Menus, Tool Bars, and Dialogs in Swing

Direct use of menus and menu bars in Swing is generally straightforward:

- Menu bars are implemented via JMenuBar, menus via JMenu, and finally menu items via JMenuItem
- You can nest JMenus within JMenus — that's handled just fine; finally, JMenuItems are just like buttons — add action listeners
- There are also "check box" and "radio button" versions of JMenuItem

Tool bars are generally the same:

- Create a JToolBar
- Add whatever components you like — it generally acts like a box
- Add whatever listeners are needed for those components if you need interaction
- JToolBars have a free feature where, if they are added to JPanels with a BorderLayout, they can be dragged around that panel or even detached

For dialogs, you have two options:

- JOptionPane makes creation of common dialogs relatively easy, at the price of some flexibility
- Or, you can roll your own with JDialog

But It Can Get Complicated…

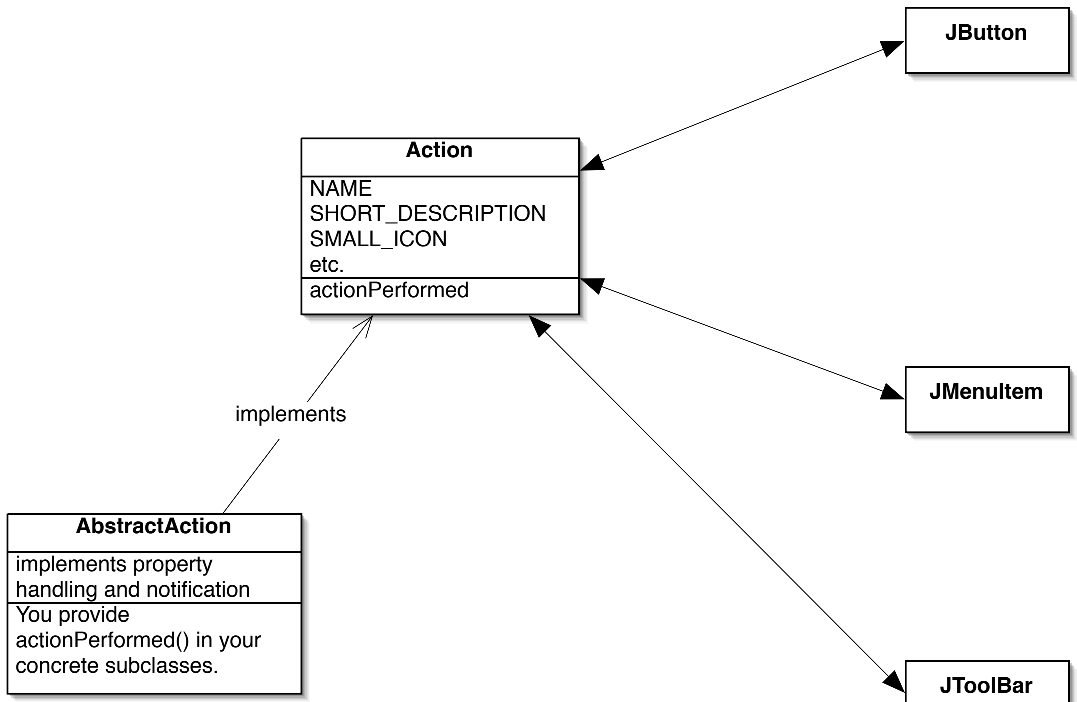
What if you want to make the same command available from both the menu bar and a tool bar, or another button in your user interface? It can get unwieldy:

```
_menuItem.addActionListener(this);
_button.addActionListener(this);
…
_menuItem.setEnabled(isCommandEnabled());
_button.setEnabled(isCommandEnabled());
```

And it gets worse if these components are all over the place (menu bars, different panels, different tool bars…)

Actions — MVC for Commands

- The *Action* interface abstracts a specific command that your program can perform, independently of the component that might trigger it
- *Action* objects can store properties like a name, tool tip text, an icon, accelerator key, and others
- When one of these properties changes, all components that are "bound" to that action will update as needed
- Best of all, *Action*s also have an *enabled* property that has the same sync-ing behavior



Actions Recipe

- Decide of your user interface is sufficiently complex to be worth the overhead of implementing Actions
- Implement some convenient mechanism for managing and accessing your Actions
- Build your components using Actions instead of lower-level Strings, Icons, and ActionListeners
- Whenever your application state changes, provide an algorithm that enables, disables, or otherwise updates affected Actions appropriately