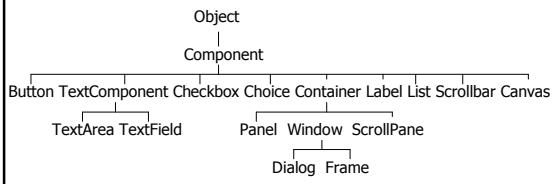
C13a, AWT

Create, display, facilitate user interaction with window objects

software framework: a way of structuring generic solutions to common problems (uses polymorphism)

AWT class hierarchy



(some) Component methods

- setEnabled(boolean)
- setLocation(int,int),getLocation()
- setSize(int,int),getSize()
- setVisible(boolean)
- setForeground(Color),getForeground()
- setBackground(Color),getBackground()
- setFont(Font),getFont()
- repaint(Graphics), paint(Graphics)
- addMouseListener(MouseListener)
- addKeyListener(KeyListener)

Some AWT classes

- Component: 2D screen for user interaction
- Container: component that can nest other components within itself:
  - setLayout(LayoutManager)
  - add(Component),remove(Component)
- Window: a Container, that can be displayed, can stack windows:
  - show()
  - toFront()
  - toBack()

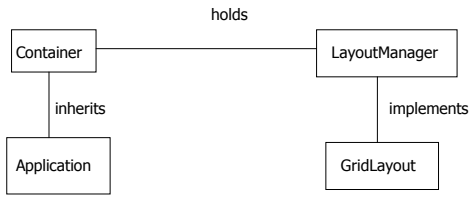
Frames

- Frame: a window with title bar, menu bar, cursor, border, …
  - setTitle(String),getTitle()
  - setCursor(int)
  - setResizable()
  - setMenuBar(MenuBar)
- Application class CannonWorld (from ch.6) uses:
  - setTitle(String) inherited from Frame
  - setSize(int,int) from Component
  - show() from Window
  - repaint() from Component
  - paint() overridden (inherited from Component)

Polymorphism again:

- Parent class code (Component, Window, …) written WITHOUT any reference to a particular application
- Application: simple override necessary bits and pieces (e.g. paint(), listeners) to define application-specific behavior
- Reuse: combine inheritance, overriding, and polymorphism

Layout manager



Again: combine inheritance, composition, interface implementation

5 standard LayoutManager types

- BorderLayout: max. 5 components
- GridLayout: rectangular array of equal-sized components
- FlowLayout: place components left-to-right, top-to-bottom, variable-sized
- CardLayout: stack components vertically, only one visible at a time
- GridBagLayout: non-uniform grid of squares, most flexible

LayoutManager code examples

```
Panel p = new Panel();
p.setLayout(new GridLayout(4,4,3,3));
p.add(new ColorButton(Color.black,"black"));

CardLayout lm = new CardLayout();
Panel p = new Panel(lm);
p.add("One", new Label("Number one"));
p.add("Two", new Label("Number two"));
…
lm.show(p,"Two");
```

User interface components

- Label:
  Label lab = new Label("score: 0 to 0");
  add("South", lab);
  getText(), setText(String)
- Canvas:
  – Simple component, can be target for drawing operations (see ScrollPane example)

Buttons and ActionListeners

```
Button b = new Button("do it!");
b.addActionListener(new DoIt());
…
private class DoIt implements ActionListener {
      public void actionPerformed( ActionEvent e) {
            // whatever
            }}
```

- **Alternative: inheritance + interface implementation**:

```
private class ColorButton extends Button implements ActionListener {
      private Color c;
      public ColorButton(Color c1, String name) {
            super(name);
            c = c1;
            addActionListener(this);   //  ← OURSELVES !!!
            }
      public void actionPerformed(ActionEvent e) { setFromColor(c);}}
```

Abstract ButtonAdapter class

```
abstract class ButtonAdapter extends Button implements
   ActionListener {
    public ButtonAdapter(String name) {
        super(name);
        addActionListener(this); }
    public void actionPerformed(ActionEvent e) {pressed();}
    public abstract void pressed(); }
```

- Use with anonymous class:

```
Panel p = new Panel();
p.add(new ButtonAdapter("Quit") {public void pressed()
            {System.exit(0);}} );
```

Scrollbar

- A slider to specify integer values
- Same trick as above:

```
private class ColorBar extends Scrollbar implements
   AdjustmentListener {
    public ColorBar(Color c) {
        super(Scrollbar.VERTICAL,40,0,0,255);
        setBackground(c);
        addAdjustmentListener(this); }// ← ourselves!!
    public void  adjustmentValueChanged(AdjustmentEvent e) {
        setFromBar(); }}  // possibly using getValue()
```

Text components

- TextField: fixed-sized block
- TextArea: uses scrollbars for text larger than the area
- setText(String), getText()
- append(String) for TextArea only
- As always need listener:

```
interface TextListener extends EventListener {
    public void textValueChanged(TextEvent e); }
```

Checkbox

- Maintain/display labeled binary state (on/off, yes/no)
- getLabel(), setLabel(String),getState(), setState(String)
- And again, a listener:
  - An ItemListener for an ItemEvent

Checkbox code example

```
class CheckTest extends Frame {
    private Checkbox cb = new Checkbox("off");
    public static void main(String[] args)
        {CheckTest w = new CheckTest(); w.show();}
    public CheckTest() {
        setTitle("CheckBox"); setSize(300,70);
        cb.addItemListener( new CheckListener());
        add("Center", cb); }
    private class CheckListener implements ItemListener {
        public void itemStateChanged(ItemEvent e) {
            cb.setLabel((cb.getState()) ? "ON" : "OFF"); }}}
```

Checkbox groups, Choices, Lists

- Select one of a number of possibilities
  - CheckboxGroup (radio buttons): only one active, use for small groups (<5)
  - Choice: display current selection, pop-up menu for alternatives
  - List: display some alternatives

```
class ChoiceTest extends Frame {
    public static void main(String[] args) {
        ChoiceTest w = new ChoiceTest(); w.show();}
    private String[] c = {"One", …, "Ten"};
    private Label display = new Label();
    private Choice theC = new Choice();
    private List theL = new List();
    private CheckboxGroup theG = new CheckboxGroup();
    private ItemListener theListener = new ChoiceListener();
```

```
public ChoiceTest() {
    setTitle("Selection example");
    setSize(300,300);
    for(int i = 0; i<10; i++) {
        theC.addItem(c[i]); theL.addItem(c[i]); }
    theC.addItemListener(theListener);
    theL.addItemListener(theListener);
    add("West",makeCheckBoxes());
    add("North",theC);add("East",theL),add("South",display); }

private class ChoiceListener implements ItemListener {
    public void itemStateChanged(ItemEvent e) {
        display.setText(theG.getSelectedCheckboxGroup().getLabel() +
            theL.getSelectedItem() + theC.getSelectedItem()); }}

private Panel makeCheckBoxes() {
    panel p = new Panel( new GridLayout(5,2));
    for(int i = 0; i < 10; i++) {
        Checkbox cb = new Checkbox(c[i], theG, false);
        cb.addItemListener(theListener); p.add(cb); }
    return p; }}
```