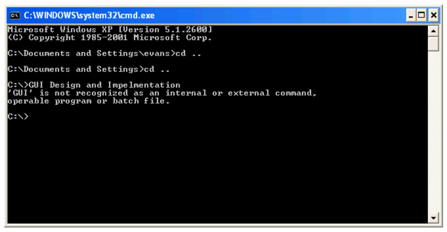
cs205: engineering software
university of virginia
fall 2006



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\evans>cd ..

C:\Documents and Settings>cd ..

C:\>GUI Design and Impelmentation
'GUI' is not recognized as an internal or external command,
operable program or batch file.

C:\>
```

GUI Design and Implementation

Schedule design meetings this week

Early Interfaces

IBM 705

Univac 1956

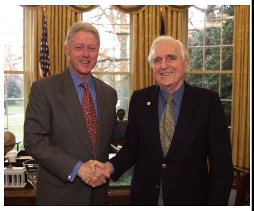
Sketchpad

- Ivan Sutherland, 1963 (PhD thesis supervised by Claude Shannon)
- Interactive drawing program
- Light pen

http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-574.pdf

Birth of the GUI

"We see the quickest gains emerging from (1) giving the human the minute-by-minute services of a digital computer equipped with computer-driven cathode-ray-tube display, and (2) **developing the new methods of thinking and working that allow the human to capitalize upon the computer's help**. By this same strategy, we recommend that an initial research effort develop a prototype system of this sort aimed at increasing human effectiveness in the task of computer programming."



Medal of Technology 2000

Douglas Engelbart, *Augmenting Human Intellect* (1962)

Computer as "Clerk"

In such a future working relationship between human problem-solver and computer 'clerk,' the capability of the computer for executing mathematical processes would be used whenever it was needed. However, the computer has many other capabilities for manipulating and displaying information that can be of significant benefit to the human in nonmathematical processes of planning, organizing, studying, etc. Every person who does his thinking with symbolized concepts (whether in the form of the English language, pictographs, formal logic, or mathematics) should be able to benefit significantly.

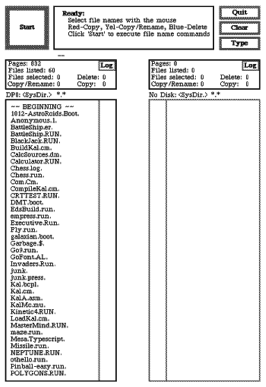
Douglas Engelbart, *Augmenting Human Intellect* (1962)

Engelbart's Demo (1968)

- First Mouse
- Papers and folders
- Videoconferencing
- Email
- Hypertext
- Collaborative editing

http://sloan.stanford.edu/MouseSite/1968Demo.html

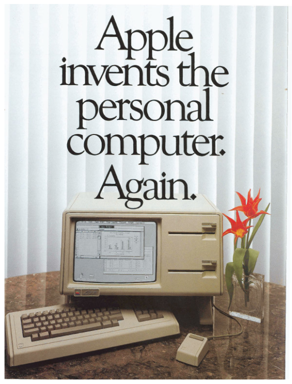
Xerox Alto



```
Start   Ready:
          Select file names with the mouse
          Red-Copy, Yel-Copy/Rename, Blue-Delete
          Click 'Start' to execute file name commands      Quit
                                                           Clear
                                                           Type
        --
Pages: 832                     Log     Pages: 0                       Log
Files listed: 60                       Files listed: 0
Files selected: 0     Delete: 0        Files selected: 0     Delete: 0
Copy/Rename: 0        Copy:   0        Copy/Rename: 0        Copy:   0
DP0: <SysDir.>  *.*                    No Disk: <SysDir.>  *.*
 ~~ BEGINNING ~~
 1012-AstroRoids.Boot.
 Anonymous.1.
 BattleShip.er.
 BattleShip.RUN.
 BlackJack.RUN.
 BuildKal.cm.
 CalcSources.dm.
 Calculator.RUN.
 Chess.log.
 Chess.run.
 Com.Cm.
 CompileKal.cm.
 CRTTEST.RUN.
 DMT.boot.
 EdsBuild.run.
 empress.run.
 Executive.Run.
 Fly.run.
 galaxian.boot.
 Garbage.$.
 Go9.run.
 GoFont.AL.
 Invaders.Run.
 junk.
 junk.press.
 Kal.bcpl.
 Kal.cm.
 KalA.asm.
 KalMc.mu.
 Kinetic4.RUN.
 LoadKal.cm.
 MasterMind.RUN.
 maze.run.
 Mesa.Typescript.
 Missile.run.
 NEPTUNE.RUN.
 othello.run.
 Pinball-easy.run.
 POLYGONS.RUN.
```

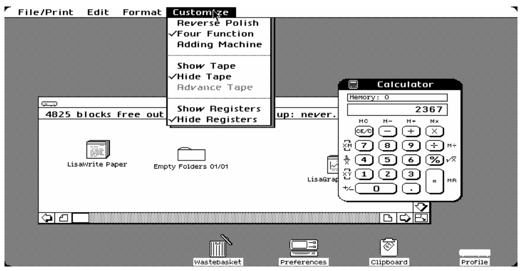
Xerox PARC, 1973

Apple Lisa



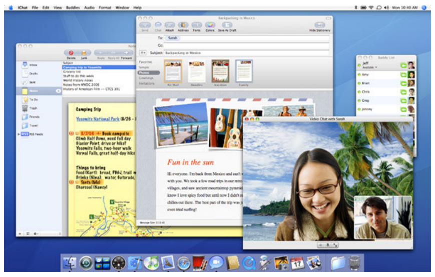
1983

Lisa Interface



http://www.guidebookgallery.org/screenshots/lisaos10

Any real progress since then?



OS X Leopard, 2006

Designing GUIs

- Requires lots of skill
- Psychology, Cognitive Science
- User studies
- **Good taste**

Read Donald Norman’s and Ed Tufte’s books

Building GUIs

- Like all Programming
  - Encapsulation, Abstraction, Specification
  - Testing: especially hard
- Unique-ish Aspects
  - Event-Driven (network programming also like this)
  - Multi-Threaded (network, others)
  - Huge APIs

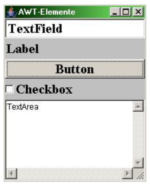
Model-View-Controller

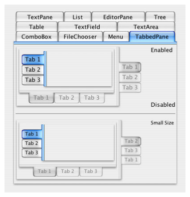
- Model: domain data and logic
- View: presents model
- Controller: receives input and alters model

Goal: **abstraction**
separate display from model
separate control interface

Invented at PARC in 1970s (Smalltalk)

Java GUI Toolkits





AWT
Abstract Window Toolkit
Looks like Java

Swing
(added JDK 1.2)
real reason for Swing coming later...

Frames

$$\begin{tikzcd}
\text{java.lang.Object} \\
\text{java.awt.Component} \arrow[u] \\
\text{java.awt.Container} \arrow[u] \\
\text{java.awt.Window} \arrow[u] \\
\text{java.awt.Frame} \arrow[u] \\
\text{javax.swing.JFrame} \arrow[u]
\end{tikzcd}$$

Main windows are JFrame objects

```
JFrame frame =
   new JFrame("Swing GUI");
```

$\uparrow$
Window Title

JFrame Methods

// inherited from java.awt.Window
public void pack()
  MODIFIES: this
  EFFECTS: Causes this Window to be sized to fit the preferred size and layouts of its subcomponents.

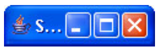
// inherited from java.awt.Component
public void setVisible(boolean b)
  MODIFIES: this, display
  EFFECTS: If b, shows this. Otherwise, hides this.

Swing Application

```
import javax.swing.*;

public class Main {
  private static void showGUI() {
      //Create and set up the window.
      JFrame frame = new JFrame("Swing GUI");
      frame.pack();
      frame.setVisible(true);
  }

   public static void main(String args[]) {
      javax.swing.SwingUtilities.invokeLater(new Runnable() {
          public void run() { showGUI(); }
      });
   }
}
```



Based on Sun's Swing tutorials:
http://java.sun.com/docs/books/tutorial/uiswing/learn/example1.html

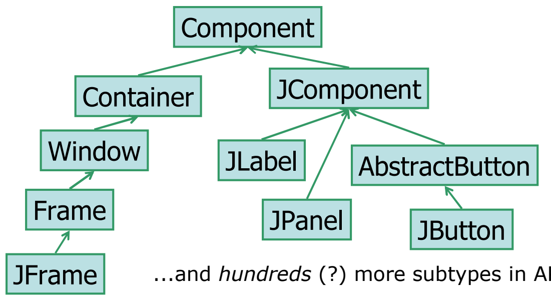
Adding to a Frame

public java.awt.Container **getContentPane**()
  EFFECTS: Returns the contentPane object for this.

in java.awt.Containter:
public Component **add**(Component c)
  MODIFIES: this
  EFFECTS: Appends *c* to the end of this container.

What can you add?

public Component **add**(Component c)



...and *hundreds* (?) more subtypes in API

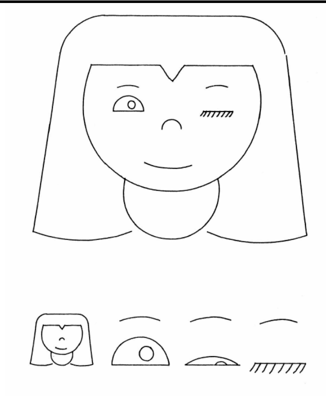
GUIs and Subtyping

In the process of making the Sketchpad system operate, a few very general functions were developed which make no reference at all to the specific types of entities on which they operate. These general functions give the Sketchpad system the ability to operate on a wide range of problems. The motivation for making the functions as general as possible came from the desire to get as much result as possible from the programming effort involved. For example, the general function for expanding instances makes it possible for Sketchpad to handle *any* fixed geometry subpicture. The rewards that come from implementing general functions are so great that the author has become reluctant to write any programs for specific jobs.

Each of the general functions implemented in the Sketchpad system abstracts, in some sense, some common property of pictures independent of the specific subject matter of the pictures themselves.

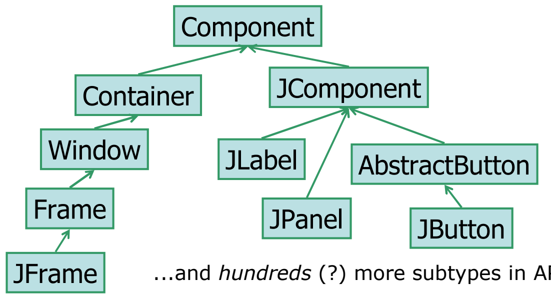
Ivan Sutherland, *Sketchpad: a Man-Machine Graphical Communication System*, 1963 (major influence on Alan Kay inventing OOP in 1970s)

Components in Sketchpad



What can you add?

public Component **add**(Component c)



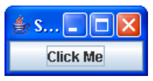
...and *hundreds* (?) more subtypes in API

Adding Components

```
import javax.swing.*;

public class Main {
  private static void showGUI() {
     //Create and set up the window.
     JFrame frame = new JFrame("Swing GUI");
     java.awt.Container content = frame.getContentPane();
     content.add(new JLabel ("Yo!"));
     content.add(new JButton ("Click Me"));
     frame.pack();
     frame.setVisible(true);
  }

  public static void main(String args[]) {
     ...
  }
}
```

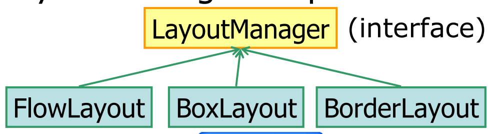


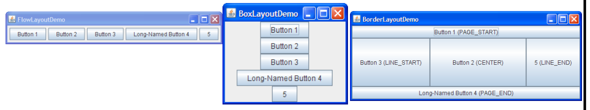
What happened to "Yo!"?

Layout

// in Container:
public void **setLayout**(LayoutManager mgr)
   MODIFIES: this
   EFFECTS: sets the layout manager to mgr
      for this container.

LayoutManager Implementations





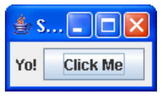
...about 30 more in API!

http://java.sun.com/docs/books/tutorial/uiswing/layout/visual.html

Adding Components

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

public class Main {
  private static void showGUI() {
      //Create and set up the window.
      JFrame frame = new JFrame("Swing GUI");
      java.awt.Container content = frame.getContentPane();
      content.setLayout(new FlowLayout());
      content.add(new JLabel ("Yo!"));
      content.add(new JButton ("Click Me"));
      frame.pack();
      frame.setVisible(true);
  }
```



Don’t try this at home?

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

public class Main {
  private static void showGUI() {
      //Create and set up the window.
      JFrame frame = new JFrame("Swing GUI");
      java.awt.Container content = frame.getContentPane();
      content.setLayout(new FlowLayout());
      content.add(frame);
      frame.pack();
      frame.setVisible(true);
  }
```

```
Exception in thread "AWT-EventQueue-0"
java.lang.IllegalArgumentException: adding container's parent to itself
```

Charge

- GUI APIs are subtyping and inheritance paradises, concurrency morasses
- GUI APIs are huge and complex
  - Java’s is especially complex because of AWT + Swing, and portability

Creating a *simpler* GUI requires *more complex* programming

Simplicity is the ultimate sophistication.
Introducing Apple II, the personal computer.