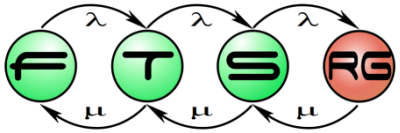
# User Interface Programming in SWT

Eclipse Based Technologies

http://inf.mit.bme.hu/edu/courses/eat



# Developing Graphical User Interfaces

- Java Graphics Toolkits
  - AWT
    - Native widgets
    - Only common subset of platform widgets available!
  - Swing
    - Manually drawn widgets
    - Superset of platform widgets
    - Extensible
  - JavaFX
    - Available with Java 7 (but not added by default!)
    - Completely rethinked UI model

# Problems with User Interfaces

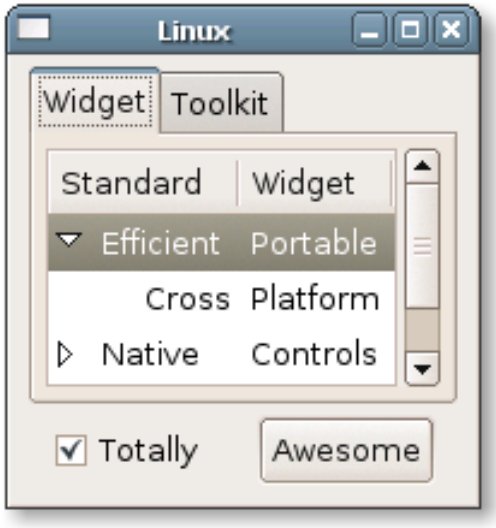
- "Does not look like Word" problem
  - Reusing platform look-and-feel
  - Internationalization settings
  - Java goal: look everywhere the same
- AWT
  - very low level
- Swing
  - memory usage and performance issues at start
  - Since Java 6 more than appropriate
- Java FX
  - Still not widely used

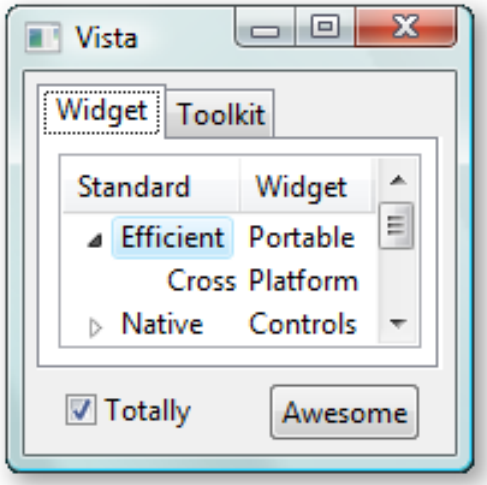
# SWT – Standard Widget Toolkit

- Developed by IBM
  - Swing was not appropriate
  - When starting the Eclipse project
    - Based on Smalltalk native widget accessing experiments
  - Goals
    - Use native widgets everywhere possible
    - Looks like a native application

# SWT – Standard Widget Toolkit

- Reusing platform widgets
  - Fast
  - Platform look-and-feel
    - Every platform service available
      - OLE, drag-n-drop, …
  - Needs porting!
    - Appears differently



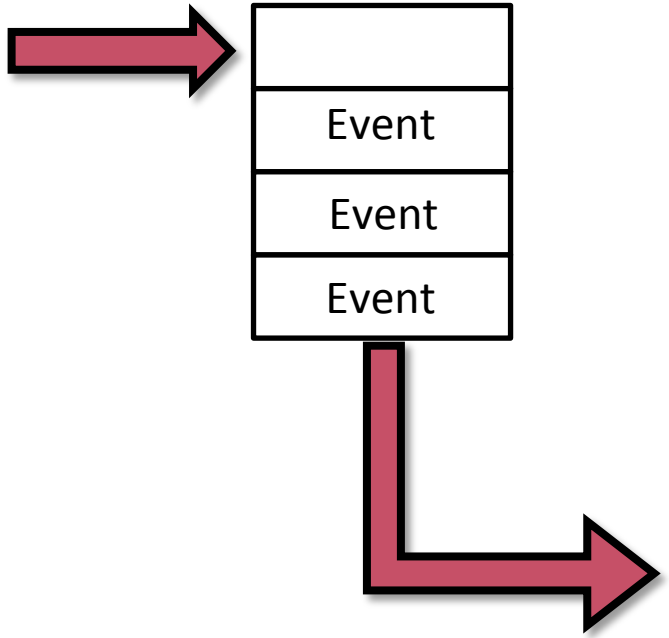


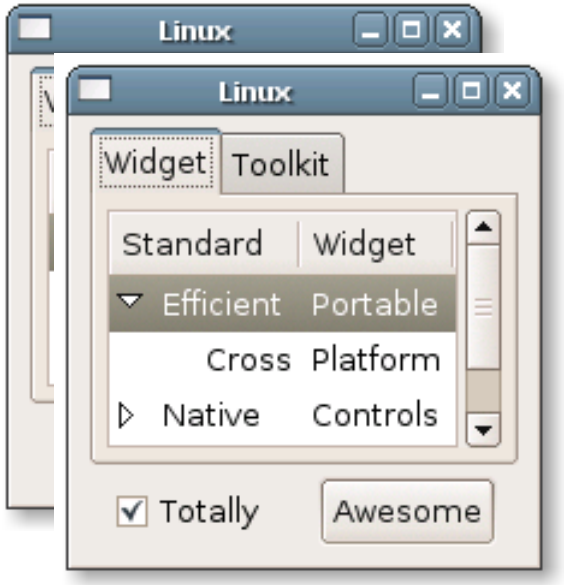
Source of pictures:
http://eclipse.org/swt

# Programming Model

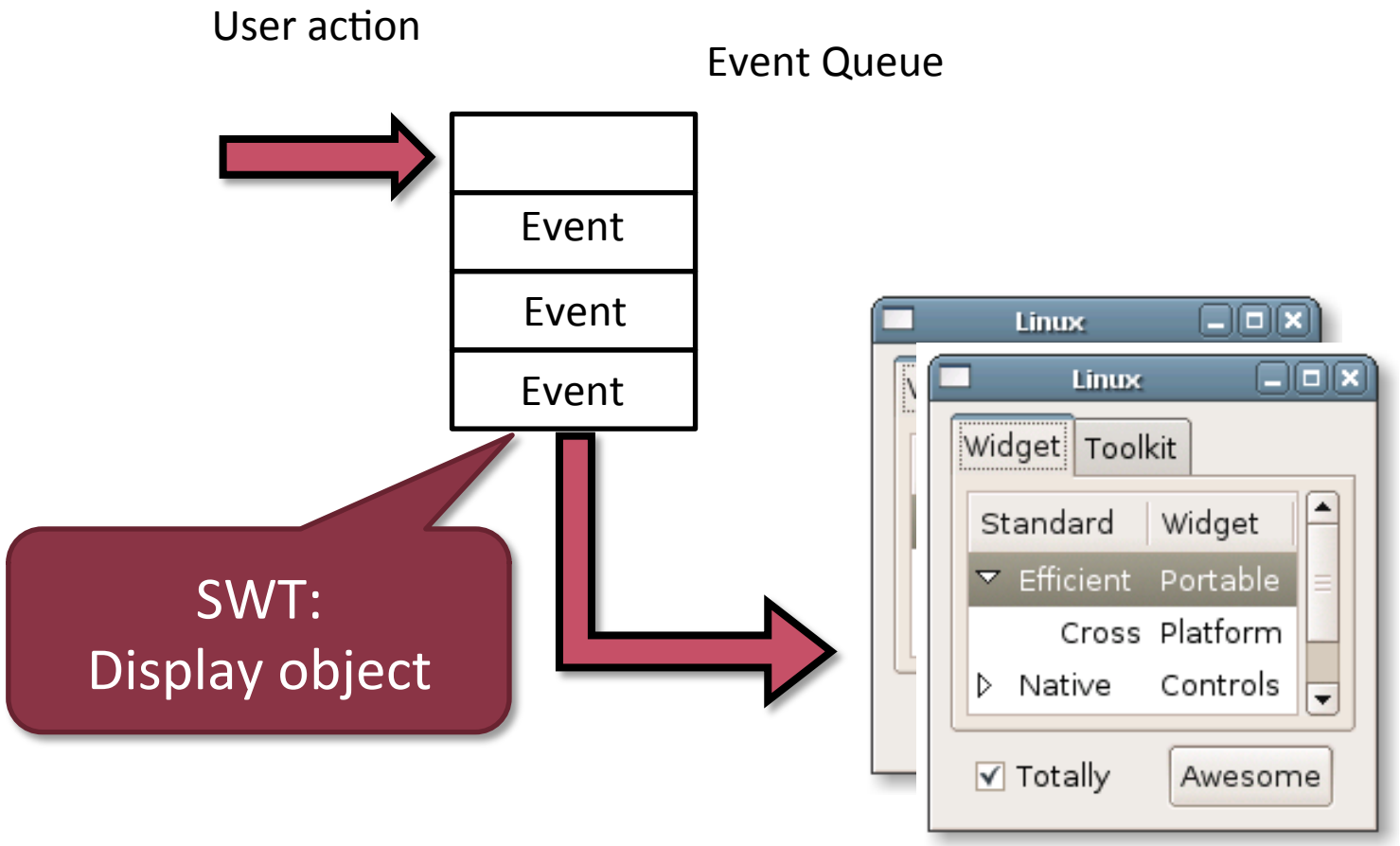
User action

Event Queue

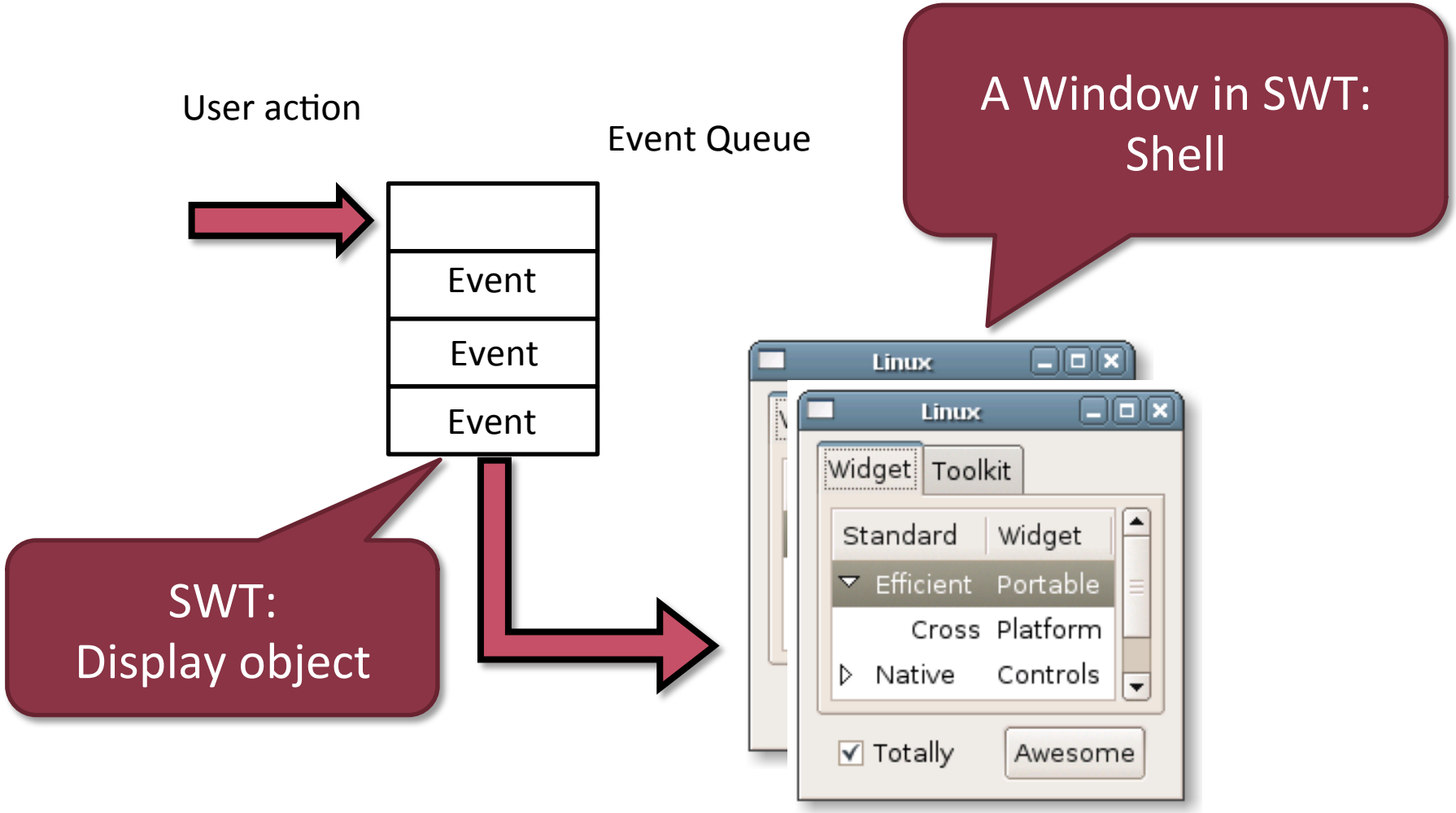


# Programming Model



# Programming Model



# SWT Event Loop

- Explicit event loop
  - The application needs to include it!
    - Collecting incoming events
    - and processing it
  - Loop termination
    - Application terminates
  - Very similar to Win32 API

# SWT Event Loop

```
public static void main(String [] args) {
        Display display = new Display();
        final Shell shell = new Shell(display);
        shell.setSize(400, 400);
        shell.open();

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

# Event Handling

- Event: something the application needs to react
  - User events
    - Mouse move
    - Key presses
    - …
  - System
    - Time passes
    - …

# Event Handling

- Assigning Event Listeners
  - Generic event listeners for every widget
    - Event information available as style bits (see later)
  - Typed listeners when applicable
    - Keyboard
    - Mouse
    - Multitouch
    - …
- Both Listener and Adapters available

# SWT widgets

- Relatively small widget hierarchy
  - E.g. as opposed to Swing
  - A class describes multiple widgets
    - Selection happens via style bits
- Associating model objects possible
  - getData()/setData() methods
  - Very useful for generic UI processing code and data bindings

# Style bits

- Additional information
  - E.g. Button->CheckBox
- Constructor parameter
  - It is not possible to change later
  - The available styles depend on specific widgets
- Implementation
  - Works with Java 1.4 -> no ‘enum’ available
  - Using **int** constants from the SWT class
    - Multiple style bits can be selected via bitwise or:
      - SWT.SEPARATOR|SWT.HORIZONTAL

# SWT widgets

- Manual instantiation
  - No factory
  - Strict containment hierarchy
- Manual cleanup
  - Garbage collection is not enough!
    - Native widgets!
  - Method dispose() needs to be called manually

# Dispose rules

# Dispose rules

1. Base rule:
   - *If you create it, you dispose it!*
2. Reverse rule:
   - *Do only dispose it if you create it!*

3. Exception:
   - Widgets hierarchies are disposed by disposing the top-level element

# Dispose rules

1. Base rule:
   - *If you create it, you dispose it!*
2. Reverse rule:
   - *Do only dispose it if you create it!*

3. Exception:
   - Widgets hierarchies are disposed by disposing the top-level element

# Dispose rules

1. Base rule:
   - *If you create it, you dispose it!*
2. Reverse rule:
   - *Do only dispose it if you create it!*

3. Exception:
   - Widgets hierarchies are disposed by disposing the top-level element

# Common widgets

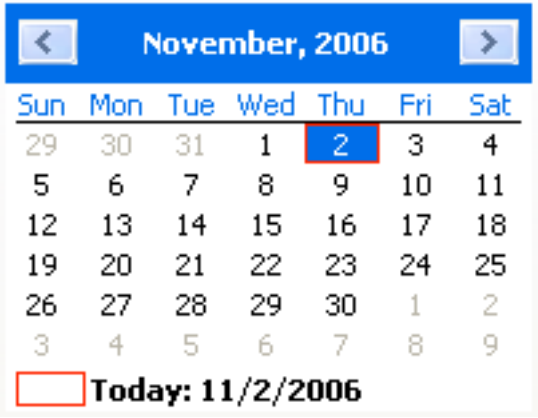
- Button
  - Push, radio, combo buttons
- Label
  - Read-only text display field (w or w/o icons)
- Text
  - Writeable text fields (single line, multiline)
- StyledText
  - Custom drawn multiline text field (e.g. Eclipse editors)

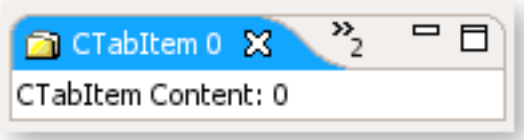
# Common widgets

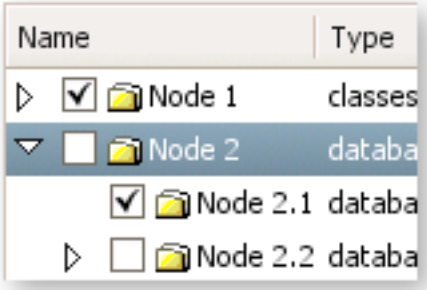
- Composite
  - Stores other widgets
  - Allows setting layouts
- Canvas
  - Manual drawing
- Menu, Toolbar
- List, Tree, Table
  - Specific widgets for displaying large amounts of data
  - Avoids creating a huge number of buttons, etc.

# Common Widgets

- And many more: http://eclipse.org/swt/widgets/

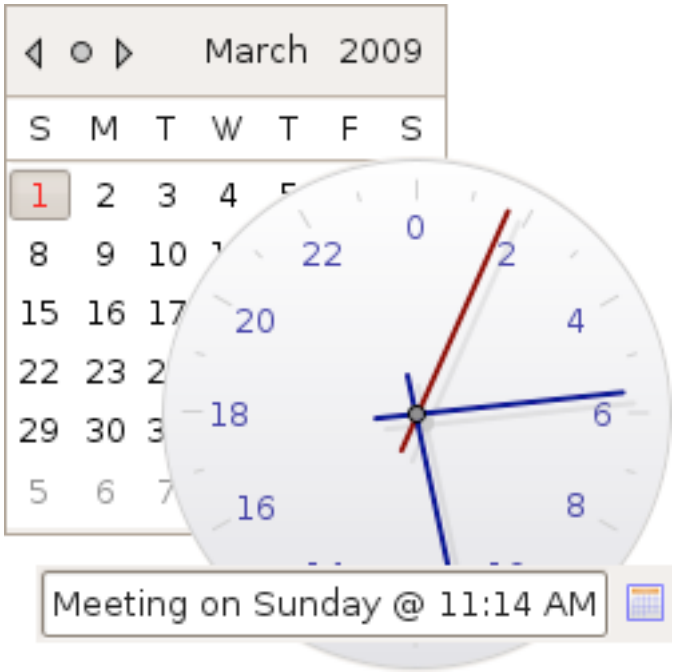



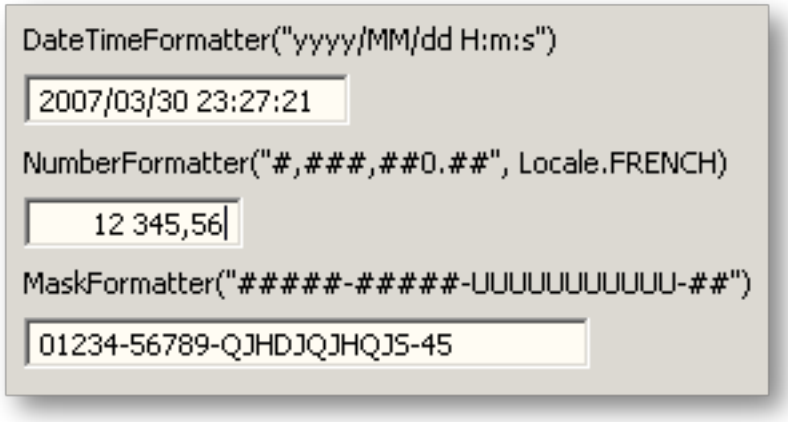



# Additional widget – Nebula project

- http://eclipse.org/nebula/
- Date-time handling

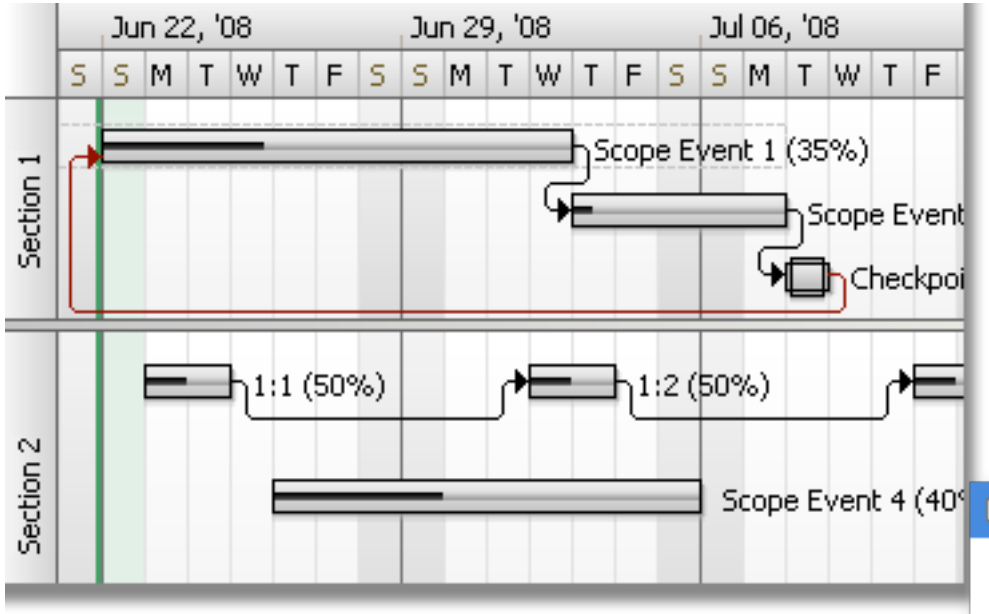



- FormattedText

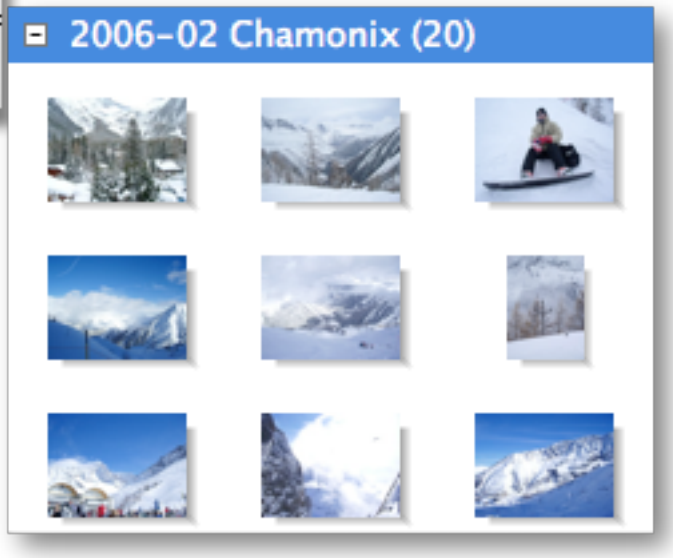


# Additional widgets – Nebula project

- Gantt diagram




- Galery

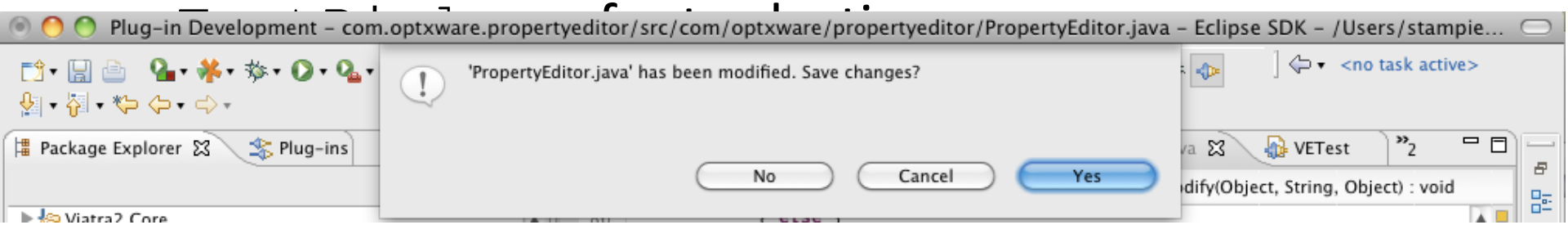


# Dialog windows

- Types
  - `MessageBox`- displaying messages
  - `ColorDialog` – color choosing
  - `DirectoryDialog` – directory structure
  - `FileDialog` – file selection/save
  - `FontDialog` – font selecting
  - `PrintDialog` – printing
  - These are not widgets!
- Reuses operating system dialogs
  - Specific dialog settings available
    - Pl. SWT.SHEET

# Dialog windows

- Types
  - `MessageBox`- displaying messages
  - `ColorDialog` – color choosing
  - `DirectoryDialog` – directory structure
  - `FileDialog` – file selection/save



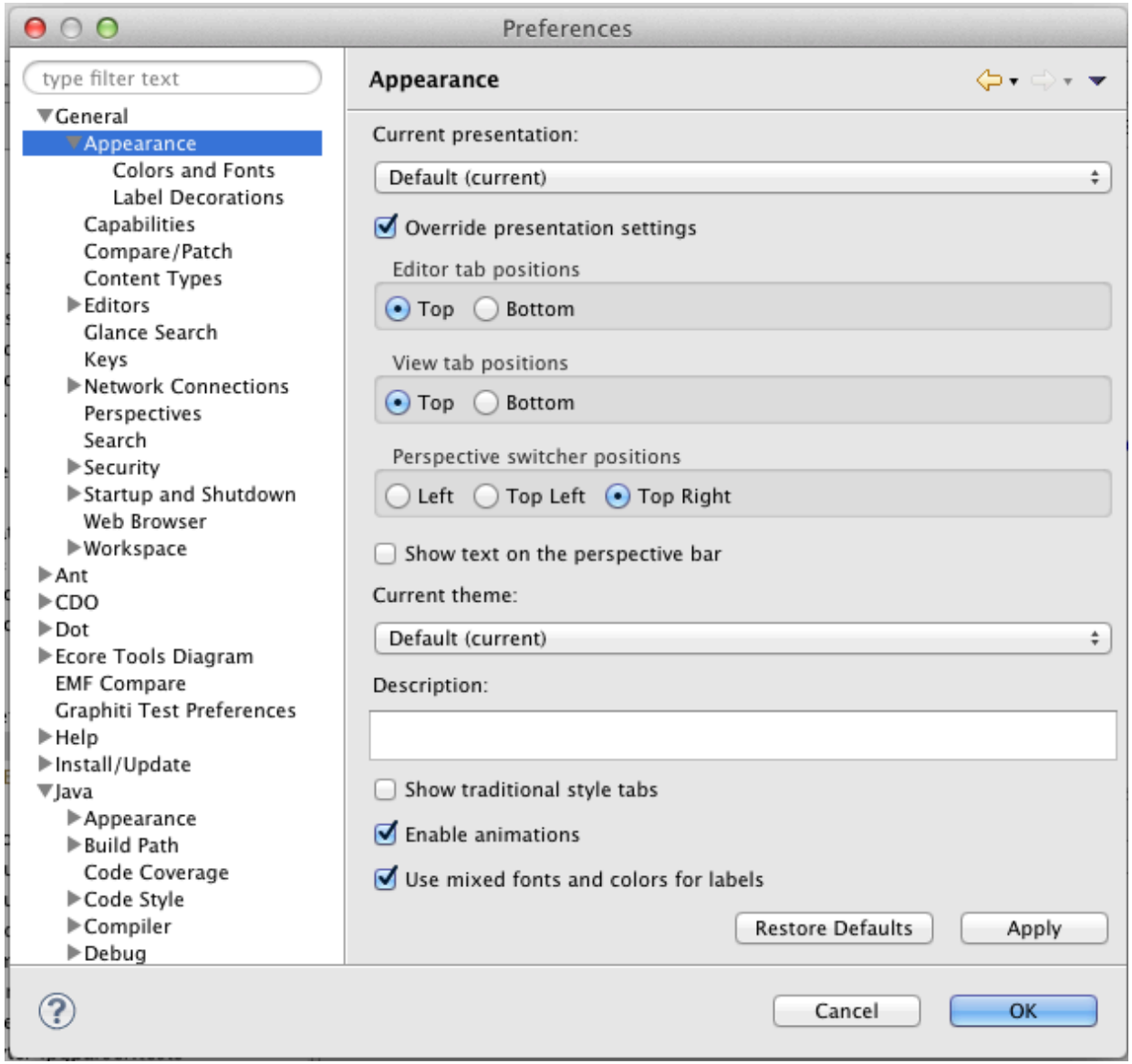

- Reuses operating system dialogs
  - Specific dialog settings available
    - Pl. SWT.SHEET

# Complex Form Design

Layouts

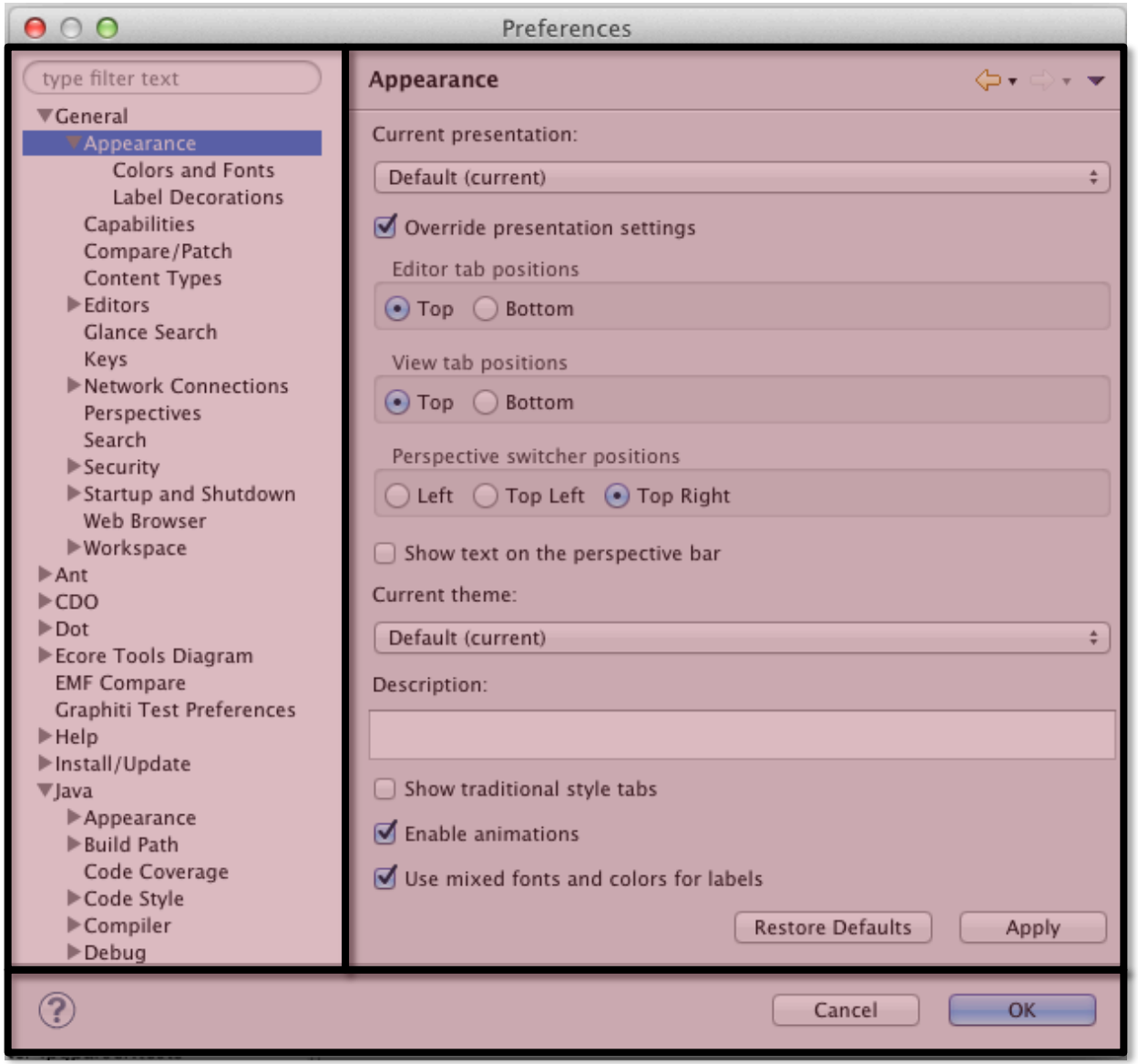
# Complex Forms

How many widgets does the window contain?



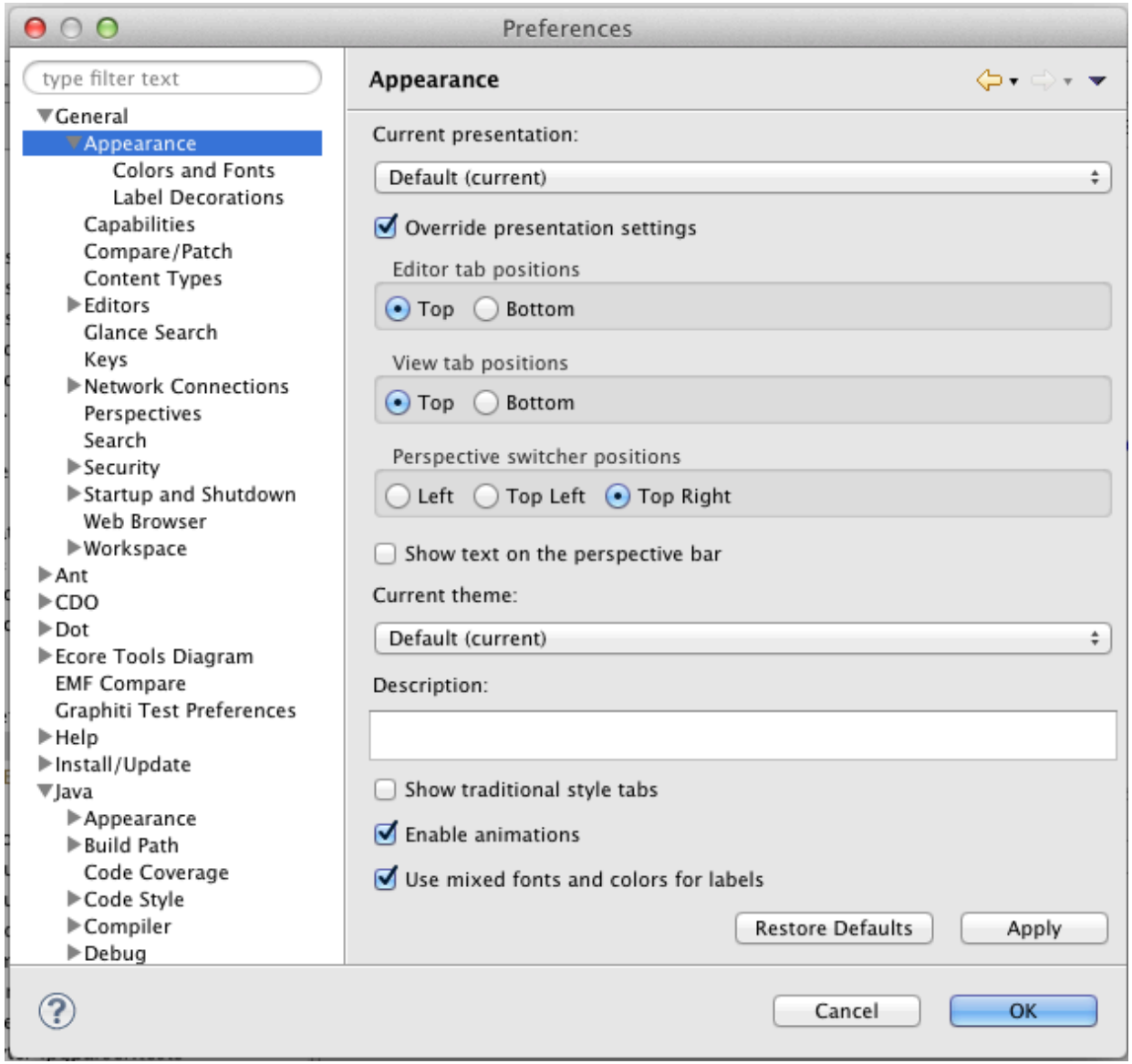
# Complex Forms

How many widgets does the window contain?



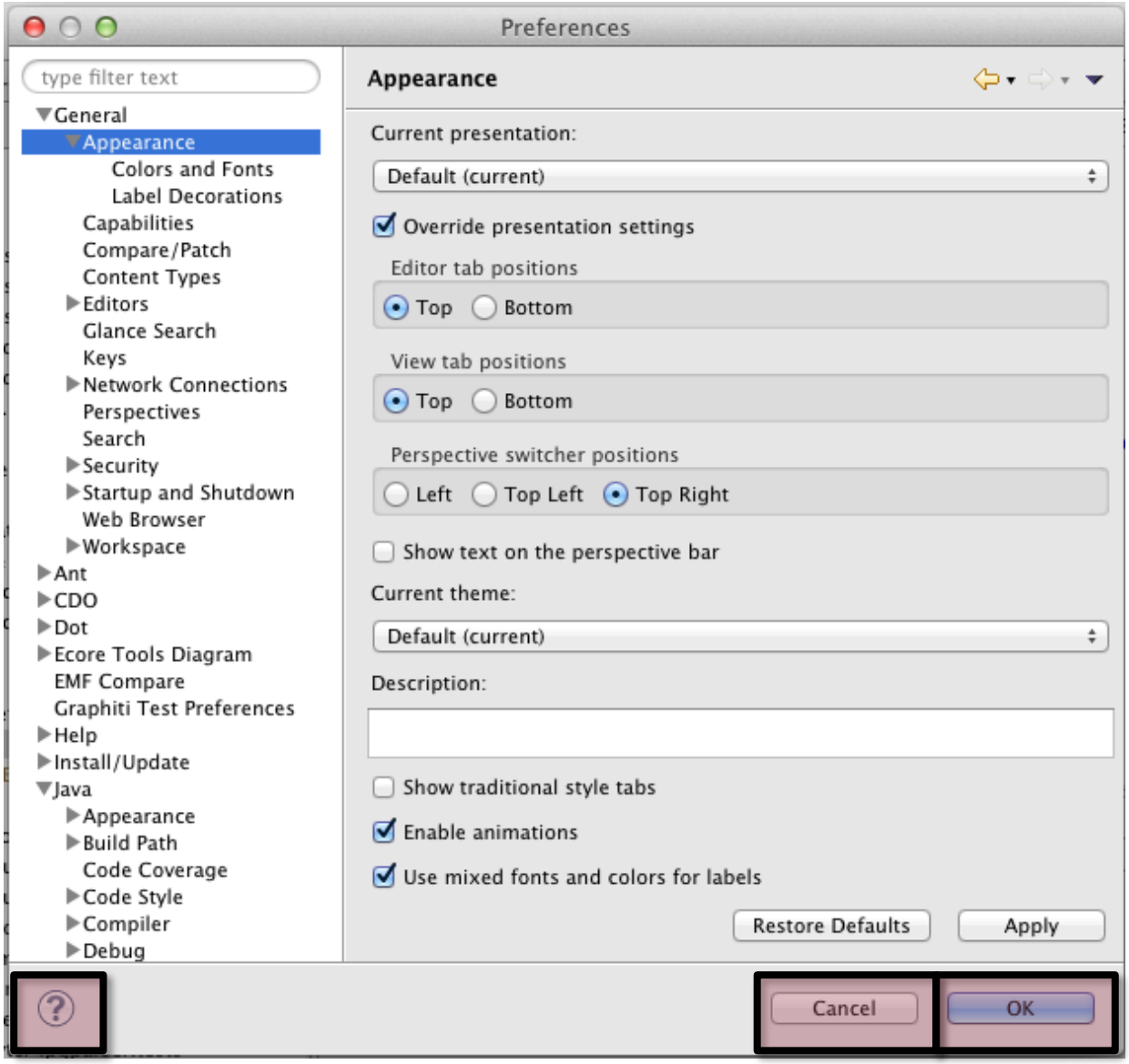
# Complex Forms

How many widgets does the window contain?



# Complex Forms

How many widgets does the window contain?



# Widget Containment Hierarchy

- Strict containment hierarchy
  - Every widget has a single parent
  - Exception: Shell (window)
- Composite widget
  - Contains other widgets
  - Layout can be created

# Layout

- Separation of arrangement and content
  - Decides positioning
  - Relative to container
  - Követi a konténer méretének változását
- Abstract Base class: Layout
  - Do not call it manually

# Layout Hints

- Every widget might give some positioning information
  - Use #setLayoutData
  - Different data for different layouts
    - Inconsistent setting results in runtime error

# Layouts in SWT

- Built-in layouts available
  - FillLayout
  - RowLayout
  - GridLayout
  - FormLayout
  - StackLayout
- Default layout
  - Set up coordinates and size for each widget!

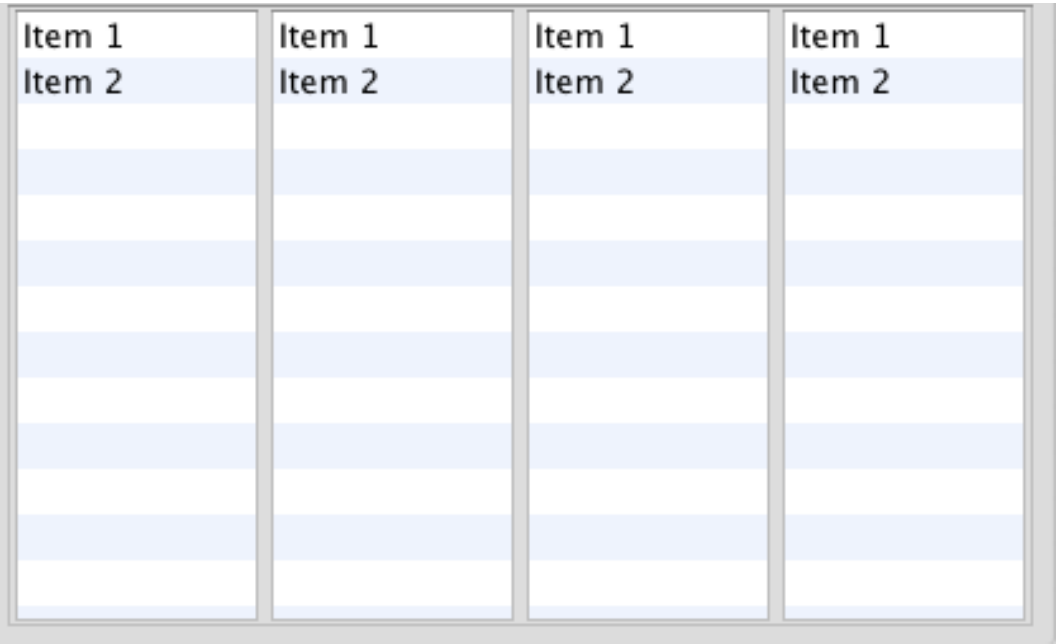
# Layouts in SWT

- Built-in layouts available
  - FillLayout
  - RowLayout
  - GridLayout
  - FormLayout
  - StackLayout
- Default layout
  - Set up coordinates and size for each widget!

If no layout is defined and no size/positioning is set up, the widget will not appear!

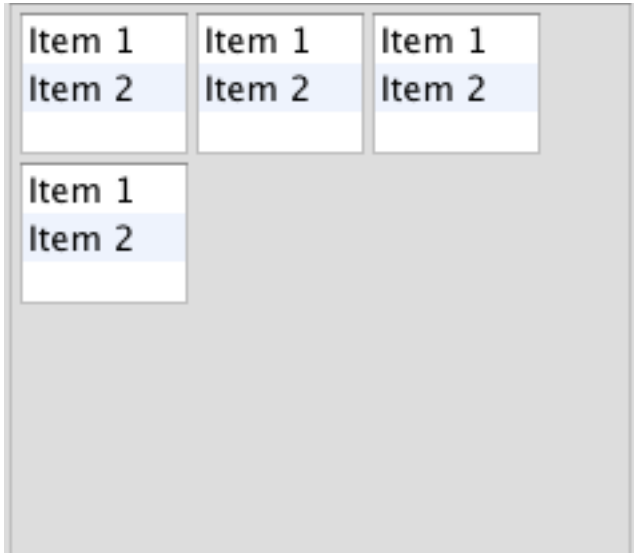
# FillLayout

- Fill all space
  - Places all elements next to each other
  - Horizontal or vertical
  - Primitive layout
    - Ignores suggested size of widgets!
- May be useful for nested composites



# RowLayout

- Similar to FillLayout
  - Arranges elements into rows or columns
  - Considers widget size
- Hint object: RowData (LayoutData) : height, width
  - Sets the preferred size of the widget

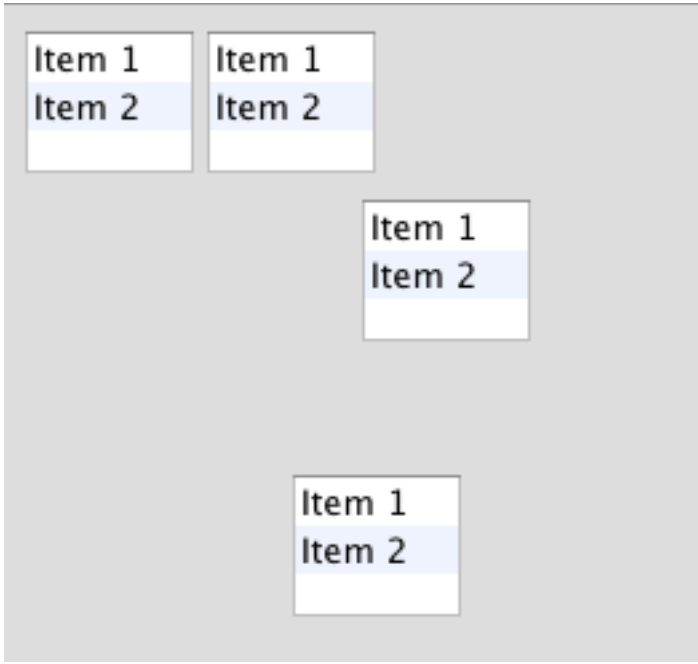


# GridLayout

- Grid arrangement
  - Fixed number of columns
- Important properties
  - horizontalSpacing
  - makeColumnsEqualWidth
  - marginHeight
  - marginWidth
  - numColumns
  - verticalSpacing

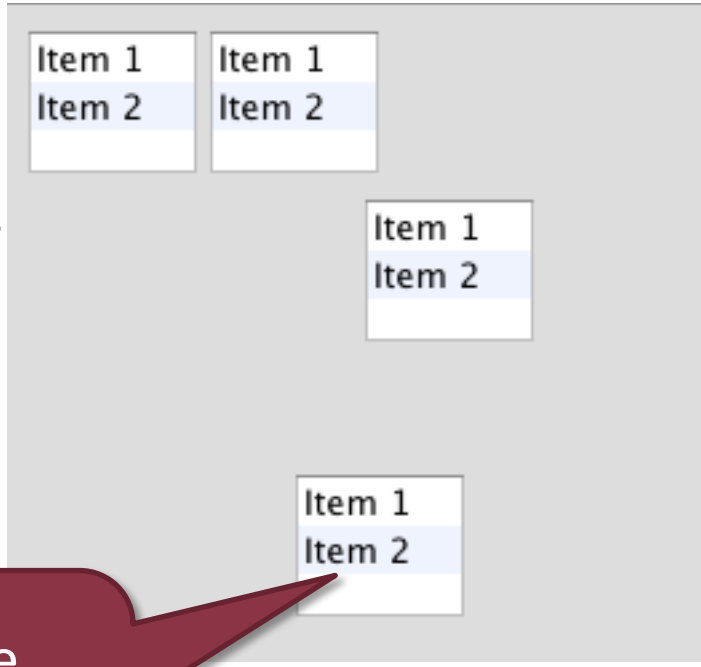
# FormLayout

- Complex layout
- Layout data stores attachments
  - Relative positioning for a selected side
  - Definition
    - y=ax+b
    - y: height, x: width
    - a: relative positioning, b: ofset



# FormLayout

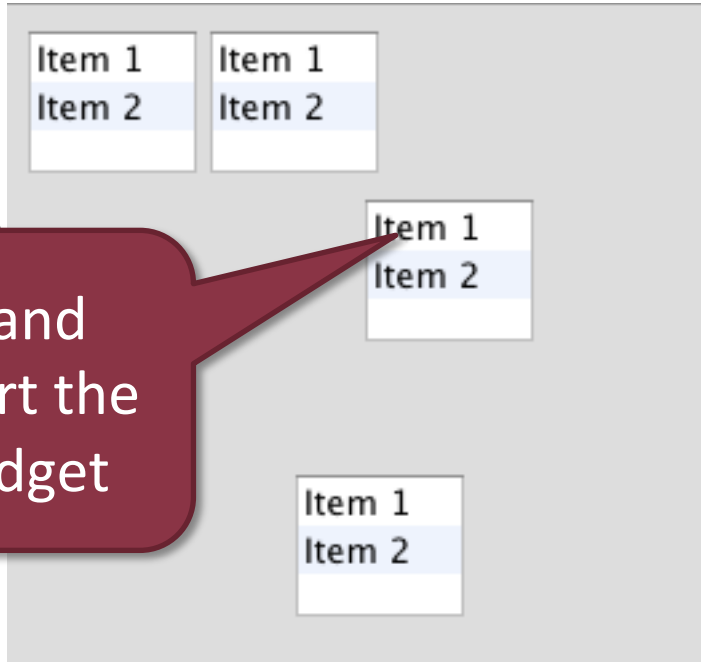
- Complex layout
- Layout data stores attachments
  - Relative positioning for a selected side
  - Definition
    - y=ax+b
    - y: height, x: width
    - a: relative positioning, b: ofset




Absolute position wrt container

# FormLayout

- Complex layout
- Layout data stores attachments
  - Relative positioning for a selected side
  - Definition
    - y=ax+b
    - y: height, x: width
    - a: relative positioning, b: ofset



# StackLayout

- Each element has the same size and position
- Only the top control will be visible
  - `StackLayout.topControl`
  - After setting, `layout()` needs to be called for UI update
- Margin settable
  - `marginHeight`
  - `marginWidth`

# Layout

# Layout

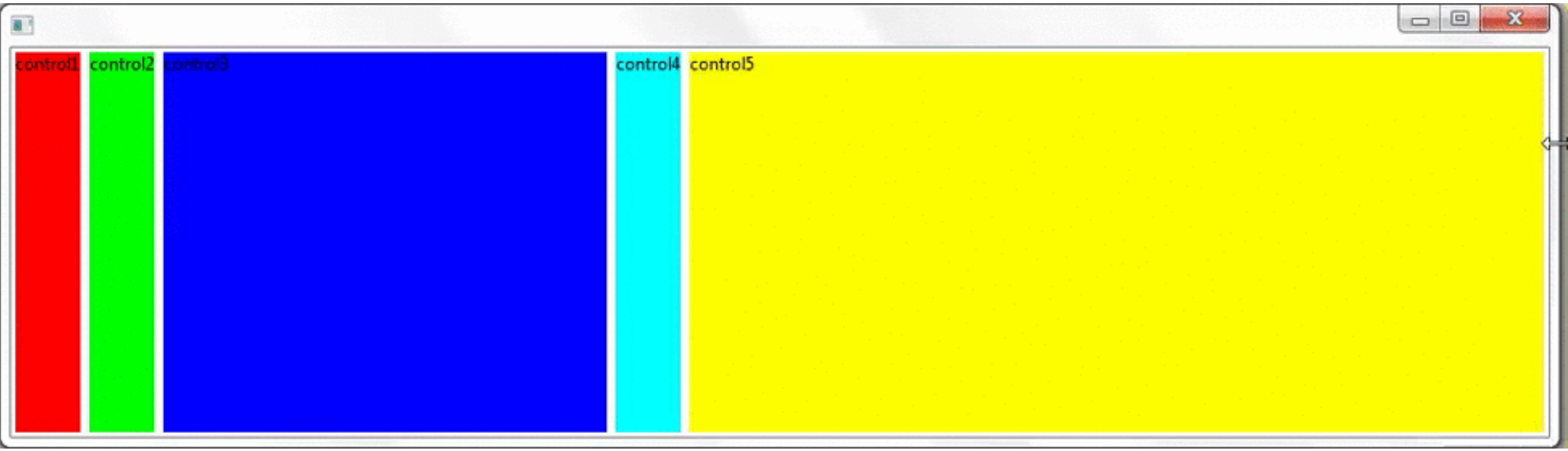
- Many layouts available
- Custom layouts possible
  - Create new layout implementation
  - E.g. responsive layout from http://www.codeaffine.com/2014/02/24/responsive-uis-with-eclipse-and-swt/
- Not required to use
  - Widget#setBound(x,y,w,h)
    - Container-relative positioning
    - Size

# Layout

- Many layouts available
- Custom layouts possible
  - Create new layout implementation
  - E.g. responsive layout from http://www.codeaffine.com/2014/02/24/responsive-uis-with-eclipse-and-swt/
- Not required to use
  - Widget#setBound(x,y,w,h)
    - Container-relative positioning
    - Size

# Layout

- Many layouts available
- Custom layouts possible
  - Create new layout implementation
  - E.g. responsive layout from http://www.codeaffine.com/2014/02/24/responsive-uis-with-eclipse-and-swt/



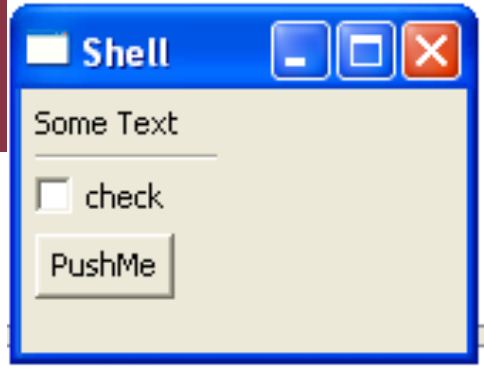
# Layout

- Many layouts available
- Custom layouts possible
  - Create new layout implementation
  - E.g. responsive layout from http://www.codeaffine.com/2014/02/24/responsive-uis-with-eclipse-and-swt/
- Not required to use
  - Widget#setBound(x,y,w,h)
    - Container-relative positioning
    - Size

# Layout

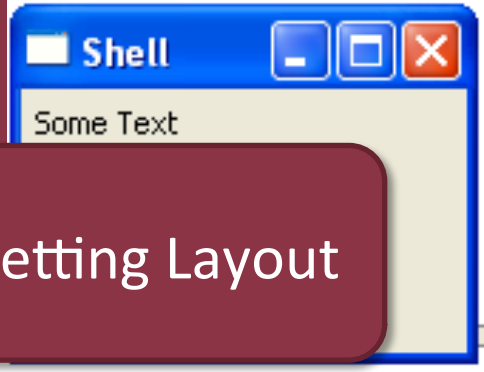
- Many layouts available
- Custom layouts possible
  - Create new layout implementation
  - E.g. responsive layout from http://www.codeaffine.com/2014/02/24/responsive-uis-with-eclipse-and-swt/
- Not required to use
  - Widget#setBound(x,y,w,h)
    - Container-relative positioning
    - Size

# Sample application



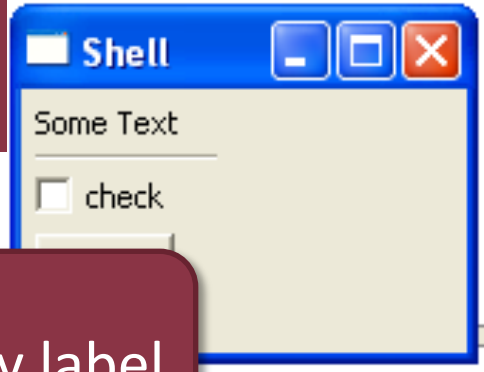

```
private void createSShell() {
        sShell = new Shell();
        sShell.setText("Shell");
        sShell.setLayout(new GridLayout());
        sShell.setSize(new Point(90,127));
        label1 = new Label(sShell, SWT.NONE);
        label1.setText("Some Text");
        label2 = new Label(sShell, SWT.SEPARATOR |
SWT.HORIZONTAL);
        label2.setText("Label");
        checkBox = new Button(sShell, SWT.CHECK);
        checkBox.setText("check");
        button = new Button(sShell, SWT.NONE);
        button.setText("PushMe");
}
```

# Sample application




```
private void createSShell() {
        sShell = new Shell();
        sShell.setText("Shell");
        sShell.setLayout(new GridLayout());
        sShell.setSize(new Point(90,127));
        label1 = new Label(sShell, SWT.NONE);
        label1.setText("Some Text");
        label2 = new Label(sShell, SWT.SEPARATOR |
SWT.HORIZONTAL);
        label2.setText("Label");
        checkBox = new Button(sShell, SWT.CHECK);
        checkBox.setText("check");
        button = new Button(sShell, SWT.NONE);
        button.setText("PushMe");
}
```

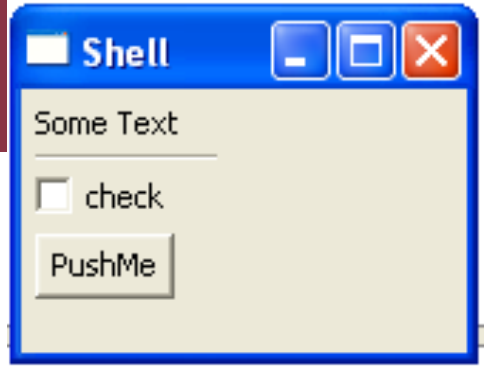
# Sample application




Read only label

```
private void createSShell() {
	sShell = new Shell();
	sShell.setText("Shell");
	sShell.setLayout(new GridLa
	sShell.setSize(new Point(9
	label1 = new Label(sShell, SWT.NONE);
	label1.setText("Some Text");
	label2 = new Label(sShell, SWT.SEPARATOR |
SWT.HORIZONTAL);
	label2.setText("Label");
	checkBox = new Button(sShell, SWT.CHECK);
	checkBox.setText("check");
	button = new Button(sShell, SWT.NONE);
	button.setText("PushMe");
}
```

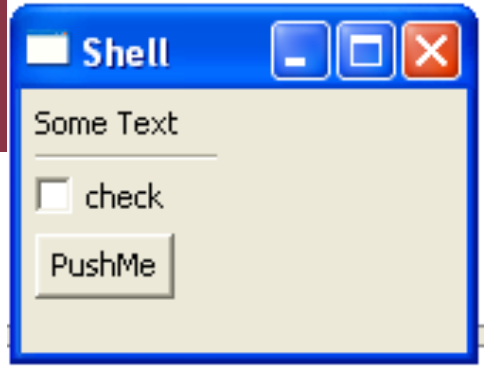
# Sample application




‘SEPARATOR’ stylebit

```
private void createSShell() {
        sShell = new Shell();
        sShell.setText("Shell");
        sShell.setLayout(new GridLayout());
        sShell.setSize(new Point(90,127));
        label1 = new Label(sShell, SWT.NON
        label1.setText("Some Text");
        label2 = new Label(sShell, SWT.SEPARATOR |
SWT.HORIZONTAL);
        label2.setText("Label");
        checkBox = new Button(sShell, SWT.CHECK);
        checkBox.setText("check");
        button = new Button(sShell, SWT.NONE);
        button.setText("PushMe");
}
```

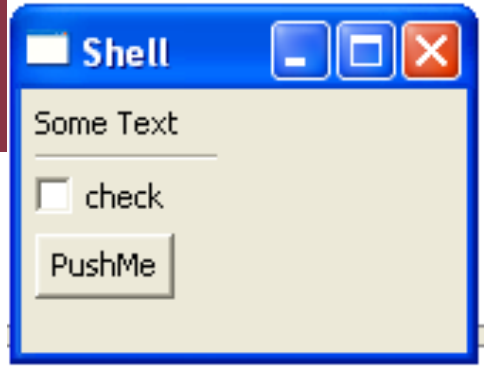
# Sample application




```
private void createSShell() {
        sShell = new Shell();
        sShell.setText("Shell");
        sShell.setLayout(new GridLayout());
        sShell.setSize(new Point(90,127));
        label1 = new Label(sShell, SWT.NONE);
        label1.setText("Some Text");
        label2 = new Label(sShell, SWT.SE
SWT.HORIZONTAL);
        label2.setText("Label");
        checkBox = new Button(sShell, SWT.CHECK);
        checkBox.setText("check");
        button = new Button(sShell, SWT.NONE);
        button.setText("PushMe");
}
```

Checkbox uses 'CHECK' style bit

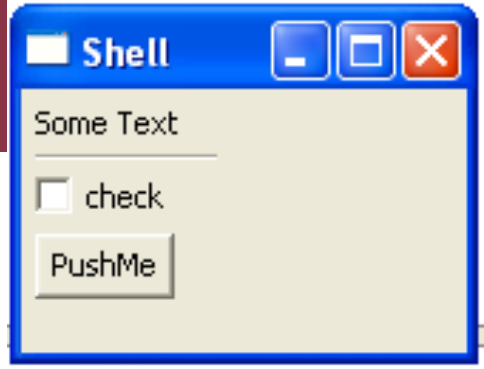
# Sample application




```
private void createSShell() {
        sShell = new Shell();
        sShell.setText("Shell");
        sShell.setLayout(new GridLayout());
        sShell.setSize(new Point(90,127));
        label1 = new Label(sShell, SWT.NONE);
        label1.setText("Some Text");
        label2 = new Label(sShell, SWT.SEPARATOR |
SWT.HORIZONTAL);
        label2.setText("Label");
        checkBox = new Button(sShell, SW
        checkBox.setText("check");
        button = new Button(sShell, SWT.NONE);
        button.setText("PushMe");
}
```

Simple button

# Sample application




```
private void createSShell() {
        sShell = new Shell();
        sShell.setText("Shell");
        sShell.setLayout(new GridLayout());
        sShell.setSize(new Point(90,127));
        label1 = new Label(sShell, SWT.NONE);
        label1.setText("Some Text");
        label2 = new Label(sShell, SWT.SEPARATOR |
SWT.HORIZONTAL);
        label2.setText("Label");
        checkBox = new Button(sShell, SWT.CHECK);
        checkBox.setText("check");
        button = new Button(sShell, SWT.NONE);
        button.setText("PushMe");
}
```

# Developing User Interface

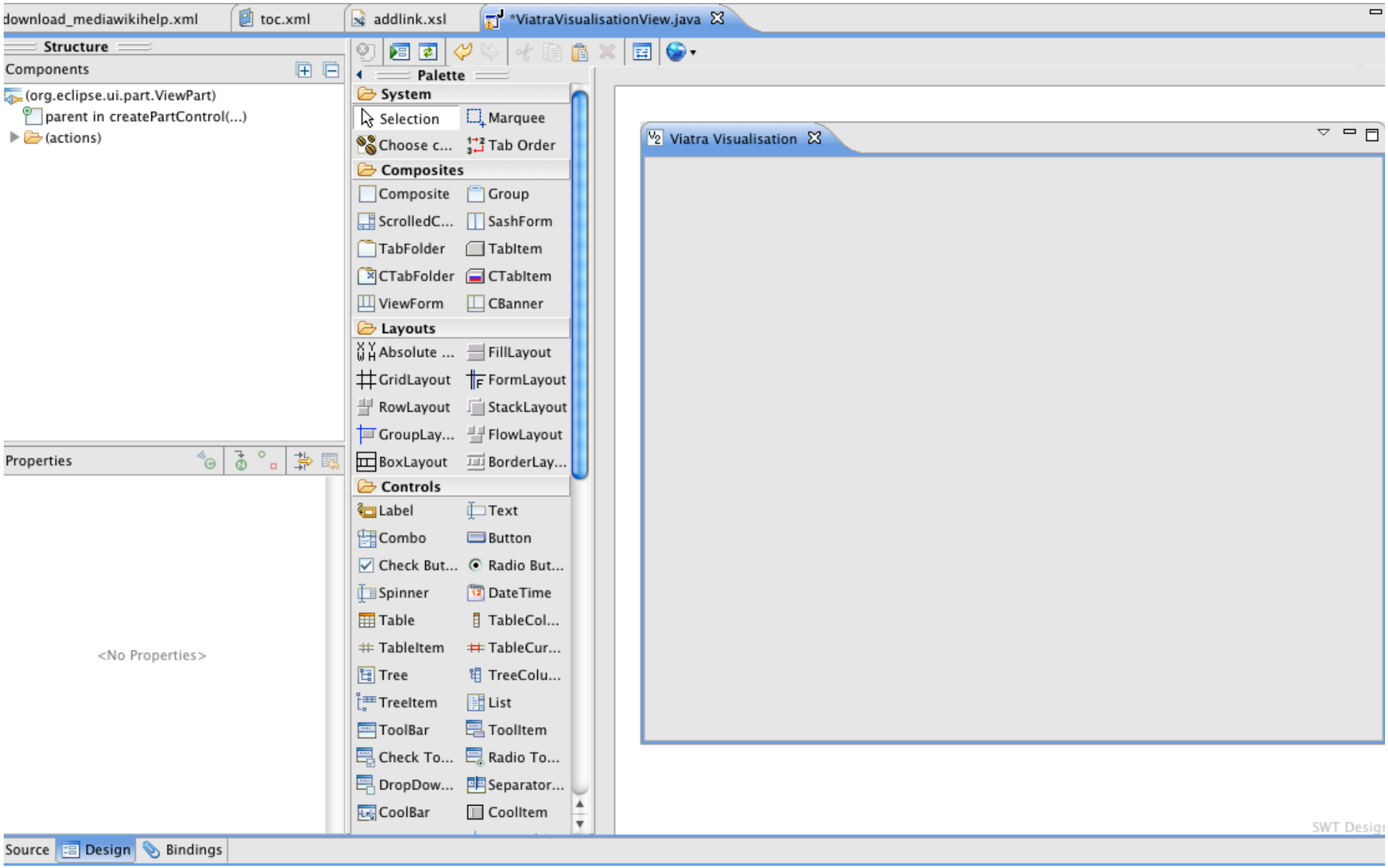
# Designing user interfaces

- Required features
  - Direct code editing
  - Layout support
  - Both source and UI editing
- Multiple tools available
  - Eclipse WindowBuilder is one of the best

# WindowBuilder

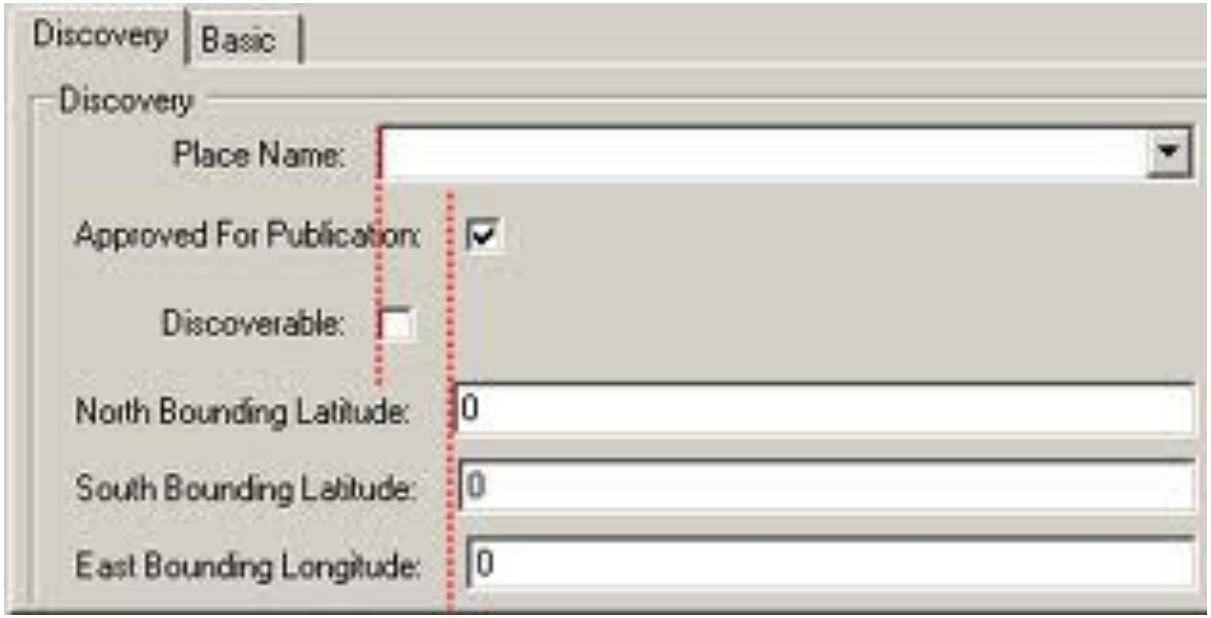
- History
  - Instantiations -> Google -> Eclipse
  - Roundtrip engineering
  - SWT, JFace, Forms API
  - Eclipse Workbench
  - BUT: memory requirements
    - Pro tip: do not use WindowBuilder as default Java editor
  - BUT: Smaller bugs, missing features

# WindowBuilder



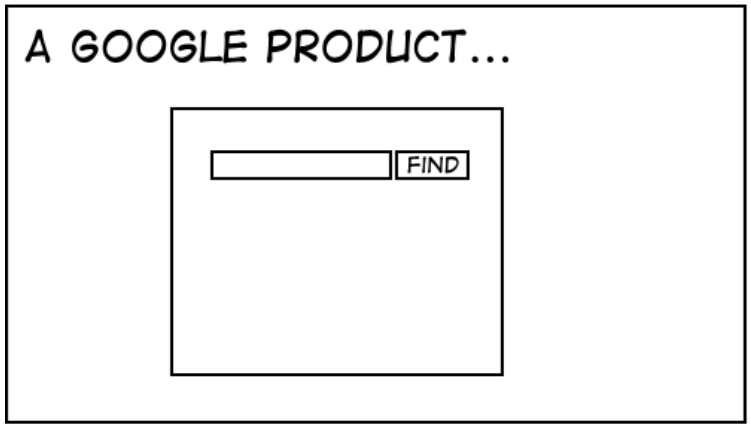
# Pitfall 1.

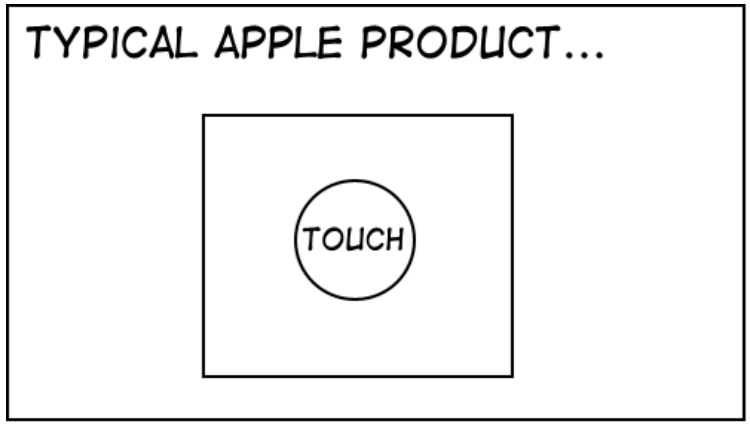
- Use automatic layouting if possible
  - Avoids alignment errors

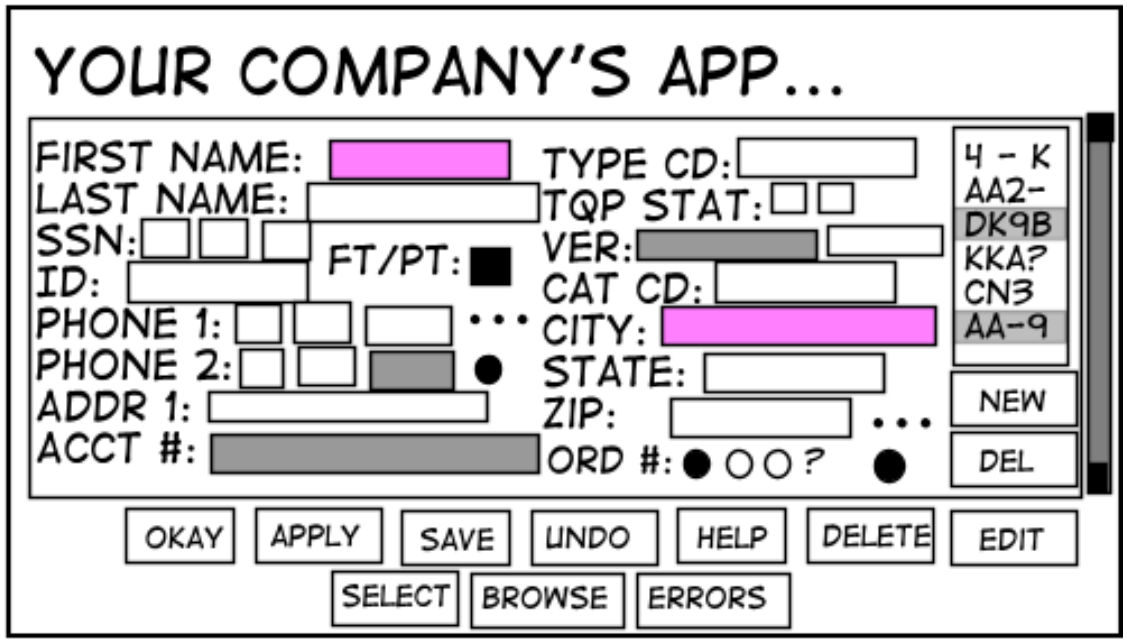



Source: http://uxmatters.com/mt/archives/2009/02/reviewing-user-interfaces.php

# Easy to create != Easy to use




STUFFTHATHAPPENS.COM BY ERIC BURKE

# Summary

# SWT - Summary

- Native graphical user interface framework
  - Fast
  - Simple
- Different form elements
  - **Complex forms**
  - Dialogs
  - Menus
  - Drawing
  - Printing
  - …

# Further reading

- Understanding Layout in SWT
  - http://www.eclipse.org/articles/article.php?file=Article-Understanding-Layouts/index.html
  - Describes Layouts
- User Interface Guidelines – Eclipsepedia
  - http://wiki.eclipse.org/User_Interface_Guidelines
- SWT Snippets:
  - http://www.eclipse.org/swt/snippets/
  - Grouped samples for coding SWT