5COSC019W - Tutorial 8 Exercises

1 A Simple Swing Program

1. Compile and run the following program.

```
import javax.swing.*;

public class SimpleSwingExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("SimpleSwingExample");
        frame.setSize(400, 400);
        frame.setVisible(true);
    }
}
```

2. What happens when you press the "window closing" button on the top right of the window? Does the program terminate is it still running?

3. Add a listener object to the frame, so that when the "window closing" button is pressed, the program terminates with the message `Program Exiting...`.

   Hint: Look at the lecture notes and make sure you understand the code involved.

2 Event Handling

1. What does the following code do? Make sure you understand fully every single line of the code.

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

// window event Handler class
class MyWindowListener extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
        System.out.println("Closing window!");
        System.exit(0);
    }
}

// button event handler class
```

```
class MyActionListener implements ActionListener {
    private int i=1;
    JFrame frame;
    MyActionListener(JFrame f) {
        frame = f;
    }

    public void actionPerformed(ActionEvent e) {
        System.out.println("Pressed Button " + i++ + "th time!");

        if (i % 2 == 0)
            frame.getContentPane().setBackground(Color.red);
        else
            frame.getContentPane().setBackground(Color.white);
    }
}

public class ComponentExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("ComponentExample");
        JButton button = new JButton("press me");
        JPanel jp = new JPanel();
        jp.setBackground(Color.white);

        // set the content pane to be the newly created JPanel
        frame.setContentPane(jp);

        frame.getContentPane().add(button);

        // register an event handler for frame events
        frame.addWindowListener(new MyWindowListener());

        // register an event handler for button events
        button.addActionListener(new MyActionListener(frame));

        frame.setSize(400, 400);
        frame.setVisible(true);
    }
}
```

2. Add a second button which when pressed sets the background of the main frame to yellow.

   Hint: Look at the `Color` class in the Java API.

3. Modify the code of the previous subquestion so that it uses only one `MyActionListener` class. The single listener class should differentiate the task based on which button generated the event (**Hint**: modify the given code of the `actionPerformed()` method so that it uses the `getSource()` method of the `ActionEvent` class).

3 Layout Managers

The purpose of this exercise is to familiarise yourselves with different layout managers.

1. Create a Java Swing program and replace the content pane of the application frame with a `JPanel` which follows the `BorderLayout` manager. Add 4 buttons in the panel.

   Resize the frame of the application and see what happens.

2. Do the same as above but use the `BoxLayout` manager following the x-direction.

   Resize the frame of the application and see what happens.

3. Do the same as above but use the `BoxLayout` manager following the y-direction.

   Resize the frame of the application and see what happens.

4. Do the same as above but use the `FlowLayout` manager.

   Resize the frame of the application and see what happens.

5. Do the same as above but use the `GridLayout` manager. The first 2 buttons should be in the first row and the last 2 button in the second row.

   Resize the frame of the application and see what happens.

4 JLabel and JTextField

1. What does the following code do? Make sure you understand fully every single line of the code.

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

// window event Handler class
class MyWindowListener extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
        System.out.println("Closing window!");
        System.exit(0);
    }
}

// textfield event handler class
class MyActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        System.out.println("You entered: " + e.getActionCommand());
    }
}

public class LabelFieldExample {
    public static void main(String[] args) {
```

```
        JFrame frame = new JFrame("JLabel and JTextField Example");

        JLabel label = new JLabel("Enter your name: ");
        // create a field with 25 chars width
        JTextField field = new JTextField(25);

        // put components next to each other in the x-direction
        Container c = frame.getContentPane();
        c.setLayout(new BoxLayout(c, BoxLayout.X_AXIS));

        // add label and field in the frame
        c.add(label);
        c.add(field);

        // register an event handler for frame events
        frame.addWindowListener(new MyWindowListener());

        // register an event handler for button events
        field.addActionListener(new MyActionListener());

        //frame.setSize(400, 400);
        frame.pack();
        frame.setVisible(true);
    }
}
```

2. Add a second textfield in the above GUI. When `enter` is pressed in the second textfield, its contents should be displayed followed by the string `Enter pressed!`.

5 Creating Professionally Looking Layouts

What does the following code do? Make sure you understand fully every single line of the code.

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

// window event Handler class
class MyWindowListener extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
        System.out.println("Closing window!");
        System.exit(0);
    }
}

// radio event handler class
class MyActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
```

```
        System.out.println("Selected: " + e.getActionCommand());
    }
}

// checkboxes event handler class
class MyCheckBoxListener implements ItemListener {
    public void itemStateChanged(ItemEvent e) {
        JCheckBox chk = (JCheckBox) e.getItem();
        String label = chk.getText();

        if (e.getStateChange() == e.SELECTED)
            System.out.println(label + " selected");
        else
            System.out.println(label + " de-selected");
    }
}


public class ButtonGroupExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Radio Buttons and CheckBoxes Example");

        // create a JPanel to hold the checkboxes
        JPanel topPanel = new JPanel();
        // JPanel has BoxLayout in x-direction
        topPanel.setLayout(new BoxLayout(topPanel, BoxLayout.X_AXIS));

        JCheckBox chk1 = new JCheckBox("GPS");
        JCheckBox chk2 = new JCheckBox("Alloys");
        JCheckBox chk3 = new JCheckBox("Power Steering");
        JCheckBox chk4 = new JCheckBox("Convertible");
        JLabel label = new JLabel("Extras: ");

        // add label and checkboxes in JPanel;
        topPanel.add(label);
        topPanel.add(chk1);
        topPanel.add(chk2);
        topPanel.add(chk3);
        topPanel.add(chk4);

        // create a JPanel to hold the checkboxes
        JPanel leftPanel = new JPanel();
        // JPanel has BoxLayout in x-direction
        leftPanel.setLayout(new BoxLayout(leftPanel, BoxLayout.Y_AXIS));

        JRadioButton rd1 = new JRadioButton("CD Player");
        JRadioButton rd2 = new JRadioButton("DVD Player");
        JRadioButton rd3 = new JRadioButton("Cassette Player");
```

```
        // group radio buttons together
        ButtonGroup group = new ButtonGroup();
        group.add(rd1);
        group.add(rd2);
        group.add(rd3);

        // add radio buttons in the JPanel
        leftPanel.add(rd1);
        leftPanel.add(rd2);
        leftPanel.add(rd3);

        // add JLabels in the frame
        frame.getContentPane().add(topPanel, "North");
        frame.getContentPane().add(leftPanel, "West");

        // register an event handler for frame events
        frame.addWindowListener(new MyWindowListener());

        // register an event handler for checkboxes
        MyCheckBoxListener chkListener = new MyCheckBoxListener();
        chk1.addItemListener(chkListener);
        chk2.addItemListener(chkListener);
        chk3.addItemListener(chkListener);
        chk4.addItemListener(chkListener);

        // register an event handler for radio buttons
        ActionListener radioListener = new MyActionListener();
        rd1.addActionListener(radioListener);
        rd2.addActionListener(radioListener);
        rd3.addActionListener(radioListener);

        frame.setSize(400, 400);
        //frame.pack();
        frame.setVisible(true);
    }
}
```

6 Choosing a Colour

Write a Java Swing program which the user can use to choose any colour he/she would like based on preselected colours and RGB (reg/green/blue) chosen components. As soon as the user selects a colour and clicks on a button, the background of the application changes to that colour.

Hint: The `JColorChooser` swing component should be used. Study the documentation and the link given to the Java online tutorial for its usage.

7 Displaying Images

Write a Java Swing program which asks the user to select an image which is contained in a file. As soon as the user types/chooses the corresponding file, the image in that file is displayed to the user.

Add scrollbars to the application or the area displaying the image.

8 Working with Files

1. Study the following example which introduces you in how to write to and read from a file some text data. Make sure that you understand the code fully.

```
import java.io.*;
import java.util.Scanner;

public class FileExample {
    File fp = new File("my_data.txt");

    // do some file writing
    void write() {
        try {
            PrintWriter pw = new PrintWriter(fp);

            pw.print(1 + " ");
            pw.print(2);
            pw.println(3);
            pw.print(4 + "\n");
            pw.close();
        }
        catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    // do some file reading
    void read() {
        Scanner sc = null;
        try {
            sc = new Scanner(fp);
            while (sc.hasNext()) {
                int i = sc.nextInt();
                System.out.println(i);
            }
        }
        catch (FileNotFoundException ex) {
            System.err.println("Exception: " + ex);
        }
        finally {
```

```
            if (sc != null)
                sc.close();
        }
    }

    public static void main(String[] args) {
        FileExample fileTesting = new FileExample();
        fileTesting.write();
        fileTesting.read();
    }
}
```

2. Which numbers are read back from method `read`? $1, 2, 3, 4$ or $1, 23, 4$? Justify your answer and observe the output of the program to verify it.
3. Study from the Java API documentation, class `Scanner`. What other useful methods it provides for reading?
4. Modify the previous example, so that the program writes and reads from a file a series of strings instead of just integers.
5. Modify the previous example, so that the program writes and reads a mixture of integers and strings to/from the same single file.

9 Serialisation of Objects

When an object is *serialised*, then it is converted to a format which can be used to write the object to a file or transmit it over the network.

- To be able to perform serialisation the class that the object belongs to must implement the `java.io.Serializable+ interface`:

To write an object of class `Date` to a file:

```
// first create the file
FileOutputStream fos = new FileOutputStream("serial.bin");
ObjectOutputStream oos = new ObjectOutputStream(fos);

// create a Date and write it to the file
java.util.Date d = new java.util.Date();
oos.writeObject(d);
```

To read the object back from the file:

```
FileInputStream fis = new FileInputStream("serial.bin");
ObjectInputStream ois = new ObjectInputStream(fis);

Object o = ois.readObject();
// cast it back to the original object type
java.util.Date savedDate = (java.util.Date) o;
```

Note that only instance data are being serialised, i.e. the values of static data are not saved...

- Run the following code and make sure that you understand the source code. Ask you tutor for anything that might not be clear. What does the program do?

```
import java.io.*;

public class SerialisationExample {
    public static void main(String[] args) {
        try {
            // first create the file
            FileOutputStream fos = new FileOutputStream("serial.bin");
            ObjectOutputStream oos = new ObjectOutputStream(fos);

            // create a Date and write it to the file
            java.util.Date d = new java.util.Date();
            System.out.println("Saving into file date object: " + d);
            oos.writeObject(d);

            // read back the object from the file
            FileInputStream fis = new FileInputStream("serial.bin");
            ObjectInputStream ois = new ObjectInputStream(fis);

            Object o = ois.readObject();
            // cast it back to the original object type
            java.util.Date savedDate = (java.util.Date) o;

            // print the object read from the file
            System.out.println("Date object read from file: " + o);
        }
        catch (IOException ioex) {
            ioex.printStackTrace();
        }
        catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```