GUIS —Graphical User Interfaces

Read Chap. 17 of the text. The ProgramLive CD is a better way to learn about GUIs. See the CD for examples of code.

You will not be tested on this material.

Their mouse had a mean time between failure of … a week, at which time it would jam up irreparably, or ... It would jam up on the table-- ... It had a flimsy cord whose wires would break. Steve Jobs said "... Xerox says it can't be built for less than $400, but I want a $10 mouse that will never fail and can be mass-produced, because it's going to be the primary interface of the computer of the future."

... Dean Hovey ... came back and said, "I've got some good news and some bad news. The good news is, we've got a new project with Apple. The bad news is, I told Steve we'd design him a mouse for ten bucks."

... A year … later ... we had a design, filed a patent, and were granted a patent, on the electro-mechanical-optical mouse of today, which is still the reference design for PC mice. ... and ... we ended up ... [making] the mouse as invisible to people as it is today.

Interview with Steve Sachs on Apple and the Mouse in 1979 and the first computer with a GUI, the Apple Lisa (about $9,999 in about 1982).
http://library.stanford.edu/mac/primary/interviews/sachs/trans.html

Putting components in a JFrame

```
import java.awt.*;                import javax.swing.*;
/** Demonstrate placement of components in a JFrame. Use BorderLayout.
   It places five components in the five possible areas:
     (1) a JButton in the east,
     (2) a JLabel in the west,
     (3) a JLabel in the south,
     (4) a JTextField in the north, and
     (5) a JTextArea in the center.   */
public class ComponentExample extends JFrame {
   /** Constructor: a window with title t and 5 components */
   public ComponentExample(String t) {
      super(t);
      Container cp=  getContentPane();
      cp.add(new JButton("click me"), BorderLayout.EAST);
      cp.add(new JTextField("type here", 22), BorderLayout.NORTH);
      cp.add(new JLabel("label 1"), BorderLayout.SOUTH);
      cp.add(new JLabel("label 2"), BorderLayout.WEST);
      cp.add(new JTextArea("type\nhere", 4, 10), BorderLayout.CENTER);
      pack();
   }
}
```


What components can go in a JFrame

Packages that contain classes that deal with GUIs:
java.awt javax.swing

Component: Something that can be placed in a GUI window. They are instances of certain classes, e.g.

JButton, Button: Clickable button
JLabel, Label: Line of text
JTextField, TextField: Field into which the user can type:
JTextArea, TextArea: Many-row field into which user can type
JPanel, Panel: Used for graphics; to contain other components
JCheckBox: Checkable box with a title
JComboBox: Menu of items, one of which can be checked
JRadioButton: Same functionality as JCheckBox
Container: Can contain other components
Box: Can contain other components

JFrame's content pane

Layout manager: An instance controls the placement of components.

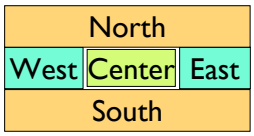
JFrame layout manager default: BorderLayout.

BorderLayout layout manager: Can place 5 components:

```
Container cp= getContentPane();
JButton jb= new JButton("Click here");
JButton jl= new JLabel( "label 2");

cp.add(jb, BorderLayout.EAST);
cp.add(jl, BorderLayout.WEST);

pack();
setVisible(true);
```



MazeGUI

```
Box JenBox= new Box(BoxLayout.Y_AXIS);   // Jen's title
scoreJLabel JenScore= new JLabel("   0",  SwingConstants.CENTER);

Box southBox= new Box(BoxLayout.Y_AXIS); // South panelJLabel
sproutsLeft= new JLabel("     Brussel sprouts left: 5555");

Box northBox= new Box(BoxLayout.Y_AXIS); // North panelJLabel
endOfGame= new JLabel("       Steve won!");
```

Box: components appear in order added. Y_AXIS mean vertical, X_AXIS mean horizontal

```
/** Constructor: ... */
public MazeGUI(Maze maze) {
   this.maze= maze;                        Container con= this.getContentPane();

   JenBox.add(new JLabel("  Jen "));       JenBox.add(new JLabel("  ate "));
   JenBox.add(JenScore);                   con.add(JenBox, BorderLayout.WEST);

   ... similar for Steve ...               con.add(SteveBox, BorderLayout.EAST);

   northBox.add(endOfGame)                 northBox.add(new JLabel(" "));
                                             con.add(northBox, BorderLayout.NORTH);

   ... similar for South ...               con.add(southBox, BorderLayout.SOUTH);
```


MazeGUI

```
JPanel mazePanel= new JPanel();  // The panel on which the maze is placed

/** Constructor: ... */
public MazeGUI(Maze maze) {
   ... Container con= this.getContentPane();

   ...
   mazePanel.setLayout(new GridLayout(maze.nRows(), maze.nCols()));

   // Create the array of JLabels and add them to the maze
   labels= new JLabel[maze.nRows()][maze.nCols()];
   for (int r= 0; r != labels.length; r= r+1) {
      for (int c= 0; c != labels[0].length; c= c+1) {
         labels[r][c]= new JLabel("" + maze.WALL);
         mazePanel.add(labels[r][c]);
      }
   }
   update();                        con.add(mazePanel, BorderLayout.CENTER);
   ...
```

new GridLayout(r,c)
A table of components with r rows and c columns

Listening to events

An **event** is a mouseclick, a mouse movement into or out of a window, a keystroke, etc.

Basically, to "listen to" a kind of event, you have to

- Write a method that will listen to the event.
- Register an instance of the class that contains the method as a *listener* for the event.

We show you how to do this for clicks on buttons and keystrokes in the next two slides.

Listening to a Button

- Write a procedure

```
/** Process click of button */
public void actionPerformed(ActionEvent ae) {
    ...
}
```

- Have the class implement interface ActionListener --write the class heading as

```
public class C extends JFrame implements ActionListener {
  ...
}
```

We have not discussed interfaces, and we won't. Wait for CS 211!

- Add an instance of this class as an "action listener" for the button:

```
button.addActionListener(this);
```


Listening to the keyboard

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

public class AllCaps extends KeyAdapter {
  JFrame capsFrame= new JFrame();
  JLabel capsLabel= new JLabel();

  public AllCaps() {
    capsLabel.setHorizontalAlignment(SwingConstants.CENTER);
    capsLabel.setText(":)");
    capsFrame.setSize(200,200);
    Container c= capsFrame.getContentPane();
    c.add(capsLabel);
    capsFrame.addKeyListener(this);
    capsFrame.show();
  }

   public void keyPressed (KeyEvent e) {
    char typedChar= e.getKeyChar();
    capsLabel.setText("" + typedChar + "");
  }

}
```

1. Extend this class.

3. Add this instance as a key listener for the frame

2. Override this method. It is called when a key stroke is detected.