Swing Layout Epilogue: GridBagLayout

• Now that you’ve logged some decent hours with Swing layout, let’s cap it off with the mother of all layout managers — GridBagLayout

• The main advantage of GridBagLayout is that it can replicate virtually any layout that you can imagine — it is the most general of the built-in layout managers

• The main disadvantage is that generality comes at a price — GridBagLayout code can be difficult to read, error-prone, and difficult to modify

• But, there are ways to minimize this difficulty…

How I Like to Use GridBagLayout…

• Prepare multiple GridBagConstraints objects for commonly-used component setups

• Should it fill its area or not?

• Should a component be “greedy” about space?

• How much white space should surround the component?

• Use GridBagConstraints.RELATIVE (i.e. relative positioning) so you can rearrange components by changing the order with which you add them
GridBagLayout Becomes Necessary When…

- The desired layout has a great degree of two-dimensional interdependence — that is, multiple components influence each other in terms of both their width and height

- You want per-component control of alignment, fill, “greediness” for space (a.k.a. “weight” in GridBagConstraints terms), and margins (a.k.a. “insets” in overall Swing-speak)

Parting Shot: TableLayout

- We have only covered the “built-in” layout managers, and not even all of them — but as mentioned before, simply by implementing the LayoutManager2 interface, you can create your own layout managers

- This is generally a daunting task, and is not recommended unless you have a Really Brilliant layout manager strategy

- One good example is TableLayout — check it out if you feel like it and see what you think:
  
  http://www.clearthought.info/software/TableLayout