the Frame.
import javax.swing.*;
import java.awt.event.*;

public class MenuExample extends JFrame {
    public MenuExample() {
        setTitle("Menu Example");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a menu bar
        JMenuBar menuBar = new JMenuBar();

        // Create a menu
        JMenu fileMenu = new JMenu("File");

        // Create menu items
        JMenuItem newItem = new JMenuItem("New");
        JMenuItem openItem = new JMenuItem("Open");
        JMenuItem saveItem = new JMenuItem("Save");
        JMenuItem exitItem = new JMenuItem("Exit");

        // Add menu items to the menu
        fileMenu.add(newItem);
        fileMenu.add(openItem);
        fileMenu.add(saveItem);
        fileMenu.addSeparator(); // Add a separator between Save
and Exit
        fileMenu.add(exitItem);

        // Add action listener to exit menu item
        exitItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });

        // Add the menu to the menu bar
        menuBar.add(fileMenu);

        // Set the menu bar to the frame
        setJMenuBar(menuBar);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new MenuExample().setVisible(true);
            }
        });
    }
}
```

```
/* PRO 5.9 Write a program to display Load and Save file dialog
& display the name of a selected file from the file dialog.*/


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

public class FileDialogExample extends JFrame implements
ActionListener {
    JButton loadButton, saveButton;
    JLabel statusLabel;

    public FileDialogExample() {
        setTitle("File Dialog Example");
        setSize(300, 200);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        loadButton = new JButton("Load File");
        saveButton = new JButton("Save File");
        statusLabel = new JLabel("No file selected");

        loadButton.addActionListener(this);
        saveButton.addActionListener(this);

        JPanel panel = new JPanel();
        panel.add(loadButton);
        panel.add(saveButton);
        panel.add(statusLabel);
        add(panel);

        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == loadButton) {
            JFileChooser fileChooser = new JFileChooser();
            int returnValue = fileChooser.showOpenDialog(this);
            if (returnValue == JFileChooser.APPROVE_OPTION) {
                File selectedFile =
fileChooser.getSelectedFile();
                statusLabel.setText("Selected file: " +
selectedFile.getName());
            }
        } else if (e.getSource() == saveButton) {
            JFileChooser fileChooser = new JFileChooser();
            int returnValue = fileChooser.showSaveDialog(this);
            if (returnValue == JFileChooser.APPROVE_OPTION) {
                File selectedFile =
fileChooser.getSelectedFile();
                statusLabel.setText("Saved file: " +
selectedFile.getName());
            }
        }
    }

    public static void main(String[] args) {
        new FileDialogExample();
    }
}

/*PRO 5.10 Write a program to explain the concept of adapter
class for window listeners */

import java.awt.*;
import java.awt.event.*;

 /* Custom frame that listens for window events using
 WindowAdapter*/

class MyFrame extends Frame {
    MyFrame(String title) {
        super(title);

        // Add a window listener
        addWindowListener(new MyWindowAdapter());

        // Set frame size and make it visible
        setSize(300, 200);
        setVisible(true);
    }
}
// Custom adapter class extending WindowAdapter

class MyWindowAdapter extends WindowAdapter {
    // Override only the window closing event

    public void windowClosing(WindowEvent e) {
        System.out.println("Window is closing");
        System.exit(0);
    }
}
// Main class to demonstrate the usage
public class AdapterDemo {
    public static void main(String[] args) {
        MyFrame frame = new MyFrame("Adapter Demo");
    }
}
```