Chapter 3: Making the Menus Work

Making the Menus Work

Now that our window is visually complete, we need to make the rest of the menu options work. There are four menu items, and right now only two of them work (Exit and About GUI Pizza). The other two (New Order and Save Order) don't do anything yet. We want the New Order menu item to reset all the buttons and fields to their original values, as if we had just opened the window. When users want to erase everything they've done and start the order over, they can use this item. The Save Order button will write all the order information to a text file that can be printed or displayed. We're going to start with the New Order option, since it's the easier of the two. Then we'll finish up with the Save Order option.

To initiate our processes, we're going to use event listeners, but we'll take a somewhat different approach to how we use them than we did earlier. In previous projects, we assigned listeners to every item in the window that we wanted to interact with. We could do that this time, too, and attach a listener to each radio button, check box, and text field. Then, every time the user changed any one of them, we could check its contents and keep track of what our user is doing on the screen.

But this time there's one big difference: We don't care how many times a user changes any of the fields or buttons in our window. We don't need to trigger an event and check values each time something changes. If we have users who can't decide which crust they want or which toppings they like, and who keep clicking different options or changing side order numbers, we would be triggering events all the time that won't matter.

All that matters in this window is what the values are when our user finally clicks the Save Order menu item. The About GUI Pizza and Exit menu items completely ignore what is on the screen. Our New Order menu option will set the values of the components, but it doesn't care what is in them when a user clicks it. The only time the settings of the fields and buttons matter is when the user clicks the Save Order menu option.

So here's what we'll do: We won't attach listeners to any of the buttons or fields in the window. We don't need to. In fact, a component can have any number of listeners, including none, but that's a topic for another time. We're just going to set up event listeners for the two menu items we still need to finish. The New Order listener will simply reset the values of all the buttons and fields in the menu. The Save Order listener will check the values of all the buttons and fields in the window and then create its output file according to their values. We can do that, if you remember, because even though our listeners are objects of another class, that class is an *inner class* in our program, so it has access to all the data in our outer class, not just to the component it is listening to.

The New Order Option

No matter what changes our user has made to the screen, when he or she clicks the New Order menu option, we want to set all the buttons and fields back to the way they were when the window opened.

That means the Regular Crust radio button should show up as selected, all the check boxes should be unselected, and all the text fields should be empty.

What we need to do, then, is delete the current contents of the actionPerformed() method in NewListener, the listener class for the New Order option. That gives us an empty listener class that looks like this:

```
private class NewListener implements ActionListener

{

     public void actionPerformed(ActionEvent e)

     {

     }

}
```

Now we're ready to add the code that will reset everything.

Let's start with our radio buttons. Radio buttons created from the JRadioButton class all have a method named *setSelected()* that turns their selections on or off. The method expects a Boolean parameter, so if I give it a value of *true*, the radio button appears selected. If I give it a value of *false*, the button appears unselected. Remember that only one radio button in a group can be selected at a time.

To set the Regular Crust radio button on and all the other radio buttons off, I only need one line of code:

```
regularCrustButton.setSelected(true);
```

Now for the check boxes. The JCheckBox class also contains a setSelected() method, and it works on check boxes the same way as on radio buttons, with one difference. Since any number of check boxes can be selected at a given time, changing one won't affect the others. This means we'll need to go through and turn each one off individually, like this:

```
pepperoniBox.setSelected(false);

sausageBox.setSelected(false);

cheeseBox.setSelected(false);

pepperBox.setSelected(false);

onionBox.setSelected(false);

mushroomBox.setSelected(false);

oliveBox.setSelected(false);

anchovyBox.setSelected(false);
```

That leaves only the text fields. Text fields built from the JTextFieldclass also have a method to set their contents. In this case, it's the setText() method, and it requires a String argument with the value we want to put into the text box. To clear a text field, all I need to do is give it an empty string for an argument. We'll need to clear each field individually:

```
breadSticksText.setText("");

buffaloWingsText.setText("");

nameText.setText("");

addressText.setText("");

cityText.setText("");
```

That takes care of our NewListener class definition. Only one left, our SaveListener class, which is the event listener for the Save Order option.