COIT 11222 – Visual Programming

Tutorials: Week 8

Author: Mike O’Malley (Course Coordinator)

Please Note:

• Keep all of your attempts at all questions in well named files (e.g. Week 1, Question 2, a good file name would be "W01_Q2_Hello_World.java").

• The tutorial questions in this course are often based on the questions for prior weeks. In other words, to save you time and effort, you will use your prior week's attempt(s) at question(s) as a starting point for questions in future weeks.

• Tutorial Solutions may be provided below. Only look at solutions when you have made good attempts at the questions - otherwise you will NOT learn!

• This document is an encrypted PDF, which means students can view and print it, but they cannot copy and paste the answers into TextPad. In other words, if you want to copy the solutions, you will need to type them in, which at least means you will be learning something and fixing the typos you make, rather than just copying and pasting my hard work and learning nothing.

Questions

1. Applet #1 (Arrays Driving Applications): Change the Example 1 Applet from this week's lecture to use buttons instead of radio buttons. Review the code carefully to make sure you know what is going on and why.

   • Estimated Amount of Work to do this: 30 mins.
   • This will help you understand using arrays and parallel arrays to drive applications will therefore help you with Assignment 2 !! (Even though it is for an Applet and NOT a Windowed Application, but the principles are still the same).

2. Applet #2 (Arrays Driving Applications): Change the Example 2 Applet from this week's lecture (or adapt your attempt at the above question) to use buttons instead of radio buttons.

   • Estimated Amount of Work to do this: 15 mins.
   • This will help you understand using arrays and parallel arrays to drive applications will therefore help you with Assignment 2 !! (Even though it is for an Applet and NOT a Windowed Application, but the principles are still the same).
3. **Applet #3 (Arrays Driving Applications):** Combine Example 2 Applet from this week's lecture and your attempt at Question 2 above, so that the radio buttons are in the top half of the screen and the buttons are in the bottom half of the screen.

Here is what we are trying to achieve:

- **Estimated Amount of Work to do this:** 15 mins.
- **Hint:** Create 2 GridLayout panels, add radio buttons to one and the buttons to the second, and then add the panels to the user interface.
- **Hint #2:** It may be best to also have a 2 rows x 1 column GridLayout panel for the user interface, so you can add the GridPanels containing the components to each cell of the grid.
- This will help you understand Layout Managers, Panels, etc and will therefore help you with Assignment 2!! (Even though it is for an Applet and NOT a Windowed Application, but the principles are still the same).

4. **Applet #4 (Assignment 2):** Create an Applet version of the Assignment 2 for this term. You should have already created the user interface layout (in last week’s tutorial). So, what you need to do now is add the parallel arrays, use these to create the required GUI components, add any other GUI components you need (labels, etc), activate the GUI components (that need to react to mouse clicks), and write the code to do the required processing and generate the required output.

- **Estimated Amount of Work to do this:** 2 - 4 hours (approx).
- **Even though this is an Applet and NOT a Windowed Application, it will still help you enormously with Assignment 2.**
- The amount of work to convert an Applet to a Windowed Application is very small (which we cover next week), then all you will need to do is add and activate the menus (which is also a small amount of work and which we also cover next week).
Tutorial Solutions

Please Note:

- ATTENTION: SOLUTIONS ARE BELOW !!!
- Only look at these solutions when you have made good attempts at the questions - otherwise you will NOT learn!

1. Applet #1 (Arrays Driving Applications):

- The code below is based on Example 1, which was presented in this week’s lecture;
- All changes below are highlighted in YELLOW.

```java
/*
Title:         W08_Animals__Buttons_FlowLayout__Applet
Author:        Mike O'Malley
Java version:  All versions.
Description:   An example to explore the use of parallel arrays for driving the application by defining buttons.
*/

import java.awt.*;               // For GUI components
import java.awt.event.*;         // For event handling
import javax.swing.JOptionPane;  // For Swing Dialogs
import java.text.DecimalFormat;  // For DecimalFormat.
import javax.swing.*;            // For Swing components, frames, etc.
import java.applet.Applet;
import java.awt.event.ActionEvent;

public class W08_Animals__Buttons_FlowLayout__Applet  extends Applet  implements ActionListener
{
    // ***********************
    // *** Class Constants ***
    // ***********************

    // Note: These arrays drive this application - they are *VERY* important !!

    String ANIMALS_LIST [] = {"Ant", "Elephant", "Gorilla", "Frog", "Magpie",
                            "Great White Shark", "Mid Eastern Red Bellied Parrot");

    double ANIMAL_LIFE_SPANS [] // Must contain a value for every item in ANIMALS_LIST.
                                = {0.1,    100.0,      65.0,     2.0,    8.0,
                                 10.0,   4.3,        3.7,      43.0,
                                 29.5,               30.0};

    double ANIMAL_WEIGHTS []    // Must contain a value for every item in ANIMALS_LIST.
                                = {0.001,   5000.0,    200.0,    0.25,   0.5,
                                 30.0,    4.3,        3.3,     2300.0,
                                 3000.0,            0.3};

    final int    NUM_COLUMNS             = 4;

    // ******************
```
// *** Class Data ***
// ******************
// Declare GUI Components

public void init ()
{
    // Define the Layout.
    this.setLayout (new FlowLayout (FlowLayout.CENTER));

    for (int i = 0; i < ANIMALS_LIST.length; i++)
    {
        // *** Buttons:
        // Create the button and assign it a name.
        Animal_Button [i] = new Button (ANIMALS_LIST [i]);

        // Make the button active (for mouse clicks).
        Animal_Button [i].addActionListener (this);

        // Add the button to the grid on the user interface.
        add (Animal_Button [i]);
    }
}

private void Display_Selected_Animal (int Selected_Animal)
{
    // Display details of selected Animal.
    if (Selected_Animal >= 0)
    {
        JOptionPane.showMessageDialog (null,
"A "
+ ANIMALS_LIST [Selected_Animal]
+ " lives for "
+ ANIMAL_LIFE_SPANS [Selected_Animal]
+ " years, and weighs "
+ ANIMAL_WEIGHTS [Selected_Animal]
+ " Kg."
, ANIMALS_LIST [Selected_Animal] + " Information:",
JOptionPane.INFORMATION_MESSAGE);
    }
}

public void actionPerformed (ActionEvent e)
{
    int i                = 0;
    boolean Item_Found   = false;

    // Determine which Button has been clicked.
    while ((i < ANIMALS_LIST.length) && (Item_Found == false))
    {
        // If the a button's label matches an item in the array ...
        if (e.getActionCommand() == ANIMALS_LIST [i])
        {
            Display_Selected_Animal (i);
            Item_Found = true;
        }
        i++;
    }
}

// *** Class Methods ***
// *********************

// Constructor - initialise all class data and set-up the GUI components / Frame.
2. **Applet #2 (Arrays Driving Applications):**
   - No solution provided.
   - The examples presented in the lecture and the solutions already provided above cover all you need to answer this question.

3. **Applet #3 (Arrays Driving Applications):**
   - No solution provided.
   - The examples presented in the lecture and the solutions already provided above cover all you need to answer this question.

4. **Applet #4 (Assignment 2):**
   - No solution provided – this is directly related to the Assignment.
   - The examples presented in the lecture and the solutions already provided above cover all you need to answer this question.