Inside SWT

Steve Northover

SWT Team Lead
IBM Canada

eclipse.org/swt

Goal of this talk (~~to sell you a book~~)

Understand what SWT is and where it came from

Learn about the API, design decisions and implementation strategies used in the toolkit

The Agenda

What is SWT?
History
Hello World
Implementation
Philosophies
Questions

+

... Rants & Raves
… Horrible Hacks
… Design Decisions
… Widgets Exposed!
… Answers?

What is SWT?

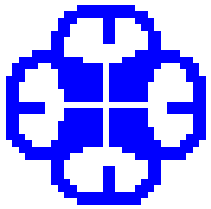
- SWT = “Standard Widget Toolkit”
- Standard UI component for the Eclipse IDE
- Used to build Eclipse “plug-ins” or stand alone applications
- Roughly equivalent to AWT/Swing

Widgets are native

History of SWT

Object Technology International (OTI)

- Virtual Machines, Class Libraries
  - Smalltalk, Java, ENVY/Image
- Version Management Systems
  - ENVY/Manager, VisualAge for Java
- Embedded Systems



Acquired by IBM in 1996

History of SWT

Smalltalk

- ParcPlace (Cincom?) VisualWorks
  - Emulated Widgets (100% Smalltalk + VM)

- Digitalk Smalltalk/V
  - Native Widgets (Smalltalk + OS + VM)
- IBM/Smalltalk ()
  - CommonWidgets (native, Smalltalk + OS)
  - Windows, OS/2, Motif, Macintosh, OpenLook

Native vs. Emulated!

History of SWT

Then … Smalltalk cratered!

History of SWT

IBM VisualAge Family

- VisualAge for Java
  - Written in IBM/Smalltalk
  - Award Winning Java IDE
- Next generation Java IDE (“Leapfrog”)
  - Prototype in Java using AWT/Swing
  - Appearance and performance problems, at the time
  - “Leapfrog” became … “The Toad”

OTI to build award losing IDE?

History of SWT

What now?

Would AWT/Swing mature fast enough before “Leapfrog” (VisualAge Micro Edition) went to market?

Would Java UI development be subject to the same negatives that Smalltalk had faced?

M-m-must … ship … code

History of SWT

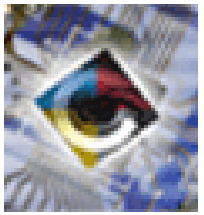
Port the widgets to Java

- Keep good design decisions
  - Focus on efficiency, run on any VM …
- Throw out bad design decisions
  - Ditch problem API, lose extra layers …
- Embrace Java
  - Strong typing, method visibility …
  - Naming conventions, overloaded methods …
  - Follow “good Java practices” …

History of SWT

VisualAge Micro Edition Ships

- Wins awards but doesn't replace VisualAge for Java
  - Used for embedded systems
  - Proprietary repository (no CVS support)
  - Missing features, areas for improvement …



- Rewrite it, improve it, rename it to … Eclipse!

Enter Eclipse and Open Source

What is SWT all about?

"The SWT component is designed to provide efficient, portable access to the user-interface facilities of the operating system on which it is implemented."

eclipse.org/swt

Hello World


```
import org.eclipse.swt.widgets.*;

public class HelloWorld {
   public static void main(String[] args) {
      Display display = new Display();
      Shell shell = new Shell(display);
      shell.setText("Hello, World");
      shell.setSize(200, 100);
      shell.open();
      while (!shell.isDisposed()) {
            if (!display.readAndDispatch())
                  display.sleep();
      }
      display.dispose();
   }
}
```

Hello World

- Display
  - Connection to the window system
  - Represents the “screen”
  - Normally a singleton
  - Contains a list of Shells

```
Display display = new Display();
```

Hello World

- Shell
  - Represents a “window” on the “screen”
  - Root of a tree of Composites and Controls

```
Shell shell = new Shell(display);
shell.setText("Hello, World");
shell.setSize(200, 100);
shell.open();
```

Hello World

- Composite
  - container that holds other Composites and Controls
- Control
  - “heavyweight” operating system control (Buttons, Lists, Labels, Tables, Trees …)
  - Shells and Composites are also Controls

Hello World

- The Event Loop
  - Explicitly coded by the programmer
  - Repeatedly reads and dispatches events
  - Yields CPU when no events are available

```
while (!shell.isDisposed()) {
   if (!display.readAndDispatch())
      display.sleep();
}
```

Hello World

- Operating system resources are explicitly released by the programmer

```
display.dispose();
```

Rule #1: *"If you created it, you dispose it."*

Rule #2: *"Disposing a parent disposes the children."*

Standard Constructors

- Widgets must be created with a parent
- Widgets always have style bits

```
Shell dialog = new Shell(shell, SWT.DIALOG_TRIM);

Button button = new Button(shell, SWT.PUSH);

Text text = new Text(group, SWT.SINGLE|SWT.BORDER);
```

Threading Model

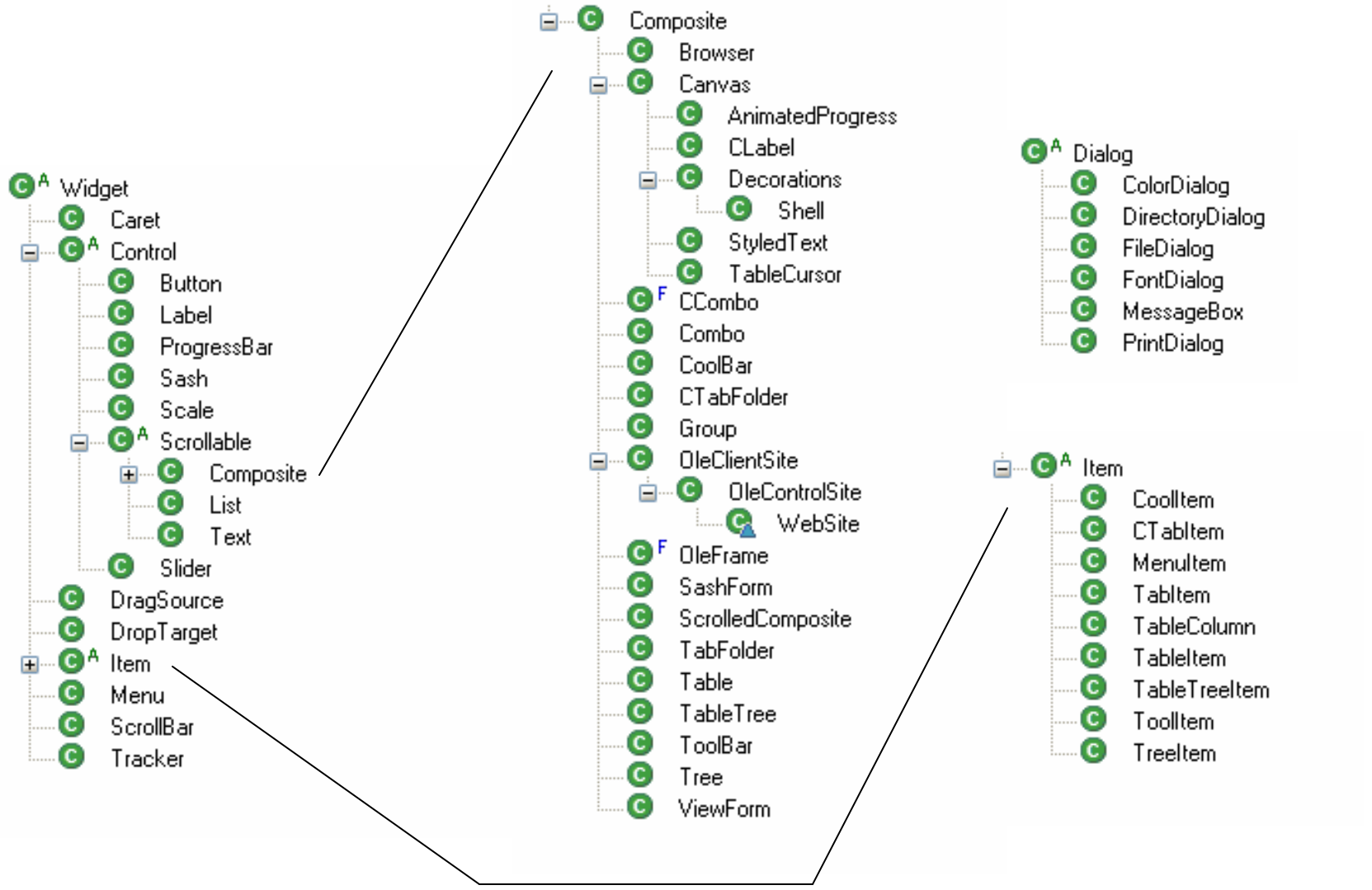
- UI-thread runs the event loop
- Widgets must be called from the UI-thread
  - Otherwise use syncExec(), asyncExec(), wake()
- Graphics can be called from any thread

Threading is enforced

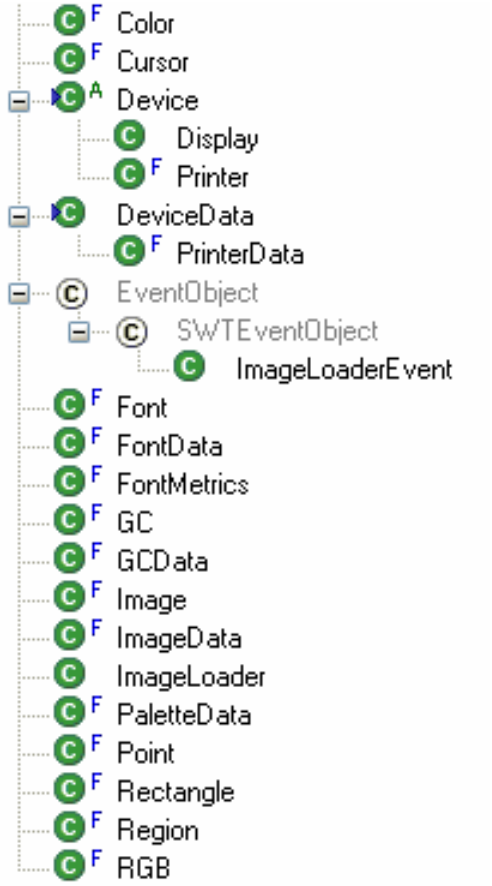
Events

- JavaBeans event model (“typed” events)
- Low level “untyped” events are available
  - Don’t need to use them or understand them
- Events come from the event loop (and from widget API)

Every Widget API Class (including D&D, OLE, Browser and Custom Controls)



Every Graphics API Class (including Printing)



That’s it! Ready to start hacking?

What classes did you show me?

- org.eclipse.swt.graphics
- org.eclipse.swt.widgets
- org.eclipse.swt.dnd
  - Drag & Drop and Clipboard
- org.eclipse.swt.browser
  - HTML Browser control
- org.eclipse.swt.custom
  - Custom controls for Eclipse
- org.eclipse.swt.ole.win32

What classes didn't you show me?

- org.eclipse.swt
  - Constants and exceptions (3 classes)
- org.eclipse.swt.events
  - JavaBeans boilerplate ☹ (tons of classes)
- org.eclipse.swt.layout
  - 4 standard layouts (fill, row, grid, form)
- org.eclipse.swt.program
  - Launch a program, find an icon (one class)

What classes didn't you show me?

- org.eclipse.swt.accessibility
  - 2 classes (plus more JavaBeans boilerplate)
- org.eclipse.swt.awt
  - Bridge between SWT and AWT/Swing (one class)
- org.eclipse.swt.internal.*
  - The implementation of SWT (don't touch)

How is SWT implemented?

Implementation of SWT

- SWT is written in Java (“100%” + DLL’s)
  - Java is a systems programming language
  - Java is fast, stable, garbage collected …
- SWT API is portable, implementation is not
  - One implementation per platform (jars + DLL’s)
  - Most of the code is platform specific

How is this organized?

Implementation of SWT

Example: Text.java

One possible implementation of Text.selectAll()

```
public void selectAll() {
   checkWidget();
   setSelection(0, getText().length());
}

static final native String getText();
```

What is wrong with this code?

Implementation of SWT

Text.java – Windows and Macintosh

… SWT/win32/org/eclipse/swt/widgets/Text.java

```
public void selectAll() {
   checkWidget();
   OS.SendMessage(handle, OS.EM_SETSEL, 0, -1);
}
```

… SWT/carbon/org/eclipse/swt/widgets/Text.java

```
public void selectAll() {
   checkWidget();
   OS.TXNSelectAll(txnObject);
}
```

Implementation of SWT

Text.java – GTK

… SWT/gtk/org/eclipse/swt/widgets/Text.java

```
public void selectAll() {
    checkWidget();
    if ((style & SWT.SINGLE) != 0) {
       OS.gtk_editable_select_region(handle, 0, -1);
    } else {
       byte[] start = new byte[ITER_SIZEOF];
       byte[] end = new byte[ITER_SIZEOF];
       OS.gtk_text_buffer_get_iter_at_offset(h, start, 0);
       OS.gtk_text_buffer_get_end_iter(h, end);
       int iMark = OS.gtk_text_buffer_get_insert(h);
       int sMark = OS.gtk_text_buffer_get_selection_bound(h);
       OS.gtk_text_buffer_move_mark(h, sMark, start);
       OS.gtk_text_buffer_move_mark(h, iMark, end);
    }
}}
```

Implementation of SWT

Text.java – Motif

… SWT/motif/org/eclipse/swt/widgets/Text.java

```
public void selectAll () {
    checkWidget();
    int position = OS.XmTextGetLastPosition(handle);
    int xDisplay = OS.XtDisplay(handle);
    if (xDisplay == 0) return;
    boolean warnings = display.getWarnings();
    display.setWarnings(false);
    int timeStamp = OS.XtLastTimestampProcessed(xDisplay));
    OS.XmTextSetSelection(handle, 0, position, timeStamp);
    OS.XmTextSetInsertionPosition(handle, 0);
    display.setWarnings(warnings);
}
```

Implementation of SWT

Text.java – Photon

… SWT/photon/org/eclipse/swt/widgets/Text.java

```
public void selectAll () {
   checkWidget();
   OS.PtTextSetSelection (handle, new int [] {0},
      new int [] {0x7FFFFFFF});
}
```

It looks just like C!

Advantages of this Approach

- Java code behaves like the equivalent C
  - One-to-one mapping of operating system API
  - Shared Libraries have no interesting code
- Uses the terminology of the platform
  - Leverages operating system programmers
  - Supports the operating system vendors
- Compiles easily to native code (GCJ, JET)
- Easier to debug, maintain and take apart

Taking SWT apart

- Compiles and runs on JDK 1.2
- J2ME compatible, compiles against CLDC 1.0
- Unused natives can be excluded
- Unreferenced packages can be removed or stubbed
  - org.eclipse.swt.custom, org.eclipse.swt.dnd, org.eclipse.swt.layout, org.eclipse.swt.accessibility
  - org.eclipse.swt.widgets (for super hackers only)

Ant scripts are available

Fundamental Philosophies

- Embrace the Operating System
- There is no good or bad, just the same and different
- Less is More
- Everything Means Something

Embrace the Operating System

- No free threads
- No free event loops
- No free resource management
- Always use the operating system
  - Don’t even roll your own Label
- Live with the limitations
  - Your program is just as good as any other application on the desktop

Embrace the Operating System

- Design API's bottom up
  - Don't force your vision, be pragmatic
- Use the thinnest possible interface
  - Minimize the distance from public API to the operating system calls that do the work
- Be a good operating system programmer
  - Use operating system naming and conventions
  - Be a systems programmer first and then a Java programmer

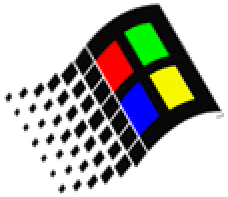
The “Operating System Hall of Shame”

1. Be brave
2. Try to laugh
3. Remain calm

No matter what I say, I love each and every one like a child!

The “Operating System Hall of Shame”

- Windows
  - OLE (ActiveX) has megalomania
    - It’s just DLL’s (and “objects”, file system, storage …)
    - Horribly verbose (QueryInterface(), check, cast ‘n call)
  - Bizarre limitations, strange design decisions
    - Scroll Bars and Menus are not Controls
    - Modal menu and resize loops
  - WM_KEYDOWN, WM_SYSKEYDOWN … barf!
  - GDI fumbling, SelectObject(), compatible HDC’s?
  - STGC_DANGEROUSLYCOMMITMERELYTODISKCACHE



Built by “Hungarians”?

“Hungarian” Notation in Windows

```
Variant[] rgvarg2 = new Variant[] {rgvarg};
int[] rgdispidNamedArgs = new int[]
{COM.DISPID_PROPERTYPUT};
int dwFlags = COM.DISPATCH_PROPERTYPUT;
if ((rgvarg.getType() & COM.VT_BYREF) ==
COM.VT_BYREF)
dwFlags = COM.DISPATCH_PROPERTYPUTREF;
Variant pVarResult = new Variant();
int result = invoke(dispIdMember, dwFlags,
rgvarg2, rgdispidNamedArgs, pVarResult);
```

But I’m not “Hungarian”! (Oh, yes you are!)

How does Microsoft create new controls?

BOOL InitCommonControlsEx()

BOOL Init Common Control Sex (!)

Daddy, where do widgets come from?

The “Operating System Hall of Shame”

- The Macintosh

  - “No way to get … always have the wrong ... if I could just”
  - Objective C? Carbon, Cocoa … (2 complete toolkits)
  - Released in 1984, now supports clipping (20 years) NEW
  - API’s on API’s on API’s … IconFamily … 20 years worth!
  - “Too high level” and “Too low level” at the same time
  - “bounds” are not the bounds, events don’t go to widgets …
  - TXNObject instead of new Text control (TXNLongRect)
  - SetAutomaticControlDragTrackingEnabledForWindow

Using Set Automatic Control Drag Tracking Enabled For Window

- Toggling “Automatic Control Drag Tracking” for a Window:

```
if (isAutomaticControlDragTrackingEnabledForWindow(window)) {
        SetAutomaticControlDragTrackingEnabledForWindow(window,
          false);
} else {
        SetAutomaticControlDragTrackingEnabledForWindow(window,
          true);
}
```

- The optimized version of the code:

```
SetAutomaticControlDragTrackingEnabledForWindow(window,
    !isAutomaticControlDragEnabledForWindow (window));
```

Using Set Automatic Control Drag Tracking Enabled For Window

- Toggling “Automatic Control Drag Tracking” for a Window:

```
if (isAutomaticControlDragTrackingEnabledForWindow(window))
    {
      SetAutomaticControlDragTrackingEnabledForWindow(wind
        ow, false);
} else {
      SetAutomaticControlDragTrackingEnabledForWindow(wind
        ow, true);
}
```

- Optimized version of the code:

```
SetAutomaticControlDragTrackingEnabledForWindow(window,
    !isAutomaticControlDragEnabledForWindow (window));
```

Using Set Automatic Control Drag Tracking Enabled For Window

- Toggling “Automatic Control Drag Tracking” for a Window:

```
if (isAutomaticControlDragTrackingEnabledForWindow(window)) {
        SetAutomaticControlDragTrackingEnabledForWindow(window, false);
} else {
        SetAutomaticControlDragTrackingEnabledForWindow(window, true);
}
```

- Optimized version of the code:

```
SetAutomaticControlDragTrackingEnabledForWindow(window,
    !isAutomaticControlDragEnabledForWindow (window));
```

Using Set Automatic Control Drag Tracking Enabled For Window

- Toggling “Automatic Control Drag Tracking” for a Window:

```
if (isAutomaticControlDragTrackingEnabledForWindow(window)) {
        SetAutomaticControlDragTrackingEnabledForWindow(window,
            false);
} else {
        SetAutomaticControlDragTrackingEnabledForWindow(window,
            true);
}
```

- Optimized version of the code:

```
SetAutomaticControlDragTrackingEnabledForWindow(window,
    !isAutomaticControlDragEnabledForWindow (window));
```

Using Set Automatic Control Drag Tracking Enabled For Window

- Toggling “Automatic Control Drag Tracking” state for a Window:

```
if
    (isAutomaticControlDragTrackingEnabledForWindow(window
    )) {
      SetAutomaticControlDragTrackingEnabledForWindow(wind
        ow, false);
} else {
      SetAutomaticControlDragTrackingEnabledForWindow(wind
        ow, true);
}
```

- Optimized version of the code:

```
SetAutomaticControlDragTrackingEnabledForWindow(window,
    !isAutomaticControlDragEnabledForWindow (window));
```

The “Operating System Hall of Shame”

- GTK

  - “It’s not X, it’s GDK … oh wait … it’s the same!”
  - Too many handles (“Dances with Handles”)
  - Widgets won’t stay where you put them
  - Event loops embedded in API (NEVER DO THIS)
  - Everything is deferred, no way to force work to happen, other than … running an event loop
  - Too many versions (different bugs)

“Come home Windows, all is forgiven!”

The “Operating System Hall of Shame”

- X/Motif
  - Ugly! (Ancient Roman look, frames and chiselling)
  - X Windows (“I am not remote, why are you pretending I am? I’m right here at my desk.”)
  - Parents don’t want focus (GP’s, hangs …)
  - FontStruct, FontSet, FontList … yetch!
  - XInitThreads() can’t be used (GP’s, hangs …)
  - XmNenableThinThickness

Will this thing ever die?

Seriously …

- Operating systems are large and complex
  - They provide an amazing amount of functionality
  - They need to differentiate themselves (“It’s a Mac!”)
  - They don’t expect portable widget toolkits to be implemented on top of them
  - They don’t expect to be programmed from Java

They are all special

Fundamental Philosophies

Embrace the Operating System

There is no good or bad, just the same and different

Less is More

Everything Means Something

But my code is better, it uses the “zlot” pattern!

- Consistency is more important than “correctness”
- Aren’t some ways of doing things better than others?
  - Yes, but “better” is a matter of opinion
- Different code:
  - Takes longer to understand
  - Harder to maintain and debug
  - Gets forgotten, unloved, hated … then rewritten

Consistency ensures correctness

"*A* benefit of consistency is that when a bug or performance improvement is identified in a code pattern, the fix can be applied everywhere. So consistency ensures correctness, which is more important to us than elegance."

- from "The SWT Manifesto", 2003

Fundamental Philosophies

Embrace the Operating System

There is no good or bad, just the same and different

Less is More

Everything Means Something

Less is More

- Minimize the number of …
  - Classes, methods, fields, constants, API names
- Processor cycles are for the application
  - Do you care how SWT is implemented?
- Always take space/speed over some cost to maintainability/flexibility/portability because there's …

"No future", Rotten, 1976

Fundamental Philosophies

- Embrace the Operating System
- There is not good or bad, just the same and different
- Less is More
- Everything Means Something

Everything Means Something

Code with intention

- A null check implies “The field can be null” (NOTE: This implies that the converse is true)
- A field implies “Set the value, no work is done”
- Lazy initialization means “The computation is expensive and the result may not be needed.”
- Workarounds are commented, for example “Bug in Windows …”

… and many more

Everything Means Something

There is no “harmless” code

- How is code that “does nothing” harmful?
  - Increases the size of the library
  - Increases the complexity of the library
  - Slows down program execution
  - Breaks when code around it changes

Every line is sacred,
Every line is great …

Successful Strategies

- Encourage operating system programming
  - We are platform hackers that happen to use Java
- Encourage a conservative and minimal implementation
- Maintain strict control of API and features
  - Prototype because documentation lies
  - API is forever, @**deprecated** is a lie

Successful Strategies

- Encourage code ownership
  - Maintaining the code you write is good for you
  - Learn from mistakes (Never use the “zlot” pattern)
  - Learn how to evolve an API
- Browse all code changes
  - Every line of code in SWT is checked by the committers when they catch up from CVS

Summary

Writing and maintaining industrial strength cross platform widget libraries is hard.

Widget systems are inherently complex.

Several basic approaches can be used to reduce the complexity.

Less is More

Q&A