THE UNIVERSITY OF AUCKLAND

COMPSCI 230

Software Design and Construction

GUI Concepts

2013-04-17

Recap: Framework

Generic software platform for a certain type of applications

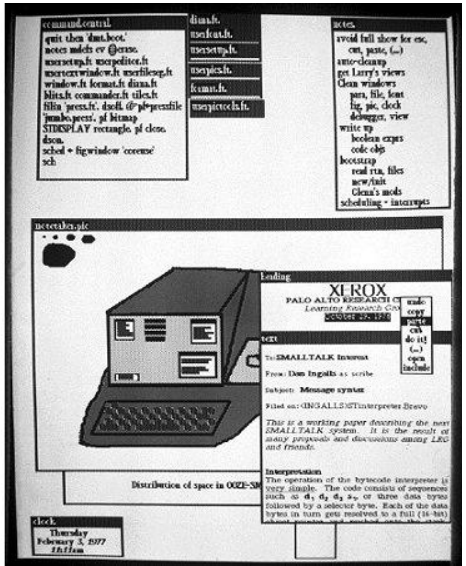
- Consists of **parts** that are found in many apps of that type
  - **Libraries** with APIs (classes with methods etc.)
  - Ready-made extensible programs ("**engines**")
  - Sometimes also **tools**
    (e.g. for development, configuration, content)
- Often evolved by developing many apps of that type and **reusing code** more and more

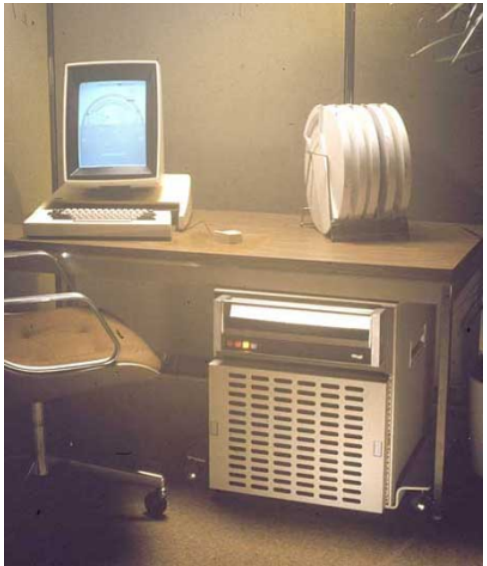
Characteristics:

- **Reusable**: the parts can be used for many apps of that type
- **Extensible**: developers can add their own app-specific code
- **Inversion of Control**: framework often calls your code

THE UNIVERSITY OF AUCKLAND

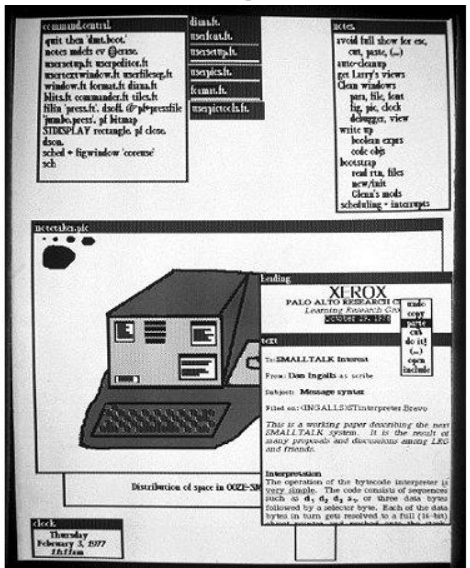
Graphical User Interface (GUI) Concepts

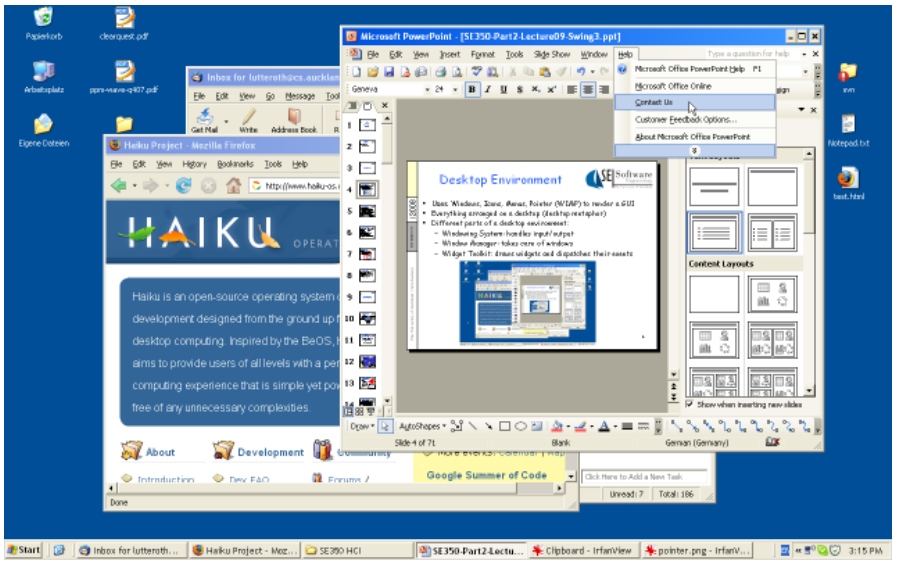




Desktop Environment

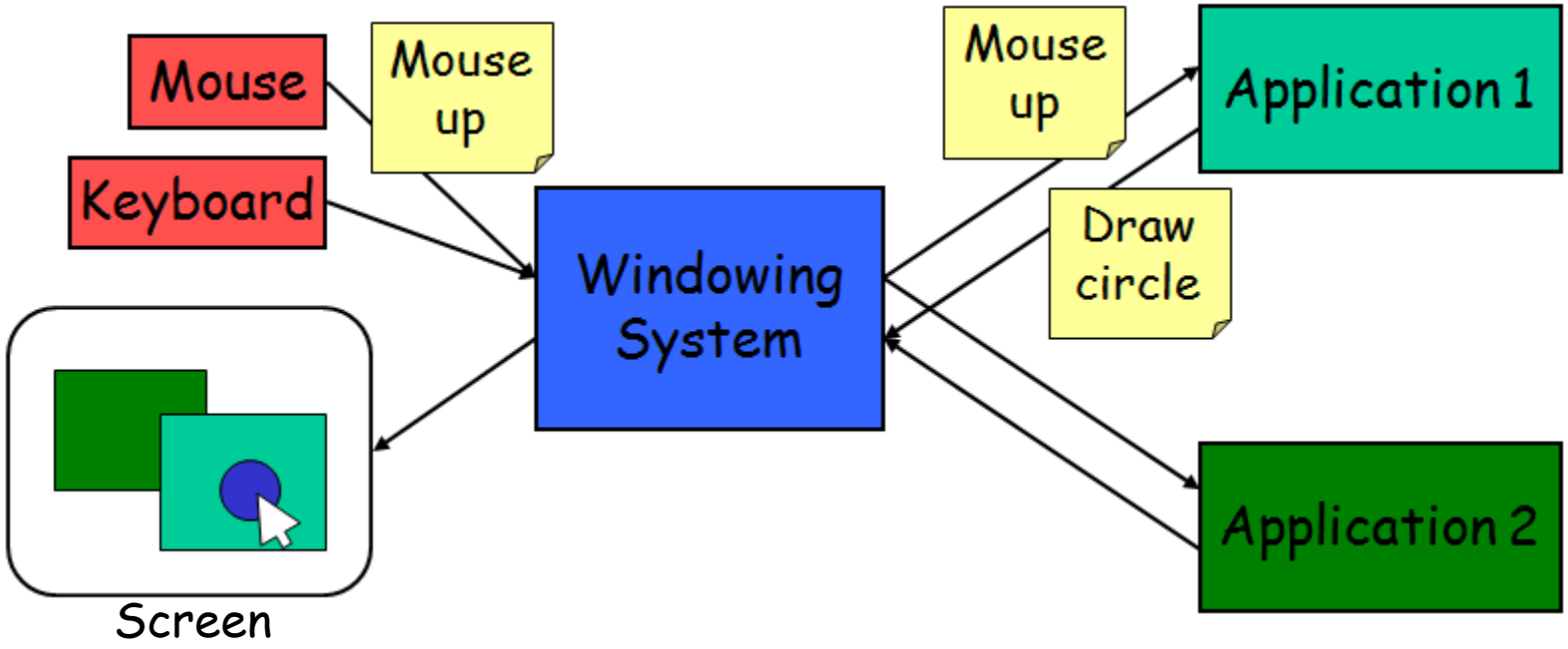
- Uses **W**indows, **I**cons, **M**enus, **P**ointer (WIMP) to render a GUI
- Everything arranged on a desktop (**desktop metaphor**)
- Different parts of a desktop environment:
  - **Windowing System**: handles input/output
  - **Window Manager**: takes care of windows
  - **GUI Framework** (aka. **GUI Toolkit**): draws widgets and dispatches their events





Windowing System

- Manages **input** and **output devices**: graphics cards, screens, mice, keyboards
- Sends **input events** from input devices to apps
- Receives and processes **drawing commands** from apps
- Often able to talk to remote applications: send input events and receive drawing commands over the network



GUI Input Events

Primitive Pointer Events

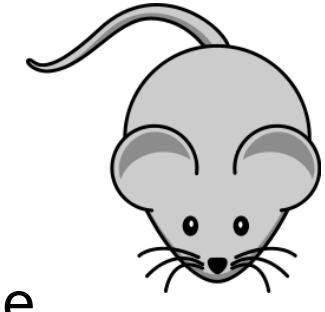
- Mouse Moved
- Mouse Down
- Mouse Up

Primitive Keyboard Events

- Key down
- Key up

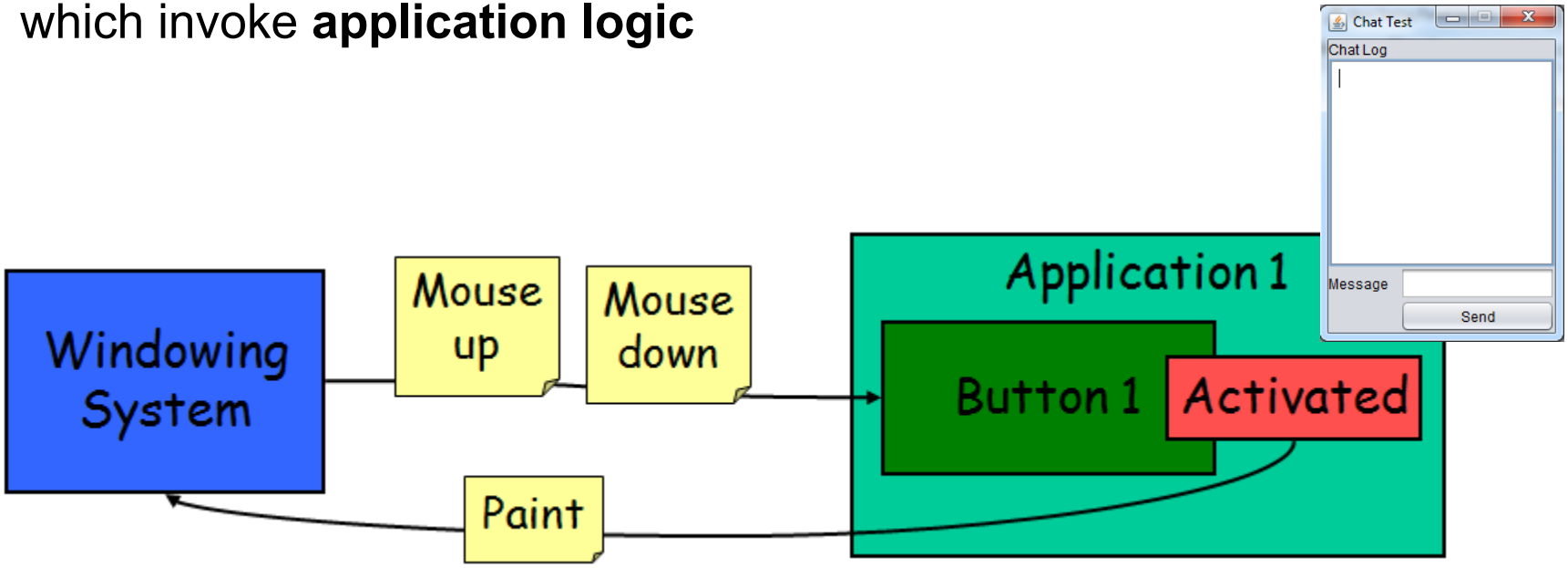
Complex Pointer Events

Click = mouse down, mouse up
Double Click = two clicks within a certain time
Enter = mouse moves into a region
Leave = mouse moves out of a region
Hover = mouse stays in region for a period of time
Drag and Drop = mouse down, mouse moved, mouse up



Input Handling in Widgets

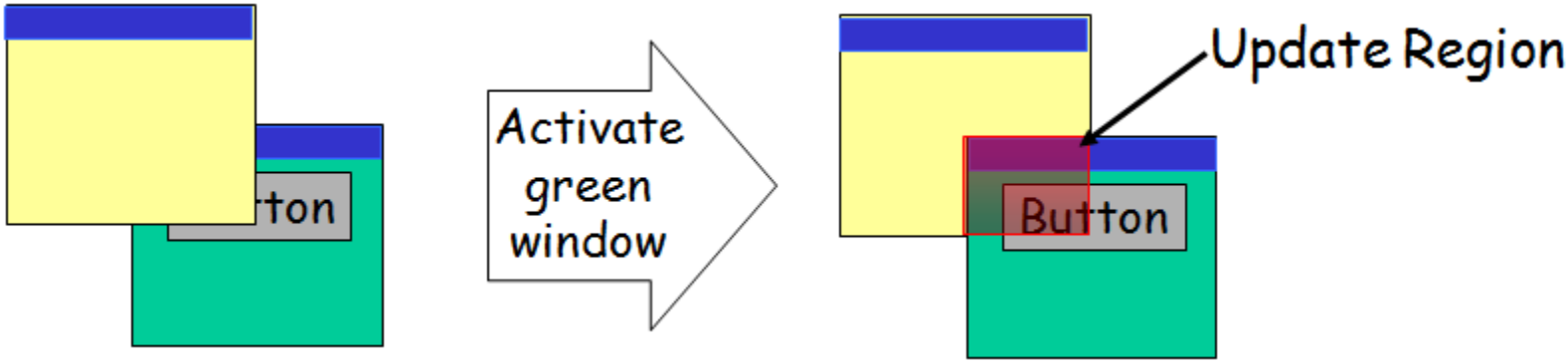
- **Input events** are dispatched to the right widgets by windowing system and GUI framework
- Keyboard events are sent to widget with **input focus** in active window
- Widgets have **event listeners** (aka. **handlers**) for input events; they can translate simple input events into more complex, specific ones (e.g. "Mouse down, mouse up" becomes "Button1 activated")
- Developers can set event listeners for widgets, which invoke **application logic**



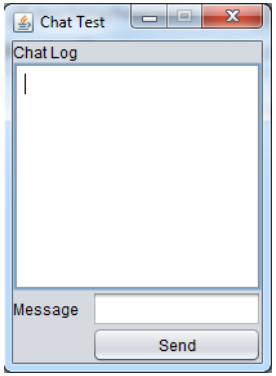
Rendering of Widgets

Widgets have a visual representation

- Widgets define "**paint**" event listener: draws the widget by sending commands to the windowing system
- Widget gets **paint events** (aka. "**update events**") from the windowing system (through GUI framework)
- Often not complete redrawing, but "**update region**"
- Application can send "**invalidate**" events to the windowing system if redrawing necessary (to trigger paint events)



The GUI Event Loop

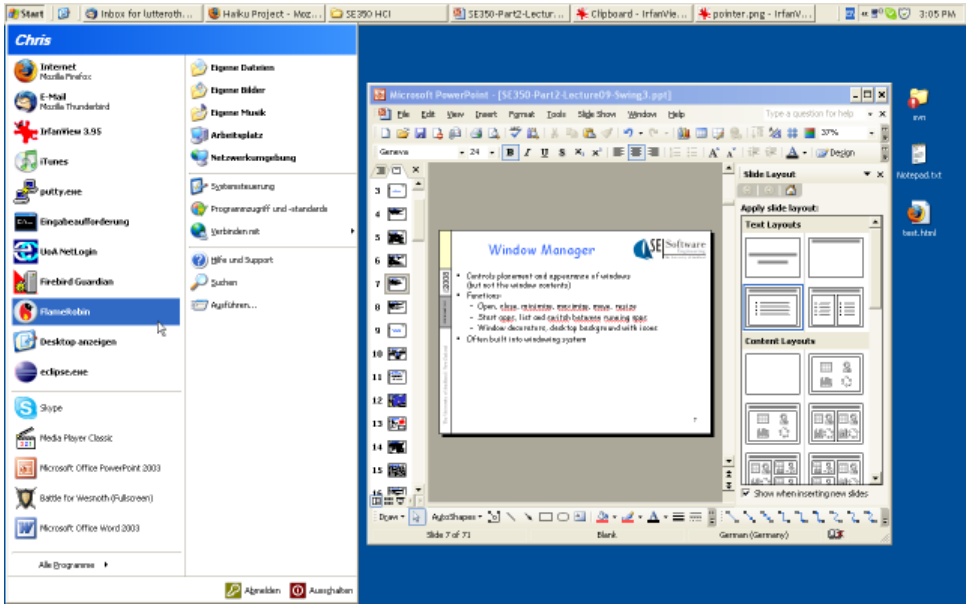


1. GUI application is started
2. Widgets are set up
3. Event loop is started
4. **Wait** for events from the windowing system (event queue)
5. **Dispatch** each event to the right widget
   a. **Input event**: call appropriate event listener (→ call to application logic)
   b. **Paint event**: call paint method
6. Go back to 4.

→ Event-Driven Programming

Window Manager

- Controls **placement and appearance of windows** (but not the window contents)
  - Open, close, minimize, maximize, move, resize
  - **Start apps**, list and switch between running apps
  - Window decorators, desktop background with icons
- Often built into windowing system
- Implemented using a GUI framework

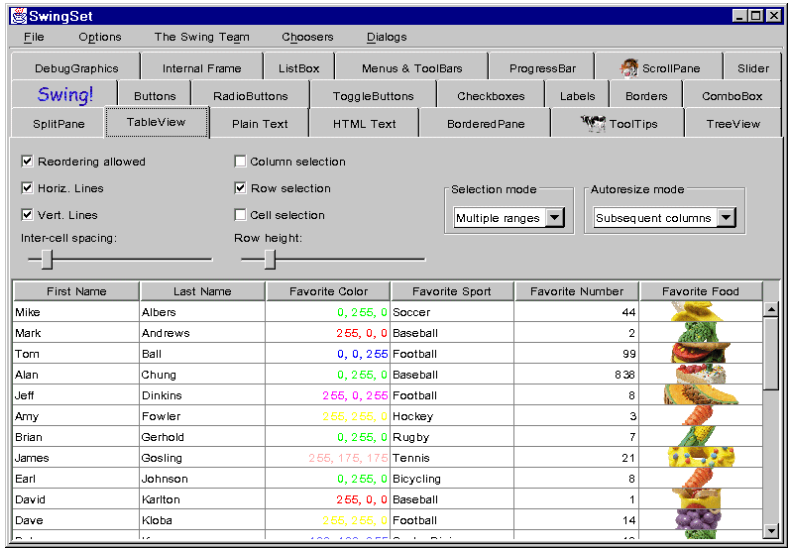


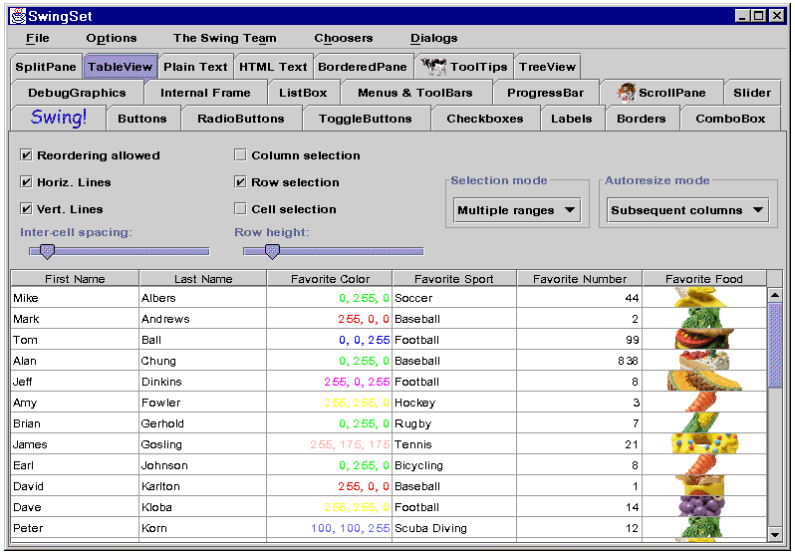
Introduction to Java Swing

THE UNIVERSITY OF AUCKLAND

Java Swing

- Standard GUI framework for Java
- Developed in the late 90s
- Based on an existing framework called AWT
- **AWT** is simply an API to the GUI framework of the operating system a Java program runs on
  - Widgets have "native" look
  - Can't change the look & feel
- Swing builds new widgets **on top of AWT**
- The **look & feel** can be changed

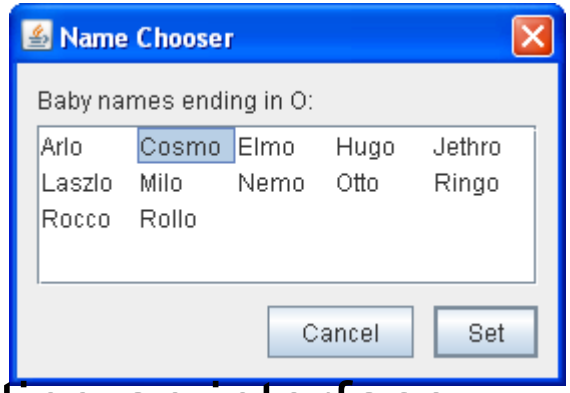




SWING DESIGN PRINCIPLES

1. GUI is built as **containment hierarchy** of widgets (i.e. the parent-child nesting relation between them)

2. Event objects and event listeners
   - **Event object**: is created when event occurs (e.g. click), contains additional info (e.g. mouse coordinates)
   - **Event listener**: object implementing an interface with an event handler method that gets an event object as argument

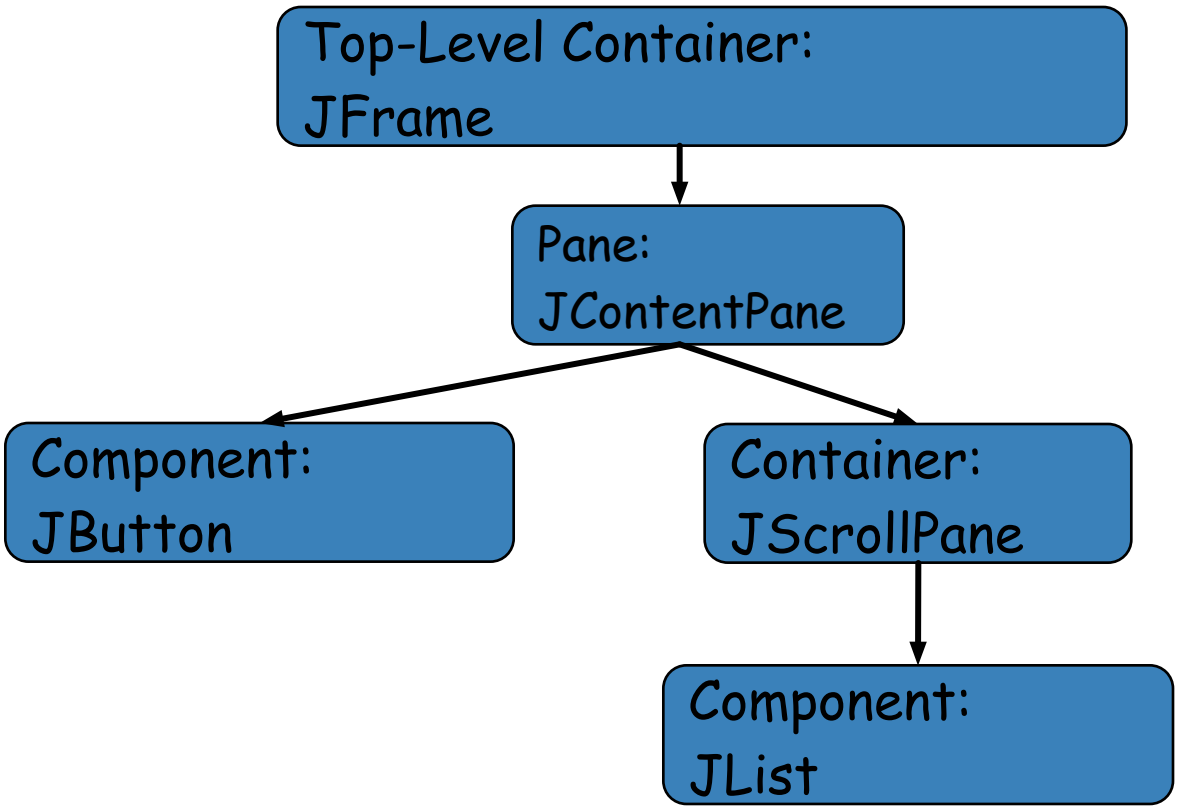


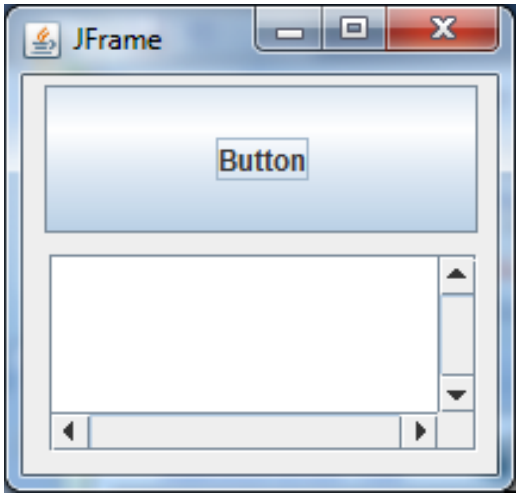
3. Separation of Model and View:
   - **Model**: the data that is presented by a widget
   - **View**: the actual presentation on the screen

Containment Hierarchy

THE UNIVERSITY OF AUCKLAND

- Most UI are created by nesting widgets into other widgets (containers)
- **Containment hierarchy**: the way the widgets of a UI are nested
- Not all controls visible; often invisible internal containers





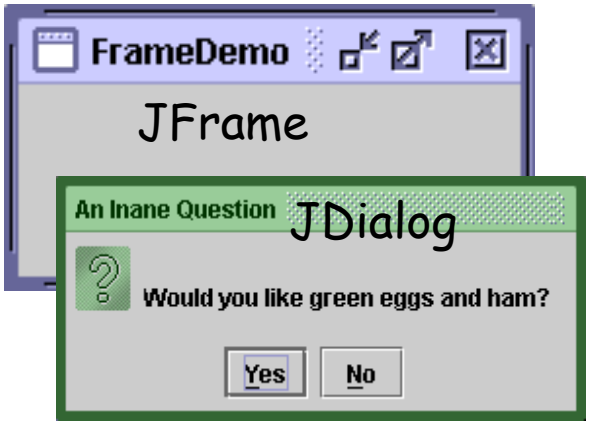
Java Swing Example GUI with containment hierarchy

*

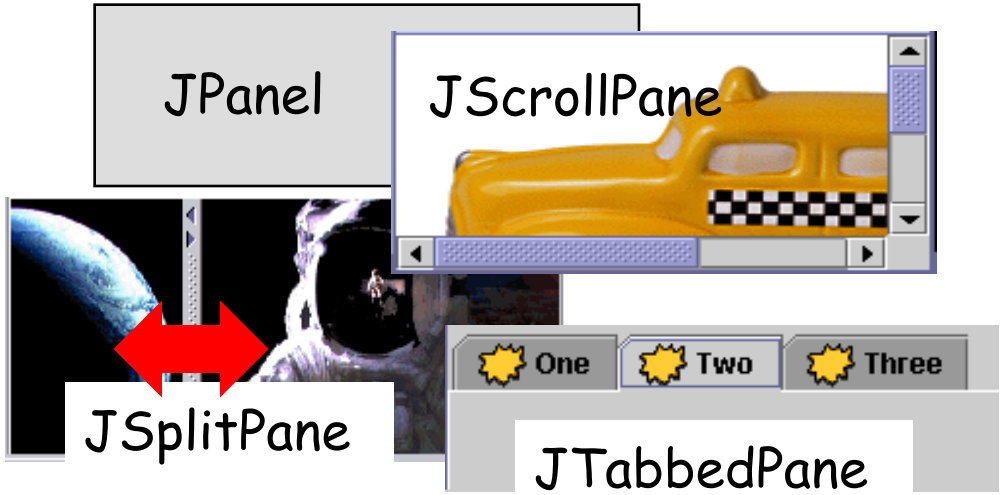
Swing Widgets

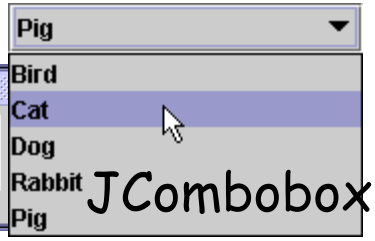
THE UNIVERSITY OF AUCKLAND

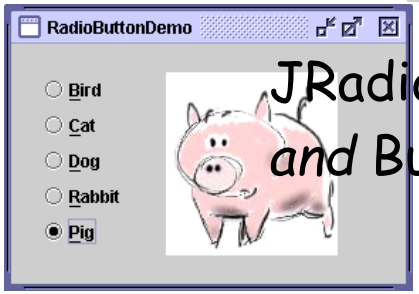
Top-Level Containers

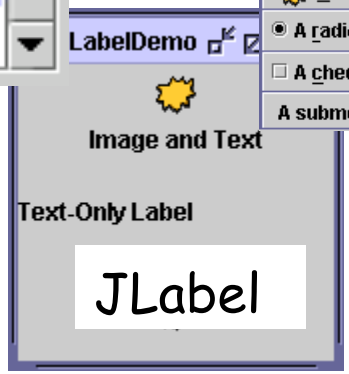


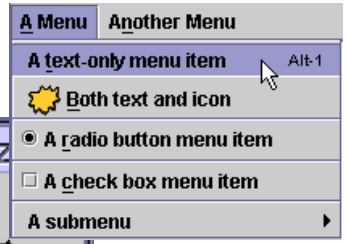
General-Purpose Containers



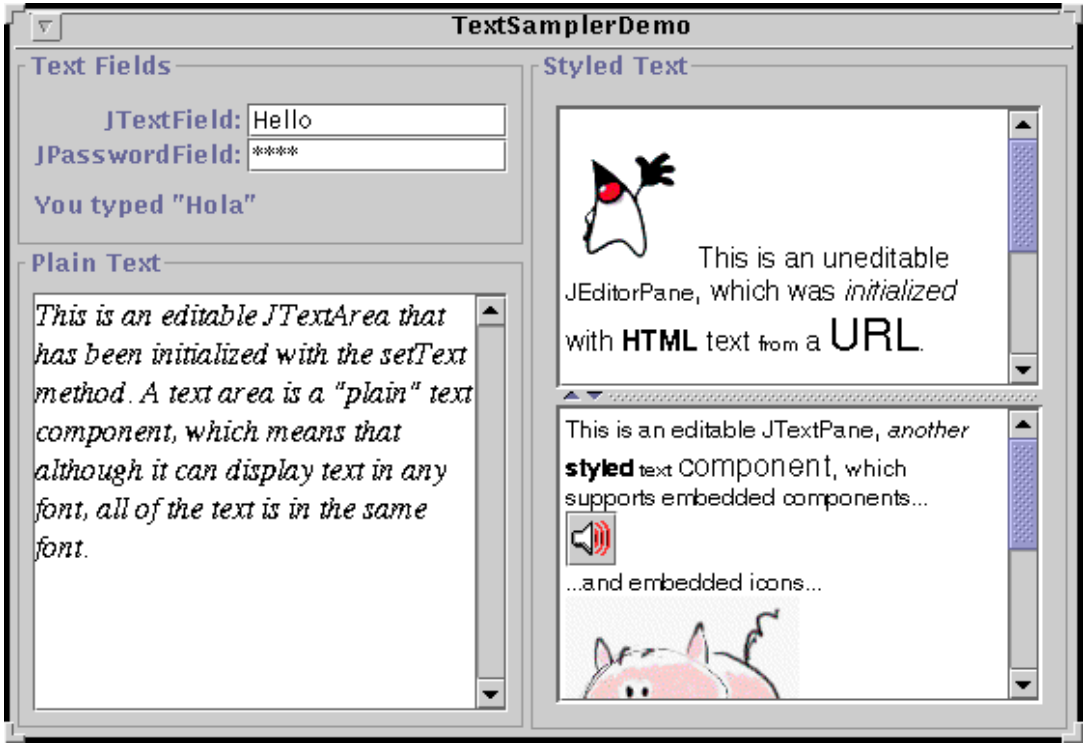


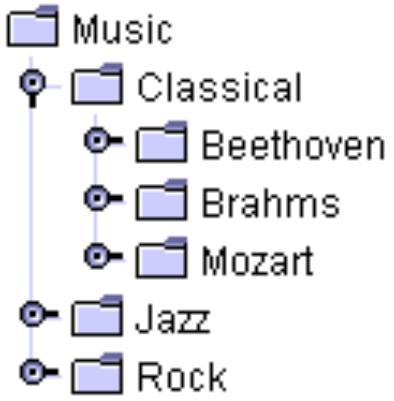






More Swing Widgets

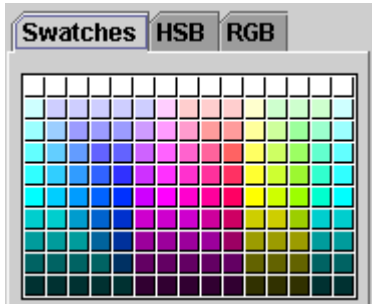




JTree

First Name	Last Name	Favorite Food
Jeff	Dinkins	
Ewan	Dinkins	
Amy	Fowler	
Hania	Gajewska	
David	Geary	

JTable



JColorChooser

Swing Hello World

THE UNIVERSITY OF AUCKLAND

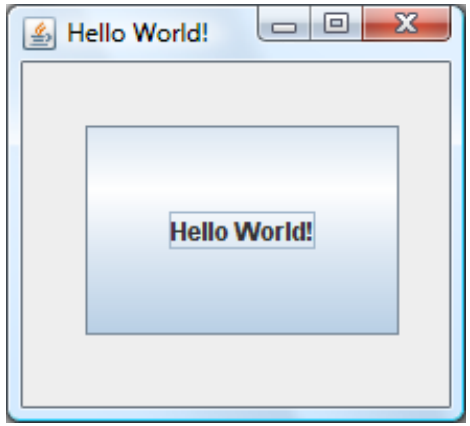
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class HelloWorld {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Hello World!");
        frame.setSize(220, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Container contentPane = frame.getContentPane();
        contentPane.setLayout(null);

        JButton button = new JButton("Hello World!");
        button.setLocation(30, 30);
        button.setSize(150, 100);
        contentPane.add(button);

        frame.setVisible(true);
    }
}
```



Swing Hello World with Events

THE UNIVERSITY OF AUCKLAND

```
...
public class HelloWorld {
   public static void main(String[] args) {
      ...
      JButton button = new JButton("Hello World!");
      button.addActionListener(new MyActionListener());
      ...
   }
}
```

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

public class MyActionListener implements ActionListener {
   public void actionPerformed(ActionEvent e) {
      Toolkit.getDefaultToolkit().beep();
   }
}
```

Summary

- **Desktop environments** consist of:
  - **Windowing System**: handles input/output
  - **Widget Toolkit**:
    draws widgets and dispatches their events
  - **Window Manager**: takes care of windows

- **Swing** is a GUI toolkit for Java
  - GUI as containment hierarchy of widgets
  - Event objects and event listeners

References:

http://java.sun.com/docs/books/tutorial/uiswing/
http://www.javabeginner.com/java-swing-tutorial.htm

Quiz

1. What does a windowing system do with input events?
2. How can applications draw their widgets?
3. What is the containment hierarchy of a GUI?
4. What is an event listener?