

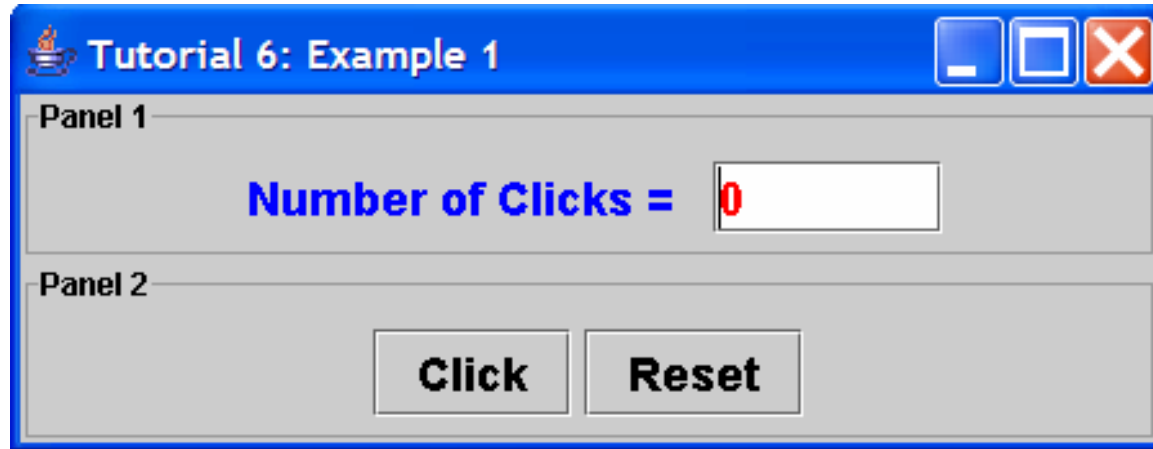
1.00/1.001 Tutorial 6

October 17 & 18, 2005

Topics

- Swing
 - Basic Concepts
 - Layouts
- Problem Set 5
 - JList
 - MVC paradigm

Basic Concepts

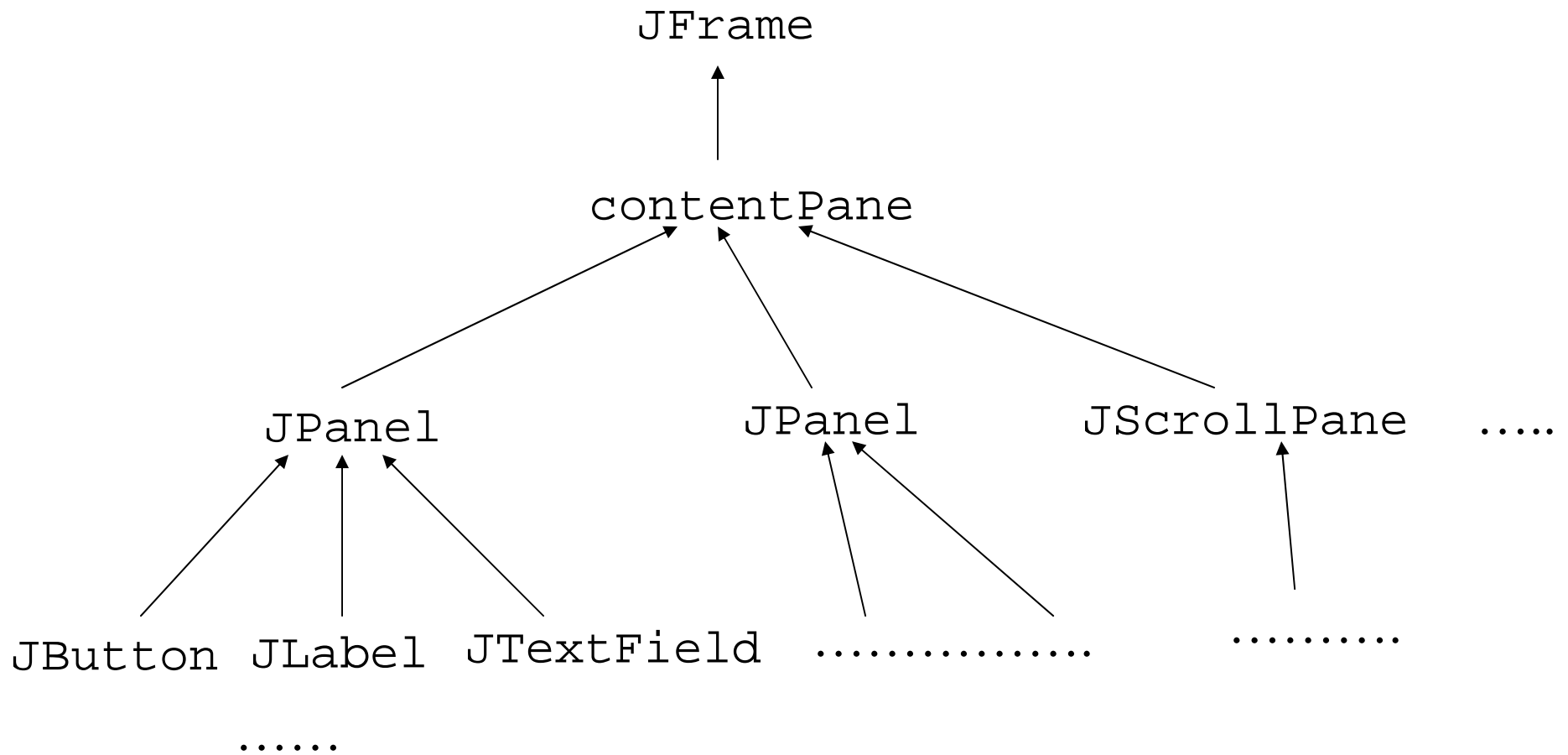


- Identify the top level window
- Identify the containers
- Identify the components

Basic Concepts

- **Top Level Window**
 - JFrame
- **Containers**
 - JPanel (Panel 1, Panel 2)
- **Components**
 - JLabel, JTextField
 - JButton

Composition of a Swing GUI



Review - Swing Toolkit

- **Top Level Windows** (`JFrame`, `JDialog`)
 - Not contained in other containers
 - Interact with native windowing system
- **Containers** (`ContentPane`, `JPanel`, `JScrollPane`)
 - Hold components
 - The `ContentPane` is of type `Container`: it is not a class.
- **JComponents** (`JButton`, `JLabel`)
 - Present GUI elements
 - Interact with user

Swing - Layouts

- `java.awt.BorderLayout`
 - Default Layout for `JFrame`
 - Has 5 areas: NORTH, SOUTH, EAST, WEST, CENTER
 - Each area can hold ONE component only
 - If you need to add more than one component to an area of a `BorderLayout`, add the components to a `JPanel` and add the `JPanel` to the desired area.

Swing - Layouts

- `java.awt.FlowLayout`
 - Default Layout for `JPanel`
 - Components are added in order from left to right and centered; contents wrap with the size of the container. (Similar to word wrapping in text editors)

Swing - Layouts

- `javax.swing.BoxLayout`
 - Allows items to be laid out in one direction only, e.g. horizontally (`BoxLayout.X_AXIS`) or vertically (`BoxLayout.Y_AXIS`)
 - Constructor takes the layout's container as an argument, e.g.

```
JPanel myPanel = new JPanel();
```

```
BoxLayout myLayout = new
```

```
    BoxLayout(myPanel, BoxLayout.X_AXIS);
```

```
myPanel.setLayout(myLayout);
```

Swing - Layouts

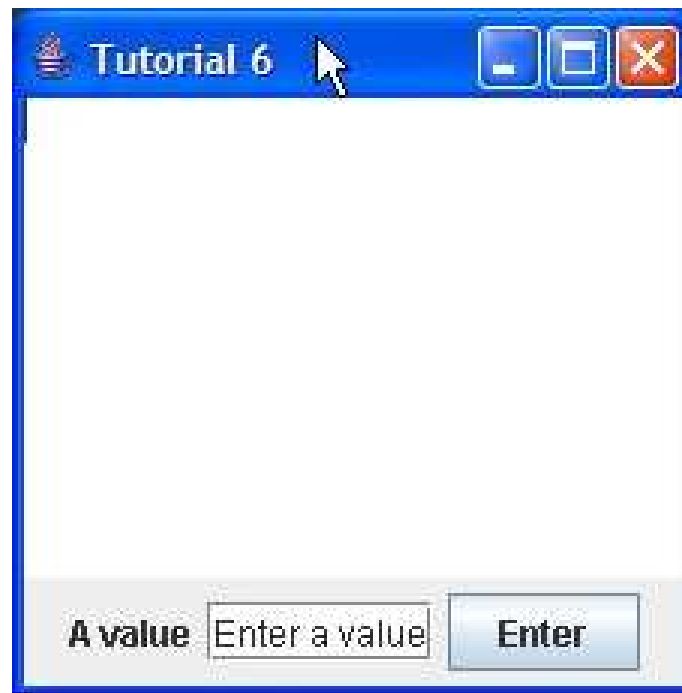
- `javax.swing.BoxLayout`
 - For more information, see <http://java.sun.com/docs/books/tutorial/uiswing/layout/box.html>.

Useful Classes

- `java.awt.Color`
 - Default colors: `Color.RED`
 - Create your own:
`Color unknown = new Color(255,27,100)`
- `java.awt.Font`
 - Try to use logical fonts to avoid computer-dependent problems. Logical fonts: `Serif`, `SansSerif`, `Monospaced`, `Dialog`, and `DialogInput`
- `javax.swing.Box`
 - Can create components like spacers (use static methods)
 - Very useful with `BoxLayout`
- See javadoc for more information on these and other classes

GUI Example

- Now we'll build the simple non-functioning GUI shown below.



GUI Example cont'd

- Create a new class called `Tutorial6.java`. This class should inherit from `JFrame`. Also, make sure there is a `main()` method in this class.
- Remember that the visual components (`JFrame`, `JLabel`, etc.) we use are found in the `javax.swing` package and that some useful classes like `BorderLayout` are in the `java.awt` package.

GUI Example cont'd

- Write a constructor for `Tutorial6` with no arguments that:
 - Makes the title of the `JFrame` read "Tutorial 6". [Hint: `JFrame` has a constructor which takes a `String` as an argument and makes that string the title of the `JFrame`.]

GUI Example cont'd

- Write a constructor for `Tutorial6` with no arguments that:
 - Places a `JTextArea` in the center of the `BorderLayout`. `JTextArea` has a constructor that takes two `int` values, one being the number of rows and the other the number of columns of text. Let this component have 10 rows and 20 columns of text.

GUI Example cont'd

- Write a constructor for `Tutorial6` with no arguments that:
 - Places a `JLabel` with the text "A value", `JTextField` with the default text "Enter a value" and a `JButton` with the text "Enter" in the south of the `BorderLayout`. Recall that only one component is allowed in search section of a `BorderLayout`. How do we add all of these items? [Hint: you need a container.]

GUI Example cont'd

- Write a constructor for `Tutorial6` with no arguments that:
 - Calls `pack()` to ensure the window has reasonable dimensions.
 - Sets the default close operation to `JFrame.EXIT_ON_CLOSE`.
- In the main method
 - Create a new `Tutorial6`.
 - Make the window visible.

GUI Example cont'd

- Play around a little with window sizing and entering values in the text components (`JTextArea`, `JTextField`)
- Now let's see what happens if we don't use a `JPanel` to hold multiple components within a portion of a `BorderLayout`.
 - Add two `JButtons` with the text “silly1” and “silly2” to the west of the `BorderLayout`.
 - What happens?

Problem Set 5

- In problem sets 5 and 6 you will build a GUI for recording TVPrograms.
- Builds on problem set 4
 - Make sure you use code that works
- Problem set 5 involves
 - Creating the graphical interface
 - Completing all code required for computation

JList

- `javax.swing.JList`
 - Allows user to select one or more objects from a list
 - By default, the list's contents can't change
 - If you want to be able to add or remove items from the list, you need to change the `ListModel` data member of the `JList`. More on this for problem set 6; this feature is not needed for problem set 5.
 - You can easily construct an empty list:

```
JList myList = new JList();
```
 - For more information, see javadoc or the Java tutorial (<http://java.sun.com/docs/books/tutorial/uiswing/components/list.html>)

MVC

- How do you structure programs so that the organization is efficient and sensible? In other words, how do you determine the best places to put the different code fragments needed for a program?
- MVC or Model-View-Controller is one strategy for organizing your program. So what does MVC mean?

MVC explained

- Model
 - Computational code is written in the model.
 - Often, you may need to do several calculations with the same information. Also, one calculation may depend on the outcome of another. Having all the computational code in one class allows minimum external access to data and minimum method calls.
 - There may be more than one model, if some data is independent of the rest

MVC explained

- View
 - This are the graphical interface of the program – the visible part of the GUI
 - Usually, the view displays or uses some information computed in the model.
 - There may be more than one view (“skin”) for one program.

MVC explained

- Controller
 - This is the “master” class. The controller synchronizes calculation (the model) with the graphical display (the view) so that they work together.
 - The controller usually handles events, for example, pressing the “C” button of the calculator (See Lecture 15 for image) should clear the JLabel (part of the view) and also clear the data that has been stored so far (in the model)

MVC and Problem Set 5

- Usually, the model is a separate class, since many kinds of computation require a lot of code.
- The view and controller are sometimes a single class, for convenience.
- This program will use the `TIVO` class as the model (it contains all the data we need and also methods that manipulate that data). We will use the new `RecordingsView` class as the view. It will also be the controller but that's what PS6 is all about – Event handling!
- You'll see more on MVC later in the term.