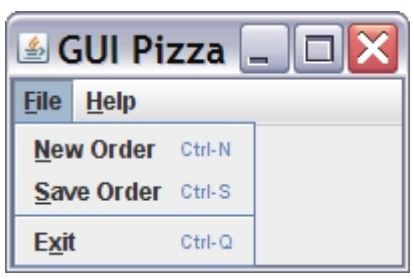
Chapter 2: Getting Set Up

Getting Set Up

I imagine that the thought running through your head right now is, "Where do I start?" If that's the case, I know how you feel. I've been there a few times myself. So here's where we'll start: Take our finished program from the previous lesson and remove everything from the content pane. Then modify the menus so they include only the following options:

For the File menu, include three options: New Order, Save Order, and Exit. That menu will look something like this:



The File menu

- The New Order menu item will clear all fields on the screen and set them back to their default values.
- The Save Order item will save the order. Since printing parameters vary so much between systems, we'll make life easier by simply saving the order to a file in text format. We'll talk more about this in the next lesson.
- The Exit item will, of course, close the window.

Our Help menu will have only the About option, and it will look like this:

The Help menu

When clicked, the About GUI Pizza item should produce a dialog box that looks something like this:

The About dialog box

Since these are all things we've done before, I'll let you get that much set up before we continue. If you have trouble getting it to work right, let me know in the Discussion Area, or take a peek at my code here.

Solution: Try not to peek!

If you looked at my code, you probably noticed that I added a few more methods than last time to organize my code hierarchically and to keep my methods small. So now my start() method calls makeMenus(), and makeMenus() calls makeFileMenu() and makeHelpMenu(). We'll continue this type of pattern as we go on because it makes the program easier to follow once you get used to it.

Our last setup task is to add variables to the program for each of the buttons, check boxes, and text fields we'll use. We'll do this now because we want to have them available to the code in more than one of our methods. We're also going to take a different approach to using these buttons and fields than we did in previous lessons. We won't assign an ActionListener to each one, even though we can.

Why not? We don't really want our program to jump in and take action every time a user clicks or changes one of these buttons or fields. After all, users may change their minds several times about which crust they want, and it doesn't matter to us until they save the order.

Instead of listening to every change of every item, then, our program will wait until the user clicks the Save Order option in the File menu. Then the program will look and see which items have information for us and which don't.

To make that happen, we're going to add an instance variable to our program for each button, box, and text field on the screen. This way, we can use them whenever we need them. I've listed the variables we'll need below in their declarations. Their names and types should make it clear which variable corresponds to which item on the screen. We'll add them at the start of the class, right after our JFrame declaration.

```
// radio buttons and button group

private JRadioButton regularCrustButton;

private JRadioButton thinCrustButton;

private JRadioButton handCrustButton;

private JRadioButton deepCrustButton;

private ButtonGroup crustButtonGroup;


// check boxes

private JCheckBox pepperoniBox;

private JCheckBox sausageBox;

private JCheckBox cheeseBox;

private JCheckBox pepperBox;

private JCheckBox onionBox;

private JCheckBox mushroomBox;

private JCheckBox oliveBox;

private JCheckBox anchovyBox;


// text fields

private JTextField breadSticksText;

private JTextField buffaloWingsText;

private JTextField nameText;

private JTextField addressText;

private JTextField cityText;
```

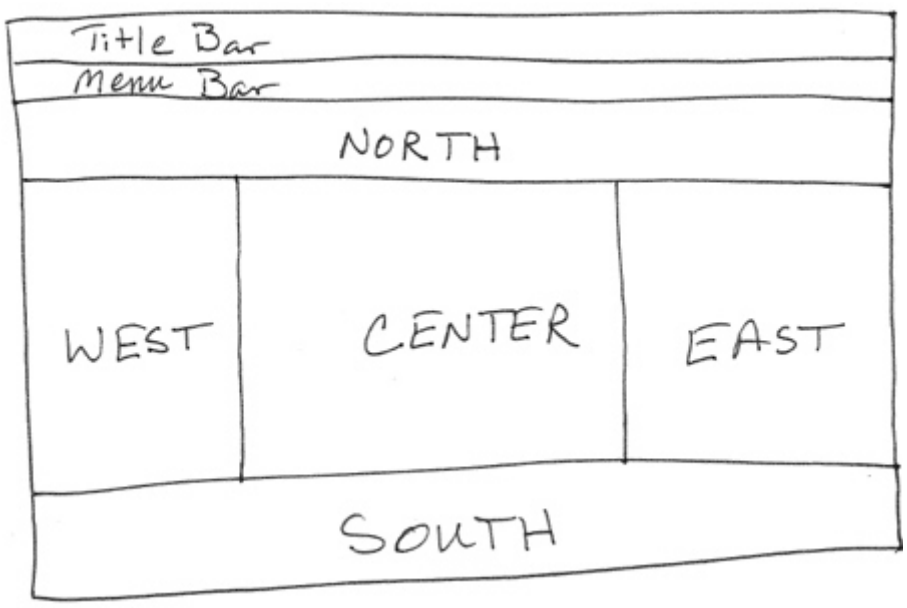
Designing Our Window

Now that we can produce a window with menus, where do we go from here?

Let's start by taking a good look at the window we want to create. What elements does it have? How are they laid out? Does their layout match up well with any of the layout managers we learned about in Lesson 6?

I hope you answered "Yes!" to the last question. These elements do match up well with one of the layout managers we studied. "Which one?" you might ask.

Do you remember the BorderLayout manager? Remember how it divided a window? It was like this:



Java's BorderLayout

If we put the pizza picture in the north region, the crust choices in the west, toppings in the center, side orders in the east, and the address entry in the south, then things line up pretty well, don't they? That's how we're going to approach our window.

To get ready to add the components, we need to initiate our frame's layout manager. If you look in the start() method, you'll see that we already have a call to makeMenus() to set up our menus. But we haven't done anything in our content pane yet. Let's initiate the content pane and get the layout manager laid out, so to speak.

To keep with our program's organization scheme, we're going to add a call in the start() method, right after the makeMenus() call. It will call a method that sets up our content pane, so let's call it makeContent(). That one-line insertion looks like this:

```
makeContent();
```

Of course, to make the call work, we have to add a new method with that name. That method is where we'll do the work to set up the body of our window. Right after the last of the menu methods, let's add a method that looks like this:

```
private void makeContent()

{

}
```

The method doesn't do anything yet, but just hang in there—we'll have it working in no time. (Well, maybe in a little while.)

We'll need a content pane in this method, too. Since we'll be working with this pane in more than one method, let's make it easy to use by declaring it at the top of our class, right after our frame. We will also need to use methods that don't belong to the Container class, but to the JPanel class. So we'll declare the variable as a JPanel object, like this:

```
private JPanel contentPane;
```

Now we're ready to go to work on our content. Here's how we get the pane and link it to our layout manager:

```
contentPane = (JPanel)frame.getContentPane();

contentPane.setLayout(new BorderLayout(6,6));
```

Note the addition of "(JPanel)" in front of the call to getContentPane(). That is called "casting" in Java, and it's telling Java that even though the getContentPane() method says it's returning a Container object to maintain its backward compatibility, we actually know it's a JPanel object and want Java to treat it like one.

There's one difference between this layout constructor and the one we used in the last lesson. The two numbers in the constructor argument list tell it that I want 6 pixels both horizontally and vertically between components in my window. If you look at the window again in the first figure up above, you'll see that there is a little spacing between the different groups of components. It looks better that way, and I wanted to introduce this aspect of the layout manager here.

Now we have the window and its layout, and we already have the menu bar set up. We're finally ready to start adding content. Let's start at the top with the picture!