CSC 133
Object-Oriented Computer Graphics Programming

GUI

Dr. Kin Chung Kwan

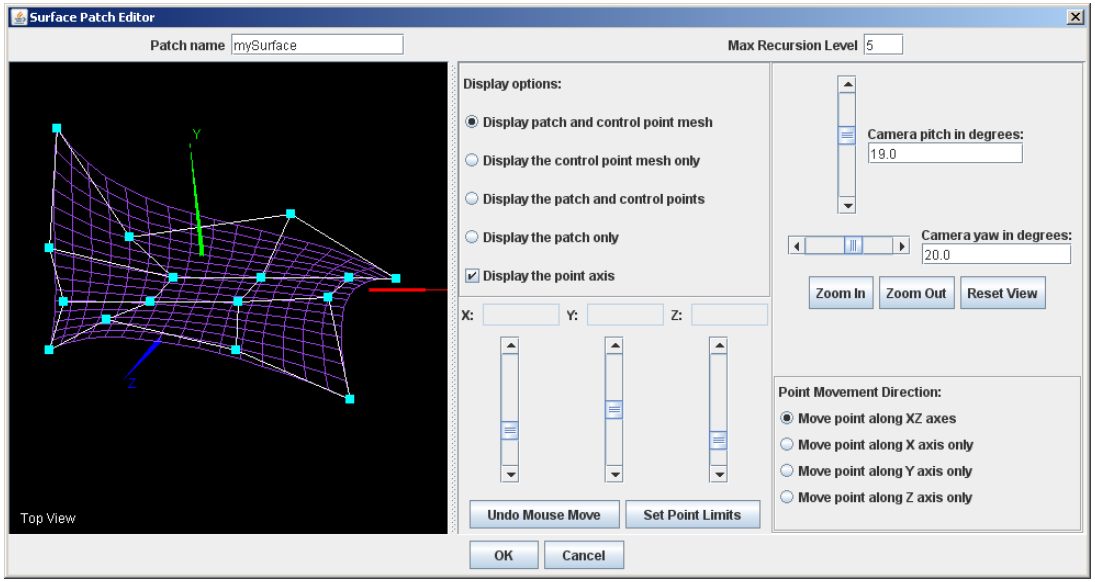
Spring 2023

Computer Science Department
California State University, Sacramento

SACRAMENTO STATE

Graphical User Interfaces

- GUI
- Input / Output



GUI Frameworks

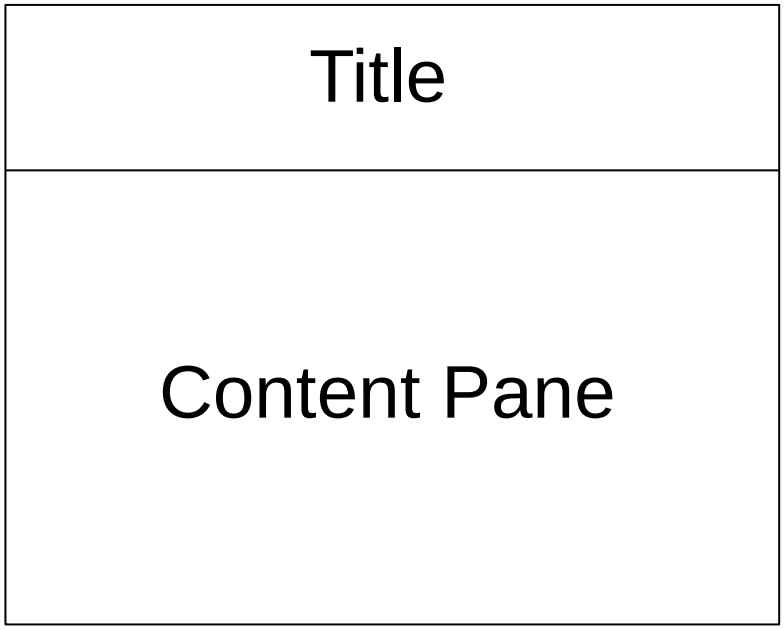
- Collection of classes that take care of low-level details of drawing on screen. Provides:
  - A *set of reusable* *screen components*
    - An object having a *graphical representation* that can *interact* with the user
  - An *event mechanism* connecting “actions” to “code”
  - *Containers* and *Layout Managers* for arranging things on screen
  - Some other packages…

Examples of GUI Frameworks

- **Microsoft Foundation Classes** (**MFC**): For C++ development on Windows (not built-in to C++)
- **AWT**: Java's first built-in GUI package
- **JFC/Swing**: Java's efficient built-in GUI package
- **UI**: CN1's GUI package

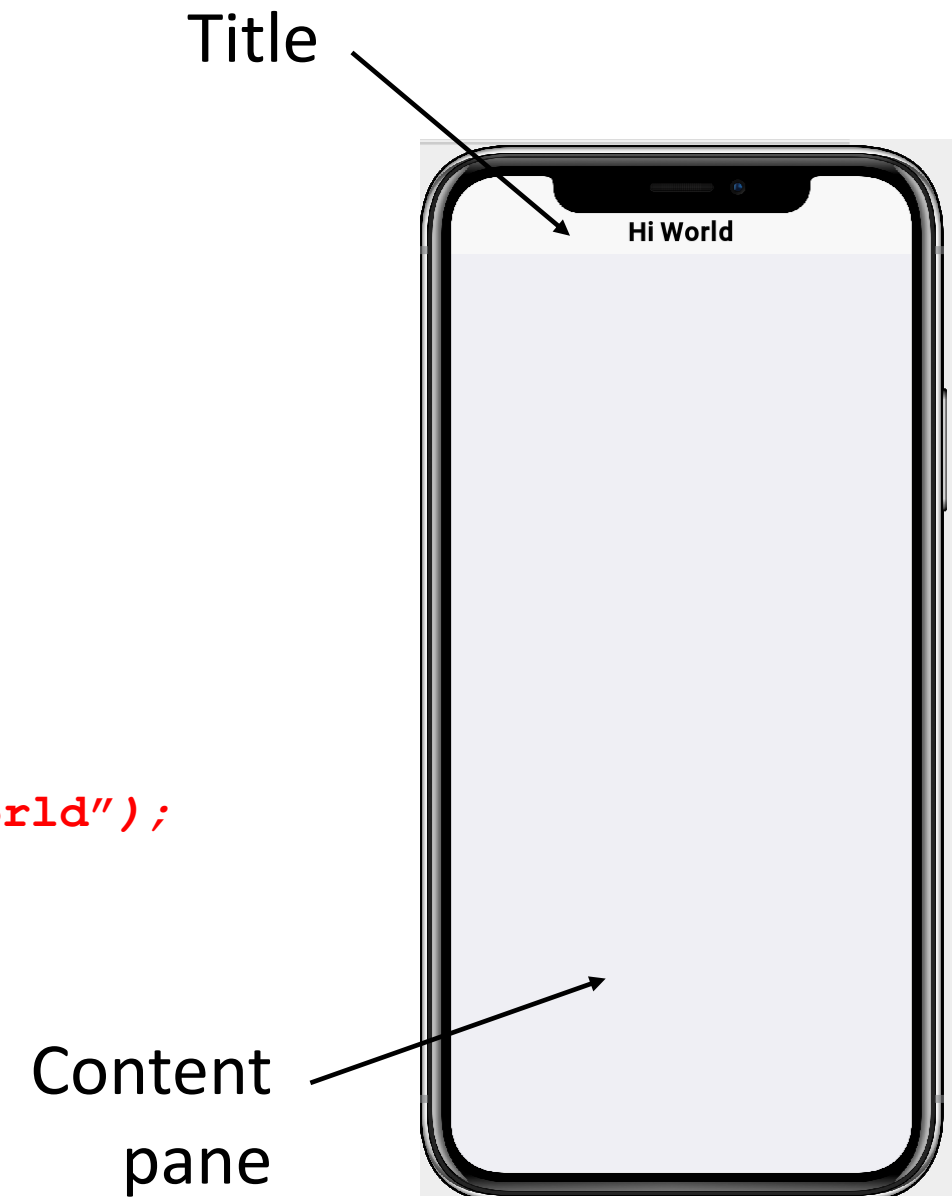
Creating a Form in CN1

- The top-level container of CN1 (like **`JFrame`** in Swing)
- Only one form can be visible at any given time
- Contains title and a content pane



Code

```
public void start() {
    if(current != null){
        current.show();
        return;
    }
    Form hi =
         new Form("Hi World");
    hi.show();
}
```



Or Extend Form

```
import com.codename1.ui.Form;
public class MyForm extends Form{
    public MyForm() {
        ...
        this.show();
        ...
    }
}
```

Add new codes here

In `start()` of the main class, call

```
new MyForm();
```

CN1 `Display` class

- Central class that manages rendering/events
- Used to place Form on the display.
- Static method `Display.getInstance()` returns the `Display` instance.
- To get the resolution:
  - `getDisplayWidth()` and `getDisplayHeight()`
- `getCurrent()` return the form currently displayed on the screen
  - null if no form is currently displayed.

Code of Display

```
import com.codename1.ui.Form;
public class MyForm extends Form{
    public MyForm() {
        this.show();
        Display d = Display.getInstance();
        System.out.println(d.getCurrent());
        System.out.println(d.getDisplayWidth());
        System.out.println(d.getDisplayHeight());
    }
}
```

```
MyForm[x=0 y=0 width=1125 height=2436 name=null,
1125
2436
```

Dialog

- Use `Dialog.show()`
- Parameter
  - Title
  - Message
  - OK msg
  - Cancel msg
- Return
  - Boolean

Closing App in CN1

Click OK to quit the application

```
Boolean bOk = Dialog.show("Confirm quit?", "Are you
sure you want to quit?", "OK", "Cancel");
if (bOk) //check if ok or cancel
    Display.getInstance().exitApplication(); //exit
```

Type of Dialog

- There are many type of `Dialog` in CN1
- Use `show()` with different parameters

- See

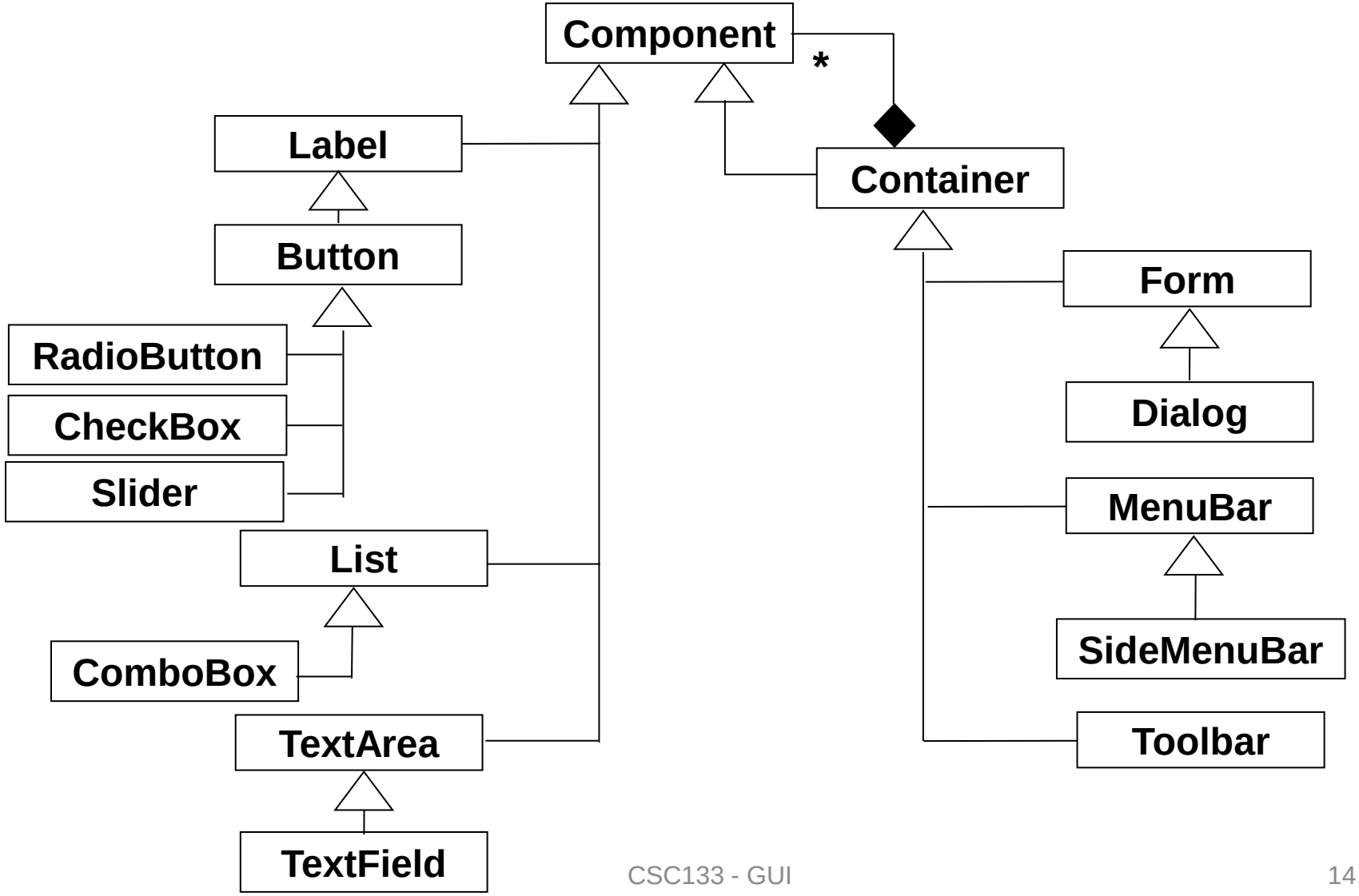
https://www.codenameone.com/javadoc/com/codename1/ui/Dialog.html#show-java.lang.String-com.codename1.ui.Component-com.codename1.ui.Command...-

Adding Components

- You can add any components onto the content pane of the `Form`

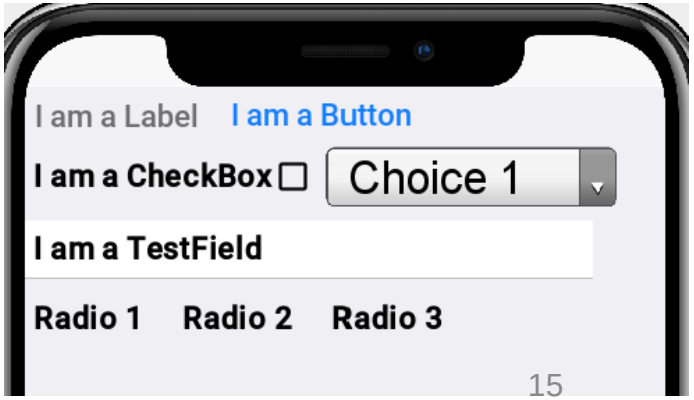
- Call `addComponent()` or `add()`

Important CN1- UI Components



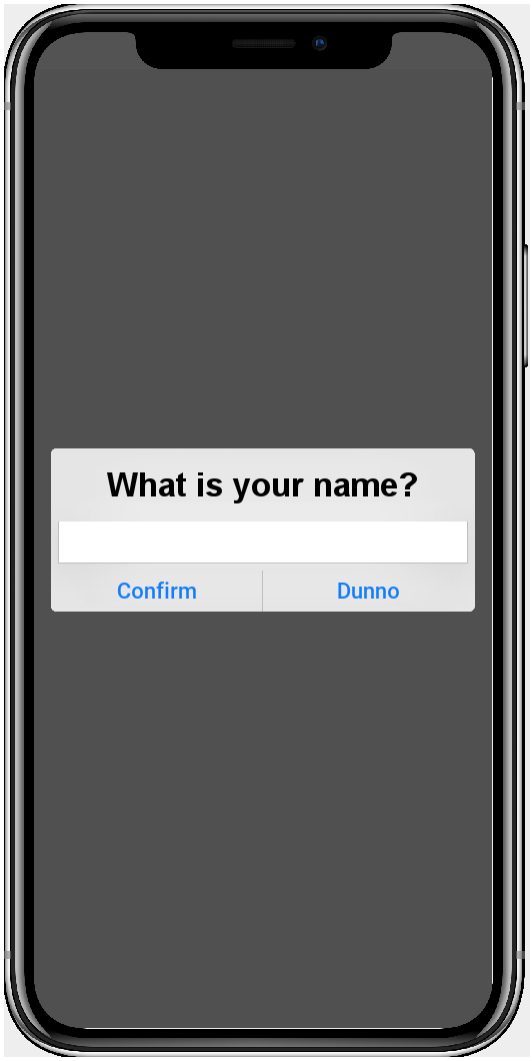
Adding Components

```
Label myLabel = new Label("I am a Label");

addComponent(myLabel);

Button myButton = new Button("I am a Button");

addComponent(myButton);

CheckBox myCheck = new CheckBox("I am a CheckBox");

addComponent(myCheck);

ComboBox myCombo = new ComboBox("Choice 1","Choice
2","Choice 3");

addComponent(myCombo);

TextField myTF = new TextField("I am a TestField");

addComponent(myTF);

RadioButton myRB1 = new RadioButton("Radio 1");

RadioButton myRB2 = new RadioButton("Radio 2");

RadioButton myRB3 = new RadioButton("Radio 3");

addComponent(myRB1);

addComponent(myRB2);

addComponent(myRB3);
```



Input Dialog

- Dialog.show() with Textfield
- Input Component and Command[]
- Return another Command



Code of Input Dialog

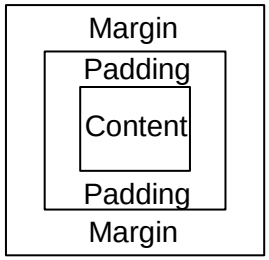
```
Command cOk = new Command("Confirm");
Command cCancel = new Command("Dunno");
Command[] cmds = new Command[]{cOk, cCancel};
TextField myTF = new TextField();
Command c = Dialog.show("What is your name?", myTF, cmds);
if (c == cOk) //check if ok or cancel
    System.out.println(myTF.getText());
else
    System.out.println("You dunno your name?");
```

CN1 `Style` class

Style of component:

- Colors
- Fonts
- Transparency
- Margin
- Padding
- Image
- Etc.

133 **133** **133** **133**

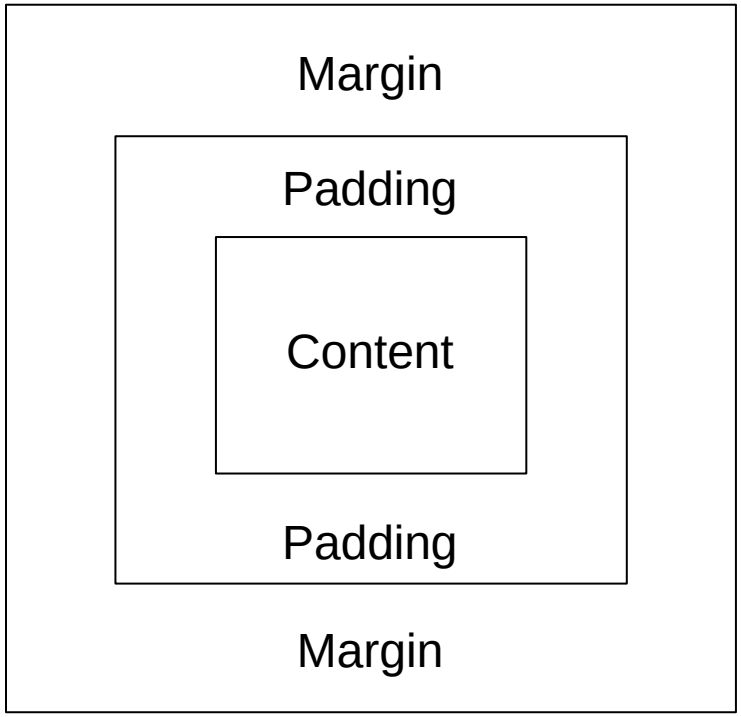


Different Condition

- Use methods in component:
  - `getStyle()`
  - `getAllStyles()`
  - `getDisabledStyle()`
  - `getPressedStyle()`
  - `getSelectedStyle()`
  - `getUnselectedStyle()`
  - Etc.

Different Style

- Use methods in style:
  - `setBgColor()`
  - `setFgColor()`
  - `setBgTransparency()`
  - `setFgAplha()`
  - `setPadding()`
  - `setMargin()`
  - Etc.

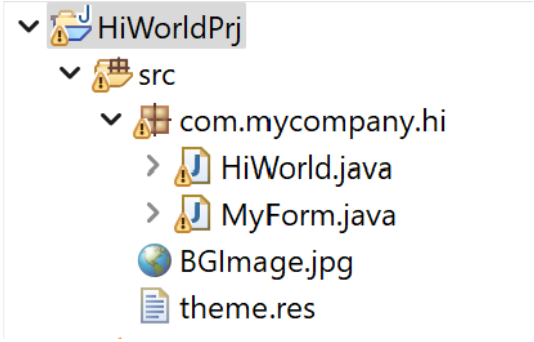


Example Code

```
button.getStyle().setBgTransparency(255);
button.getStyle().setBgColor(ColorUtil.BLUE);
button.getStyle().setFgColor(ColorUtil.WHITE);
button.getStyle().setBorder(
         Border.createLineBorder(3,ColorUtil.BLACK));
button.getAllStyles().setPadding(Component.TOP, 10);
button.getAllStyles().setPadding(Component.BOTTOM, 10);
```

Using an Image

```
try { //catch exception
    Image img =
      Image.createImage("/BGImage.jpg");
    addComponent(img);
} catch (IOException e) {
    e.printStackTrace();
}
```



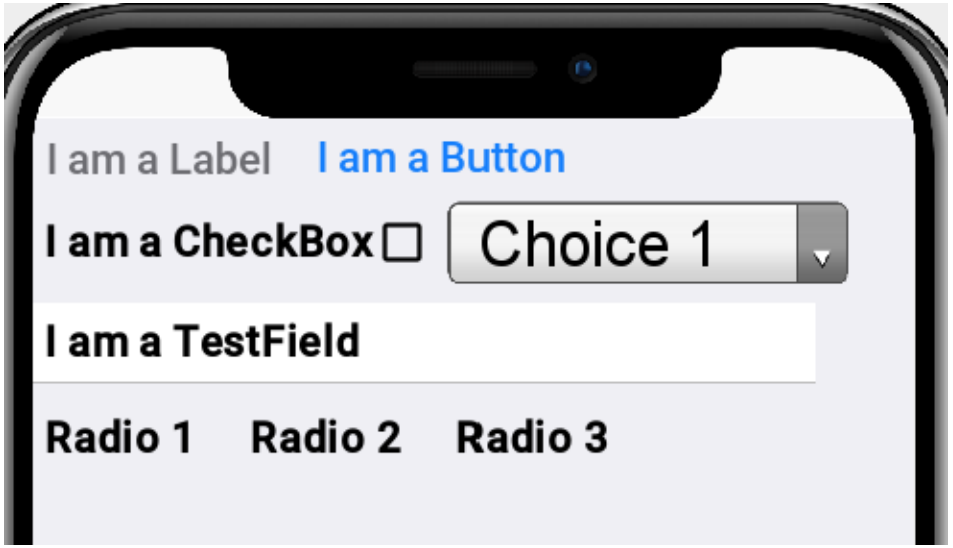
Put your image into src folder

Layout

No Layout?

- Add the component one by one.

- Stack together



Layout Managers

- Determine rules for positioning components in a container
  - Components which do not fit according to the rules may be hidden !!
- Layout Managers are classes
  - Must be instantiated and attached to their containers:

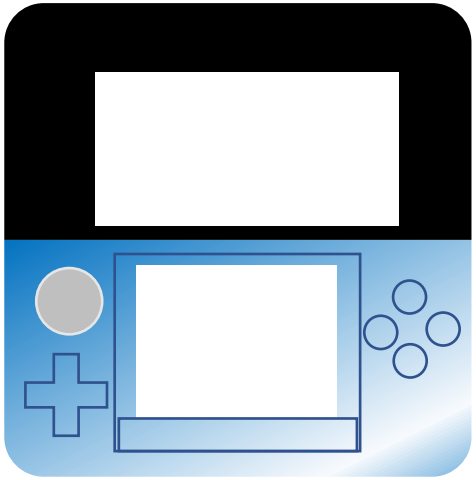
```
myContainer.setLayout( new BorderLayout() );
```

- Components can have a preferred *size*
  - Override `calcPeferredSize()` of `Component` to reach similar functionality (do not use this in the assignments)
  - Layout managers *may or may not* respect preferred size either entirely or partially

Different Layout

- FlowLayout
- BorderLayout
- BoxLayout
- GridLayout
- Etc.

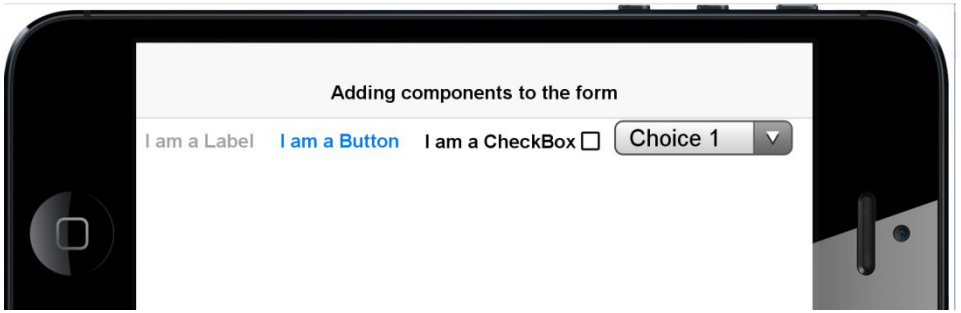
Strategy Design Pattern!



FlowLayout

- Arranges left-to-right, top-to-bottom (by default)
- Appear in the order they are added
- Respects preferred size
- Center components in the component by using:

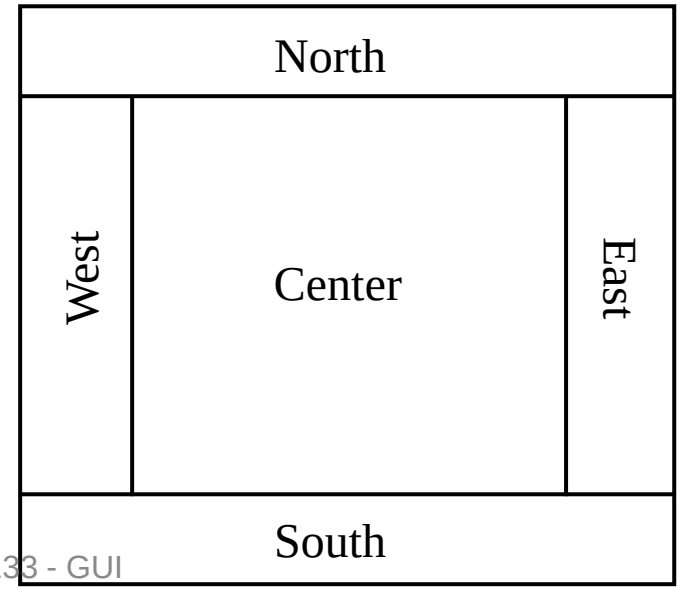
```
myContainer.setLayout(new FlowLayout(Component.CENTER));
```



BorderLayout

- Provide five “regions” of the container:
  - North, South, East, West, or Center
  - Only one component for each region

```
myContainer.add(BorderLayout.CENTER,myComponent);
```

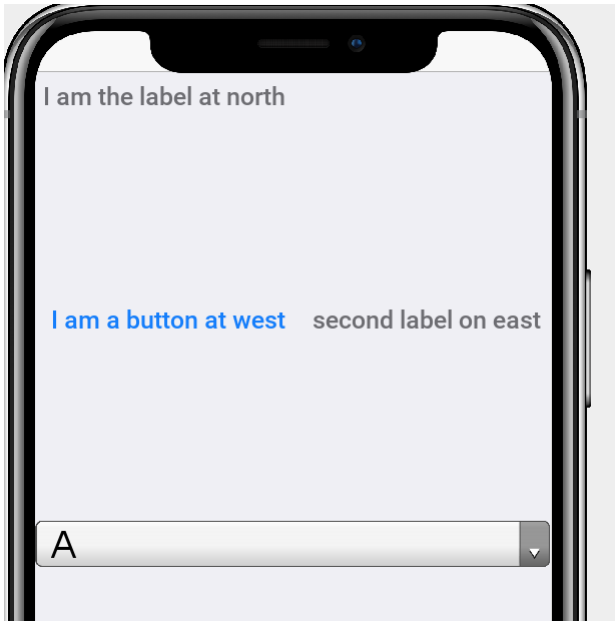


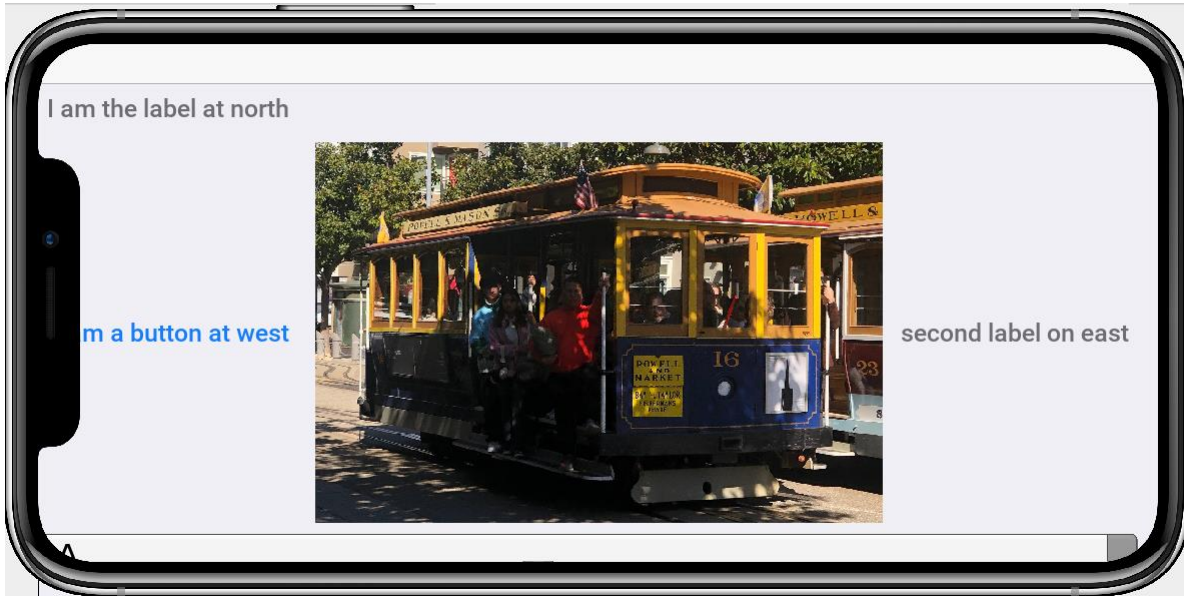
Code of BorderLayout

```
try {
   setLayout(new BorderLayout());
   Label myLabel = new Label("I am the label at north");
   Button myButton = new Button("I am a button at west");
   Image img = Image.createImage("/img.jpg");
   Label myLabel2 = new Label("first label on east");
   Label myLabel3 = new Label("second label on east");
   ComboBox myComboBox = new ComboBox("A","B","C");
   add(BorderLayout.NORTH, myLabel);//add them to the display
   add(BorderLayout.CENTER, img);
   add(BorderLayout.WEST, myButton);
   add(BorderLayout.EAST, myLabel2);
   add(BorderLayout.EAST, myLabel3);
   add(BorderLayout.SOUTH, myComboBox);
} catch (IOException e) { e.printStackTrace(); }
```

BorderLayout Result

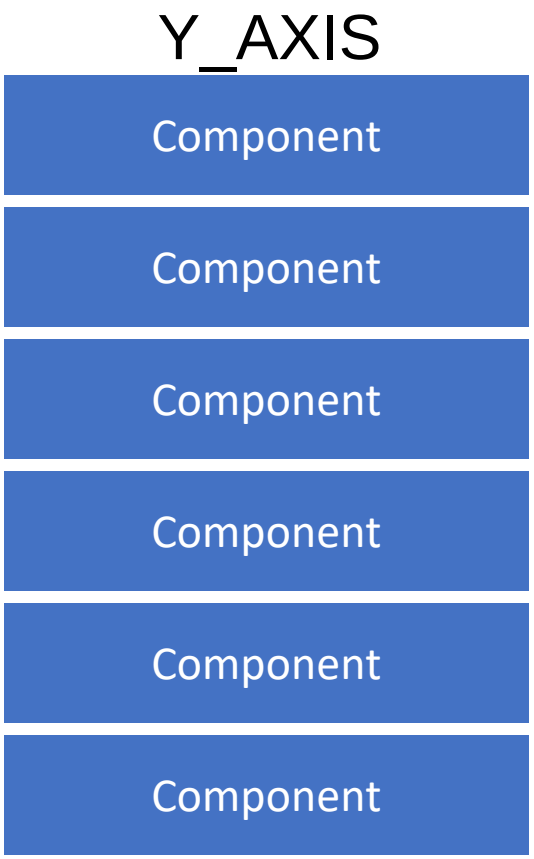
- Stretches North and South to fit, then East and West, centre gets what space is left
  - Hide component if no space

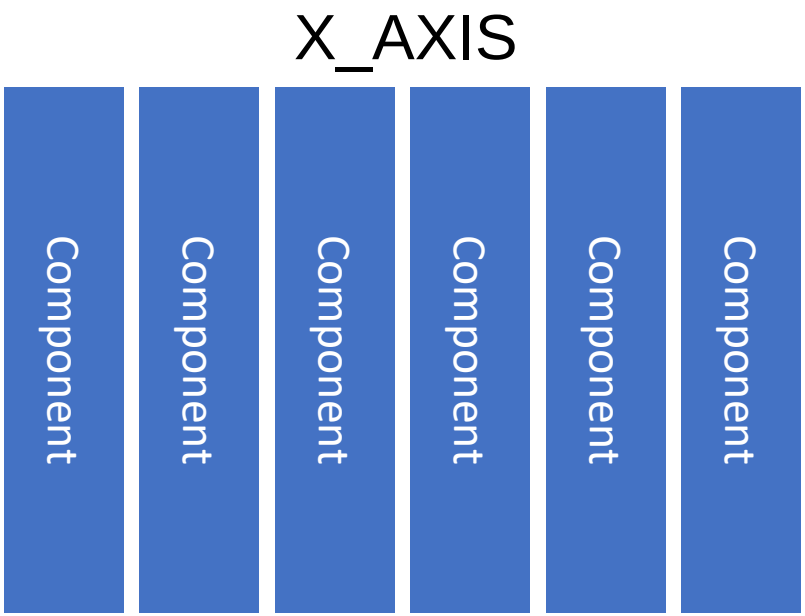




BoxLayout

- Aligned to one direction





BoxLayout

- Adds components to a horizontal or a vertical line that doesn't break the line
- Box layout accepts an axis in its constructor:

```
myContainer.setLayout(new BoxLayout(BoxLayout.X_AXIS));
myContainer.setLayout(new BoxLayout(BoxLayout.Y_AXIS));
```

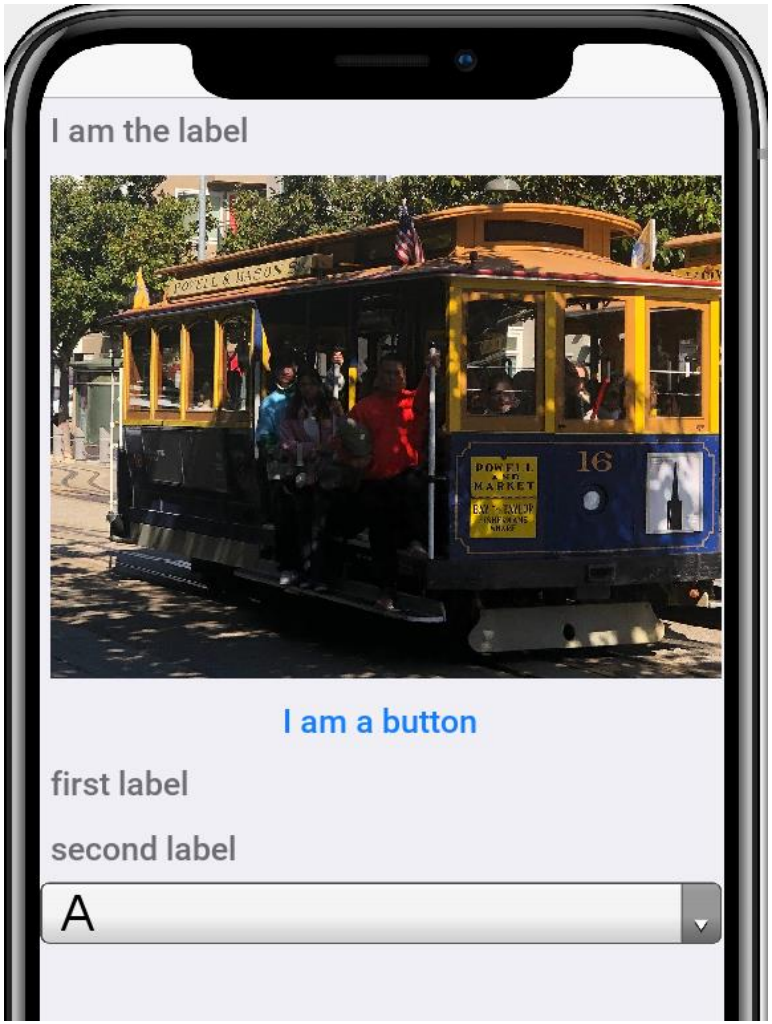
- Components are stretched along the opposite axis, e.g. X_AXIS box layout will place components horizontally and stretch them vertically.

Code for `BoxLayout`

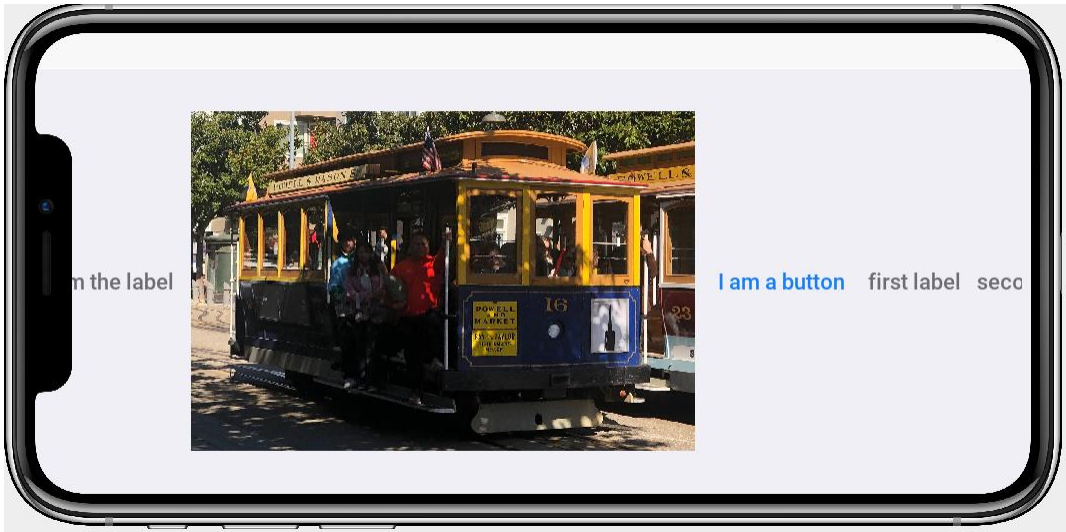
```
try {
   setLayout(new BoxLayout(BoxLayout.Y_AXIS));
   Label myLabel = new Label("I am the label");
   Button myButton = new Button("I am a button");
   Image img = Image.createImage("/img.jpg");
   Label myLabel2 = new Label("first label");
   Label myLabel3 = new Label("second label");
   ComboBox myComboBox = new ComboBox("A","B","C");
   add(myLabel).add(img).add(myButton);
   add(myLabel2).add(myLabel3).add(myComboBox);
} catch (IOException e) { e.printStackTrace(); }
```

Result

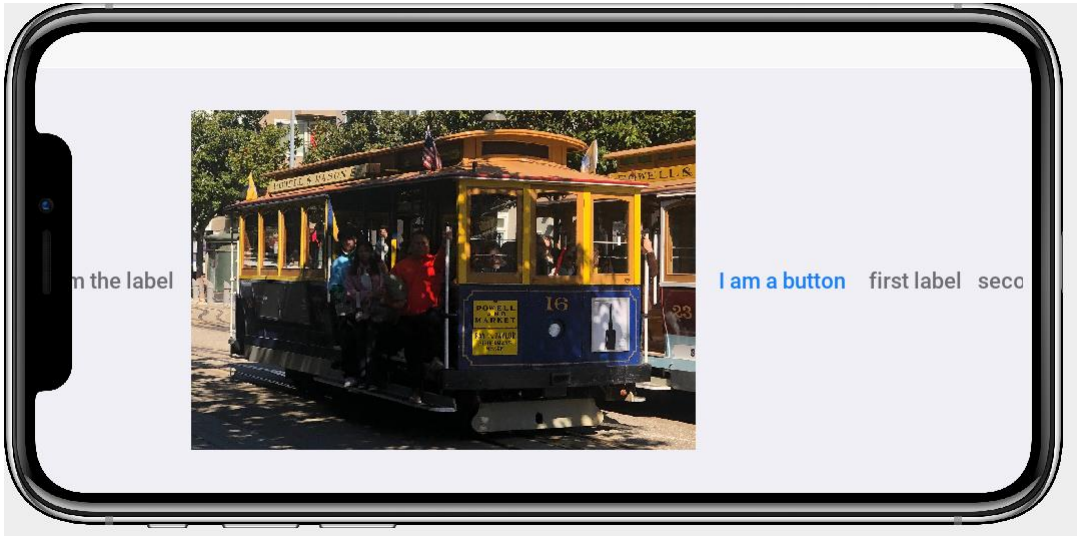
Y_AXIS



X_AXIS



The Size

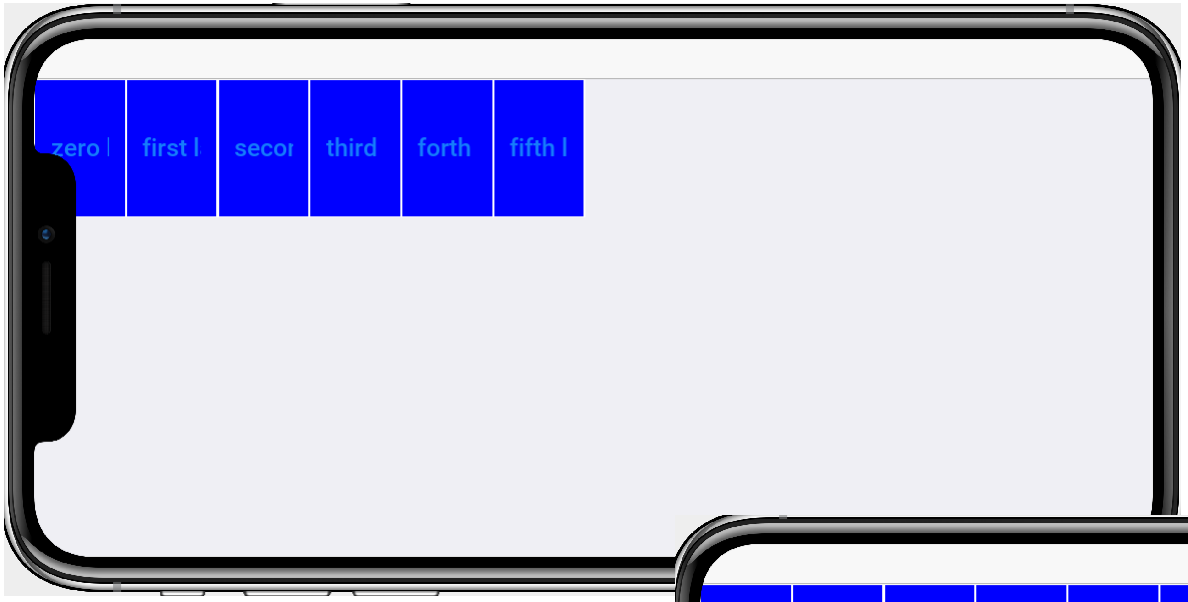


- The size depends on the content
- Some component are out-of-screen
- Solution?

Preferred Size

```
class ShortButton extends Button{
    @Override
    protected Dimension calcPreferredSize(){
        return new Dimension(200, 300);
    }
    public ShortButton(String string) {
        super(string);
        getAllStyles().setBgColor(ColorUtil.YELLOW);
    }
}
```

Result of Preferred Size



BoxLayout (below) Respects preferred width but not height.

FlowLayout (above) respects both preferred width and height.

GUI Layout

GUIs usually have multiple “areas”

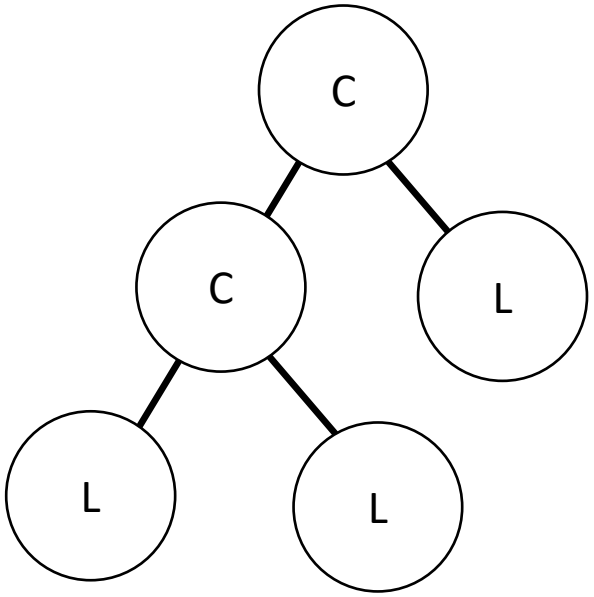
CN1 Container Class

- **`Container`** (like **`JPanel`** in Swing): an *invisible* component that
  - Can be assigned to an area
  - Can have a layout manager assigned to it
  - Can hold other components (**`Container`** is-a **`Component`** and has-a **`Component`**)

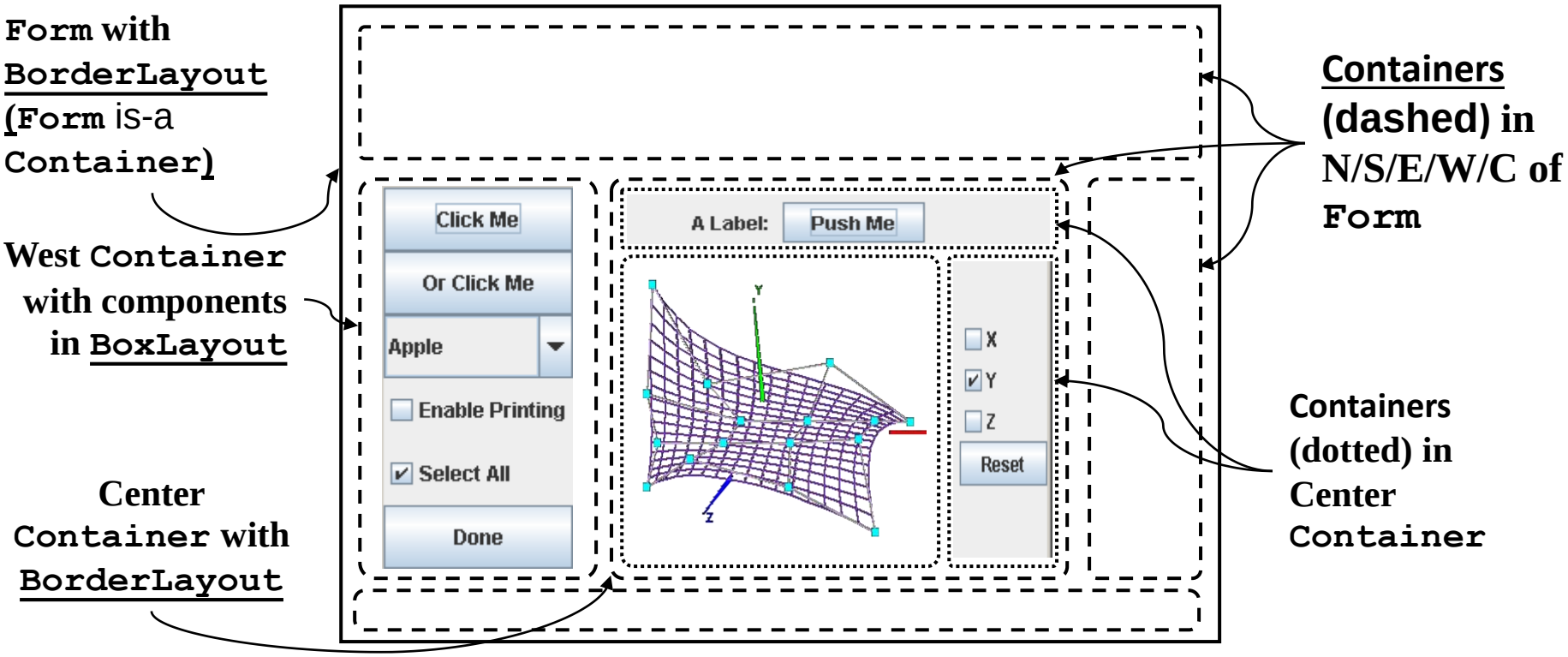
Container

- The Composite Pattern

- Container: Group
- Component: Primary objects



CN1 Container Class



Container Example

```
/* Code for a form with containers in different layout arrangements */
setLayout(new BorderLayout());
//top Container with the GridLayout positioned on the north
Container topContainer = new Container(new GridLayout(1,2));
topContainer.add(new Label("Read this (t)"));
topContainer.add(new Button("Press Me (t)"));
//Setting the Border Color
topContainer.getAllStyles().setBorder(Border.createLineBorder(4,
ColorUtil.YELLOW));
add(BorderLayout.NORTH,topContainer);
//left Container with the BoxLayout positioned on the west
Container leftContainer = new Container(new BoxLayout(BoxLayout.Y_AXIS));
//start adding components at a location 50 pixels below the upper border of the container
leftContainer.getAllStyles().setPadding(Component.TOP, 50);
leftContainer.add(new Label("Text (l)"));
leftContainer.add(new Button("Click Me (l)"));
leftContainer.add(new ComboBox("Choice 1","Choice 2","Choice 3"));
leftContainer.add(new CheckBox("Enable Printing (l)"));
leftContainer.getAllStyles().setBorder(Border.createLineBorder(4,
ColorUtil.BLUE));
add(BorderLayout.WEST,leftContainer);
```

... continued

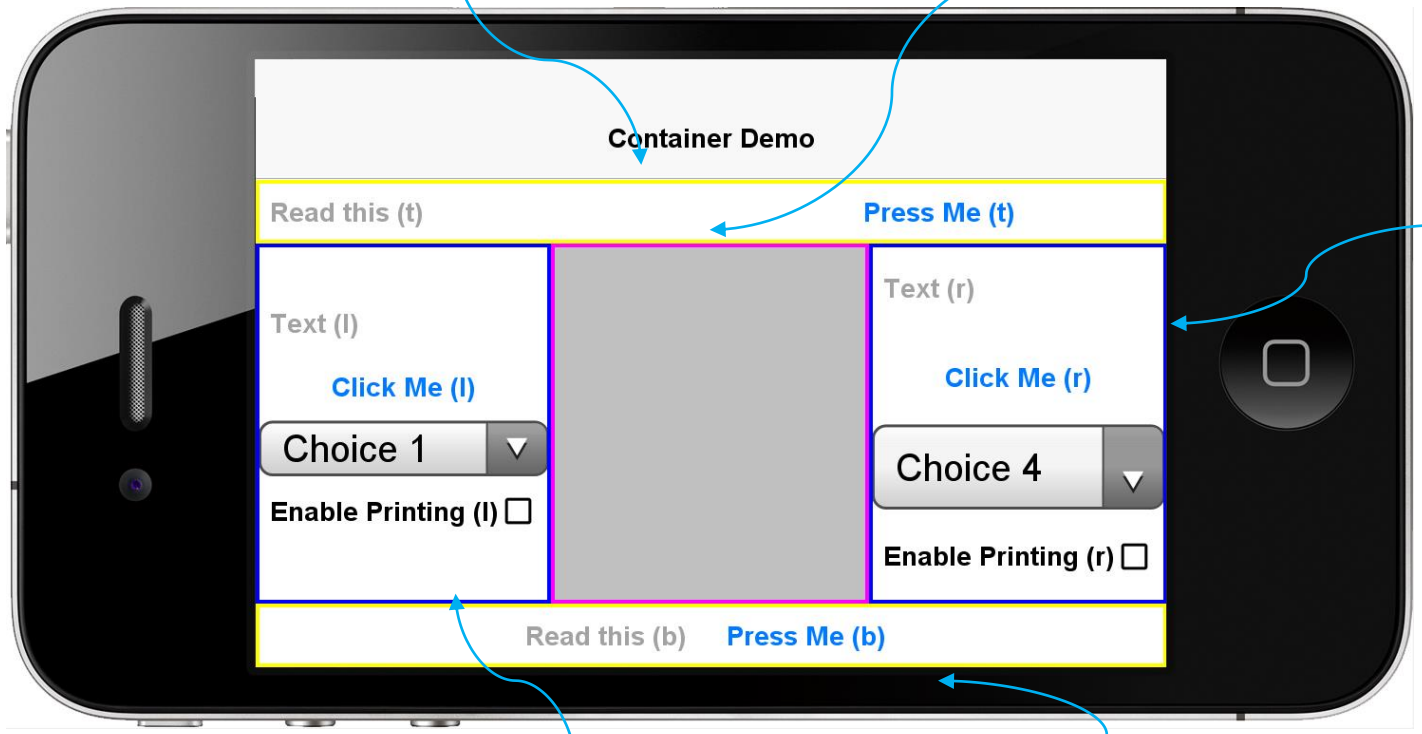
Container Example (cont.)

```
... continued
//right Container with the GridLayout positioned on the east
Container rightContainer = new Container(new GridLayout(4,1));
//...[add similar components that exists on the left container]
add(BorderLayout.EAST,rightContainer);
//add empty container to the center
Container centerContainer = new Container();
//setting the back ground color of center container to light gray
centerContainer.getAllStyles().setBgTransparency(255);
centerContainer.getAllStyles().setBgColor(ColorUtil.LTGRAY);
//setting the border Color
centerContainer.getAllStyles().setBorder(Border.createLineBorder(4,
                                                        ColorUtil.MAGENTA));
add(BorderLayout.CENTER,centerContainer);
//bottom Container with the FlowLayout positioned on the south, components are laid out
//at the center
Container bottomContainer = new Container(new FlowLayout(Component.CENTER));
//...[add similar components that exists on the top container]
add(BorderLayout.SOUTH,bottomContainer);
```

Container Example – Output

`Container` in North with `GridLayout` (space is divided to two equally-sized cells)

Empty `Container` in Center with (with light gray background)



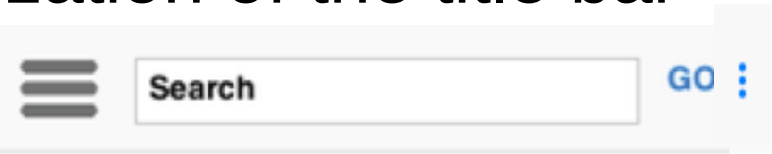
`Container` in East with `GridLayout` (space is divided to four equally-sized cells)

`Container (with padding)` in West with `BoxLayout` (components are positioned 50 pixels below the top border of the container)

`Container (with padding)` in South with `FlowLayout` (components are positioned at the center)

CN1 `Toolbar` class

- Provides deep customization of the title bar area of your form.



- Set it to your form with:

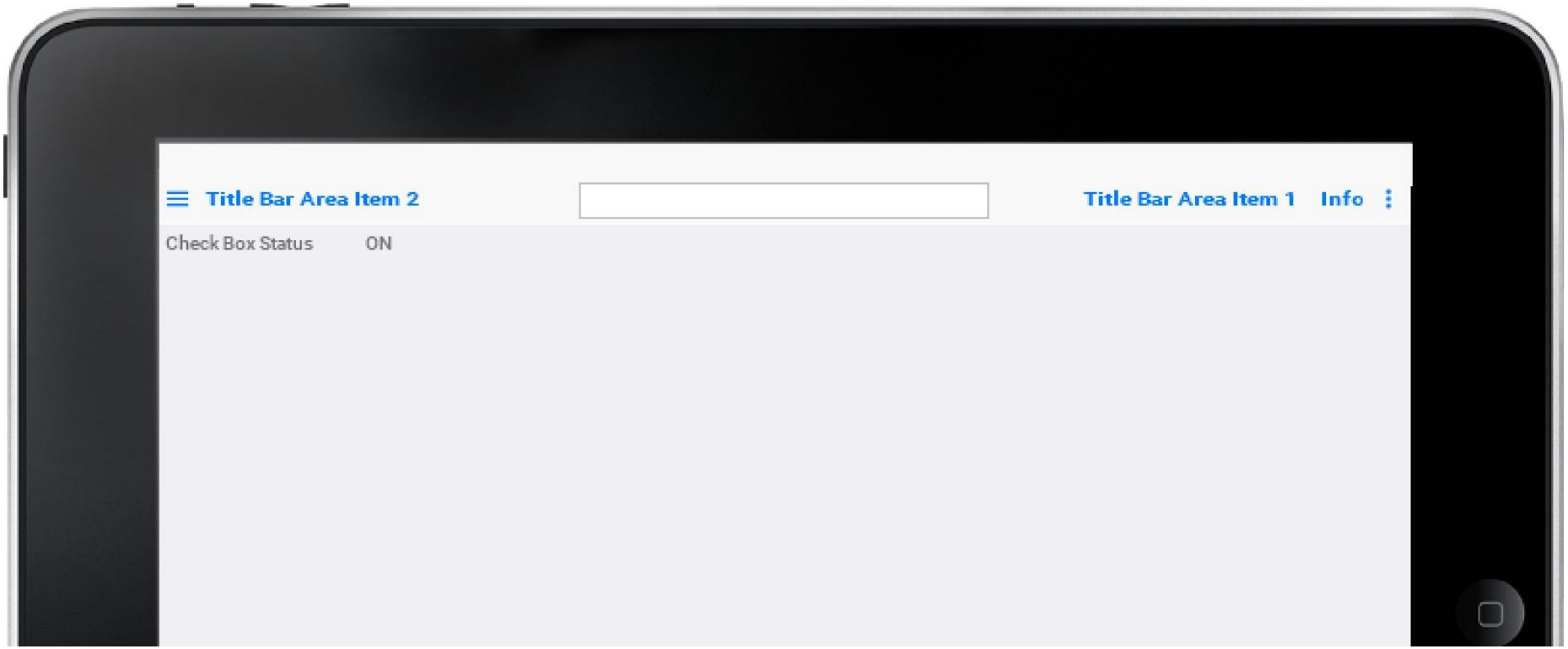
```
setToolbar(toolbar)
```

- Adding commands to different locations:
  - `addCommandToSideMenu()` (to side menu: ≡)
  - `addCommandToOverflowMenu()` (to Android style menu ⋮)
  - `addCommandToRightBar()` (to right of the title bar area)
  - `addCommandToLeftBar()` (to left of the title bar area)

Adding Items to Title Bar

```
/* Code for a form with a toolbar */
Toolbar myToolbar = new Toolbar();
setToolbar(myToolbar);//make sure to use lower-case "b", setToolBar() is deprecated
//add a text field to the title
TextField myTF = new TextField();
myToolbar.setTitleComponent(myTF);
//[or you can simply have a text in the title: this.setTitle("Adding Items to Title Bar");]
//add an “empty” item (which does not perform any operation) to side menu
Command sideMenuItem1 = new Command("Side Menu Item 1");
myToolbar.addCommandToSideMenu(sideMenuItem1);
//add an “empty” item to overflow menu
Command overflowMenuItem1 = new Command("Overflow Menu Item 1");
myToolbar.addCommandToOverflowMenu(overflowMenuItem1);
//add an “empty” item to right side of title bar area
Command titleBarAreaItem1 = new Command("Title Bar Area Item 1");
myToolbar.addCommandToRightBar(titleBarAreaItem1);
//add an “empty” item to left side of title bar area
Command titleBarAreaItem2 = new Command("Title Bar Area Item 2");
myToolbar.addCommandToLeftBar(titleBarAreaItem2);
//...[add other side menu, overflow menu, and/or title bar area items]
```

Adding Items to Title Bar (cont.)



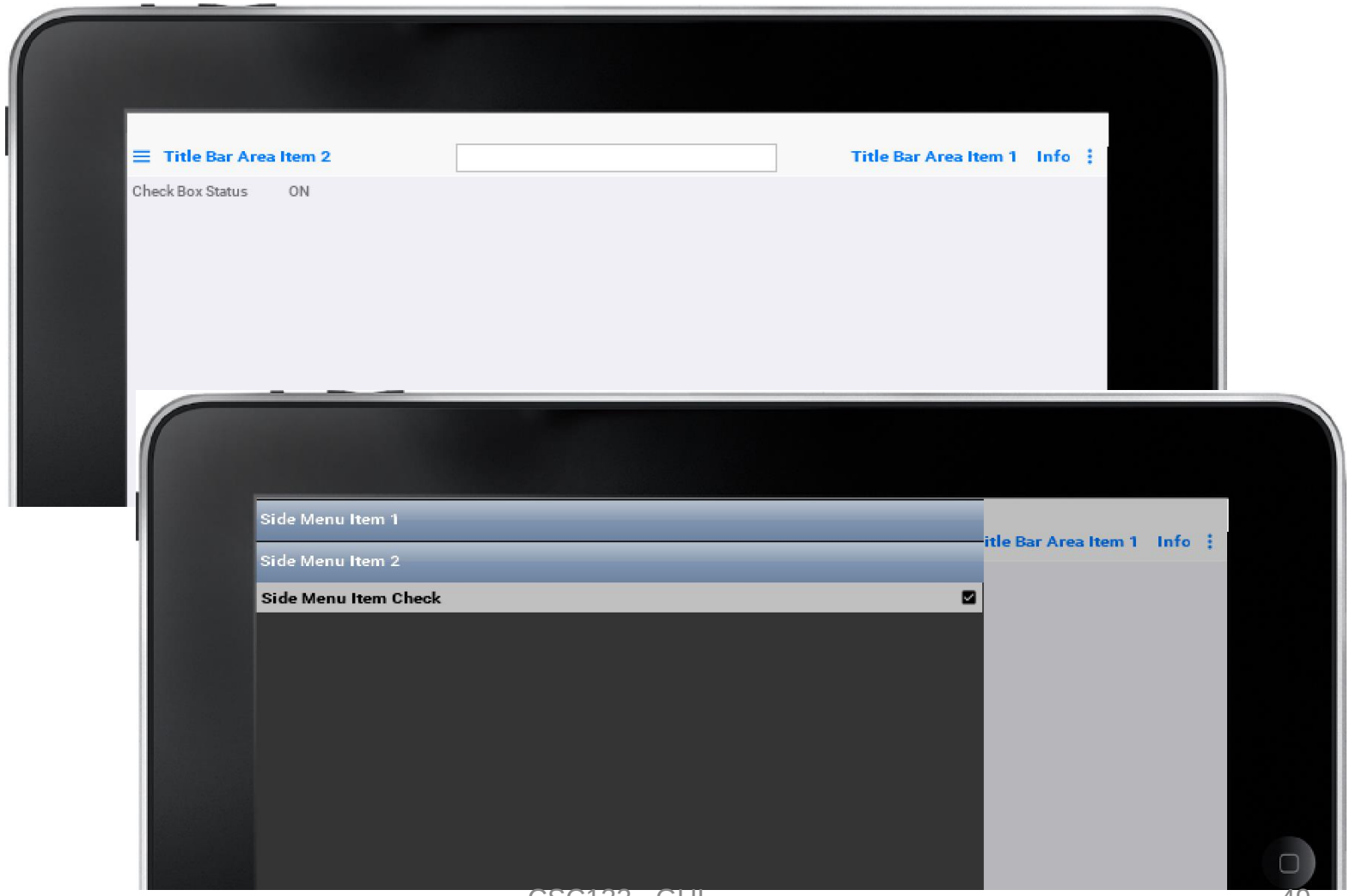
Complex Menus

- Menu items can contain components (like the title area):

```
/* Code for a form which has a CheckBox as a side menu item*/
//add a check box to side menu (which does not perform any operation yet..)
CheckBox checkSideMenuComponent = new CheckBox("Side Menu Item Check");
//set the style of the check box
checkSideMenuComponent.getAllStyles().setBgTransparency(255);
checkSideMenuComponent.getAllStyles().setBgColor(ColorUtil.LTGRAY);
//add the CheckBox component as a side menu item
myToolbar.addComponentToSideMenu(checkSideMenuComponent);
```

- We will later see how to attach operations (set commands) to the components in menus…

Complex Menus (cont.)



Any Questions?