You can add this interface to an existing class or create a JComponent with these methods and use it as an object that you can add to either a JMenu or a JToolBar. The most effective way is simply to extend the AbstractAction class. The JMenu and JToolBar will then display it as either a menu item or a button, respectively. Further, since an Action object has a single ActionListener built in, you can be sure that selecting either one will have exactly the same effect. In addition, disabling the Action object has the advantage of disabling both representations on the screen.

Let's see how this works. We start with a basic abstract ActionButton class and use a Hashtable to store and retrieve the properties.

```
public abstract class ActionButton extends AbstractAction {
    Hashtable properties;

    public ActionButton(String caption, Icon img) {
        properties = new Hashtable();
        properties.put(DEFAULT, caption);
        properties.put(NAME, caption);
        properties,put(SHORT_DESCRIPTION, caption);

        properties.put(SMALL_ICON, img);
    }
    public void putValue(String key, Object value) {
        properties.put(key, value);
    }
    public Object getValue(String key) {
        return properties.get(key);
    }
    public abstract avoid actionPerformed(ActionEvent e);
}
```

Following are the properties that Action objects recognize by key name:

```
DEFAULT
LONG_DESCRIPTION
NAME
SHORT_DESCRIPTION
SMALL_ICON
```

The `NAME` property determines the label for the menu item and the button. The `LONG_DESCRIPTION` is not used at present. The `SMALL_ICON` property does work correctly.

Now we can derive an ExitButton from the ActionButton like this.

```
public class ExitButton extends ActionButton {

    JFrame fr;
    public ExitButton(String caption. Icon img) {
```