



# HCI and Design

---

SPRING 2016

# Topics for today

---

- User interface toolkits
- Next lecture: Design theory
  - Three week component of the class
  - Real Live Designers from Google
  - Visual design
  - Interaction design

# User Interface Software

---

What support is provided for building graphical user interfaces?

- UI toolkits
- GUI builder tools

Let's examine some background...

# In the beginning...

```
bash-2.05b$ pwd
/home/dstone
bash-2.05b$ cd /usr/portage/app-shells/bash
bash-2.05b$ ls -al
total 68
drwxr-xr-x  3 root root  4096 May 14 12:05 .
drwxr-xr-x 26 root root  4096 May 17 02:36 ..
-rw-r--r--  1 root root 13710 May  3 22:35 ChangeLog
-rw-r--r--  1 root root  2924 May 14 12:05 Manifest
-rw-r--r--  1 root root  3720 May 14 12:05 bash-2.05b-r11.ebuild
-rw-r--r--  1 root root  3516 May  2 20:05 bash-2.05b-r9.ebuild
-rw-r--r--  1 root root  5083 May  3 22:35 bash-3.0-r11.ebuild
-rw-r--r--  1 root root  4038 May 14 12:05 bash-3.0-r7.ebuild
-rw-r--r--  1 root root  3931 May 14 12:05 bash-3.0-r8.ebuild
-rw-r--r--  1 root root  4267 Mar 29 21:11 bash-3.0-r9.ebuild
drwxr-xr-x  2 root root  4096 May  3 22:35 files
-rw-r--r--  1 root root   164 Dec 29 2003 metadata.xml
bash-2.05b$ cat metadata.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pkgmetadata SYSTEM "http://www.gentoo.org/dtd/metadata.dtd">
<pkgmetadata>
<herd>base-system</herd>
</pkgmetadata>
bash-2.05b$ sudo /etc/init.d/bluetooth status
Password:
* status:  stopped
bash-2.05b$ ping -q -c1 en.wikipedia.org
PING rr.chtpa.wikimedia.org (207.142.131.247) 56(84) bytes of data.

--- rr.chtpa.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 112.076/112.076/112.076/0.000 ms
bash-2.05b$ grep -i /dev/sda /etc/fstab | cut --fields=-3
/dev/sda1      /mnt/usbkey
/dev/sda2      /mnt/ipod
bash-2.05b$ date
Wed May 25 11:36:56 PDT 2005
bash-2.05b$ lsmod
Module                Size  Used by
joydev                 8256   0
ipw2200               175112   0
ieee80211             44228   1 ipw2200
ieee80211_crypt       4872    2 ipw2200,ieee80211
e1000                 84468   0
bash-2.05b$ █
```

Interaction controlled  
by system, user  
queried for input when  
needed by system

# A bit later...

---



# Even later...

---



# Event-Driven UIs

---

Old model (e.g., command line, UNIX shell, DOS)

- Interaction controlled by system, user queried for input when needed by system

Event-Driven Interfaces (e.g., GUIs)

- Interaction controlled by user
- System waits for user actions and then reacts
- More complicated programming and architecture

# What is a UI toolkit?

---

The tools that application programmer typically programs with.

Combination of interface objects and management behaviors.

Library of software components and routines that programmer puts together.

- Macintosh: Mac Toolbox, MacApp
- Windows: Windows Developers' Toolkit
- Java: Swing
- Android: Android UI toolkit



# Separation of concerns

---

## Application

- Core functionality
- Operations
- Data

## Interface

- Interface components
- Graphics
- I/O

Should these be separated?


Why?

Why not?

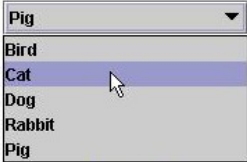
# Toolkit Example: Java Swing

GUI toolkit with a widget set and an API


### Basic Controls



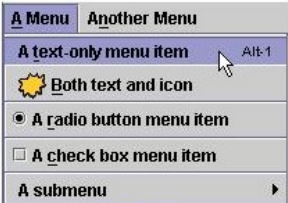
[Buttons](#)




[Combo box](#)




[List](#)




[Menu](#)



[Slider](#)



[Spinner](#)



[Text field or Formatted text field](#)

### JButton (Java 2 Platform SE v1.4.2) - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JButton.html

java swing api

#### Constructor Summary

<code>JButton ()</code>	Creates a button with no set text or icon.
<code>JButton (Action a)</code>	Creates a button where properties are taken from the Action supplied.
<code>JButton (Icon icon)</code>	Creates a button with an icon.
<code>JButton (String text)</code>	Creates a button with text.
<code>JButton (String text, Icon icon)</code>	Creates a button with initial text and an icon.

#### Method Summary

protected void	<code>configurePropertiesFromAction (Action a)</code> Factory method which sets the AbstractButton's properties according to values from the Action instance.
AccessibleContext	<code>getAccessibleContext ()</code> Gets the AccessibleContext associated with this JButton.
String	<code>getUIClassID ()</code> Returns a string that specifies the name of the L&F class that renders this component.
boolean	<code>isDefaultButton ()</code> Gets the value of the defaultButton property, which if true means that this button is the current default button for its JRootPane.
boolean	<code>isDefaultCapable ()</code> Gets the value of the defaultCapable property.
protected String	<code> paramString ()</code> Returns a string representation of this JButton.
void	<code>removeNotify ()</code> Overrides JComponent.removeNotify to check if this button is currently set as the default button on the RootPane, and if so, sets the RootPane's default button to null to ensure the RootPane doesn't hold onto an

Done

# What should toolkits do?

---

Help **design** the interface given a specification of the tasks.

Help **implement** the interface given a design.

Help **evaluate** the interface after it is designed.

Create easy-to-use interfaces.

Allow the designer to rapidly investigate different designs.

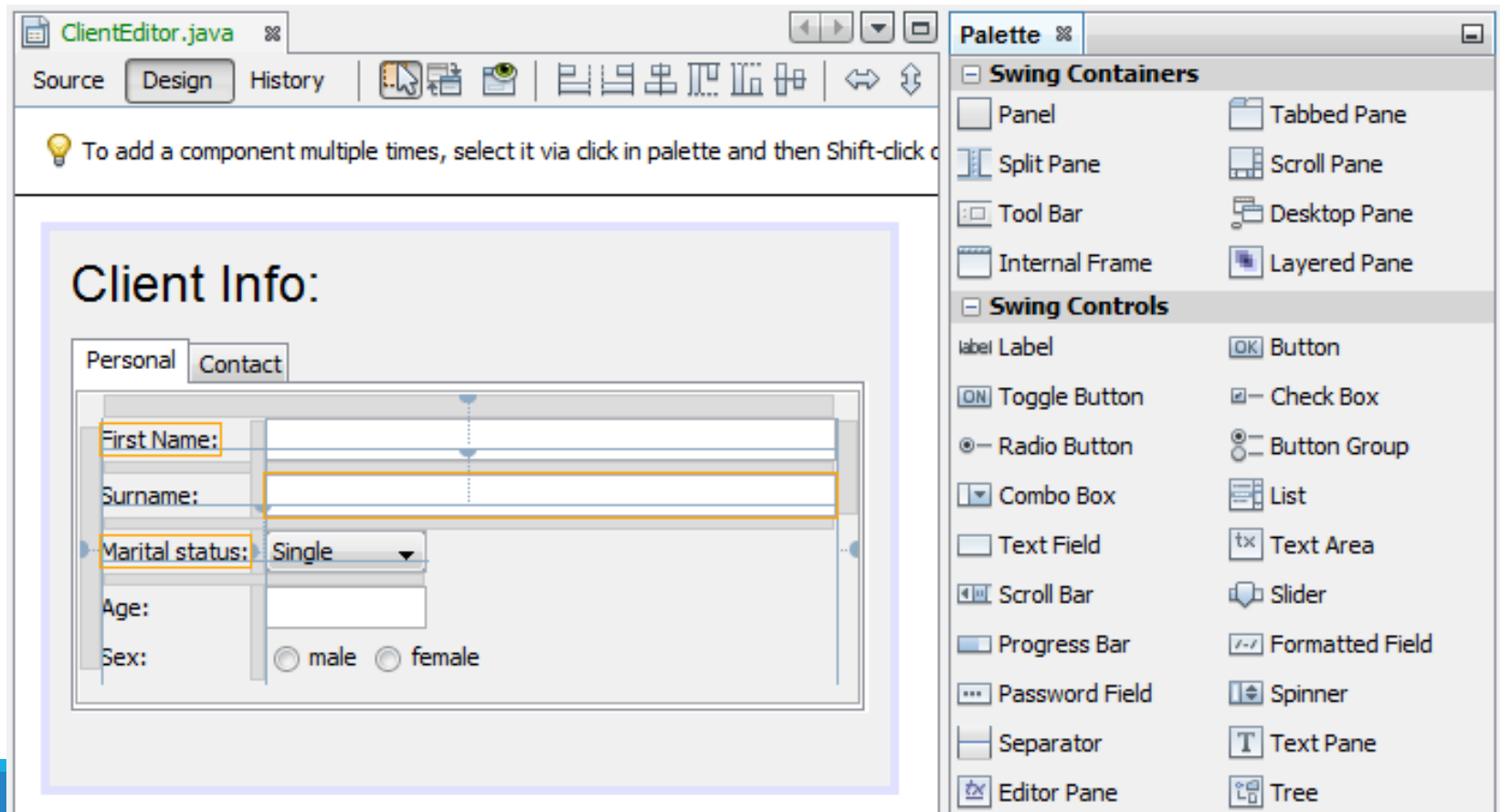
Allow non-programmers to create user interfaces.

Provide portability across different machines and devices.

Be easy to use themselves.

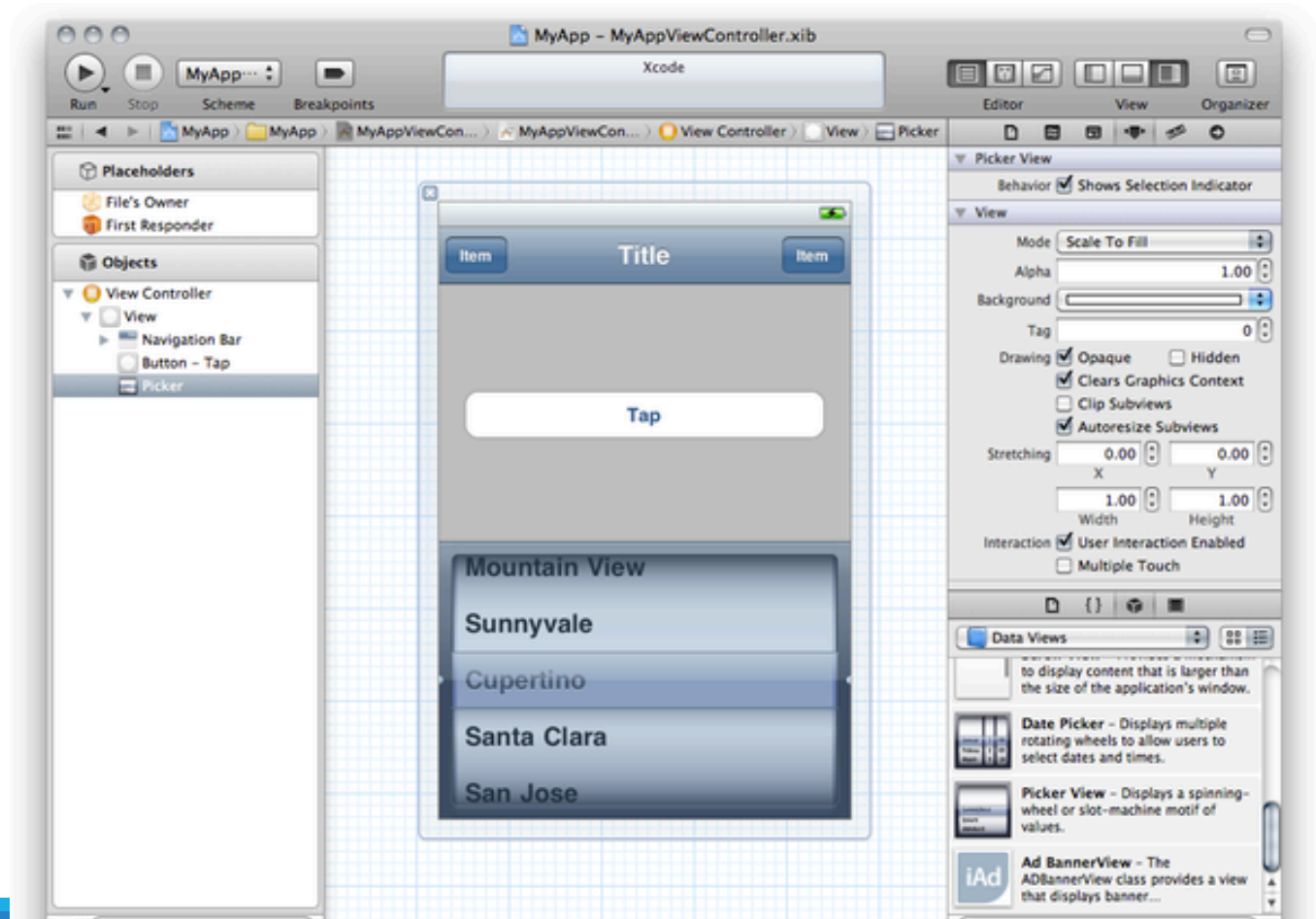
# User Interface Builders

## Java Swing



# User Interface Builders

iPhone



# User Interface Builders

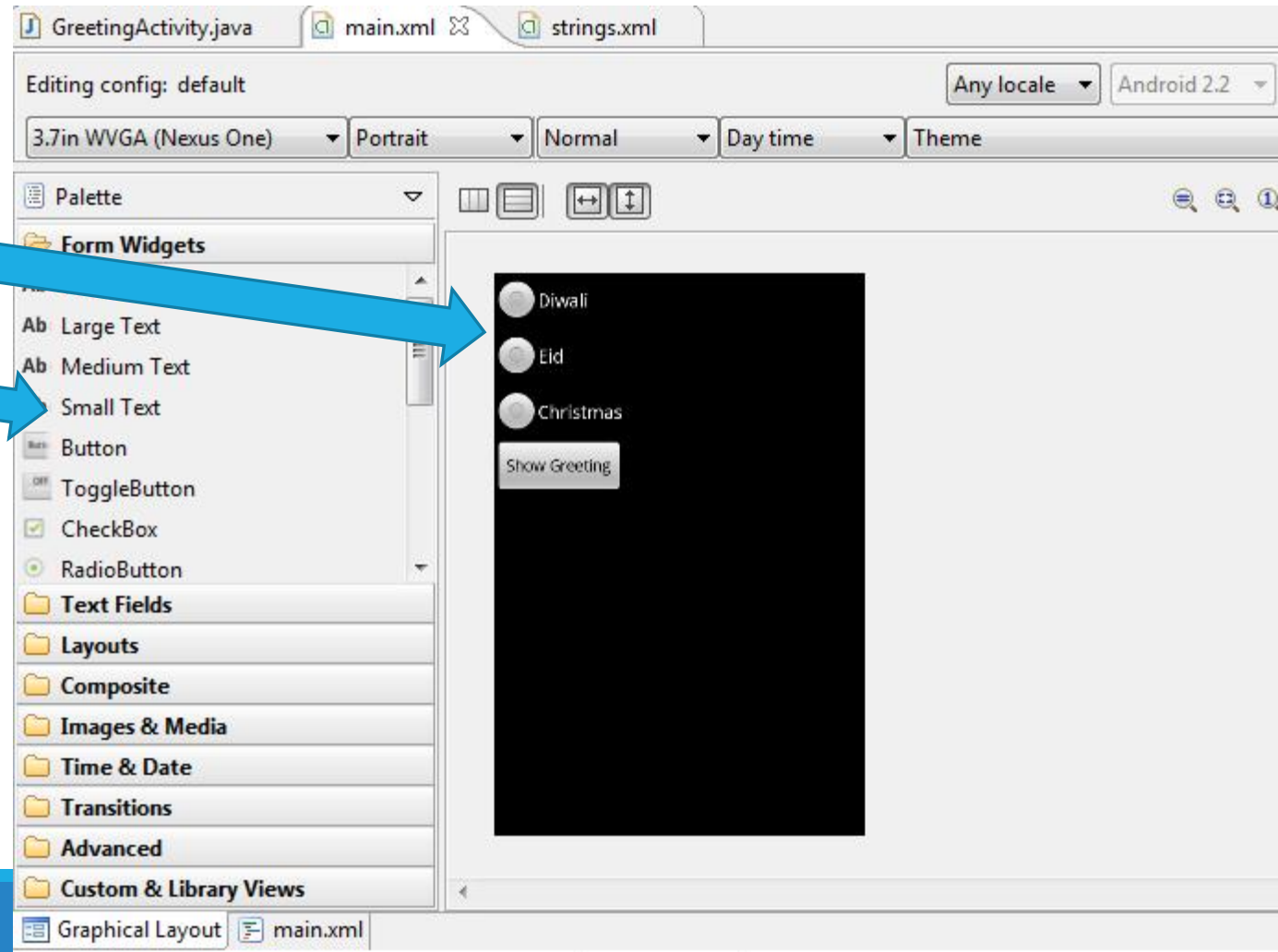
## Android

Work area  
(interface being built)

Drag and drop  
“widgets” onto  
work area

Specify position,  
color, look, etc.

Often provide  
Build/Test modes



# How Does a Toolkit Work?

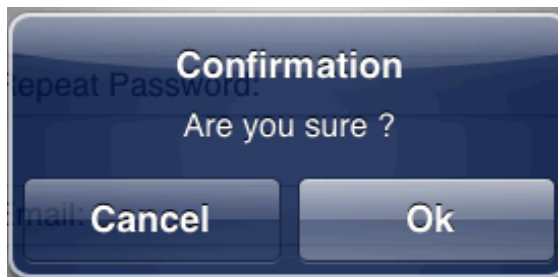
---

User takes actions, interacts with interface

Those actions must be delivered to application in meaningful ways

Application takes appropriate actions based on user input

- updates display
- produces output, etc.



# Event-Driven Cycle

---

Initialize display & system

Repeat

- Wait for and get next user action
- Decipher action
- Take appropriate action (via Callback routine)
- Update display

Until Done/Exit/Shut down



# Callback Routine

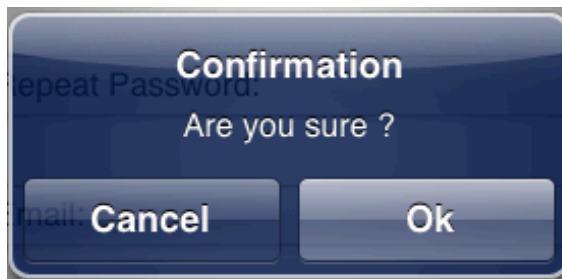
---

Software procedure, part of application

Invoked when particular action occurs to UI component, such as pressing a button, clicking, typing.

- Tells the application what to do based on the user action that has occurred.

Often invoked with event parameters (x, y, etc.)

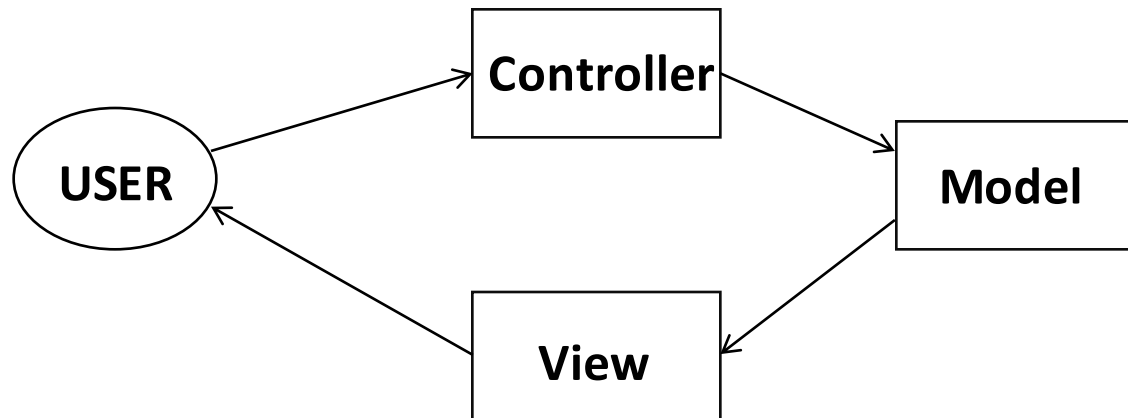


# How to manage interaction?

---

## Model-View-Controller (MVC) model

- Design pattern found, in some form, in most UI toolkits



# Model-View-Controller (MVC)

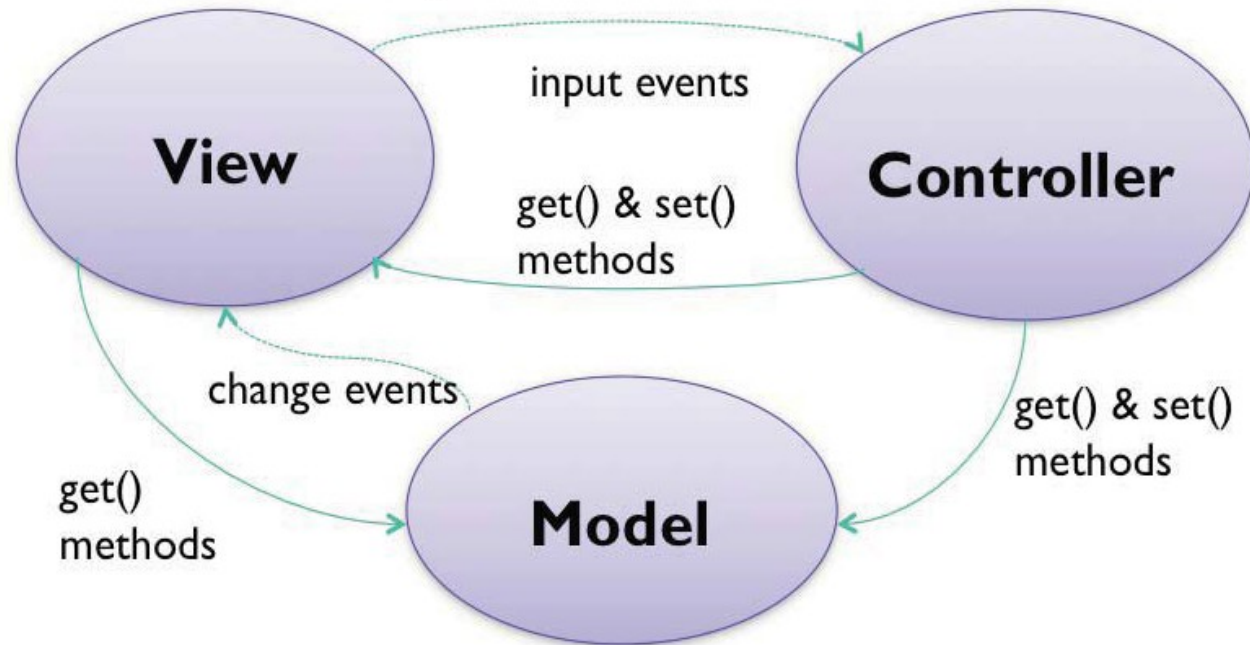
---

View handles output

- gets data from the model to display it
- listens for model changes and updates display

Controller handles input

- listens for input events on the view tree
- calls mutators on model or view



Model maintains application state

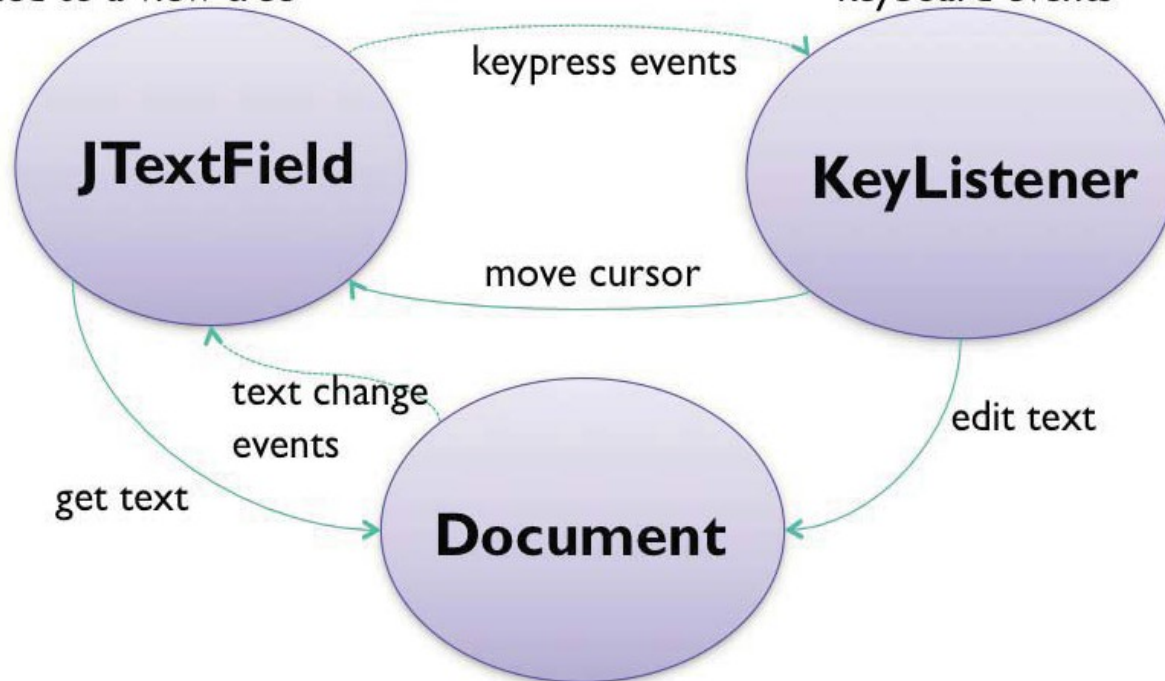
- implements state-changing behavior
- sends change events to views

# MVC Example

---

JTextField is a Component that can be added to a view tree

KeyListener is a listener for keyboard events



Document represents a mutable string of characters

# Advantages of MVC

---

- Separation of responsibilities
  - Each module is responsible for just one feature
    - Model: data
    - View: output
    - Controller: input
- Decoupling
  - View and model are decoupled from each other, so they can be changed independently
  - Model can be reused with other views
  - Multiple views can simultaneously share the same model
  - Views can be reused for other models, as long as the model implements an interface

# But... sometimes hard to separate view and controller

---

- Controller often needs output
  - View must provide **affordances** for controller (e.g. scrollbar thumb)
  - View must also provide **feedback** about controller state (e.g., depressed button)
- State shared between controller and view: Who manages the selection?
  - Must be displayed by the view (as blinking text cursor or highlight)
  - Must be updated and used by the controller

# Model - View(Widget)

---

- The MVC idea has largely been superseded by an MV (Model-View) idea
- A widget is a reusable view object that manages both its output and its input
  - Widgets are sometimes called components (Java, Flex) or controls (Windows)
- Examples: scrollbar, button, menubar



# Widget

---

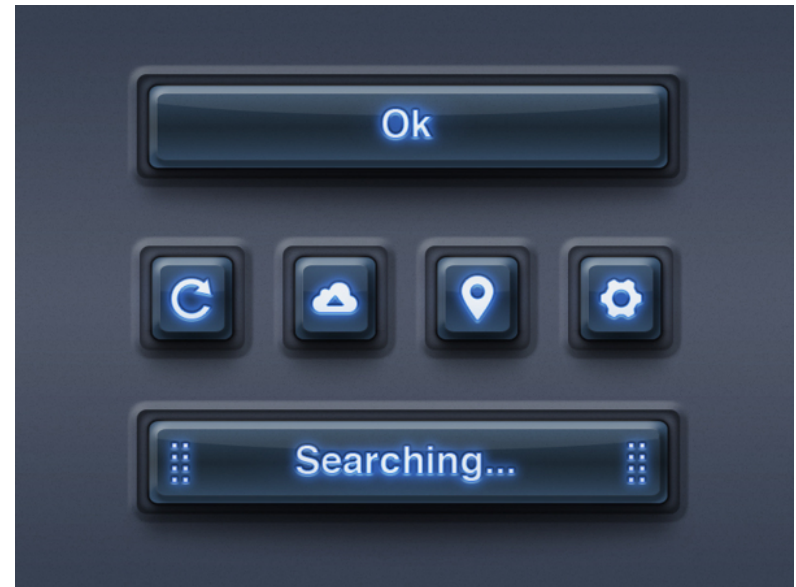
Visual appearance

Interactive behavior

Set of customizable attributes

Button:

- Color BackGround;
- int MarginLeft;
- int MarginRight;
- int BorderWidth;
- Pixmap ArmPixmap;
- Boolean FillOnArm;
- CallbackList ActivateCallback;



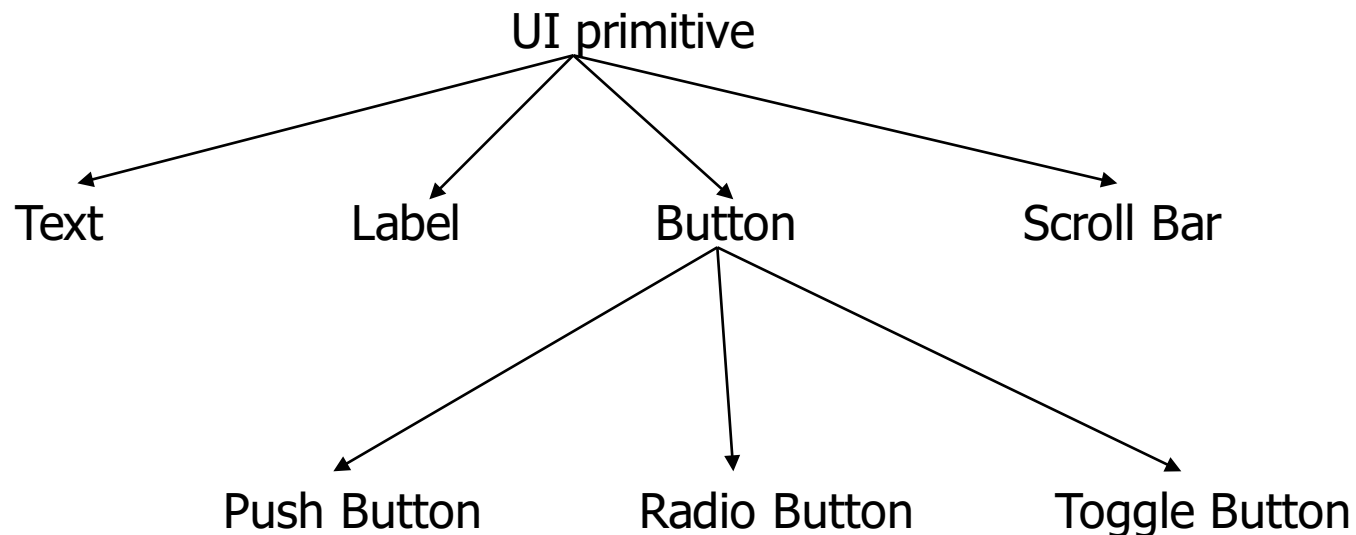


# Widget Hierarchy

---

Widgets organized into inheritance hierarchy

- Reusable components



# Why use UI Toolkits?

---

Provide sets of standard UI components

Guide the implementation

Help with screen layout and graphic design

Deal with field scrolling and editing

Validate user inputs, handle user errors and undoing of operations

Provide help and prompts

Insulate the application from all device dependencies and the underlying software and hardware systems

Support features that allow end users to customize the interface

# Why use UI Toolkits?

---

The quality of the interfaces will be higher.

## Why?

Rapid prototyping.

Easier to incorporate changes motivated by evaluation.

Re-use affords investment in high quality tools.

Consistency of interface design.

Enable collaboration among specialists.

# Why use UI Toolkits?

---

The user interface will be easier to create and maintain.

## Why?

Less code to write due to component re-use.

Better modularization, separation of concerns

Tools may abstract complex systems or algorithms.

Easier to port an application to different hardware or software environments if device dependencies are isolated in the user interface tool.

# Evaluating User Interface Tools

---

Factors to think about

- Expressiveness
- Learning Rate (to become skilled)
- Development Rate (of skilled user)
- Performance
- Portability

# The Future of UI Toolkits

---

## Emerging interface and interaction styles

- mobile
- recognition-based UIs (gestures, speech, pens, etc.)
- multiple devices

## Complex design space

- e.g., do we have to update the toolkit every time someone creates a new sensor or actuator?

## Ambiguous input

- Speech, gestures, computer vision, etc. aren't recognized as accurately as mouse clicks. Should the toolkit handle the recognition?

# Summary

---

Toolkits provide reusable interface components to simplify user interface development

Beware of the toolkit trap

- It's tempting to only make UIs that the toolkit makes easy, instead of making what's best for a specific app

Toolkits exist for most “major” computing platforms

- Mac, Windows
- Web platforms (twitter, facebook)
- Android, iPhone, etc.

# Next time...

---

- Design theory... really!