Swing UI elements

Swing Containers
- JApplet
- JDialog
- JFrame
- JWindow
- JPanel
- JScrollPane
- JSplitPane
- JTitledBorder
- JToolBar
- JInternalFrame
- JLayeredPane
- JRootPane

Swing Components
- JButton, w/ icon labels
- JMenu
- JList
- JSlider
- JTextField, JPasswordField
- JComboBox
- JLabel
- JProgressBar
- JToolTip
- JColorChooser
- JFileChooser
- JTable -- database records
- JTextArea, JEditorPane,
- JTree

Displays
How to place components inside containers.

Containers have insets and default layout managers.

Insets (top, right, left, bottom)
  layout mgt. place components in container w/o overlaying insets

AWT layout managers:
  FlowLayout  left to right, top to bottom
  BorderLayout  north, south, east, west, center
  CardLayout  1 panel @ time selected from deck
  GridLayout  place on and stretch to row column grid
  GridBagLayout  constraint based layout on grid

Swing layout manager
  BoxLayout  single row or column, w/ glue, rigid fillers

Interface builders  GroupLayout, SpringLayout

Internal managers  used by swing (ScrollPaneLayout JScrollPane)

Custom managers  write your own
Flow Layout

Layout Managers can be constructed, and set for containers. Components are added to layout managers

```java
aContainer.setLayout( new layoutManager(...));
aContainer.add(aComponent);
aContainer.add(aComponent, aConstraint);
```

FlowLayout

JPanel default
no constraints

FlowLayout constructor takes 3 args for alignment (left, center, right), horizontal, and vertical gaps.

```java
new FlowLayout(FlowLayout.Left, 10, 5);
setAlignment(FlowLayout.CENTER)
setHgap(int), setVgap(int)
```
BorderLayout

default for contentPane
(JApplet, JFrame, JDialog)
uses position constraints

Add components to positions:
north, south, west, east, center
default position is center

    aContentPane.add(aPanel, BorderLayout.NORTH);
    aContentPane.add(aList);

Center position grows with resizing and fills all space not used by other positions.

Constructs layout w/o gaps between components

    new BorderLayout();

To set gaps between components

    new BorderLayout(int Hgap, int Vgap);
or
    setHgap(int)  setVgap(int)
BorderLayout as an application’s layout manager.

- tool panel JPanel, WEST
- status line JLabel, SOUTH,
- main component, CENTER

**JTabbedPane** (updates AWT card layout encapsulates a Card mgr with tab controls!

```java
addTab(String, Icon , Component // last arg tooltip
addTab(String, Icon, Component)  
addTab(String, Component)   ...
setComponentAt(int, Component)  
Component getComponentAt(int)  ...
```

```
status line
```

```
menu bar
```
**Grid, Box**

**GridLayout** manages components in a row by column layout. All components (grids) are equal sized.

```java
GridLayout(int rows, int cols);
GridLayout(int rows, int cols, int Hgap, int Vgap);
```

**BoxLayout** places components in a single row or column. Uses invisible fillers to help layout.

```java
thisContainer.setLayout(new BoxLayout, thisContainer,
    BoxLayout.X_AXIS);  // row, use Y_AXIS for column
```

BoxLayout uses its component's alignment and size attributes to size and position its components.

- With column layout height sizes are used, all width size ignored all components same width.
- With restricted minimum and maximum width size the components X alignment (left, center, right) is used
BoxLayout can use invisible filler components.

add invisible filler components in order with visible components.

`Box.createRigidArea(size)` enables spaces to be placed in the layout

`Box.createHorizontalGlue()` and `Box.createVerticalGlue()` specify where excess space in the layout should go.

Custom filler can also be used to space objects

```
new Box.Filler(minSize, prefSize, maxSize)
```
Layout for irregular constraint based layouts most complex layout mgr

GridBagLayout uses GridBagConstraints.

- set constraint values,
- set constraints for component’s layout
- add component

... 

GridBagLayout gbl = new GridBagLayout();
GridBagConstraints gbc = new GridBagConstraints();
...
setLayout(gbl);
... // set constraints
    gbc.anchor = GridBagConstraint.NORTHEAST;
    gbc.fill = GridBagConstraint.BOTH;
    gbc.gridx = 2;
    gbc.gridy = 0;
    gridheight = 3;
    ...
    gbl.setConstraints(aComponent, gbc);
    add(aComponent);
    ...
Components can span gridcells, be smaller than gridcell, and have padding between gridcell edge and component edge.
displayarea = gridcells component is using

**GridBagConstraints:**
- anchor: where to anchor component within its grid cells
- fill: how to fill gridcells
- gridx, gridy: position of component’s upper left hand cell
- gridwidth, gridheight: number cells allotted to component
- ipadx, ipady: internal padding -- little used, changes component's size
insets external padding between cell edge and gridcell edge

weightx relative weight of component’s grid cells compared to others applied on row/column basis interacts w/ fill. Extra space in display area

Need to experiment with setting to see effects of constraints.

```java
void addComponent(Component comp, int r, int c, int w, int h){
gbc.gridx = c; // column
gbc.gridy = r; // row
gbc.gridwidth = w; // width
gbc.gridheight = h; // height
container.add(comp); }
```

```java
addComponent(ta1,0,0,1,3);
addComponent(b1,0,1,2,1);
addComponent(b2,1,1,1,1);
addComponent(b3,1,2,1,1);
addComponent(cb,2,1,2,1);
addComponent(tf,3,0,2,1);
addComponent(ta2,3,2,1,1);
```
Group

Interface builder layouts: Group and Spring

GroupLayout is a layout manager that was developed for NetBeans IDE. Can also be used with manual design/implementation.

- works with horizontal and vertical (sub) layout managers separately
- sequential is like Box for Flow
- parallel places components on top of each other

horizontal layout = sequential group \{ c1, c2, c3 \}
vertical layout = parallel group (BASELINE) \{ c1, c2, c3 \}
Spring layout specifies relationships between the edges of its components.

It lays out its children with a set directional relationships, or *constraints*, between the edges of components. For example, you might define that the left edge of one component is a fixed distance (5 pixels, say) from the right edge of another component. SpringLayout does not automatically set the location of the components it manages. Hand-coded SpringLayout must initialize component locations by constraining the west/east and north/south locations. You may also need to set the size of the container explicitly.
// Get content pane, set spring layout add components
Container contentPane = frame.getContentPane();
SpringLayout layout = new SpringLayout();
contentPane.setLayout(layout);
// Create and add the components.
JLabel label = new JLabel("Label: ");
JTextField textField = new JTextField("Text field", 15);
contentPane.add(label);
contentPane.add(textField);
// Adjust constraints for the label so it's at (5,5).
layout.putConstraint(SpringLayout.WEST, label, 5, SpringLayout.WEST, contentPane);
layout.putConstraint(SpringLayout.NORTH, label, 5, SpringLayout.NORTH, contentPane);
// Adjust text field's constraints to label's right edge + 5, 5).
layout.putConstraint(SpringLayout.WEST, textField, 5, SpringLayout.EAST, label);
layout.putConstraint(SpringLayout.NORTH, textField, 5, SpringLayout.NORTH, contentPane);
// Code that sets the container's springs:
layout.putConstraint(SpringLayout.EAST, contentPane, 5, SpringLayout.EAST, textField);
layout.putConstraint(SpringLayout.SOUTH, contentPane, 5, SpringLayout.SOUTH, textField);
Trouble shooting layouts

Watch effects of resize events

Use many, nest layout managers -- “layers” of management

Use component setSize(...) and set its size hints with
  setMaximumSize(...), setMinimumSize(...), setPreferredSize(...)

Use component’s revalidate() to make layout / size visible
  and propagate layout management reconfiguration

Color component’s background to differentiate between nested
  layout managers (design / debugging only)

Intermediate containers

**JPanel** flexible, base useful to build your own container or layout layer. FlowLayout default
Simple diet calorie counter program

menu
   File    new, open, save, saveAs
   Diet    diet list, foods, graph
   Help    usage, author

scrollable editable table for food items
   calories, fat cals, item description

textfield search string target for food data model

buttons
   search food database -- dialog with scrollable table, add to table
   add ("+") , delete ("-") items

today, week, diet day averages both numeric and graphical

status / message line

this design does not include the dialogs used for scrolling or searching food database, graph, help, or File
### Diet Helper

<table>
<thead>
<tr>
<th>File</th>
<th>Diet</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **enter search string here**

<table>
<thead>
<tr>
<th>Calories</th>
<th>Fat</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>nnnnn</td>
<td>nnn</td>
<td>aaaaaaa</td>
</tr>
<tr>
<td>nnnnn</td>
<td>nn</td>
<td>aaaaaaa</td>
</tr>
<tr>
<td>nnnnn</td>
<td>nnn</td>
<td>aaaaaaa</td>
</tr>
<tr>
<td>nnn</td>
<td>nn</td>
<td>enter new item here</td>
</tr>
</tbody>
</table>

### Average Daily Calories

<table>
<thead>
<tr>
<th>day</th>
<th>nnnnn</th>
<th>nnn</th>
</tr>
</thead>
<tbody>
<tr>
<td>week</td>
<td>nnnnn</td>
<td>nnn</td>
</tr>
<tr>
<td>diet</td>
<td>nnnnn</td>
<td>nnn</td>
</tr>
</tbody>
</table>

**status line**
Containment Hierarchy

Diet JFrame
  menu JMenuBar
    file JMenu
      new, open, save, saveAs JMenuItem
      diet JMenu
      history, foods, graphWeeks JMenuItem
    food JMenu
  mainPane componentPane (BorderLayout)
  tools JToolBar [North]
    delete, search, add JButton
  target JTextArea
  today JScrollPane [Center]
  table JTable

southPanel JPanel or Box [South]
  averages JLabel
  statistics JPanel (Box - H) -- next page
  status JLabel
southPanel    Box  (Y_AXIS)
averages JLabel
statistics    Box  (X_AXIS)
statLabels    Box  (Y_AXIS)
    day, week, diet JLabel  -- set sizes
statDay       Box  (Y_AXIS)
    calDay, calWeek, calDiet JLabel  -- set sizes
statFat       Box  (Y_AXIS)
    fatDay, fatWeek, fatDiet JLabel  -- set sizes
histogram JPanel  -- set sizes
    -- do 2D graphics drawing here  -- how to update?
status       JLabel
### GridBag example

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>day</td>
<td>nnnnnn</td>
<td>nnn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>week</td>
<td>nnnnnn</td>
<td>nnn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>diet</td>
<td>nnnnnn</td>
<td>nnn</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**southPanel JPanel (GridBag 5 x 4)**

- **averages JLabel (0,0, 4, 1)**
- **day JLabel (1,0,1,1) set sizes** # (x, y, rows, cols)
- **week JLabel (1,1,1,1) set sizes**
- **diet JLabel (1,2,1,1) set sizes**
- **calDay JLabel (2,0,1,1) set sizes**
- **calWeek JLabel (2,1,1,1) set sizes**
- **calDiet JLabel (2,2,1,1) set sizes**
- **fatDay JLabel (3,0,1,1) set sizes**
- **fatWeek JLabel (3,1,1,1) set sizes**
- **fatDiet JLabel (3,2,1,1) set sizes**
- **histogram JPanel (4,1,3,1) size sizes**
- **status JLabel (0, 3, 1, 4)**
Combination example

<table>
<thead>
<tr>
<th></th>
<th>day</th>
<th>week</th>
<th>diet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nnnnnn</td>
<td>nnn</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
southPanel    Box    (Y_AXIS)
averages      JLabel
statistics     Box    (X_AXIS)
stats          JPanel GridBag (fill Both)
day, week, diet JLabel    column 1  -- set sizes
calDay, calWeek, calDiet  JLabel column2  -- set sizes
fatDay, fatWeek, fatDiet  JLabel column3  -- set sizes
histogram       JPanel    -- set sizes
                  -- do 2D graphics drawing here  -- how to update?
status          JTextField
```
Initial display
GridBag demo is smaller horizontally because Box and Combo demos have two rigidAreas

Note: Box doesn't align labels to the right

Demos have been resized horizontally and vertically.

Horizontal resize is more critical test.

Combo has better display properties (w/ horizontal ridge)
Event Sequences

File Menu, actLis --- JFileChooser

Diet Menu
  diet list, actLis --- JDialog, JTable displays diet history by days
  foods actLis --- JDialog, JTable displays food data file
  graph, actLis --- JDialog, JPanel displays histogram of daily avgs

HelpMenu
  author, actLis --- JOptionPane
  usage, actLis --- start html help in a browser

"-" button, actLis --- delete selected row from table

"+": button, actLis --- add empty row to table, show row for edit input

"search" button, actLis --- JDialog, JTable with all items in foods, user selects item, "ok" adds item and shows item in table

target, textField, actLis --- getText(), JTable with all hits from foods dataModel that match target string, user item, "ok" adds item and shows item in table
MVC

Model/ViewController architecture more obvious in selection and view oriented UIs: JScrollPane, JLists, JComboBox, JTable, JEditorPane (JTextPane).

Scrolling viewports present a smaller "clipped" view of larger data set (model).

Model must keep
- horizontal and vertical base offsets
- horizontal and vertical lengths
- horizontal and vertical % of total area

View/Controller updates offsets and lengths on scrollbar modification events.

Selection UIs maintain collections and sets of selected items
Tables and text panes maintain text, formatting, embedded objects...
Scroll Panes

JScrollPane container manages the “viewport” on it’s component. Viewport is smaller window displaying parts of the larger scene. Scroll bar has a knob, buttons, and a track.

ScrollPolicy = {*_AS_NEEDED, *_ALWAYS, *_NEVER}
  VERTICAL_SCROLLBAR_AS_NEEDED
  *_AS_NEEDED is default

JScrollPane(Component);

To set vertical and horizontal scroll policy

JScrollPane(Component, int , int);
JScrollPane(int, int);
new JScrollPane(
  VERTICAL_SCROLLBAR_ALWAYS,
  HORIZONTAL_SCROLLBAR_NEVER)
JScrollPane display area

center component display

optional 4 corners & 4 sides.

scrollbars or custom “rules” in sides

adjacent sides needed for corner to be visible

Some API features:

setAutoscrolls(boolean)    
drags in component cause scrolling

component must support scrolling

setVisibleRowCount(int)    
number visible rows

ensureIndexIsVisible(int)    
for list, scrolls to show specified index

Other intermediate containers:

JSplitPane, JTabbedPane, JToolBar, JInternalFrame, JLayeredPane
JList

Container with selectable items (see also JComboBox, JTable)

JLists use a ListModel, or array of objects, or vector
  JList(ListModel) presents a dynamic set of items
  JList(Object[]) or JList(Vector) presents static list items.
  static lists == menu of choices (colors, fonts, sizes, zip codes...)

list have a selection model

  SINGLE_SELECTION   one item
  SINGLE_INTERVAL_SELECTION one range adjacent items
  MULTIPLE_INTERVAL_SELECTION multiple ranges

aList.addListSelectionListener(this) requires implementation of method

  public void valueChanged(ListSelectionEvent e) {....}

with single selection getSelectedIndex() returns the index of the item
setSelectedIndex(anInt) selects item at index anInt

<< walkthrough ObjectListDemo.java >>
ObjectListDemo

Maintains a list of objects.

Each object has an item and a rank (randomly set)

Items sortable by rank.

Icon button labels

Floating toolbar
DefaultListModel

int getSize(); // list count

void setSize(int size);

Object get(int index); // returns index element
Object getElementAt(index);

void setElementAt(Object object, index);

Object set(index, object); // replace index object
  // returns old object

void add(index, object);
void addElement(object); // add at end of list

void insertElement(object, index); // add at index

Object remove(index); // cuts && returns object

void removeElementAt(index);

boolean removeElement(object); // cuts 1st object found

void removeAllElements();
Event Diagram

**listSelectionEvent**

- **ListSelectionListener** valueChanged

**ActionEvent**

- **MoveListener moveIt**

**ActionEvent**

- **EditListener**

**ActionEvent**

- **SortListener**

**ActionEvent**

- **DoneListener**

**ActionEvent**

- **ToDoListener**

**WindowEvent**

- **EXIT_ON_CLOSE**

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listSelectionEvent</td>
<td>Set selected index, buttons item → textfield</td>
</tr>
<tr>
<td>ActionEvent</td>
<td>Move item up or down</td>
</tr>
<tr>
<td>ActionEvent</td>
<td>Modify item string</td>
</tr>
<tr>
<td>ActionEvent</td>
<td>Sort items</td>
</tr>
<tr>
<td>ActionEvent</td>
<td>Delete item</td>
</tr>
<tr>
<td>ActionEvent</td>
<td>Add item</td>
</tr>
<tr>
<td>WindowEvent</td>
<td>Exit app.</td>
</tr>
<tr>
<td>Event</td>
<td>Listener</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>ListSelectionEvent</td>
<td>ListSelectionListener</td>
</tr>
<tr>
<td></td>
<td>valueChanged</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ActionEvent</td>
<td>MoveListener</td>
</tr>
<tr>
<td></td>
<td>moveIt</td>
</tr>
<tr>
<td>ActionEvent</td>
<td>EditListener</td>
</tr>
<tr>
<td>ActionEvent</td>
<td>SortListener</td>
</tr>
<tr>
<td>ActionEvent</td>
<td>DoneListener</td>
</tr>
<tr>
<td>ActionEvent</td>
<td>ToDoListener</td>
</tr>
<tr>
<td>WindowEvent</td>
<td>EXIT_ON_CLOSE</td>
</tr>
</tbody>
</table>
JTable

Displays model by rows with columns
row cells can have renders.

Displays column headers.
row cell values can be editable.
can have custom cellEditors -- column editors

Model can be derived from AbstractTableModel if subclass is needed or
DefaultTableModel can be instantiated (no subclass needed).

TableModelListener detects changes to data.

```
void tableChanged(TableModelEvent e) {...}
```

DefaultTableModel can manipulate data order
```
moveRow(int start, int end, int to) // from to
```

TableSorter can be use to order the model
Cell editors use UI controls (JComboBox, JCheckBox) to select values for cells.

Cell renderers use UI controls (JLabel, JCheckBox) to represent the values of cells.

Cell renderers can be for a column, Class type, or individual cells.

Date column used a customized DefaultTableCellRenderer
Calories and Fat columns uses a JComboBox cell editor.
Calories column uses a customized DefaultTableCellRenderer
Fat column uses the DefaultTableCellRenderer
Exercise column uses a JCheckbox cell editor and renderer.
class MyCustomRenderer extends DefaultTableCellRenderer {
    public Component getTableCellRendererComponent(
        JTable t, Object value, boolean isSelected,
        boolean hasFocus, int row, int col) {
        ...
        setValue(...);
        ...
        return this;
    }
}

class MyCheckboxRenderer extends JCheckBox implements TableCellRenderer { ... }
If not JTextField, JCheckBox, or JComboBox you need to:

- substituted the component to be the editor.
- set the default editor – setDefaultEditor or setCellEditor

see Oracles's [TableDialogEditDemo.java](http://example.com) example and description on the Sun Java GUI trail (under Tables).

Substitutes a button (for a JCheckbox) that is used to display a JColorChooser to get a color for the cell's background color.
TextComponents

![Diagram of text components]

- **JTextComponent**
  - **JTextField**
  - **JTextArea**
  - **JPasswordField**
  - **JEditorPane**
  - **JTextPane**

**JTextComponents**

- **document**, the model, manages components content
- **view** displays content
- **editor kit**, implements editing of content (**controller**) supports infinite undo / redo
- **pluggable caret and caret change listeners**

**JEditorPane** easy way to have *.rtf files created to load document for help dialogs – can also be hyper text (**html editor kit**).
Persistence

Java supports object persistence with serialization (short term) and XML encoding/decoding (long term).

Serialized objects of JTable (and others) will not be compatible with future Swing releases.

XML persistence recommended.

Using XMLEncoding, XML encoding API, and XML decoding API

JTable's public `TableModel` `getModel()` can get the current model

DefaultTableModel's public `Vector` `getDataVector()` returns the Vector of Vectors that contains the table's data values.

The vectors contained in the outer vector are each a single row of values. In other words, to get to the cell at row 1, column 5:

```
((Vector)getDataVector().elementAt(1)).elementAt(5);
```

see XMLListDemo.java
Java J2SE and J2EE SDKs provide the JDBC Java Database Connection API in the `java.sql` package.
2 tier architecture

Java Application

JDBC

DBMS

Client machine

DBMS proprietary protocol

Database server

3 tier architecture

Java applet or HTML browser

Java Application Server

JDBC

DBMS

Client machine (GUI)

HTTP, RMI, CORBA, … calls

Server machine (business logic)

DBMS proprietary protocol

Database server
The java.sql package defines the JDBC API for sending SQL queries and update statements to a database, and for obtaining query results. This example demonstrates how to use JDBC to connect to a database and obtain a java.sql.Connection object, how to execute a SQL query using a java.sql.Statement object, and how to work with the query results returned in a java.sql.ResultSet object.

JTable can display any type of tabular data as long as it implements the javax.swing.table.TableModel interface. The examples in this article include a TableModel implementation that interprets a JDBC ResultSet object, so that the queried data can be displayed by a Jtable.

O'Reilly Java in a nutshell example