

# Informatics 134

Software User Interfaces  
Spring 2023

---

Mark S. Baldwin

*baldwinm@ics.uci.edu*

5/22/2023

# Agenda

1. Upcoming
2. Layout and Geometry Management
3. User Interface Layout Tools
4. References

## Upcoming

---

# Upcoming

- Today:
  - Layout and Geometry Management
  - Introduce A4
  - Tonight: A3 is DUE
- This Week:
  - Work on user evaluations
  - Work on prototypes/final product

# Layout and Geometry Management

---

# Layout and Geometry Management

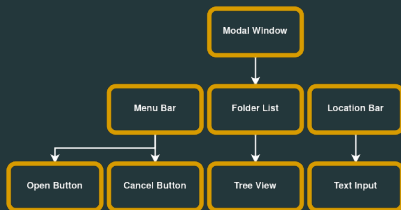
## From Primitives to Containers

### Quick Recap

Graphical toolkits are hierarchical

Build widgets with graphical primitives

Build widgets with widgets

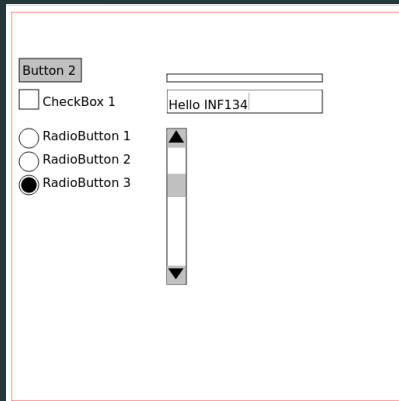


# Layout and Geometry Management

## From Primitives to Containers

Building widgets with widgets...

Think about what you have built so far...  
How might you build new widgets with  
existing widgets?



# Layout and Geometry Management

## From Primitives to Containers

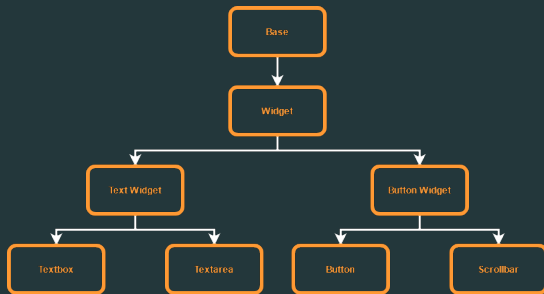
Some examples...

Scrollbar -> Scroll Pane

Button -> Scrollbar button

Textbox -> Text Area, other  
text input widgets

Checkbox and Radiobutton ->  
Selection or boolean widget





# Layout and Geometry Management

## From Primitives to Containers

Containers...store and manage individual widgets

Individual widgets are placed in containers (like our 'window' ex.)

Containers can be placed in containers

Design patterns...



## From Primitives to Containers

Decorator pattern: add behavior to an existing [graphical] object

[Wikipedia, 2021b]

- Extend functionality of object

- Does not change expected behavior of object

- Examples ???

# Layout and Geometry Management

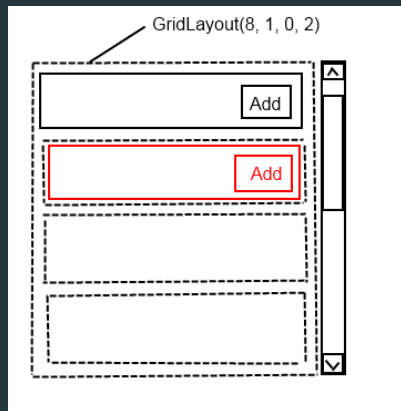
## From Primitives to Containers

Decorator pattern: add behavior to an existing [graphical] object [Wikipedia, 2021b]

- Extend functionality of object

- Does not change expected behavior of object

- Example: Scrollable list (a list widget decorated with a scroll pane)



[java2s.com, 2021]

## From Primitives to Containers

Composite pattern: a [graphical] object that can behave as a single object or a collection of objects [Wikipedia, 2021a]

Conceptually similar to recursive logic, lists of lists...

Containers of containers can lead to more complex interfaces, but easier to maintain and reason about

# Layout and Geometry Management

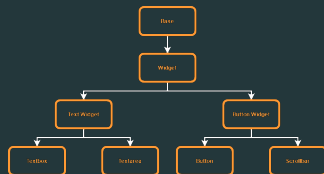
## From Primitives to Containers

This approach is common across nearly all graphical toolkits

Take advantage of OOP concept of inheritance

Can build parallel hierarchies for themes, resources, etc.

Support layout!!!



## From Containers to Layout Managers

As a GUI grows in complexity, there will be a need for layout and geometry management!

Must support:

- Different devices

- Resolutions

- Screen sizes

- Font sizes

- Accessibility

- Internationalization

## From Containers to Layout Managers

### Managing layout

- Layout is controlled by a manager rather than the widget

- Layout types represent a collection of equations to position widgets

- Rules can be applied to individual widgets (min width, left align, etc.)

- Conceptually similar to HTML and CSS, rules vary

## From Containers to Layout Managers

Managing layout

- Packing (1D, borders boxes)

- Gridding (2D, grids tables)

- Other (springs, dynamic algorithm)

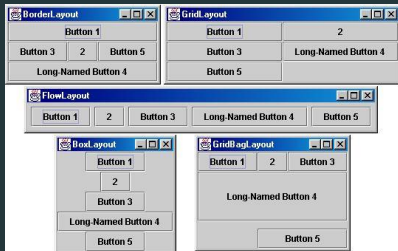


# Layout and Geometry Management

## From Containers to Layout Managers

A good example from Java Swing

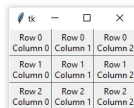
Layout types apply different algorithms to arrange widgets automatically



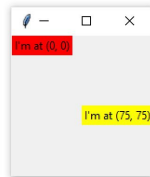
# Layout and Geometry Management

## From Containers to Layout Managers

TKinter's Grid/Pack/Place



Row 0 Column 0	Row 0 Column 1	Row 0 Column 2
Row 1 Column 0	Row 1 Column 1	Row 1 Column 2
Row 2 Column 0	Row 2 Column 1	Row 2 Column 2



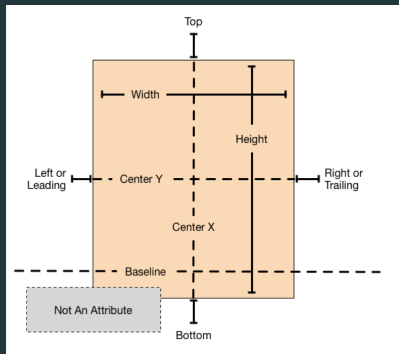
# Layout and Geometry Management

## From Containers to Layout Managers

Apple's Auto Layout (the gold standard?)

Define objects, attributes, and relationships

Attributes define constraints, the layout engine updates accordingly



# Layout and Geometry Management

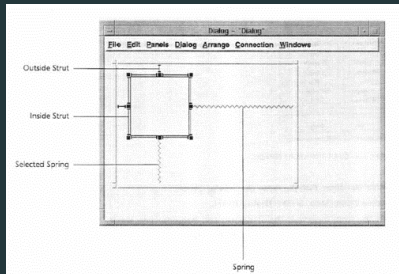
## From Containers to Layout Managers

Implementations vary across time and toolkit

Historically "Struts and Springs" most prevalent

Most toolkits offer variations on grid, fixed, placed (a3 is fixed-though, in practice we'd just use CSS)

Most are constraint-based (program rules, let engine adjust based on external criteria)



[Hudson and Mohamed, 1990]

# Layout and Geometry Management

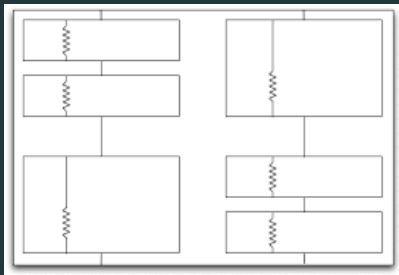
## From Containers to Layout Managers

Struts and Springs: a constraint based layout

Struts are rigid points of attachment to a nearby object

Springs are flexible points of attachment to a nearby object

What happens when the window pictured here is resized?



## From Containers to Layout Managers

### Constraints...

- Widget relationship expressed by programmer

- Expressed relationship maintained by the toolkit

- Toolkit maintains algorithms depending on type of constraint system

- Algorithms vary by constraint system, "constraint solving algorithm", there are MANY!

Many solvers today focus on ML, adaptive UI, learning models



**Why do we need layout?**



# Layout and Geometry Management

## From Containers to Layout Managers

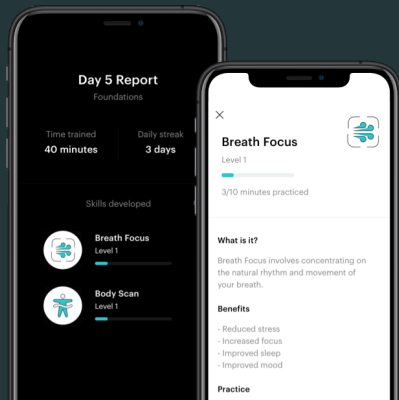
Reduce code complexity

Consistency

Add flexibility to UI

Visual appeal?

**Usability**



[balanceapp.com, 2021]

# User Interface Layout Tools

---

## Managing Layout Graphically

Building layout and geometry managers is hard

Writing code that uses layout managers is less hard, but hard

# User Interface Layout Tools

## Managing Layout Graphically

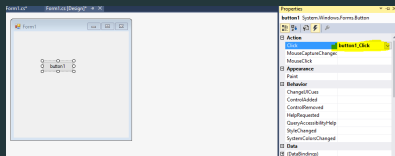
### User Interface Tools...

- Support rapid prototyping (pre-coding)

- Reusability (can apply to multiple platforms)

- Add consistency across platforms

- Bring designers, developers, and researchers together through a single tool



## User Interface Layout Tools

## Managing Layout Graphically

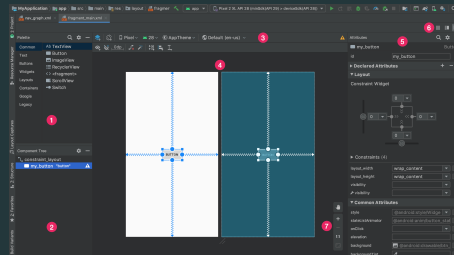
## User Interface Tools...

## Automate much of the coding process

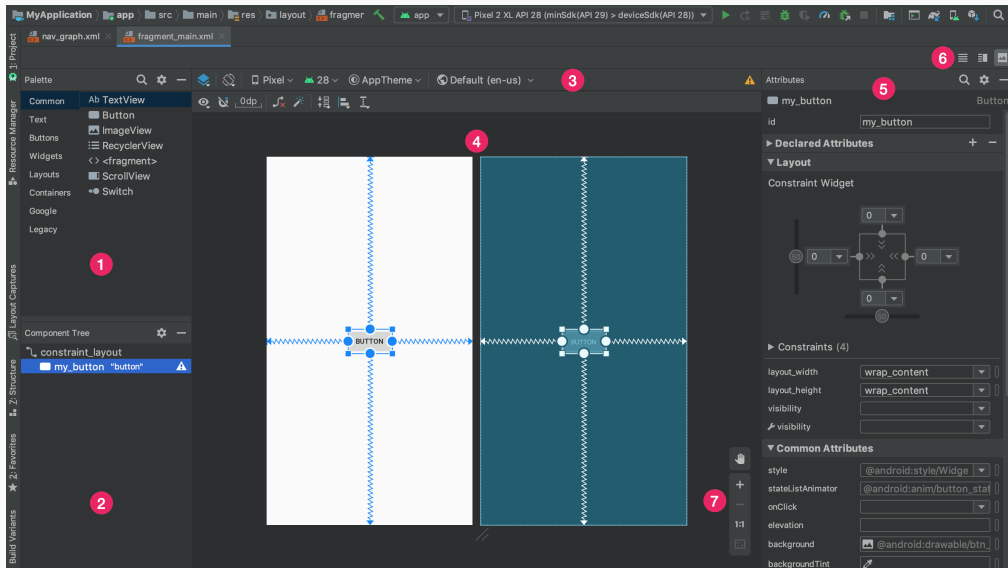
## Replace programming steps with graphical configuration

Lower level of expertise to create

## Raise level of reliability



[android.com, 2021]



[android.com, 2021]

# User Interface Layout Tools

## Managing Layout Graphically

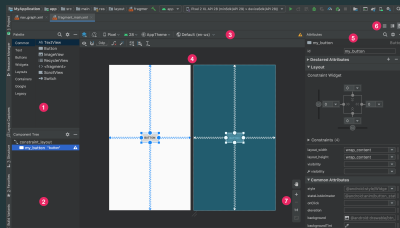
Lower the level of expertise and raise reliability...

Make creating a UI easy and easy to use

Invite non-programmers into the process

Support validation

Can drive important processes like undo, error recovery, and accessibility



[android.com, 2021]

# User Interface Layout Tools

## Some takeaways

As computational systems evolve, so will UIs and the tools that we use to build them

These types of tools are critical for building effective software interfaces

How will we build for future user interfaces?



# User Interface Layout Tools

## Some takeaways

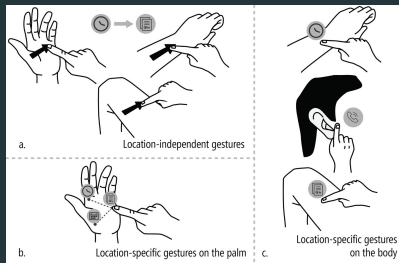
Future user interfaces?

Wearables?

Augmented Reality?

Voice or Conversational agents?

On Body, Eyewear?



# **Individual Assignment 4**

## References

---

## References i



android.com (2021).  
**Layout editor.**



balanceapp.com (2021).  
**Balance - personalized meditation.**



Hudson, S. E. and Mohamed, S. P. (1990).  
**Interactive specification of flexible user interface displays.**  
*ACM Trans. Inf. Syst.*, 8(3):269–288.



java2s.com (2021).  
**Jpanel « jscrollpane « java swing qa.**



Wikipedia (2021a).  
**Composite pattern.**



Wikipedia (2021b).  
**Decorator pattern.**