ICS 111
GUI programming

- History of Graphical User Interfaces
- Frames
- Labels
- subclassing JFrame

Historical note: Non-Graphical User Interfaces

- The first computers were programmed using switches toggled by hand
- By the 1970s, computers could use disks and magnetic tapes for storage
- Users entered programs and data into computers with switches, punched cards, and paper tape
- Output was on printers, blinking lights, punched cards, and paper tape
- Users could also interact with the computer via text-oriented terminals
  - the Scanner and System.out methods in Java work just fine with teminals
  - terminal windows provide all of the functionality of a hardware terminal
  - in addition, terminal windows allow resizing, moving around the screen, etc

Historical note: Early Graphics

- The graphic technique in common use today is known as bitmap graphics:
  - each pixel (picture element – each dot) in the display is saved as a bit in memory
    - multiple bits per pixel for color or grayscale displays
  - a program in the computer reads and writes to a giant array of bits (or ints) to change what the display shows
  - bitmap displays are only affordable when memory is cheap
- Early graphics used commands sent to specialized hardware that could draw lines on a screen: **vector graphics**
  - no pixels!
  - an image could only be drawn with straight lines
  - vector graphics has improved a lot since the 1960s!
- Less fancy graphics were created using characters printed to a printer.

  This later became known as **ASCII art**


```
  _
>(o)__
  (_~_/
~~~~~~~
```

```
  |\=/|.-"""-.
  /6  6\       \
=\_Y_/=   (_   ;\
    ^//_/-/__///
               ((
```

Historical Note: Graphical User Interfaces

- In the 1970s, the Xerox Alto was developed as a personal computer
  - instead of computers being in a machine room and shared by many users, individual users could interact with their own computer
  - the Alto had a keyboard, a 3-button mouse, and a (black and white) bitmap graphic screen
- In the 1980s, Apple developed the Lisa, soon followed by the Macintosh, both with 1-button mouse and bitmap graphic screen
- Microsoft also developed its Windows system for the IBM PC
- with the resulting large number of users, many developers produced programs that would only work on bitmap graphical displays:
  - word processors
  - spreadsheets
  - drawing programs
  - games
  - email clients
  - later, web browsers

Historical Note: Human-Computer Interaction

- computers were never easy to use
- computer scientists and others started studying ways to make this better:
  - how to make computers easier to use: recognize, not recall
    - most users find it easier to click on a file rather than remember or look up a file name
  - how to reduce the number of errors
    - have a universal “undo” rather than prompt “are you sure?”
  - graphical, audio, and video input and output
    - fingerprints instead of passwords
- This is a field straddling computer science and other fields such as psychology and cognitive science

Review: GUIs in Java

- JOptionPane dialog boxes: Sep 2nd lecture, book special topic 2.5, homework 2
  - just call JOptionPane.showInputDialog, jOptionPane.showMessageDialog, ...
- JFrame and graphics: Sep 23rd lecture, book special topic 4.3, homework 5
  - the book (and we) used a fixed main method that:
    - creates a Jframe object
    - sets its size
    - creates a Jcomponent that calls your draw method, and adds it to the frame
    - makes the frame visible
    - the draw method takes as parameter an object of type Graphics
  - The java.awt.Graphics class has many methods, including:
    - drawRect and fillRect
    - drawLine
    - drawOval and fillOval
    - drawString

Model-View-Controller

- The internal representation of what is to be displayed is the **model**
- What is actually displayed is the **view**
  - the view may represent only part of the model
  - methods such as javax.swing.JFrame.setVisible() and java.awt.Component.repaint() make the view match the model
- The **controller** accepts input from the user and modifies the model accordingly

javax.swing.JFrame

- javax is a collection of e**X**tensions to the Java standard library
- swing is an update of the Abstract Window Toolkit (awt) library
- graphics in Java happen inside a JFrame, displayed as a window
- before you can use a JFrame, you should set its size, which is otherwise 0x0

```
JFrame frame = new JFrame();
frame.setSize(150, 100); // 150 wide, 100 tall
```

- you may also set the frame's title:

```
frame.setTitle("frame 1");
```

- make it clear that the program should exit if the user closes the frame:

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

- finally, make the frame visible:

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

  - `setVisible(false)` hides the frame

Example: building a basic Frame

BasicFrame.java

```
public abstract class BasicFrame {
  protected javax.swing.JFrame frame;
  // must be implemented by all non-abstract subclasses
  protected abstract void addContents();
  // constructor
  public BasicFrame() {
    frame = new javax.swing.JFrame();
    frame.setSize(300, 200);
    frame.setTitle("basic frame");
    frame.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
    this.addContents(); // add the contents
    frame.setVisible(true);
  }
}
```

- subclasses of BasicFrame add their content by implementing addContents

```
public class BasicFrameSubclass extends BasicFrame {
  protected void addContents() {
    // no contents
  }
}
```

Organizing content: javax.swing.JPanel

- as stated in this tutorial, a JPanel is

  useful for grouping components, simplifying component layout, and putting borders around groups of components
- by default, a JPanel's components are added one after another, horizontally
  - to change this, give a layout to the constructor for the JPanel, or call setLayout()
- JPanel is a subclass of java.awt.Component, and as such, can be added to a JFrame
- we can add different components to a JPanel, such as
  - JLabel, displays text
  - JButton, displays a button to click
    - but we haven't yet seen how to handle button clicks
  - JScrollPane, allows scrolling of content place into the scrollpane
  - JMenu
  - JSlider

A simple example

```
public class BasicFrameWithTwoItems extends BasicFrame {
  // code to add the contents of the frame
  protected void addContents() {
    javax.swing.JPanel panel = new javax.swing.JPanel();
    frame.add(panel);
    panel.setLayout(new javax.swing.BoxLayout(panel, javax.swing.BoxLayout.Y_AXIS));
    javax.swing.JLabel h = new javax.swing.JLabel("Hello, world");
    panel.add(h);
    javax.swing.JButton b = new javax.swing.JButton("This is a button");
    panel.add(b);
  }
  // constructor -- not really needed, Java automatically calls super()
  public BasicFrameWithTwoItems() {
    super();   // calls addContents()
  }
  public static void main(String[] a) {
    BasicFrameWithTwoItems bfwti = new BasicFrameWithTwoItems();
  }
}
```

Subclassing JFrame

- There is no need for the abstract class BasicFrame, we can subclass JFrame
- the components can be instance variables of the subclass
- the constructor (or a helper method) initializes the instance variables and adds them to the frame

Summary

- Human-Computer Interaction (HCI) has helped us build systems that the average person can fruitfully use
- Java graphics appear in a window called a JFrame
  - programs creating a JFrame should also set a title and a size
- individual components can be placed in a JPanel added to the JFrame
  - the JPanel organizes the components
  - each component has its own functionality