```
        // Make all the objects
    private void makeTheObjects( )
    {
        theCanvas = new GUICanvas( );
        theCanvas.setBackground( Color.white );
        theCanvas.setSize( 100, 100 );

        theShape = new Choice( );
        theShape.add( "Circle" );
        theShape.add( "Square" );

        theColor = new List( 2, false );
        theColor.add( "red" );
        theColor.add( "blue" );
        theColor.select( 0 ); // make red default

        theXCoor = new TextField( 5 );
        theYCoor = new TextField( 5 );

        CheckboxGroup theSize = new CheckboxGroup( );
        smallPic  = new Checkbox( "Small",  theSize, false );
        mediumPic = new Checkbox( "Medium", theSize, true );
        largePic  = new Checkbox( "Large",  theSize, false );

        theFillBox = new Checkbox( "Fill" );
        theFillBox.setState( false );

        theDrawButton = new Button( "Draw" );
        theMessage = new TextField( 25 );
        theMessage.setEditable( false );
    }
```

Figure D.6 Code that constructs the objects in Figure D.1

```
Label( );
Label( String theLabel );
void setText( String theLabel );
```

Button

The *Button* is used to create a labeled button. When it is pushed, an *action event* is generated.

The *Button* is used to create a labeled button. Figure D.1 contains a `Button` with the label *Draw*. When the `Button` is pushed, an *action event* is generated. Section D.3.3 describes how action events are handled. The `Button` interface is similar to the `Label`. Specifically, a `Button` is constructed with an optional `String`. The `Button` label can be changed with the method `setText`. These methods are

```
Button( );
Button( String theLabel );
void setText( String theLabel );
```

Choice

The *Choice* is used to select a single string via a pop-up list of choices. Only a string that is one of the choices can be selected, and only one choice can be selected at any time. In Figure D.1, the type of shape is a Choice object; *Circle* is currently selected. Some of the Choice methods are

The *Choice* is used to select a single string via a pop-up list of choices.

```
Choice( );
void   add( String item );
String getSelectedItem( );
int    getSelectedIndex( );
void   select( int index );
```

A Choice is constructed with no parameters. Strings can then be added to the list of Choice options. When getSelectedItem is called, a String representing the current selected item (or null if no choice is selected) is returned. Instead of returning the actual String, its index (as computed by the order of calls to add) can be returned by calling getSelectedIndex. The first item added has index 0, and so on. This can be useful because if an array stores information corresponding to each of the choices, getSelectedIndex can be used to index this array. The select method is used to specify a default selection.

List

The *List* component allows the selection from a scrolling list of Strings. In Figure D.1, the choice of colors is presented as a List. The List differs from the Choice in three fundamental ways:

The *List* component allows the selection from a scrolling list of Strings. It can be set up to allow for either one selected item or multiple selected items.

1. The List can be set up to allow either one selected item or multiple selected items.
2. The List allows the user to see more than one choice at a time.
3. The List will take up more screen real estate than the Choice.

The basic List methods are

```
List( );
List( int rows, boolean multipleSelections );
void       add( String item );
String     getSelectedItem( );
String [ ] getSelectedItems( );
void       select( int index );
```

A List is constructed with either no parameters or two parameters. The two-parameter constructor specifies the number of visible rows (in other words, the number of rows to be displayed) and a boolean that determines if multiple selections are allowed. The methods add, getSelectedItem, and