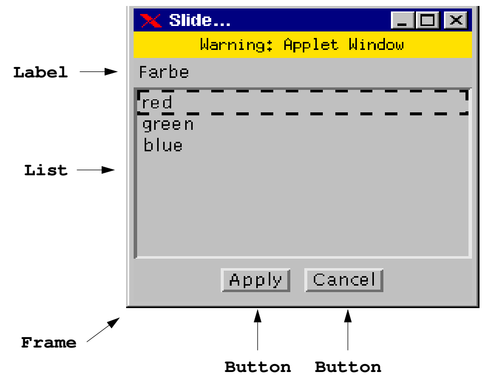
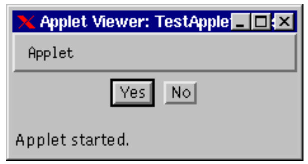
- ❏ Oft nötig: neu zeichnen eines Teilbereichs eines Fensters
- ❏ Wegen Aktionen des Fenstersystems (automatisch):
  - Fenster ist durch ein anderes verdeckt worden und wird dann nach vorn geholt
  - Fenster wird vergrößert/verkleinert
- ❏ Wegen Aktionen des Programms:
  - Der Benutzer führt eine Aktion aus, die das Fenster verändern soll (Button,...)
  - Das Programm möchte den Inhalt ändern
- ❏ Vorgehensweise: repaint( ) aufrufen: "Inhalt des Fensters ist ungültig"
- ❏ Das Fenstersystem ruft dann bei Gelegenheit "paint" auf
- ❏ Auch möglich: Teilbereiche neu zeichnen oder "explizites malen"

- ❏ X-Windows-Programme enthalten oft "Bausteine":
  - Knöpfe, Listen, Eingabefelder, Checkboxen, Radiobuttons, ...
- ❏ Dafür: vordefinierte Klassen
- ❏ Auch möglich: eigene Klassen
  - Entweder ohne Vorgabe: von Component abgeleitet
  - Dann müssen alle entsprechenden Events implementiert werden (Mausklick,...)
  - Oder durch Veränderung von existierenden Klassen: z.B. von List ableiten
- ❏ Ein "Baustein" (Instanz einer Component-Subklasse)
  - Verwaltet einen rechteckigen Bildschirmbereich
  - Bekommt alle Events, die der Bereich erhält



- ❏ Möglich: ableiten z.B. von Button, "Klick" abfangen
- ❏ Problem: Programm wird zu komplex
- ❏ daher: automatische Weiterleitung von Events an das übergeordnete Fenster
- ❏ Beispiel:



Java im World Wide Web

Komponenten

```
public class TestApplet extends Applet {

      public void init() {
            add(new Button("Yes"));
            add(new Button("No"));
      }

      public boolean action(Event evt, Object arg) {
            System.out.println("TestApplet::action "+evt);
            if (evt.target instanceof Button) {
                  if ("Yes".equals(arg)) {
                        System.out.println("TestApplet: Yes ");
                        return true;
                  }
                  if ("No".equals(arg)) {
                        System.out.println("TestApplet: No ");
                        return true;
                  }
            }
            return false;
      }
}
```


Java im World Wide Web

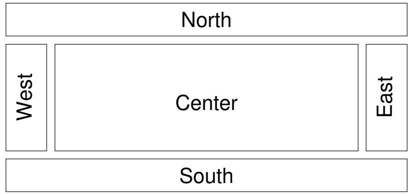
Panels

- ❏ Problem: Anordnung der Elemente
- ❏ Möglichkeit 1: Absolute Positionierung (Beispiel: MS-Windows)
  - Problem: Benutzer kann Fenstergröße nicht ändern
- ❏ Möglichkeit 2: Anordnung mit Panels
- ❏ Prinzip:
  - Ausgegangen wird von einem rechteckigen "Panel"
  - Ein Panel enthält Komponenten und möglicherweise weitere Panels
  - Jedes Panel hat eine gewisse Ausrichtungsstrategie
  - Die Komponenten werden gemäß der Ausrichtungsstrategie ausgerichtet
  - Panels selbst sind unsichtbar

Java im World Wide Web

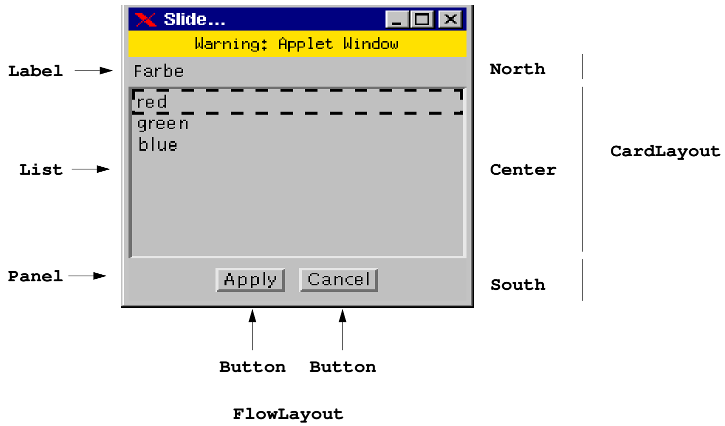
Komponenten

- ❏ Ausrichtungsstrategien:
  - FlowLayout (von links nach rechts zeilenweise, Zeilen zentriert)
  - GridLayout (Gitter)
  - BorderLayout: 5 Komponenten, oben, unten, links, rechts, mitte



Java im World Wide Web

Beispiel



Java im World Wide Web

Komponenten

```
class SliderFrame extends Frame {
      List l;                          // Die Liste als Komponente
      public boolean action(Event evt, Object arg) { ...
      }     // Auf Knöpfe reagieren
      public SliderFrame() {
            super("Slide...");

            Panel p = new Panel();       // Das "Border-Panel"
            l = new List(3,true);        // die Liste allokieren
            l.addItem("red"); ....       // und füllen
            p.setLayout(new BorderLayout());
            p.add("North",new Label("Farbe")); // Label nach oben
            Panel p2=new Panel();        // Das "Flow-Panel"
            p2.setLayout(new FlowLayout());
            p2.add(new Button("Apply"));.... // Die Knöpfe rein
            p.add("South",p2);           // p2 als South-Teil
            p.add("Center",l);           // die Liste als center
            add("Center",p);             // Das Panel als center
                                         // ins Fenster
            pack();                      // alles ausrichten...
      }
}
```


Java im World Wide Web

Komponenten

❑ Benutzung der Klasse:

```
new SliderFrame().show();
```

❑ Auch in dem Applet-Bereich lassen sich Panels, Knöpfe etc. anlegen

❑ dazu: einfach in der "init" Methode des Applets die Elemente anlegen

❑ mit "add" dem Applet hinzufügen

❑ Vorsicht bei separaten Fenstern (SliderFrame):

- Bei netscape muß man selbst die zus. Fenster entfernen, wenn die Seite gewechselt wird
- Dafür: "start" und "stop" Methode des Applets

Java im World Wide Web

Grafik

❑ Besondere Unterstützung von Grafik (nötig für WWW)

❑ Laden und anzeigen von jpg und gif Bildern

❑ Spezielle Rücksichtnahme auf WWW-Anforderungen

- Oft langsame Verbindungen
- Daher: asynchrones laden: Alle Bilder werden gleichzeitig geladen

❑ Prinzip:

- Image-Objekte enthalten je ein teilweise/vollständig geladenes Bild
- MediaTracker-Objekt überwacht das Laden der Bilder
- Synchrones Laden:
  Alle Ladevorgänge asynchron starten, dann auf alle warten
- Asynchrones Laden:
  Alle Ladevorgänge asynchron starten und während des Ladens anzeigen

Java im World Wide Web

Grafik

```
public class TGraphic extends Applet {
      public void init() {
            MediaTracker tracker = new MediaTracker(this);
            Image i,j;
            i=getImage(getDocumentBase(),"test1.jpg");
            j=getImage(getDocumentBase(),"test2.jpg");

            tracker.addImage(i,0);  // Image, load-ID
            tracker.addImage(j,1);

            tracker.checkAll(true); // mit laden anfangen
            // tracker.waitForAll(); das würde blockieren
      }
      ...
      public void paint(Graphics g) {
            g.drawImage(i,          // Grafik
                        0,0,        // Position
                        this);      // Notify beim weiterladen
      }
}
```