# The Problem

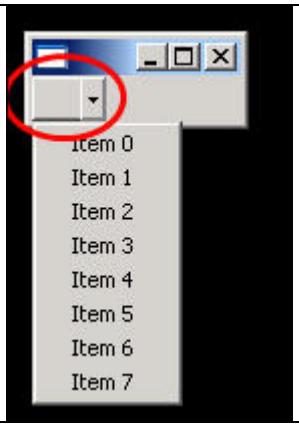
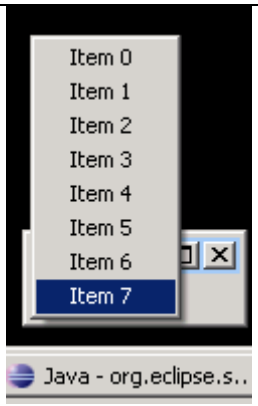
Snippet67 shows how to create a toolbar button with a drop-down menu in SWT:

```
final ToolItem item = new ToolItem (toolBar, SWT.DROP_DOWN);
item.addListener (SWT.Selection, new Listener () {
        public void handleEvent (Event event) {
                if (event.detail == SWT.ARROW) {
                        Rectangle rect = item.getBounds ();
                        Point pt = new Point (rect.x, rect.y + rect.height);
                        pt = toolBar.toDisplay (pt);
                        menu.setLocation (pt.x, pt.y);
                        menu.setVisible (true);
                }
        }
});
```

This is an awful lot of code for something as simple as a drop-down button. Wouldn't it be nice to be able to just create the button and just tell it what menu to use, like in the next sample?

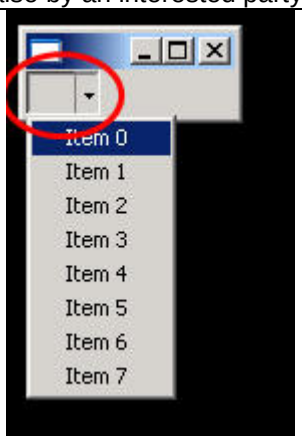
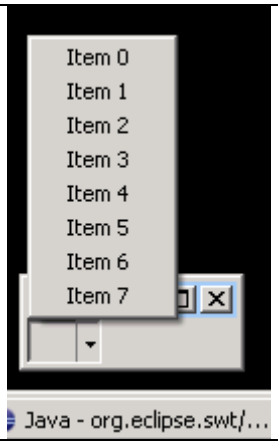
```
final ToolItem item = new ToolItem (toolBar, SWT.DROP_DOWN);
item.setDropDownMenu(menu);
```

This is the problem that my patch solves. By integrating the drop-down logic into SWT, some deficiencies of the original approach can also be corrected. For example, on win32 SWT drop-downs suffer from these 2 glitches:

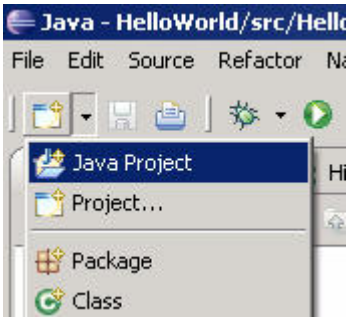
<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Drop-downs the Wrong Way 1<br>The drop-down button appears released. The button should be depressed, indicating that its drop-down menu is active.</td>
<td>Drop-downs the Wrong Way 2<br>When the menu cannot fit below the button, it is shifted up and over the button. The menu should not cover the button</td>
</tr>
</table>

# The Solution

My patch fixes both of these issues on win32. Fixes for other platforms (e.g. GTK0 can be introduced also by an interested party.

| | |
|---|---|
|  |  |
| **Drop-downs the Right Way 1**<br>The button appears depressed. Clicking it a second time will hide the menu. | **Drop-downs the Right Way 2**<br>The menu appears above the button |

Even though it is part of a separate project, I have included a small jface patch, which updates jface to use the new drop-down functionality and allows Eclipse to take advantage of the SWT fixes as can be seen here.




The patch would not have made much sense if there was no simple way for Eclipse to take advantage of it and the jface patch serves as proof of concept. Only 5 lines require change for Eclipse to take advantage of the new drop-down functionality.

# End-User Specs

1) A ToolItem created with the SWT.DROPDOWN style can have a drop-down menu attached to it via setDropDownMenu(). If the ToolItem does not have the SWT.DROPDOWN style, calls to setDropDownMenu() are silently ignored and getDropDownMenu() always returns null.
2) When the arrow of a ToolItem with the SWT.DROPDOWN style is pressed, the following process takes place:
   a) An ARROW Selection event is sent to all listeners.
      Previously, this is where clients would install a listener to display a button's drop-down menu.
      Here, listener can now also control the new drop-down behavior. setDropDownMenu() can be used to set the drop-down menu to be displayed or to prevent a drop-down menu from being displayed by passing a null argument. Display of the drop-down menu can be also prevented by setting the event object's doit flag to false.
   b) After all listeners have been notified, SWT will check if the ToolItem has a drop-down menu and the event's doit flag is true and if both conditions are met, the drop-down menu will be displayed at the appropriate location (usually under the button).

# Important Design Decisions

Here is a list of the most important design decisions, which might require careful review:

- Arrow SWT.Selection events are now delivered synchronously by using sendEvent instead of postEvent. This is required in order to implement section 2) in the specs

above. SWT.Selection events for pressing the button (as opposed to its arrow) are still delivered via postEvent.
Point 2) of the spec, is in turn required in order to make the new functionality feasible to use in jface and be able to fix the Wrong Way #1. Fixing Wrong Way #1 requires that the menu be displayed (via TrackPopupMenu) before returning from the window procedure (this has another design implication, see below). Simple jface integration requires that the menu is displayed after jface has had a chance to dynamically generate it in its Selection event listener.
I am not sure what the reason is behind using postEvent instead of sendEvent for notifying ToolItem's Selection event listeners. I did not run into any problems with sendEvent during my testing.
sendEvent is used on all platforms for consistency, even though it was primarily required for the win32 glitch fix. All platforms, except win32, use the same code for displaying drop-down, which is essentially a near-verbatim copy of the Snippet67 code. Win32 uses special native code, which in turn is a near-verbatim copy of a Windows SDK example.

- Because of the need to call TrackPopupMenu from within the window procedure (as mentioned above), I circumvent the standard SWT mechanism of deferring the display of popup menus until the next call to readAndDispatch() on win32. I am not sure what the reason behind this design decision is, but I did not run into any problems during my testing.
- Windows has a native way of correcting the Wrong Way #2 (popup menu over button glitch). It requires the use of TrackPopupMenuEx, which I had to import to OS.java, along with the TPMPARAMS struct it uses. I am not sure if this is available on WinCE – my patch might break the CE build.

# Important Considerations

- CoolBar and CoolItem, which also use the SWT.DROP_DOWN style in an equivalent way would have to be updated as well.
- The possibility that the switch from postEvent to sendEvent breaks existing code is probably quite real. One way that I can think of addressing this issue is introducing a new flag, say SWT.DROP_DOWN_MENU and activating the new behavior only when this flag is set, in addition to the DROPDOWN flag. If this flag is not set then the dropdownmenu property of ToolItem can be ignored and Arrow Selection events transmitted via postEvent to allow legacy code to function. This will also allow regular ToolItems that do not have the DROP_DOWN (arrow) style to have drop-down menus attached to them as well through that new style.
- I ran a simple test on Windows XP using the win32 and wpf builds and on Ubuntu 7 using the motif and gtk builds. The test involved running Snippet67 and Eclipse and selecting an item from a button's drop-down menu.
- I could not get the WinCE SDK installed and so was not able to build the dlls for wince and therefore do not know if my changes are compatible with wince or not.