CSc 133 Lecture Notes

8 - GUI Basics

Computer Science Department

California State University, Sacramento

Overview

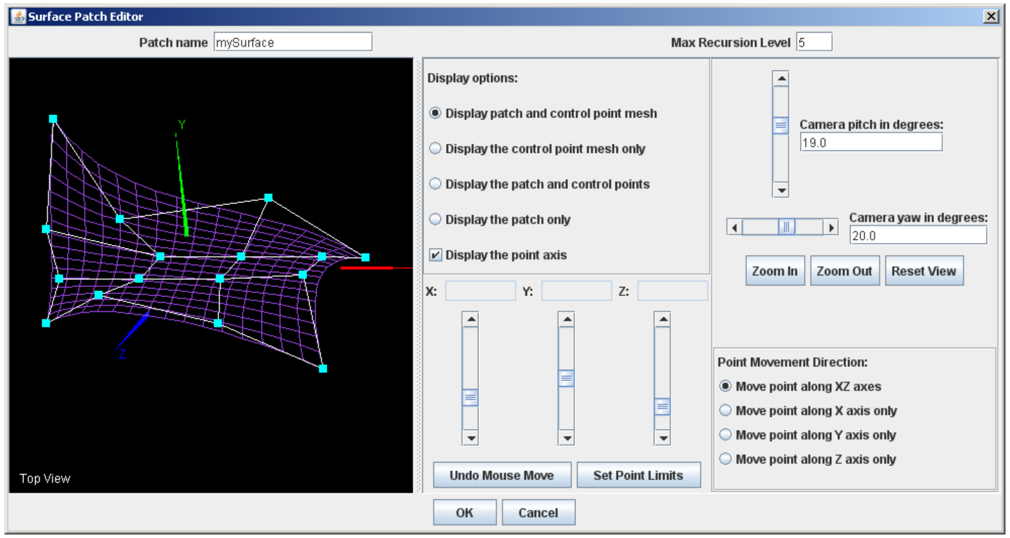
- **Displays and Color**
- **The UI Package of CN1**
- **UI Components: Form, Button, Label, Checkbox, ComboBox, TextField …**
- **Layout Managers**
- **Containers**
- **Side Menus**

Sources: (1) Codename one User Developer Guide on Canvas.

(2) https://www.codenameone.com/manual/basics.html

Modern Program Characteristics

- Graphical User Interfaces ("GUIs")
- "Event-driven" interaction



Surface Patch Editor

Patch name: mySurface

Max Recursion Level: 5

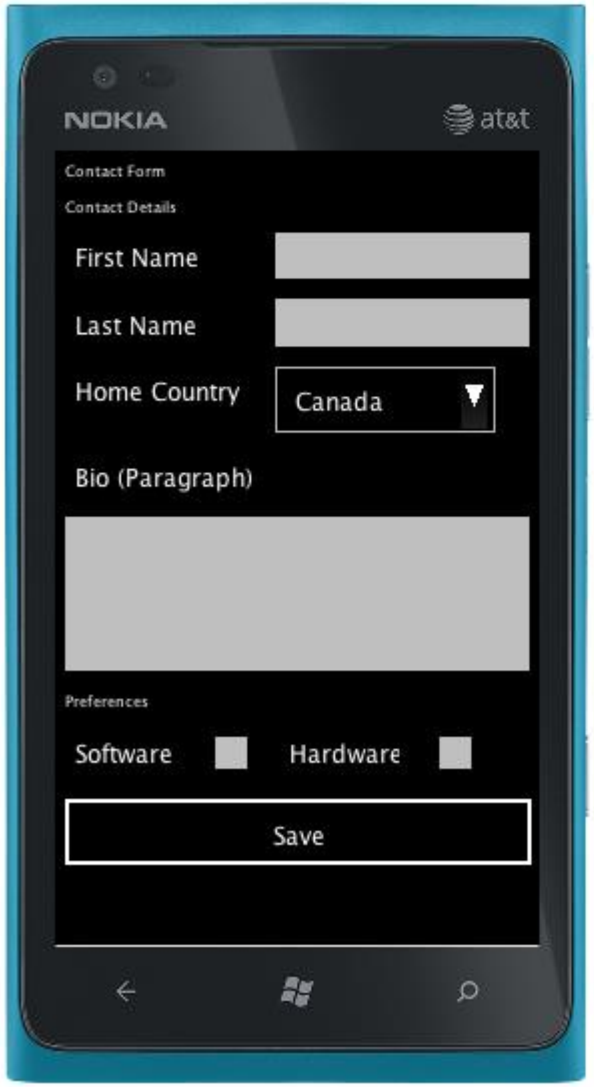
Display options:
- [✓ Display patch and control point mesh]
- Display the control point mesh only
- Display the patch and control points
- Display the patch only
- [✓ Display the point axis]

X: Y: Z:

Camera pitch in degrees: 19.0

Camera yaw in degrees: 20.0

Point Movement Direction:
- [✓ Move point along XZ axes]
- Move point along X axis only
- Move point along Y axis only
- Move point along Z axis only



Contact Form

Contact Details

First Name:

Last Name:

Home Country: Canada

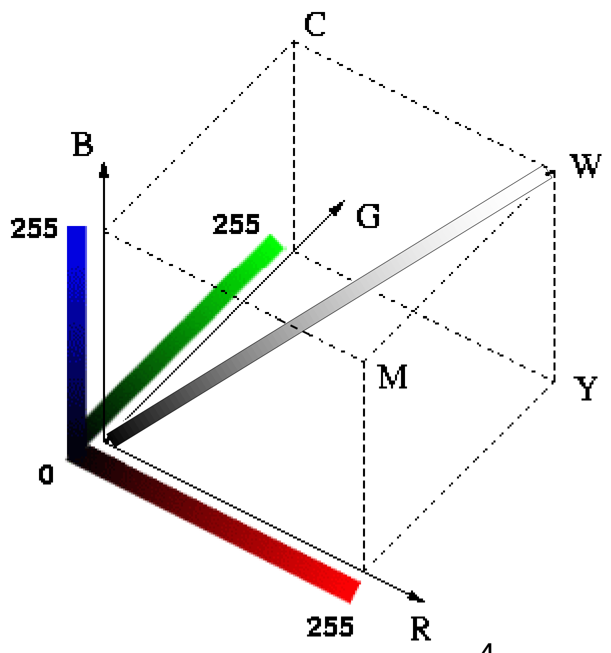
Bio (Paragraph):

Preferences

Software ☐ Hardware ☐

The RGB Color Cube

- Each axis represents one of (Red, Green, Blue)
- Distance along axis = intensity (0 to max)
- Locations within cube = different colors
  - Values of equal RGB intensity are grey



R:	red
G:	green
B:	blue
C:	cyan
M:	magenta
Y:	yellow
W:	white

Image credit: http://gimp-savvy.com

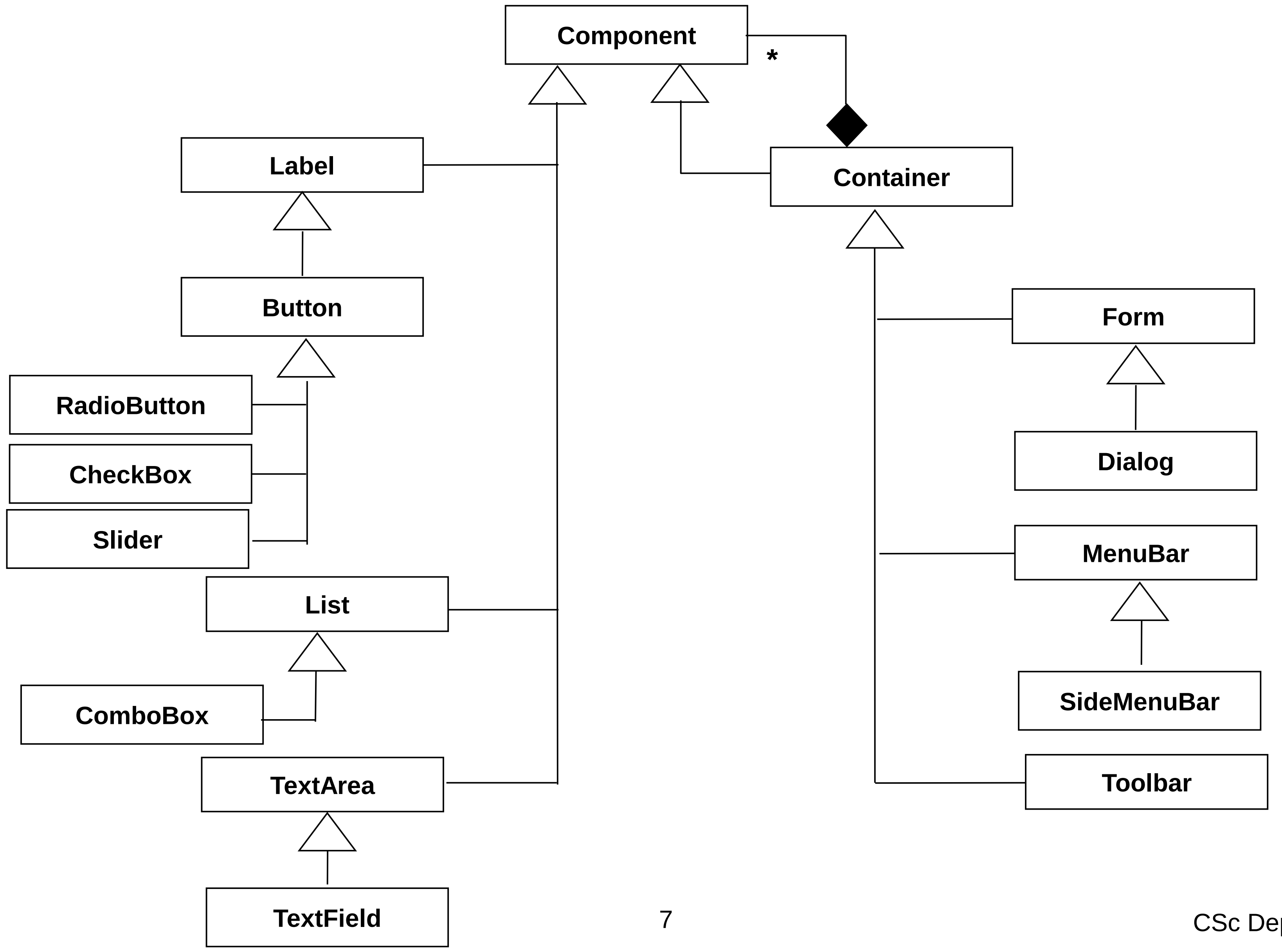
GUI Frameworks

- Collection of classes that take care of low-level details of drawing “things” on screen. Provides:
  - A *set of reusable screen components*
    - “Component”: an object having a *graphical representation*
    - Usually has the ability to *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

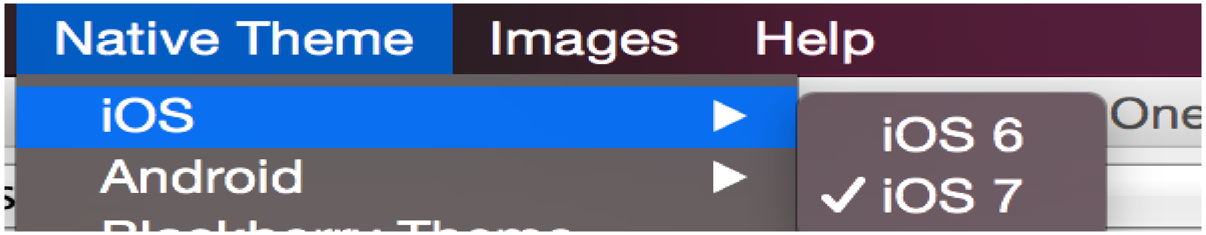
- Microsoft Foundation Classes (**MFC**): designed for C++ development on Windows (it is not build-in to C++)
- **AWT**: Java's first (inefficient) build-in GUI package
- JFC/**Swing**: Java's efficient build-in GUI package
- **UI**: CN1's GUI package (very similar to Swing)
- "Things" are called controls (MFC), components (AWT/Swing/CN1), widgets (X-Windows on Linux)

Important CN1- UI Components



CN1 Theme, Component, Style

Theme: Codename One themes are pluggable CSS like elements that allow developers to determine/switch the look of the application in runtime.



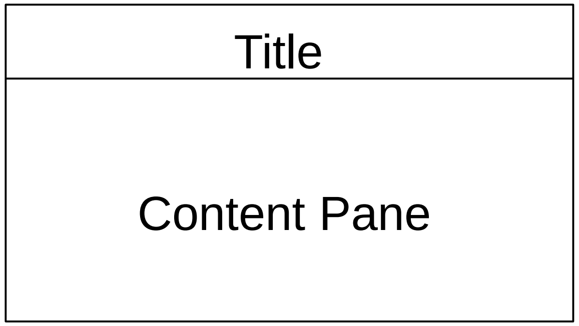
Component: Every visual element within Codename One is a Component, the button on the screen is a Component and so is the Form in which it's placed. This is all represented within the Component, which is probably the most central class in Codename One.

Style: Every component has 4 standard style objects associated with it: Selected, Unselected, Disabled & Pressed.

Only one style is applicable at any given time and it can be queried via the getStyle() method. A style contains the colors, fonts, border and spacing information relating to how the component is presented to the user

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
- Form contains title and a content pane (and optionally a menu bar which we will not utilize in the assignments):



- Calling to **`myForm.addComponent()`** is actually invoking **`myForm.getContentPane().addComponent()`**
- Hence, content pane is the "parent" container of all components you add to the form.

Creating a Form in CN1 (cont.)

```
// Contents of File  DemoSimpleForm.java:
/** This class is a driver for running the SimpleForm class. It creates a Form.
It is the "Main" class of CN1 project (created with "native" theme and "Hello
World(manual)" template).
*/
//default import statements...
public class DemoSimpleForm {
private Form current;
//default implementations of methods like init(), stop(), destroy() ...
  public void start() {
     if(current != null){
               current.show();
               return;
             }
     //change the default implementation of start()
     new SimpleForm();
  }
}
```

Recap:CN1 and Assignments (cont.)

- For instance for Assignment#1:
  - Project Name: A1Prj
  - Main Class Name: Starter (keep the same for all assignments)
  - Package: com.mycompany.a1
- Main class has the following structure:

```
public class Starter {
   …
   public void init(…) {…}
   public void start() {…}
   public void stop() {…}
   public void destroy() {}
}
```

Recap: CN1 and Assignments (cont.)

- Solve the assignment by modifying **start()** in Starter.java (**do NOT delete other methods**) and adding/modifying other necessary source files.

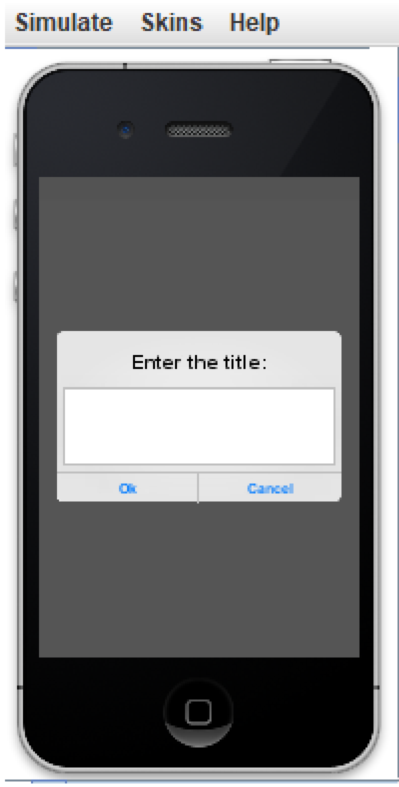
Creating a Form in CN1 (cont.)

```
// Contents of File  SimpleForm.java:
import com.codename1.ui.Form;
/** This class creates a simple "Form"  by extending an existing
 *  class "Form", defined in the CN1’s ui package.
 */
public class SimpleForm extends Form{
  public SimpleForm() {
    this.show();
  }
```

Note: Show a starter demo in class next.

Titled Form in CN1

```
import com.codename1.ui.*;
/** This class creates a "Form" that has a title specified by the user
 *  User types the title on a "TextField" on a "Dialog"
 */
public class TitledForm extends Form {
    public TitledForm() {
       Command cOk = new Command("Ok");
       Command cCancel = new Command("Cancel");
       Command[] cmds = new Command[]{cOk, cCancel};
       TextField myTF = new TextField();
       Command c = Dialog.show("Enter the title:", myTF, cmds);
       //[if you only want to display the okay option, you do not need to
       //create "cmds", just use Dialog.show("Enter the title:", myTF,
    cOk);]
       if (c == cOk)
         this.setTitle(myTF.getText());
       else
         this.setTitle("Title not specified");
       this.show();
   }
}
```



Closing App in CN1

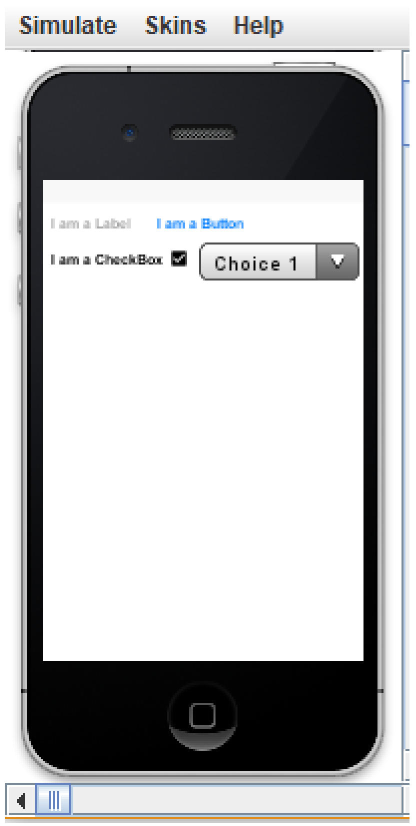
```
import com.codename1.ui.*; //not listed in the rest of the examples
/** This class creates a "Form" that has a title “Closing App Demo”
 *  Then it pops up a “Dialog” confirming closing of the application
 */
public class ClosingApp extends Form {
  public ClosingApp() {
    this.setTitle("Closing App Demo");
    Boolean bOk = Dialog.show("Confirm quit", "Are you sure you want to quit?",
"Ok", "Cancel");
    //[in a dialog if you only want to display the okay option,
    //use Dialog.show("Title of dialog", "Text to display on dialog", "Ok", null);]
    if (bOk){
         //instead of System.exit(0), CN1 recommends using:
         // This helps to quit the application
         Display.getInstance().exitApplication();
          }
    this.show();
  }
}
```

CN1 `Display` class

- Central class that manages rendering/events and is used to place top level components (Form) on the display.
- Has static **`getInstance()`** method which return the **`Display`** instance.
- To get the resolution of your display, you can call: **`Display.getInstance().getDisplayWidth()`** or …**`Height()`**
- **`Display.getInstance().getCurrent()`** return the form currently displayed on the screen or null if no form is currently displayed.

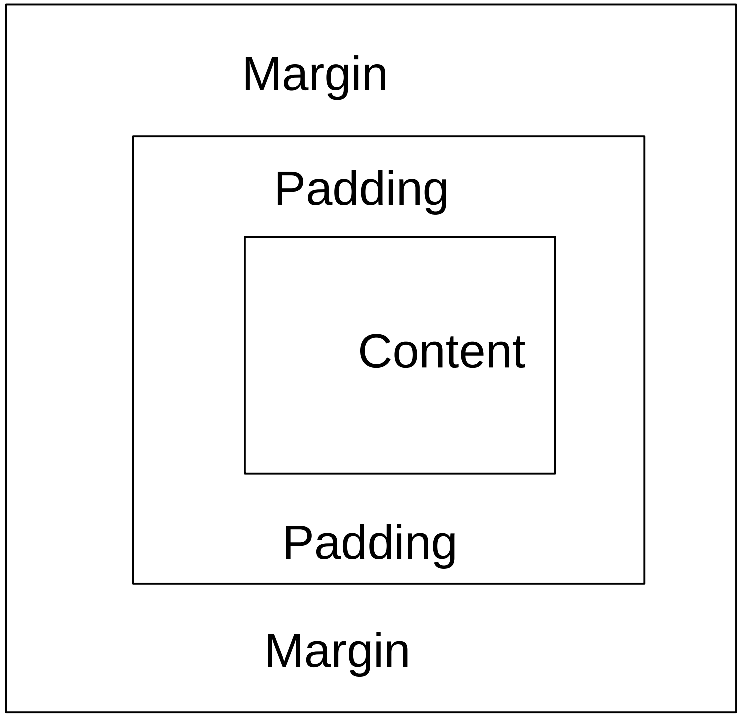
Adding Components to Form

```
public class FormWithComponents extends Form {
  public FormWithComponents () {
   // create a new label object
   Label myLabel = new Label("I am a Label");
   // add the label to the "content pane" of the form
   this.getContentPane().addComponent(myLabel);
   // [you can also call this.addComponent(myLabel)
    // or simply this.add(myLabel)] create a button and add
   Button myButton = new Button("I am a Button");
   this.addComponent(myButton);
   // create a checkbox and add
   CheckBox myCheck = new CheckBox("I am a CheckBox");
   this.addComponent(myCheck);
   // add a combo box (drop-down list) and add
   ComboBox myCombo = new ComboBox("Choice 1","Choice 2","Choice 3");
   this.addComponent(myCombo);
   this.show();
   }
}
```



CN1 `Style` class

Represents the look of a given component: **colors**, **fonts**, **transparency**, **margin** and **padding & images**.



Setting style of a Component

```
public class ComponentsWithStyle extends Form {
  public ComponentsWithStyle () {
    Button button1 = new Button("Plain button");
    Button button2 = new Button("Button with style");
    //change background and foreground colors of the unselected style of the button
    button2.getUnselectedStyle().setBgTransparency(255);
    button2.getUnselectedStyle().setBgColor(ColorUtil.BLUE);
    button2.getUnselectedStyle().setFgColor(ColorUtil.WHITE);
    //[use button2.getAllStyles() to set all styles (selected, pressed, disabled, etc.) of
the component at once]
    //add padding to all styles of button2
    button2.getAllStyles().setPadding(Component.TOP, 10);
    button2.getAllStyles().setPadding(Component.BOTTOM, 10);
    //[you can also add padding to left and right by using Component.LEFT and
Component.RIGHT]
    addComponent(button1);
    addComponent(button2);
     show(); //not listed in the rest of the examples
  }
}
```

Setting style of a Component (cont.)

```
public class ComponentsWithStyle extends Form {
  public ComponentsWithStyle () throws IOException { //for Image.createImage()
   //add button1 and button2 as shown in the previous example
  //set a background image for all styles of the form
  InputStream is = Display.getInstance().getResourceAsStream(getClass(),
                                                                  "/BGImage.png");
  Image i = Image.createImage(is);
  this.getAllStyles().setBgImage(i);
  //set an image for the unselected style of the button
  Button button3 = new Button("Expand");
  button3.getAllStyles().setPadding(Component.TOP, 10);
  //[if necessary, also add padding to bottom, left, right, etc]
  is = Display.getInstance().getResourceAsStream(getClass(), "/expand.png");
  //[copy the images directly under "src" directory]
  i = Image.createImage(is);
  button3.getUnselectedStyle().setBgImage(i);
  addComponent(button3);
  }
}
```

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:

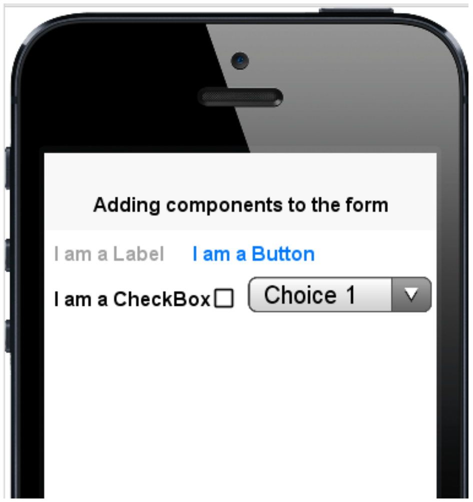
```
myContainer.setLayout( new BorderLayout() );
```

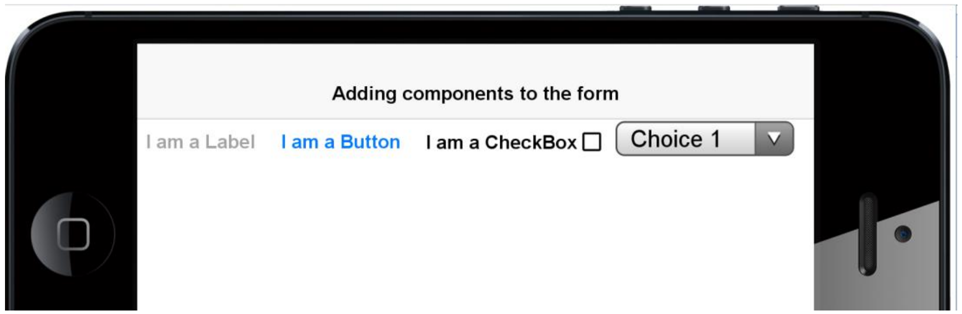
- Components can have a preferred *size*
  - **`setPreferredSize()`** of **`Component`** is depreciated
  - override **`calcPeferredSize()`** of **`Component`** to reach similar functionality (do not use this in the assignments)
  - Layout managers *may or may not* respect preferred size either entirely or partially (e.g., **`FlowLayout`** respects it whereas **`BoxLayout`** does not respect it entirely…)

Layout Managers (cont.)

- Example: **`FlowLayout`**
  - Arranges components left-to-right, top-to-bottom (by default)
  - Components appear in the order they are added
  - Respects *preferred size*
  - Components that don't fit may be *hidden*
  - You can center components in the component by using:

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

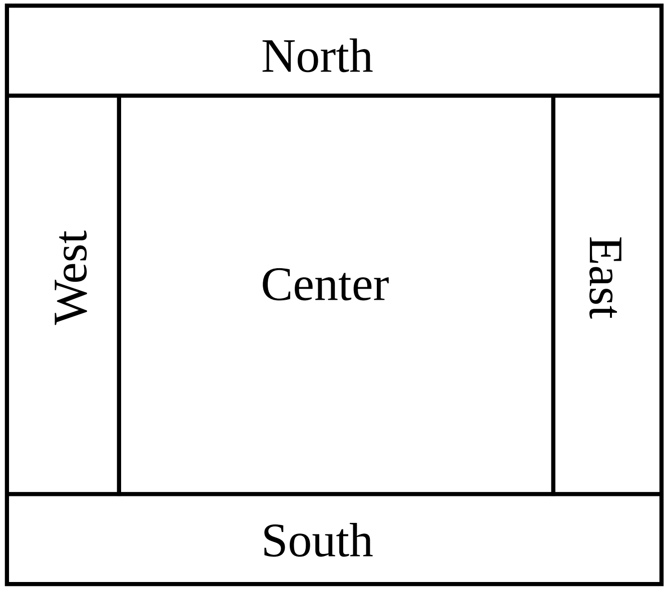




Layout Managers (cont.)

- Example: **`BorderLayout`**
  - Adds components to one of five “regions” of the container:
    North, South, East, West, or Center
  - Region must be specified when component is added

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



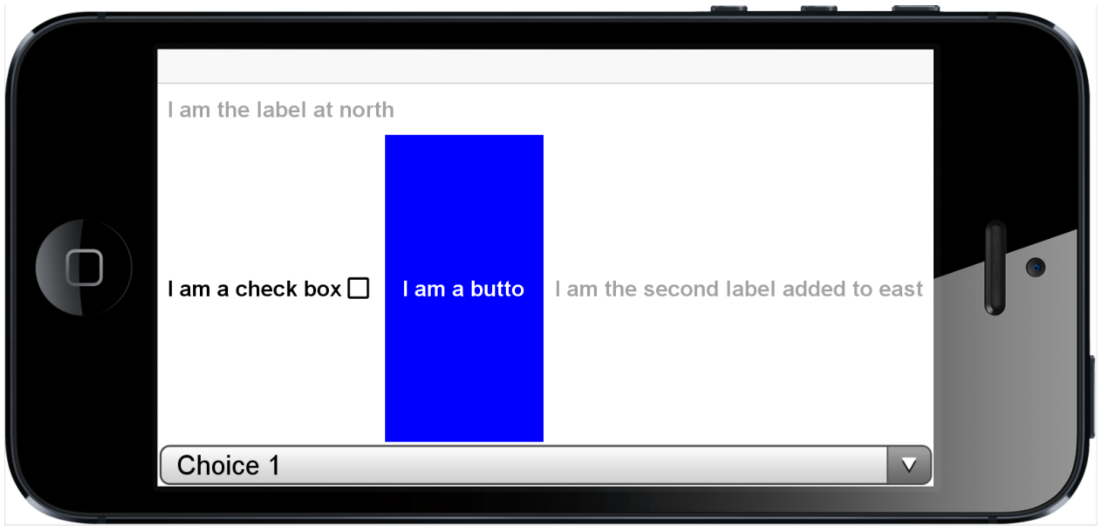
Layout Managers (cont.)

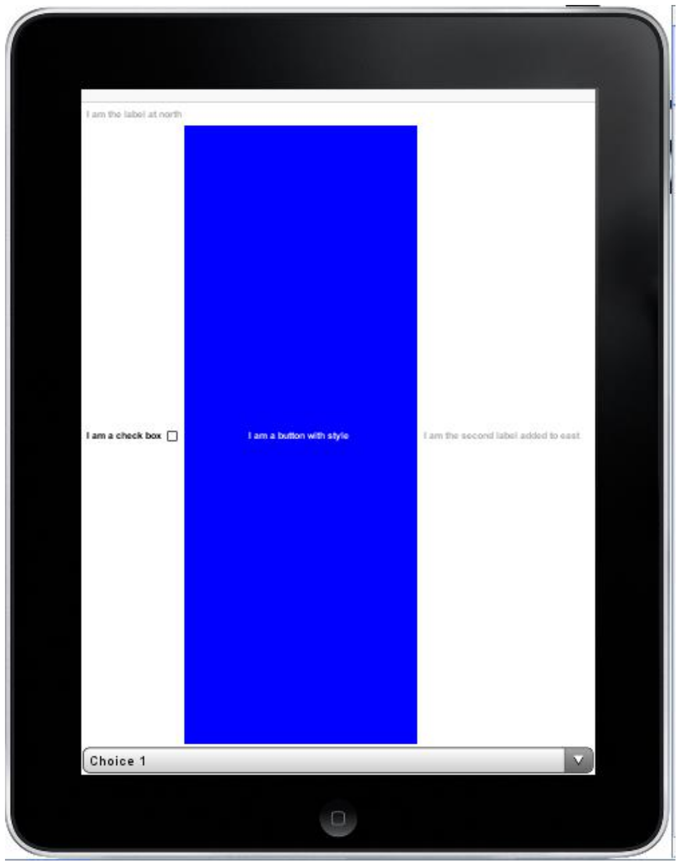
- **BorderLayout** (cont.)

```
public class BorderLayoutForm extends Form{//not listed in the rest
  public BorderLayoutForm() {              //of the examples
    //default layout for container is FlowLayout, change it to BorderLayout
    this.setLayout(new BorderLayout());
    //add a label to the top area of border layout
    Label myLabel = new Label("I am the label at north");
    this.add(BorderLayout.NORTH, myLabel);
    //... [add a check box to BorderLayout.WEST, a combo box to BorderLayout.SOUTH]
    //create a button to add to the center area
    Button myButton = new Button("I am a button with style");
    //...[set style of the button and add it to BorderLayout.CENTER]
    //add other labels to the left area of border layout
    Label myLabel2 = new Label("I am the first label added to east");
    this.add(BorderLayout.EAST, myLabel2);
    //[THIS LABEL WILL NOT BE VISIBLE, see upcoming slides for a solution]
    Label myLabel3 = new Label("I am the second label added to east");
    this.add(BorderLayout.EAST, myLabel3);}
}
```

Layout Managers (cont.)

- **BorderLayout** (cont.)
  - Stretches North and South to fit, then East and West
    - Center gets what space is left (if any!)





Layout Managers (cont.)

- Example: **`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.

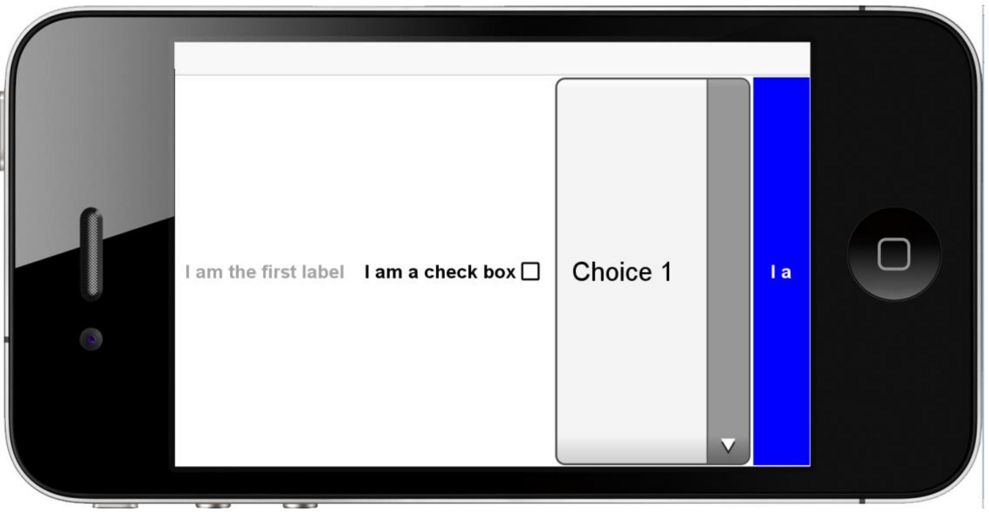
Layout Managers (cont.)

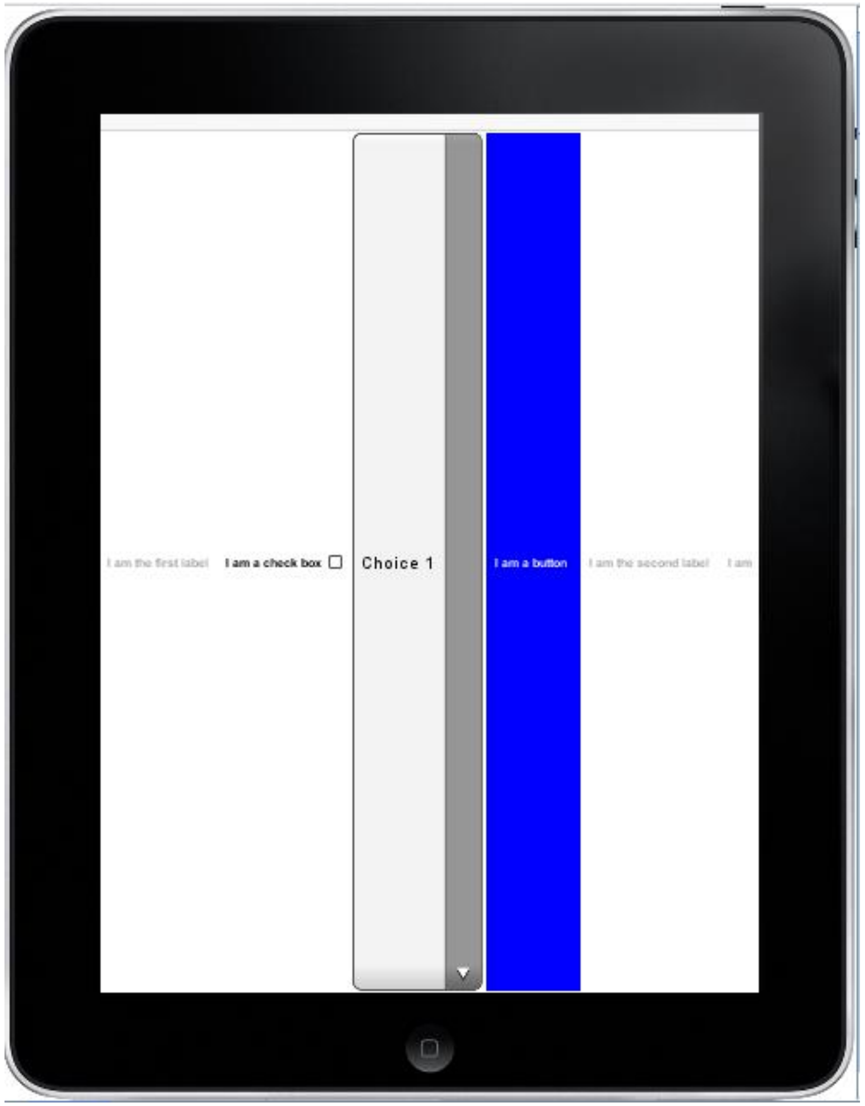
- Example: **`BoxLayout`** (cont.)

```
/* Code for a form with box layout */
setLayout(new BoxLayout(BoxLayout.X_AXIS));
//add a label as the first item
Label myLabel = new Label("I am the first label");
add(myLabel);
//... [add a check box as the second, a combo box as the third item
Button myButton = new Button("I am a button");
//...[set style of the button and add it as the fourth item]
//add other labels as fifth and sixth items
Label myLabel2 = new Label("I am the second label");
add(myLabel2);
Label myLabel3 = new Label("I am the third label");
add(myLabel3);
```

Layout Managers (cont.)

- Example: **`BoxLayout`** (cont.)





Layout Managers (cont.)

Setting preferred size (do not use this in the assignments, instead use **`setPadding()`** of **`Style`** class to change size of your buttons etc):

```
public class MyComponent extends Component{
@Override
protected Dimension calcPreferredSize(){
  return new Dimension(500, 300);}
public MyComponent() {
  //this is an empty component with a blue border
  this.getAllStyles().setBorder(Border.createLineBorder(2, ColorUtil.BLUE));}
}
```

`---------------------` below is the code for a form with default layout

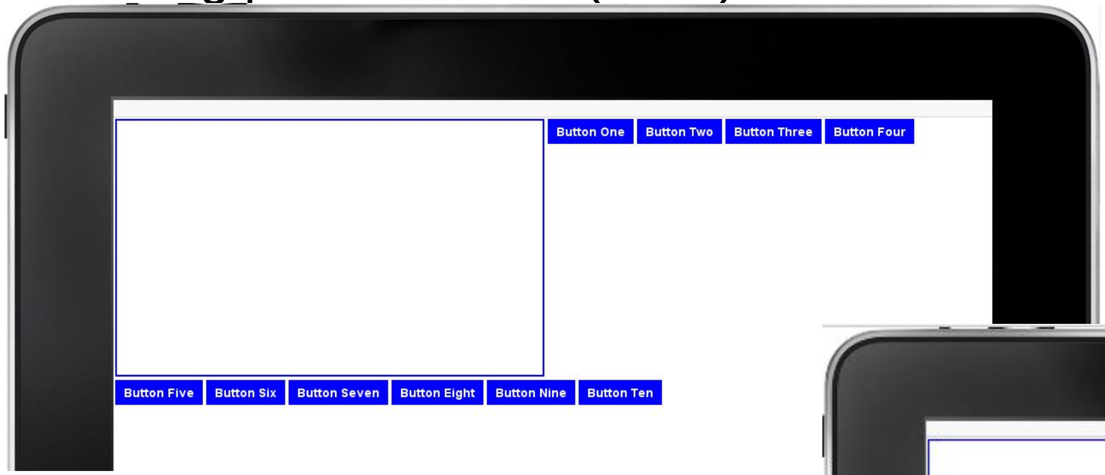
```
//using default flow layout, first add a MyComponent
MyComponent myComponent = new MyComponent();
add(myComponent);
//then add several buttons with styles
```

`---------------------` below is the code for a form with box layout

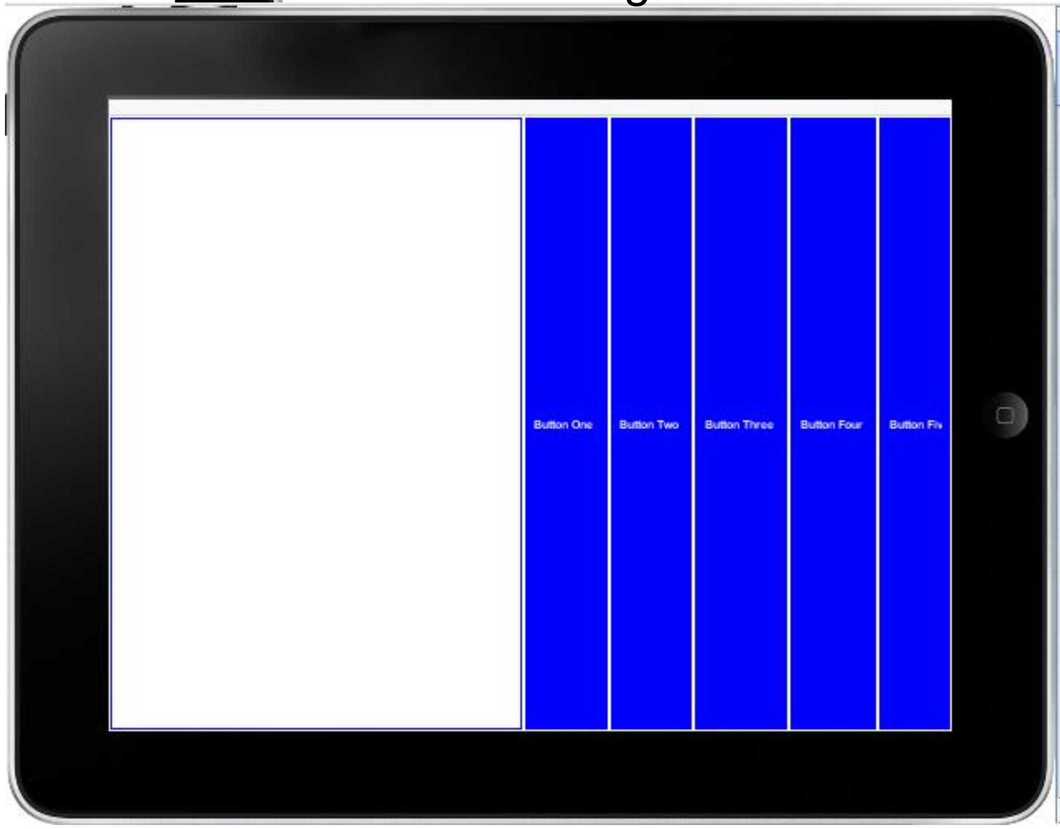
```
//using X_AXIS box layout
setLayout(new BoxLayout(BoxLayout.X_AXIS));
//add MyComponent to the first item, and then then add several buttons with styles
```

Layout Managers (cont.)

Setting preferred size (cont.):



BoxLayout (below) Respects preferred width but not height…



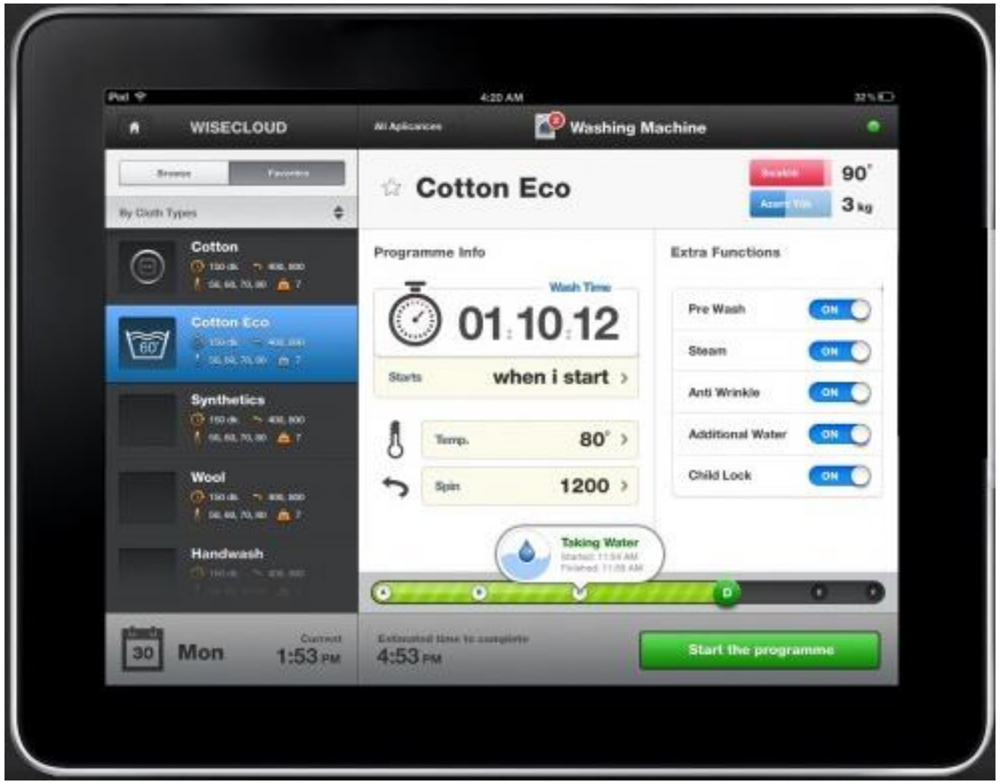
FlowLayout (above) respects both preferred width and height…

Layout Managers (cont.)

- Other Layout Managers
  - **`GridLayout`**
  - **`TableLayout`**
  - Etc..
- You can change the layout manager of the container in runtime:
  - Example of the *Strategy* Design Pattern

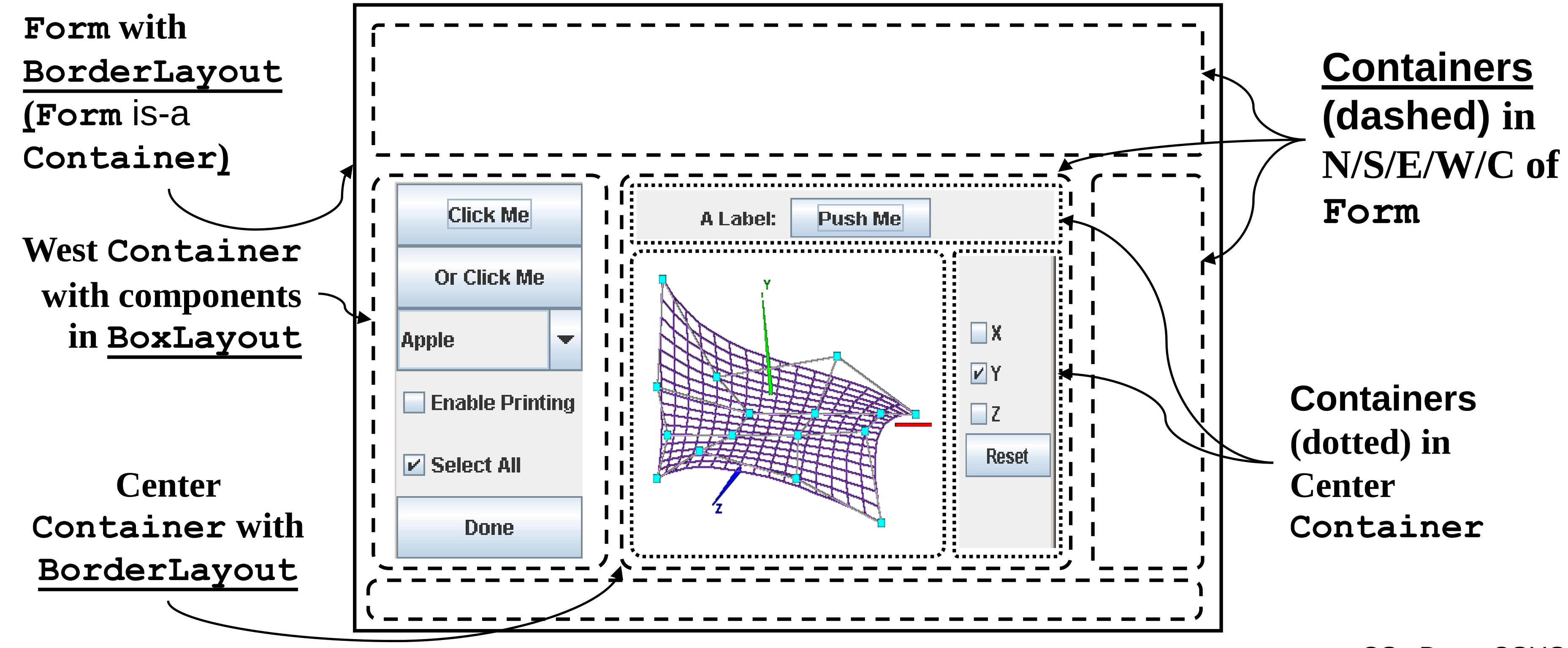
GUI Layout

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`)


Form with BorderLayout (Form is-a Container)
West Container with components in BoxLayout
Center Container with BorderLayout
Containers (dashed) in N/S/E/W/C of Form
Containers (dotted) in Center Container
Click Me
Or Click Me
Apple
Enable Printing
Select All
Done
A Label:
Push Me
Y
Z
X
Reset

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
```

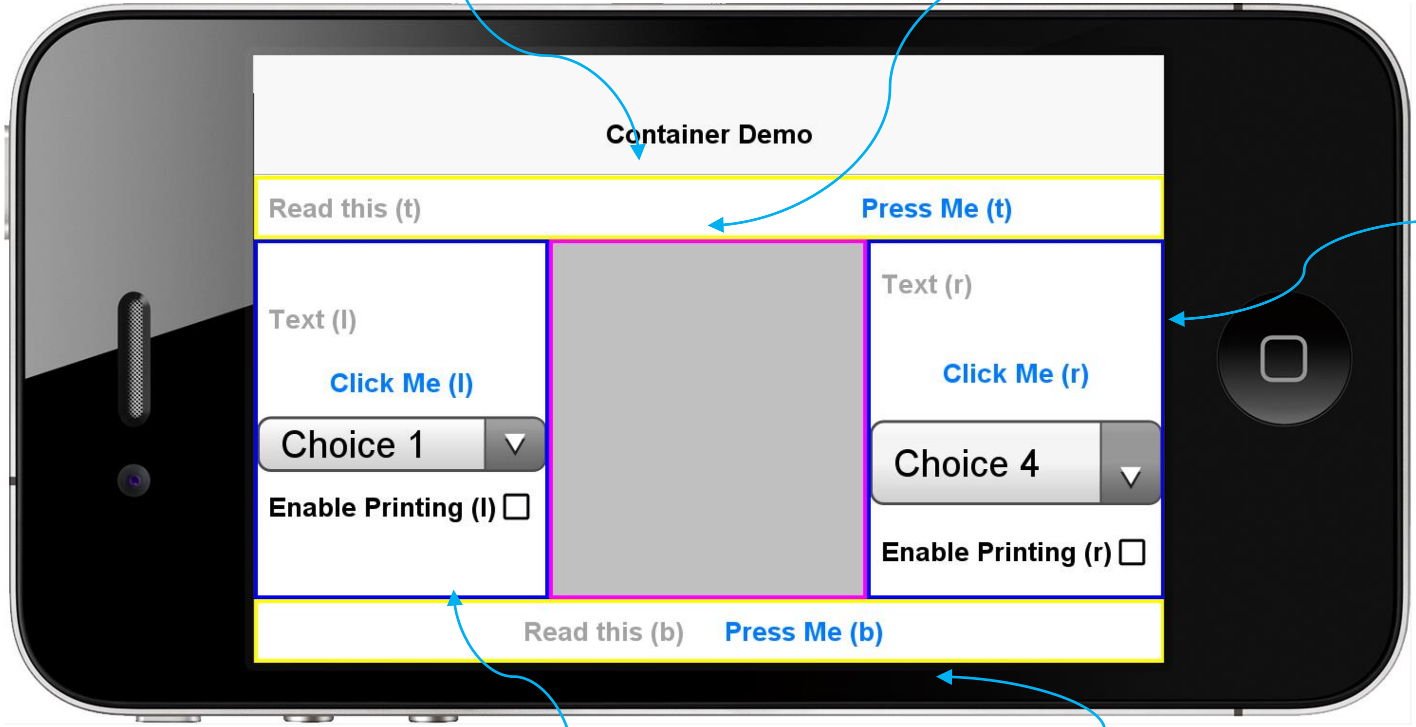
Container Example (cont.)

```
... 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)



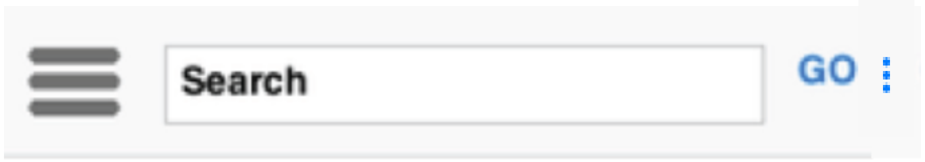
`Container` in East with `GridLayout` (space is divided to four equally-sized cells)

`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)

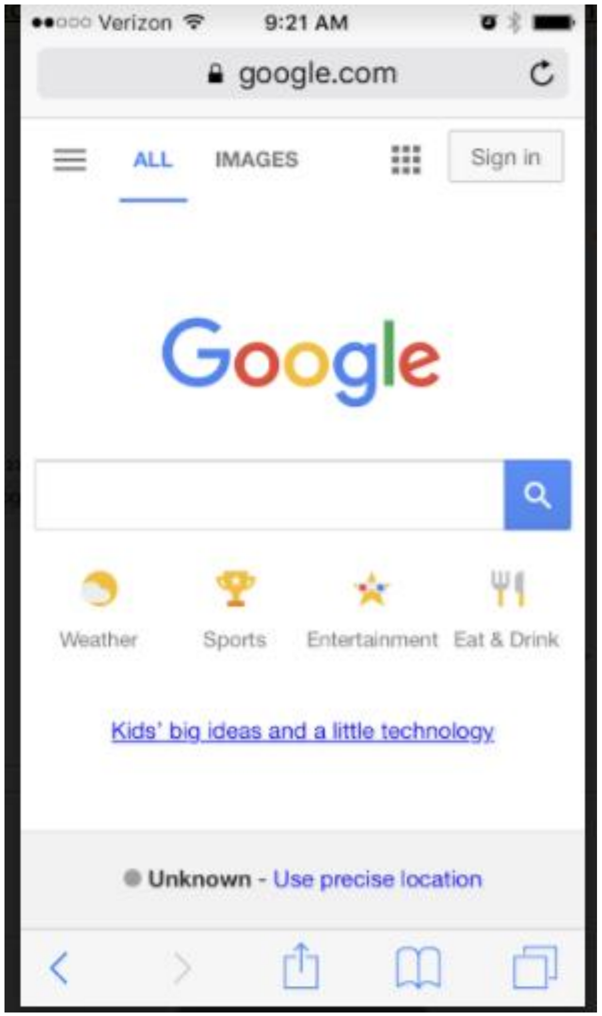
CN1 `Toolbar` class

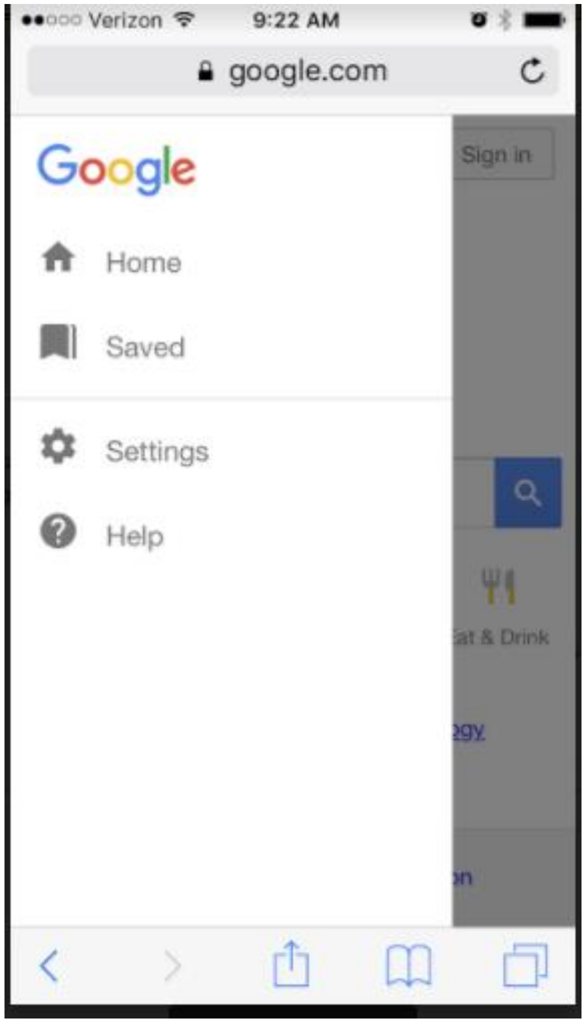
- Provides deep customization of the title bar area of your form.



- Set it to your form with: `myForm.setToolbar(toolbar)`
- Allows adding commands to four 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)

Example Toolbar

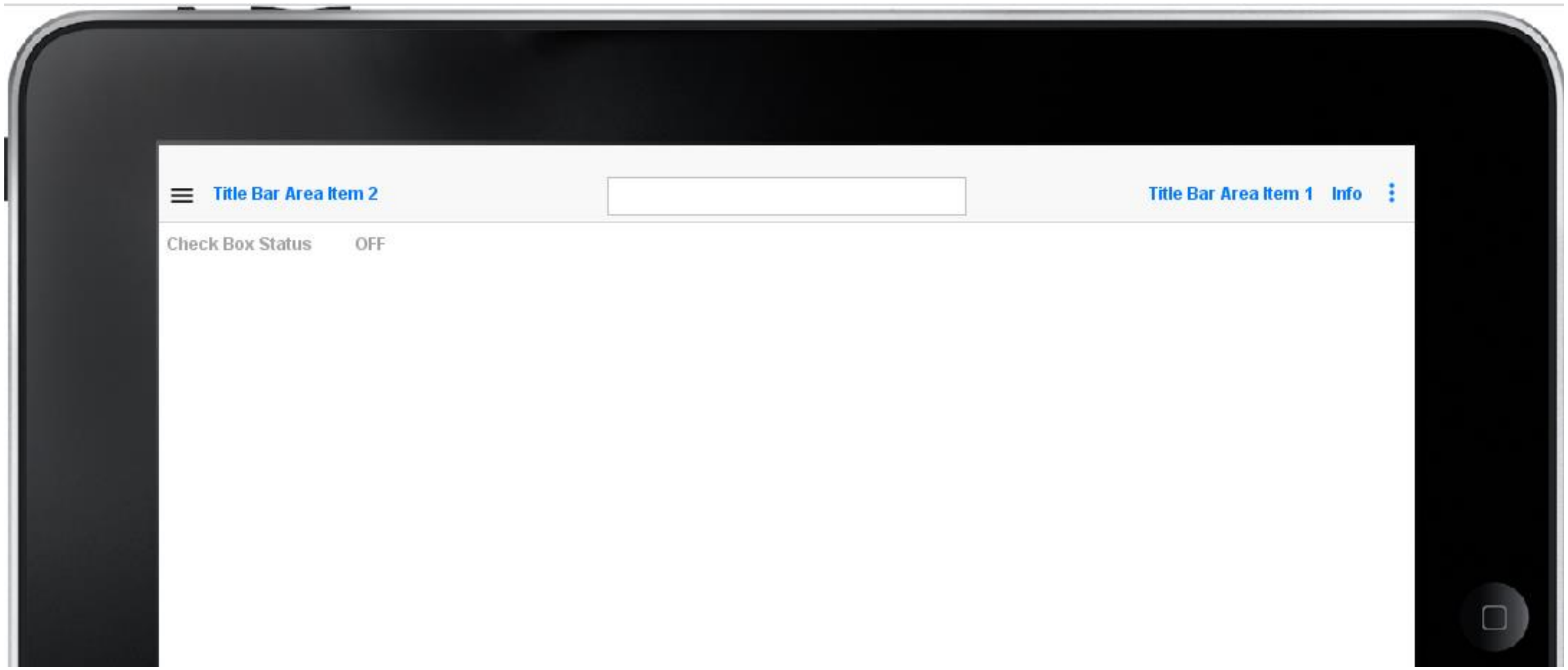




Adding Items to Title Bar

```
/* Code for a form with a toolbar */
Toolbar myToolbar = new Toolbar();
setToolbar(myToolbar);//make sure to use lower-case "b", setToolBar() is depreciated
//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.)



Check Box Status OFF

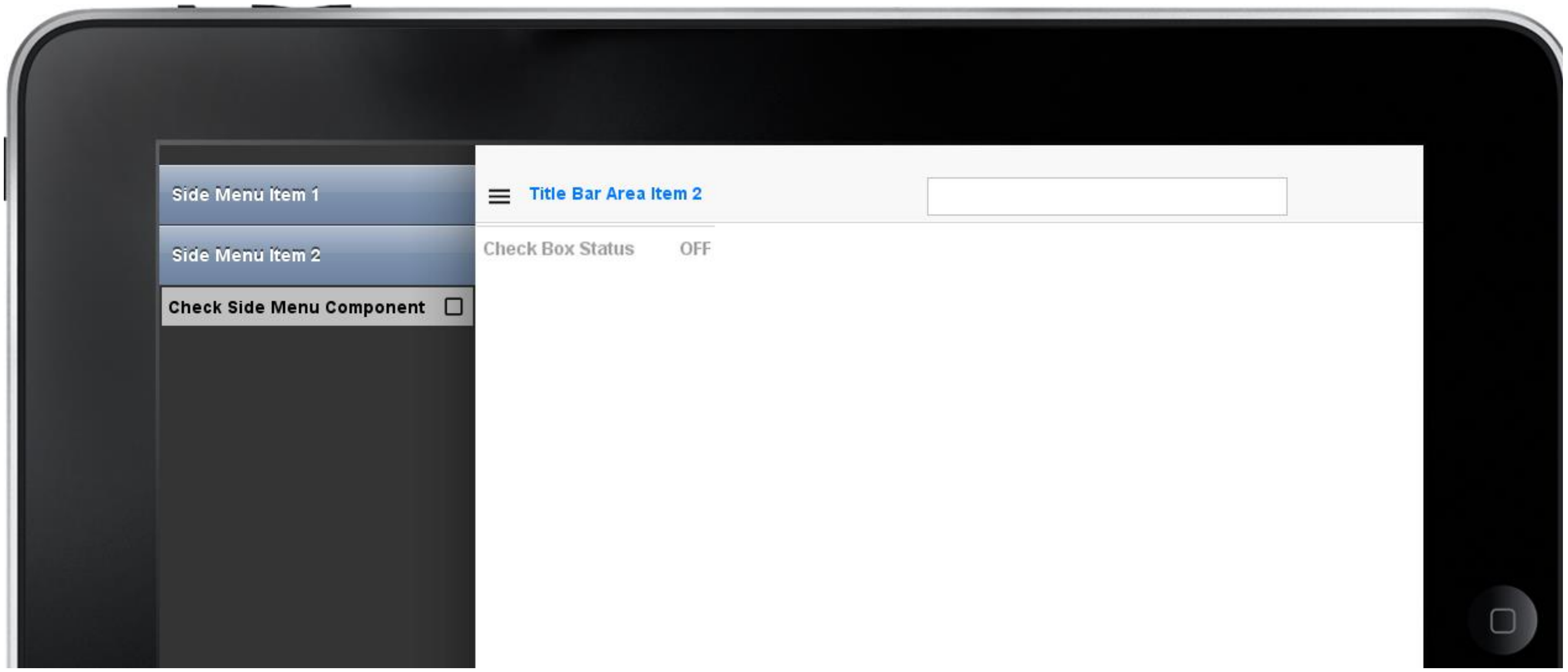
Complex Menus

- 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..)
Command sideMenuItemCheck = new Command("Side Menu Item Check ");
CheckBox checkSideMenuComp = new CheckBox("Check Side Menu Component");
//set the style of the check box
checkSideMenuComp.getAllStyles().setBgTransparency(255);
checkSideMenuComp.getAllStyles().setBgColor(ColorUtil.LTGRAY);
//set "SideComponent" property of the command object to the check box
sideMenuItemCheck.putClientProperty("SideComponent", checkSideMenuComp);
//add the command to the side menu, this places its side component (check box) in the side menu
myToolbar.addCommandToSideMenu(sideMenuItemCheck);
```

- We will later see how to attach operations to commands and link them to the components in menus…

Complex Menus (cont.)


Side Menu Item 1
Side Menu Item 2
Check Side Menu Component
Title Bar Area Item 2
Check Box Status
OFF