# Chapter 2: The Last Region

**The Last Region**

The last panel to implement in our window uses components we've seen before, so you won't need to learn anything new for it. It will be a chance to review and reinforce some of what you have already learned. Let's start with a look at what goes into the south region of the window.

We know we'll need a panel because there are multiple components to put into the south region. We have some text fields where we want users to enter data. To the left of those, we have some labels to prompt the user on what to enter. To manage those two groups of components, we'll use subpanels. We'll need one subpanel to organize the labels and another to size and line up the text fields the way we want them. Last, there's a border around the whole outer panel to tie the pieces together.

To begin, let's create a method named *makeSouthRegion()* that will set up the south region of the screen. Like we did with the other four regional methods, we'll add a call to this method to our makeContent() method, like this:

```
makeSouthRegion();
```

The first thing to do in makeSouthRegion() is to set up the outer panel, along with its layout manager and border. Here's how we'll do that:

```
private void makeSouthRegion()

{

JPanel panel = new JPanel();
panel.setLayout(new BoxLayout(panel,BoxLayout.X_AXIS));
panel.setBorder(BorderFactory.createTitledBorder("Deliver To:"));



}
```

The first line of the makeSouthRegion() method creates our outer panel, a new JPanel object named *panel*. The second line sets that panel's layout manager, and since we want our two subpanels side by side, we'll use the BoxLayout manager with its X_AXIS option. The third line puts a border around the outer panel using the same format that we used for our other regions, with the title we want it to show: *Deliver To:*.

Now we're ready to work on the subpanels. We'll start with the left one, which holds the text labels. First, we need to create the subpanel. Then we'll need to set up its layout manager. Since we want the labels to line up one above the other vertically, we'll use the BoxLayout again, but this time we'll use its Y_AXIS argument, like this:

```
JPanel smallPanel = new JPanel();

smallPanel.setLayout(new BoxLayout(smallPanel,BoxLayout.Y_AXIS));
```

The first line above creates another JPanel object and names it *smallPanel*. The next line sets its layout manager. Once we take care of those, we can add the labels, like this:

```
smallPanel.add(new JLabel("Name:"));

smallPanel.add(new JLabel("Address:"));

smallPanel.add(new JLabel("City, St, Zip:"));
```

Since we don't need to refer to these labels again, we don't need to give them names when we create them. We can add the objects directly to the subpanel.

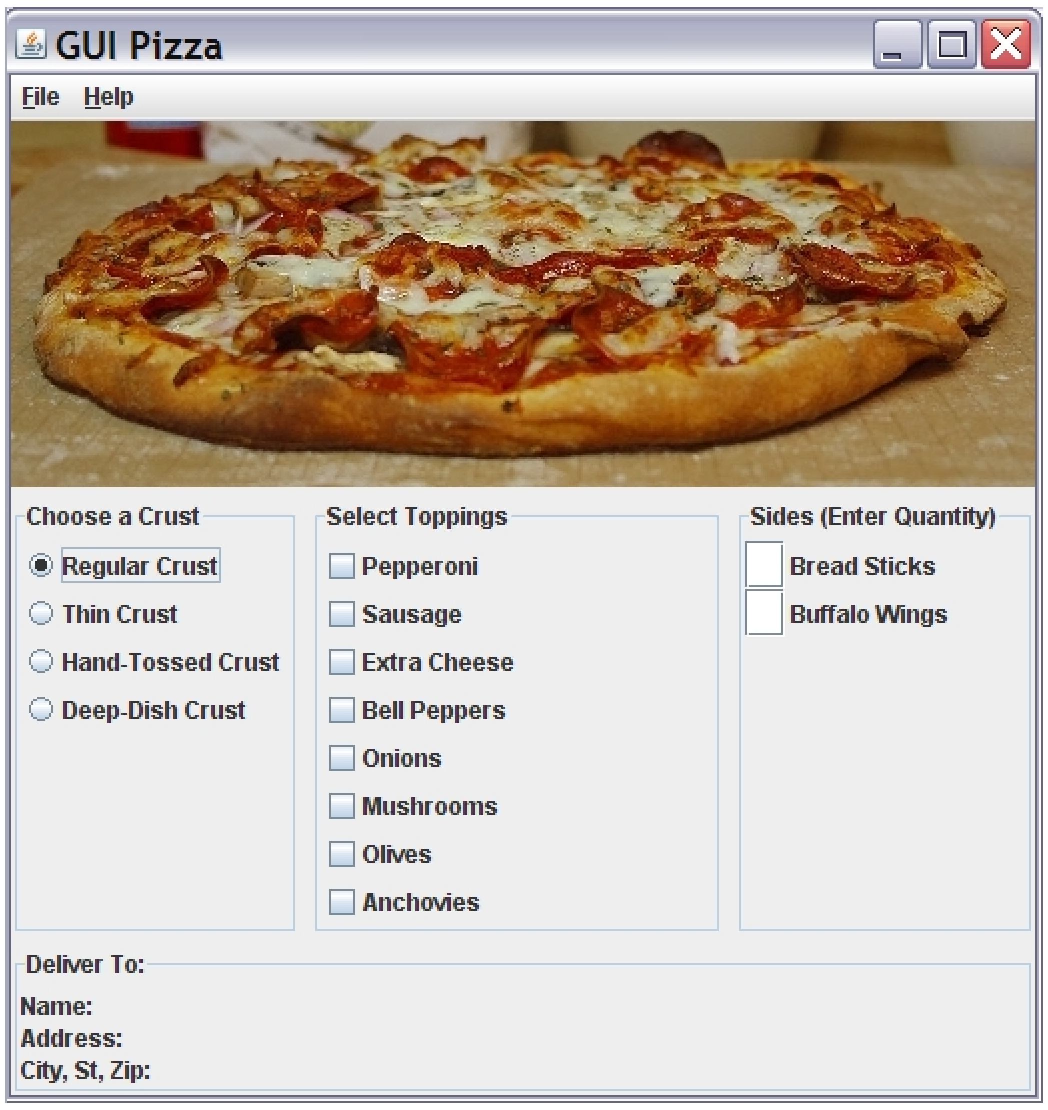
Once that's done, we're ready to add the subpanel to the outer panel with one line that looks like this:

```
panel.add(smallPanel);
```

If we want to view the window now to see how the labels look, we just need to add one more line to put our outer panel into the south region of the window:

```
contentPane.add(panel, BorderLayout.SOUTH);
```

Running the program at this point gives us this:



Window with labels

All that's left to add (visually) are the text fields to receive the name and address information. We can reuse our subpanel variable, smallPanel, since the object it refers to is already added to the window. Now we'll create a second subpanel object, give it a layout manager, and add the text fields to it, like so:

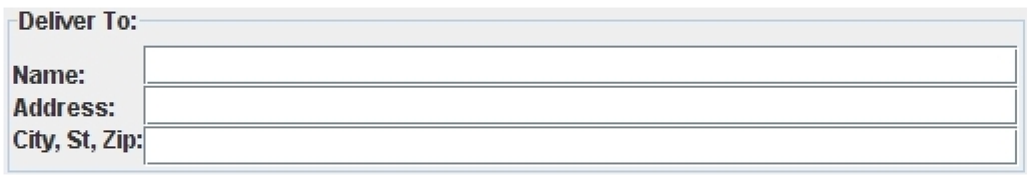
```
smallPanel = new JPanel();

smallPanel.setLayout(new BoxLayout(smallPanel,BoxLayout.Y_AXIS));

nameText = new JTextField();

addressText = new JTextField();

cityText = new JTextField();

smallPanel.add(nameText);

smallPanel.add(addressText);

smallPanel.add(cityText);
```

Since we'll refer to the text fields again in another method, we had to connect them to available names. We used the variable names we declared at the start of the class for that purpose: nameText, addressText, and cityText.

Once we've created the JTextField objects and added them to the subpanel, we add the subpanel to the outer panel with this line again:

```
panel.add(smallPanel);
```

The south panel now looks like this if we run the program:



Panel with labels and text fields

It's all there now, but it's a little crowded, especially between the longest label (City, St, Zip) and the text field to its right. So the last thing we'll do in the south region is add a little space between the components with an empty border, one with no titles, lines, or anything else to mark its edges. It just adds some space to distribute the components better. We'll add the empty border to both subpanels with this line of code:

```
smallPanel.setBorder(BorderFactory.createEmptyBorder(3,3,3,3));
```

That line adds an empty border three pixels wide on all four sides of the subpanel. The four numeric arguments give the number of pixels for the top, left, bottom, and right borders, respectively. If we want a larger border in any direction, we can get it by changing just that number.

Remember to add the line twice, once to each subpanel, before adding the subpanels to the outer panel.

If you run the program now, it should open a window that looks almost exactly like the one at the start of the lesson. The only difference is that in the window at the beginning of the lesson, I also added a small empty border (six pixels on each side) to the entire content pane, giving it a little room around the edges so the components don't bump up against the side of the window. If you want to add that border, I'll leave it to you to do in the makeContent() method.

If you have problems with anything we've gone over so far, let me know in the Discussion Area.