C13b, AWT cont.

Panel

- Container that acts like a Component
- Can be added into other components (like Frames, other Panels)

```
private Panel makeScrollBars() {
    Panel p = new Panel();
    p.setLayout(new BorderLayout());
    p.add("West",redBar);
    p.add("Center",greenBar);
    p.add("East",blueBar);
    return p; }
```

ScrollPane

- Can hold exactly one other component (➔ no LayoutManager)
- Automatic scrollbars if Component larger than ScrollPane

```
class BigCanvas extends Frame {
    public static void main(String[] args) {
        BigCanvas w = new BigCanvas(); w.show(); }

    private Polygon poly = new Polygon();
    private Canvas cv = new Canvas();
    …
```

```
public BigCanvas() {
    setSize(300,300);
    setTitle("Scroll Pane Test");
    cv.setSize(1000,1000);
    cv.addMouseListener(new MouseKeeper());
    ScrollPane sp = new ScrollPane();
    sp.add(cv);
    add("Center",sp);
    }
public void paint(Graphics g) {
    Graphics gr = cv.getGraphics();
    gr.drawPolygon(poly);
    }
private class MouseKeeper extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        poly.addPoint(e.getX(),e.getY());
        repaint();
        }}}
```

Case study: Color display

```
private class BrightenButton extends Button implements ActionListener
   {
    private int index;
    public BrightenButton(int i) {
        super( i == 0 ? "brighter" : "darker");
        index = i;
        addActionListener(this);
        }
    public void actionPerformed(ActionEvent e) {
        setFromColor(index == 0 ? current.brighter(): current.darker());
        }
    }
```

Dialogs

- Special purpose window displayed shortly
- To notify, or ask simple questions
- Always attached to a Frame
- Modal: demands user response, prevents further actions; show() won't return until dismissed, ➔ setVisible(false)
- Nonmodal: actions often placed into separate Thread

```
Dialog dig = new Dialog(this,false);
```

```
class Dialog extends Frame {
    static public void main(String[] args) {
        DialogTest w = new DialogTest(); w.show();}
    private TextArea d = new TextArea();
    private Checkbox cb = new Checkbox("Modal Dialog?");
    public DialogTest() {
        setTitle("Dialog Test Program");
        setSize(300,220);
        add("West",cb);
        add("East", new MakeButton());
        add("South",d);
        }
    private class MakeButton extends ButtonAdapter {
        public MakeButton() { super("Make Dialog"); }
        public void pressed() { makeDialog(cb.getState());}
        }
```

```
private void makeDialog(boolean modalFlag) {
    final Dialog dlg = new Dialog(this,modalFlag);
    dlg.setSize(100,100);
    dlg.add("North",new CountButton(1));
    dlg.add("West",new CountButton(2));
    dlg.add("East",new CountButton(3));
    dlg.add("South", new ButtonAdapter("Hide") {
        public void pressed() {dlg.setVisible(false);}});
    dlg.show();
    }
private class CountButton extends ButtonAdapter {
    public CountButton(int val) {super("" + val);}
    public void pressed() {d.append("Button " + getLabel() +
                                    " pressed\n");
    }}}
```

MenuBar

- Not subclass of component, but graphical!
- Attached to Frame by setMenuBar():
  ```
  MenuBar bar = new MenuBar();
  setMenuBar(bar);
  ```
- Add menus:
  ```
  Menu HelpMenu = new Menu("Help");
  bar.add(helpMenu);
  ```
- Add menu items:
  ```
  MenuItem quitItem = new MenuItem("Quit");
  quitItem.addActionListener(new QuitListener());
  helpMenu.add(quitItem);
  ```

```
class QuitItem implements ActionListener {
    private void m1(MenuItem mItem) {
        mItem.addActionListener(this);
        }
    private void m2(Menu menu) {
        MenuItem mItem = new MenuItem("Quit");
        m1(mItem);
        menu.add(mItem);
        }
    private void m3(MenuBar mBar) {
        Menu menu = new Menu("Quit");
        mBar.add(menu);
        m2(menu);
        }
    private void m4(Frame application) {
        MenuBar mBar = new MenuBar();
        application.setMenuBar(mBar);
        m3(mBar);
        }
```

```
public QuitItem(Frame application)
    {m4(application);}
public QuitItem(MenuBar mBar)
    {m3(mBar);}
public QuitItem(Menu menu)
    {m2(menu);}
public QuitItem(MenuItem mItem)
    {m1(mItem);}
public void actionPerformed(ActionEvent e)
    {System.exit(0);}
```

- Use like in:
  ```
  class ColorTest extends Frame { …
      public ColorTest() { …  new QuitItem(this);  … }}
  ```