

```
//
// FILENAME: Inspector.m
// SUMMARY: Implementation of a NeXTSTEP-style inspector using <Inspectable>
// SUPERCLASS: Object:Inspector
// PROTOCOLS: <Inspectable>
// INTERFACE: Inspector.nib
// AUTHOR: Rohit Khare
// COPYRIGHT: ©1993,94 California Institute of Technology, eText Project
//
// Implementation Comments
// Still uses the neat self-inspection trick for "No Selection"
// This panel uses Andrew Stone's WidePopupMenu to mediate.
//
// History
// 10/30/93: Added refresh: (see setDocPath in eTDocInfo)
// 10/30/93: Removed invalidate: (Obsolete, not called)
// 10/30/94: Modified to implement <InspectorTarget> protocol.
// 09/27/94: Revamped for eText5; added WidePopupMenu.
// 02/04/94: Modified invalidate to "tunnel" through to display etDocInfoUI
// 01/12/94: Reworked header file dependency
// 08/22/93: Added invalidate:
// 08/18/93: Created. Derived almost wholly from Version 1.0
// Version 1.0 tracking notes:
// 06/29/93: Added conformance for touch (currentDoc)
// 06/27/93: Created (Rohit Khare)
//
// Imported Interfaces
//
//import "Inspector.h"
//import "Kludges.subproj/WidePopupMenu.h"
//
//@implementation Inspector
//
// Class Management
//
// - init {
//     [super init];
//     current = nil;
//     currentTarget = nil;
//     currentView = nil;
//     currentType = NULL;
//     typesForInspector = NULL;
//     [NXApp loadNibSection:"Inspector.nib" owner:self withNames:NO];
//     //if ([[NXApp userModel] boolQuery:"InspectorFloat"])
//     [panel setFloatingPanel:YES];
//     [panel setExcludedFromWindowsMenu:YES];
//     [panel setFrameAutosaveName:"etInspectorPanel"];
// }
```

```
[panel setFrameUsingName:@"etInspectorPanel"];  
[self inspect:self];  
return self;  
}  
  
- free {  
    panel = [panel free];  
    invalidPanel = [invalidPanel free];  
    infoPanel = [infoPanel free];  
    return self = [super free];  
}  
  
- awakeFromNib {  
    cover = popup;  
    if (![popup isKindOfClass:[PopUpList class]]) popup = [popup target];  
    [[popup setTarget:self] setAction:@selector(swap:)];  
    [[WidePopupManager alloc] initWithButton:cover andList:popup];  
    invalidView = [invalidPanel contentView];  
    infoView = [infoPanel contentView];  
    return self;  
}  
  
//DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD  
// User Interface  
//  
- showInspector {  
    [panel makeKeyAndOrderFront:self];  
    return self;  
}  
  
- swap:sender {  
    if(strcmp([popup selectedItem], currentType)) {  
        [panel disableDisplay];           // Disable display  
        [current resignInspector:currentView ofType:currentType];  
        [currentView removeFromSuperview];  
        // Swap in the view for the selected cell of the popup  
        currentType = (char *) [popup selectedItem];  
        currentView = [current inspectorForType: currentType];  
        [currentView sizeTo:INSPECTOR_W :INSPECTOR_H];  
        [[panel contentView] addSubview:currentView];  
        [current activateInspector:currentView ofType: currentType];  
        [panel reenableView];             // Reenable display  
        [panel display];  
    }  
    if(strcmp([popup selectedItem], [cover title])) {  
        [cover setTitle:[popup selectedItem]];  
    }  
}
```

[illegible]

[illegible]

```

//  Introspection: <Inspectable> Obligations
//
#define NOSEL  "No Selection"
#define INFO   "Info"

- (const NXAtom *) types {
    if (!typesForInspector) {
        typesForInspector = malloc(3*sizeof(char *));
        typesForInspector[0] = (char *) NXUniqueString(NOSEL);
        typesForInspector[1] = (char *) NXUniqueString(INFO);
        typesForInspector[2] = NULL;
    }
    return typesForInspector;
}

- inspectorForType:(const char *) type {
    if(!strcmp(type,NXUniqueString(NOSEL)))
        return invalidView;
    else if (!strcmp(type,NXUniqueString(INFO)))
        return infoView;
    NXLogError("Massive Inspector Failure: Asked Inspector for: %s", type);
    return invalidView;
}

- resignInspector: (View *) oldInspector ofType: (const char *) type {
    return self;}

- activateInspector: (View *) newInspector ofType: (const char *) type {
    return self;}

- (const char *) inspectorTitle {return NXUniqueString("eText");}

@end

```