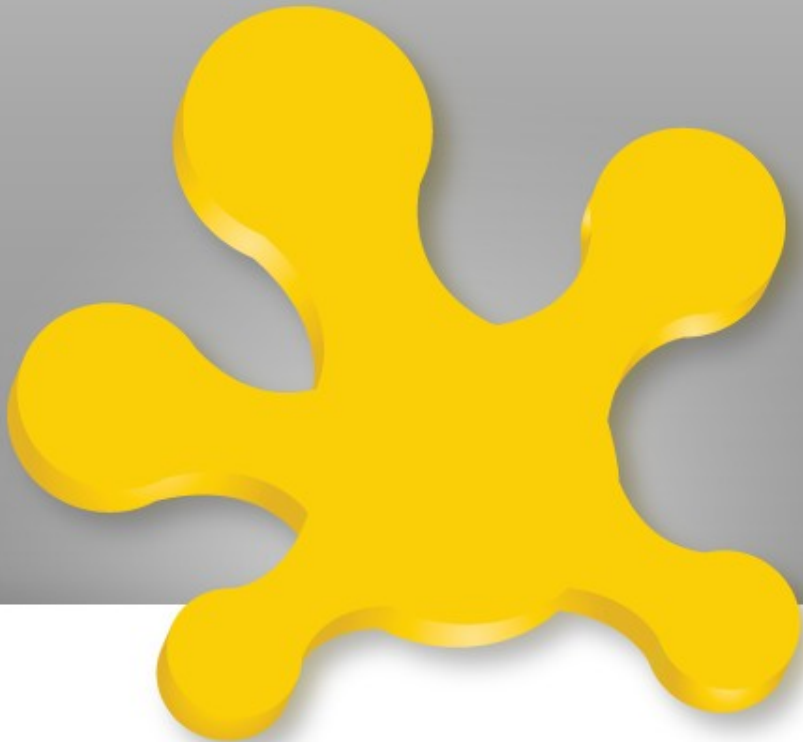


# Krzysztof Daniel, Jakub Jurkiewicz

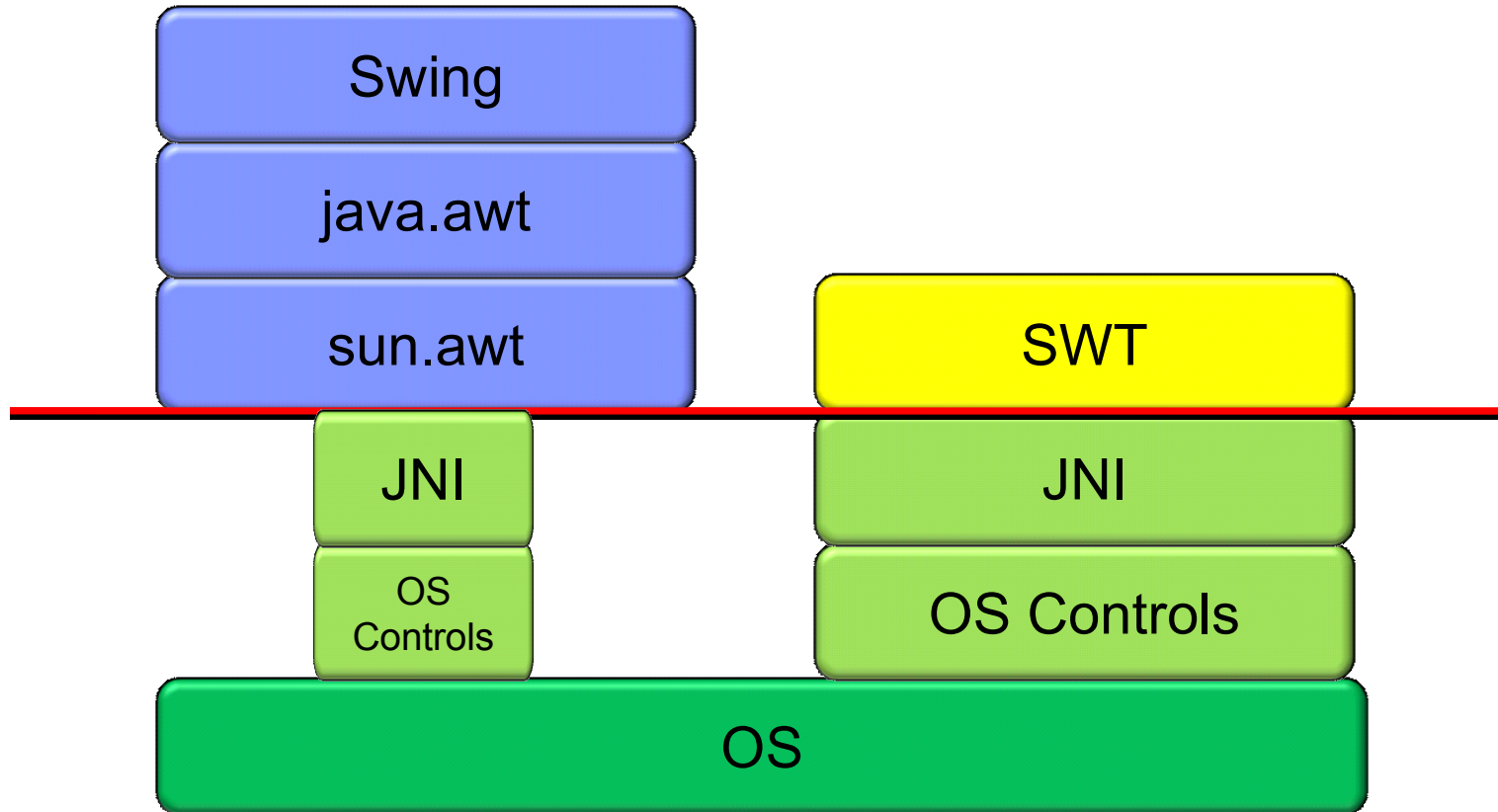
## SWT & JFace



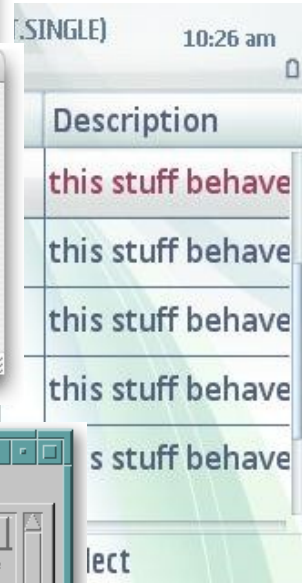
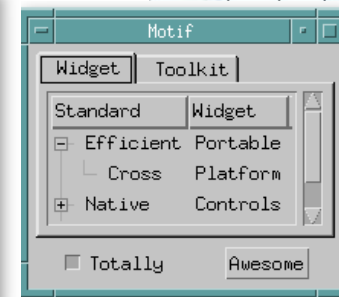
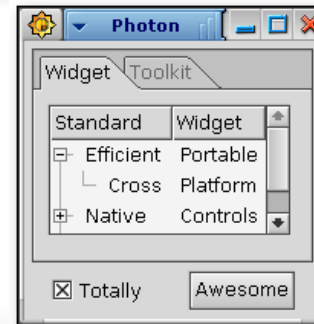
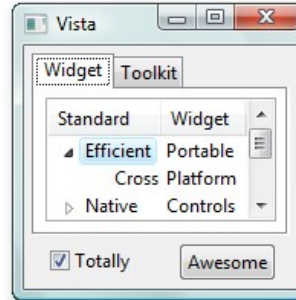
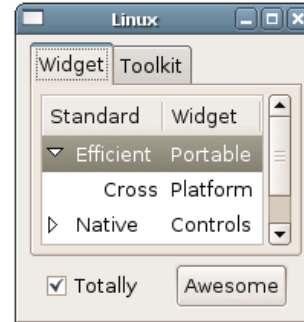
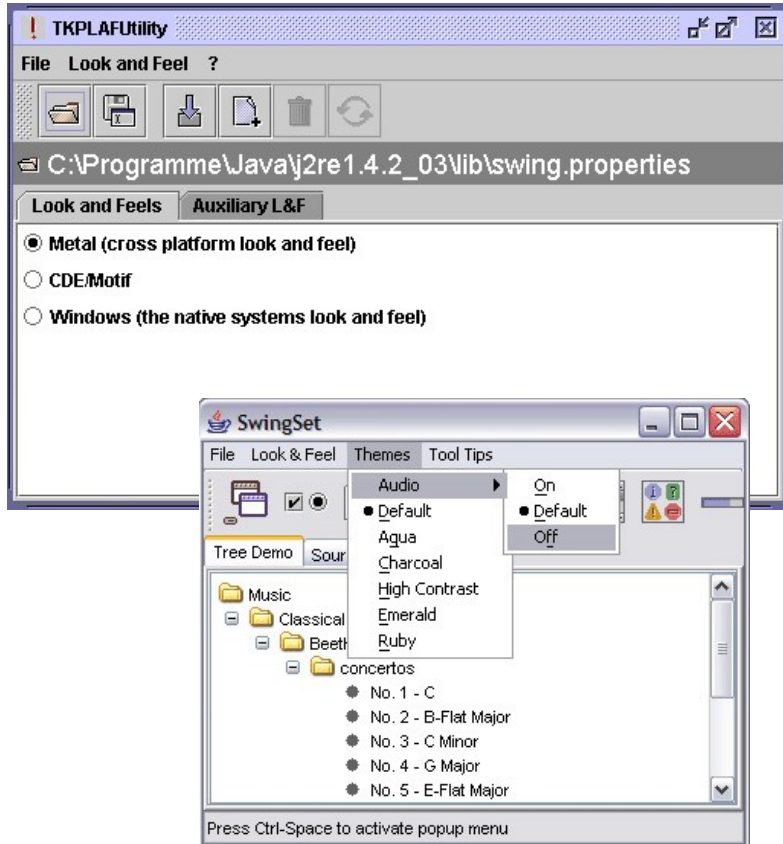
# The Simple Widget Toolkit

- Why
  - Swing was not fast enough
  - No alternatives
  - IBM's experience in SmallTalk

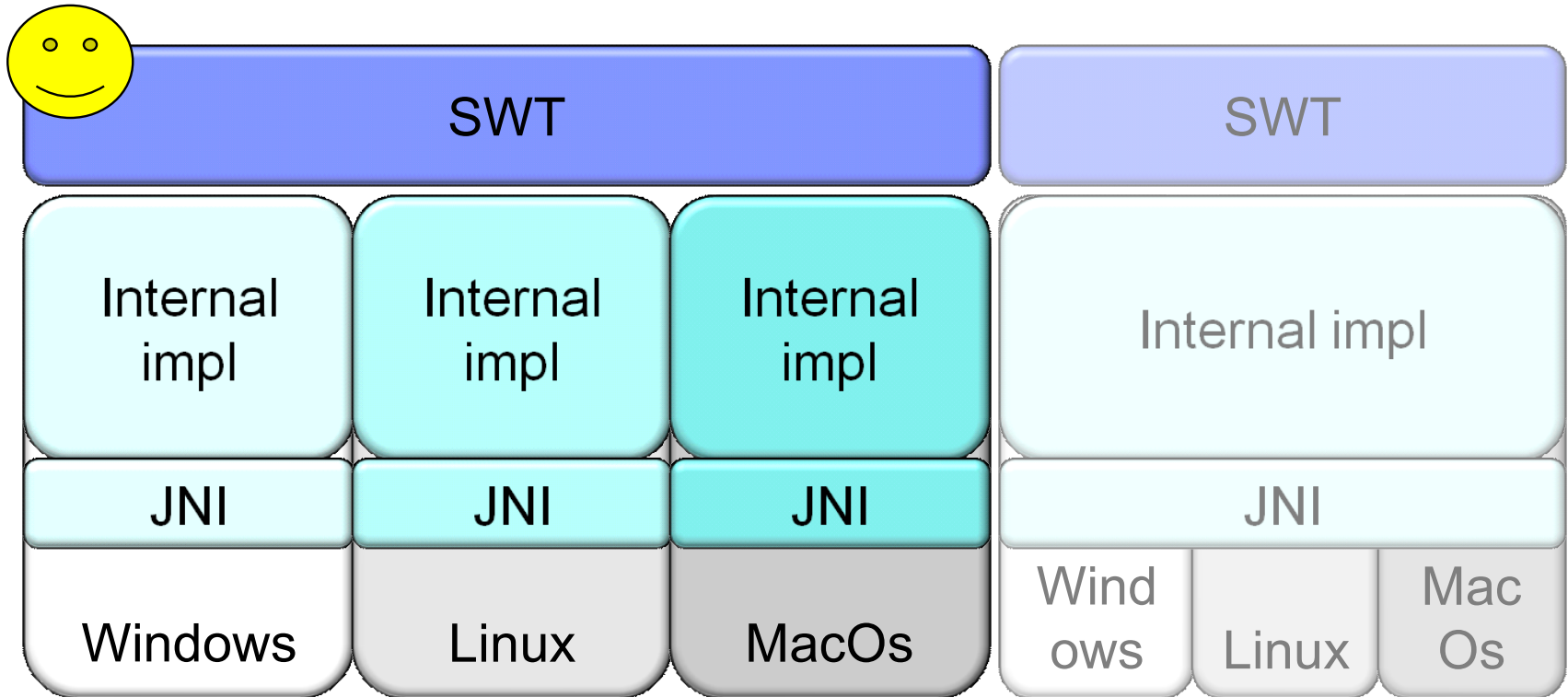
# The Standard Widget Toolkit



# The Standard Widget Toolkit



# The Standard Widget Toolkit



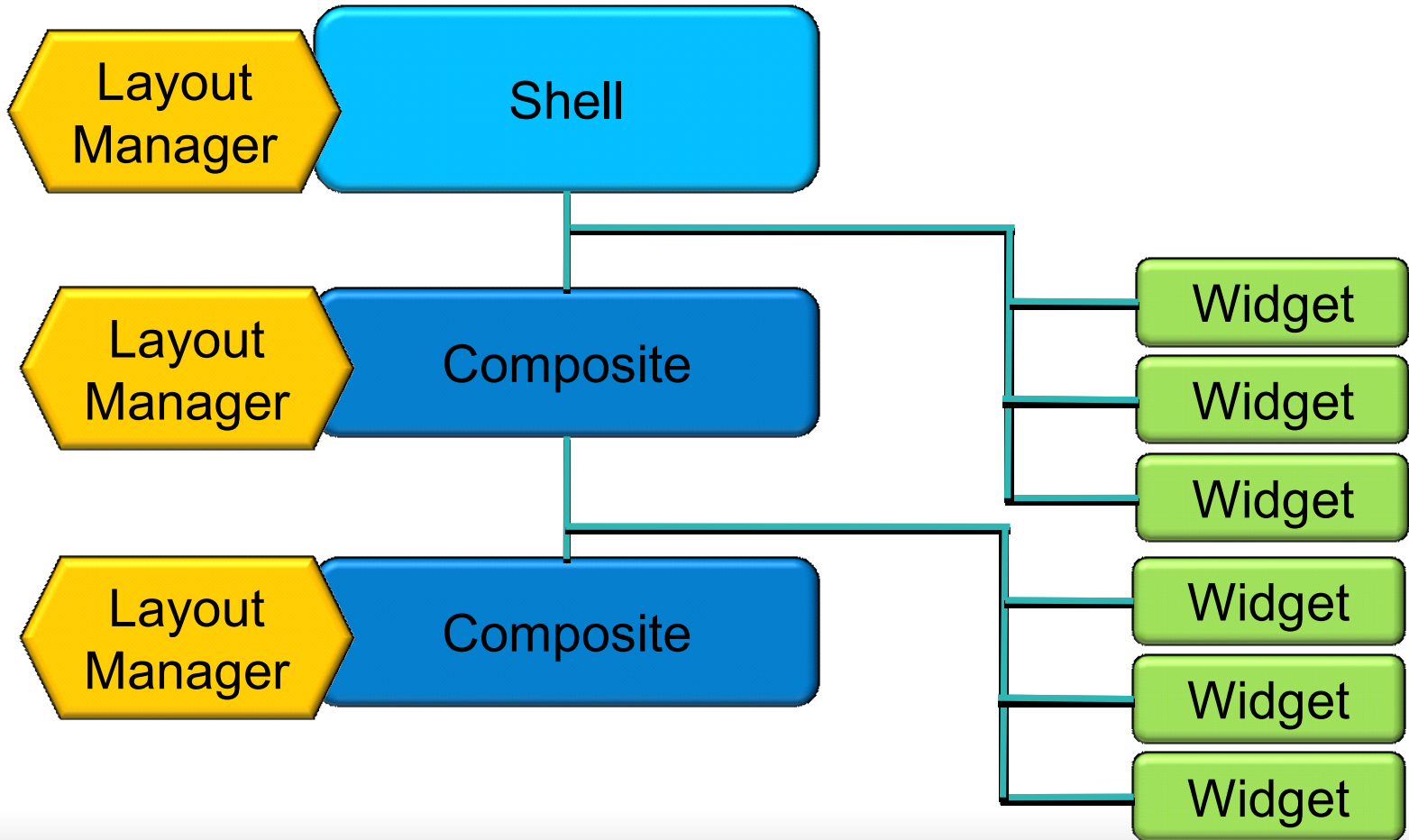
# SWT Drawbacks

- Uses native widgets
  - Memory management
  - System limitations
- Difficult to extend
- Difficult to test

# SWT: Basic terms

- Display
- Shell
- Composite
- Layout manager

# SWT: Structure



# SWT – Hello World

The image shows the Eclipse IDE interface with several windows open. The primary window is the 'New Extension' wizard, which is currently in the 'Extension Point Selection' step. The wizard's title bar reads 'New Extension'. Below the title bar, the text says 'Create a new Applications extension.' and there is a key icon. The 'Extension Point Selection' step includes an 'Extension Point filter' field and a list of extension points. The selected extension point is 'org.eclipse.core.runtime.applications'. Below the list, there is a checkbox for 'Show only extension points from the required plug-ins' which is checked. The 'Extension Point Description' is 'Applications', and the description text is highlighted in yellow. The description reads: 'Platform runtime supports plug-ins which would like to declare main entry points. That is, programs which would like to run using the platform runtime but yet control all aspects of execution can declare themselves as an application. Declared applications can be run directly from the main platform launcher by specifying the `-application` argument where the parameter is the id of an'. Below the description, there are two empty boxes for 'Available templates for applications'. At the bottom of the wizard, there are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'. In the background, other windows are visible, including 'New Plug-in Project' and 'Overview' for 'GeeConApplication1'.

**New Plug-in Project**

Plug-in Project: Create a new plug-in project.

Project name: [ ]

Use default location

Location: C:\V\

Project Settings

Create a Java project

Source folder: [ ]

Output folder: [ ]

Target Platform

This plug-in is targeted to:

Eclipse

an OSGi container

Working sets

Add project to working sets

**New Plug-in Project Content**

Enter the data required to create the plug-in project.

Properties

ID: [ ]

Version: [ ]

Name: [ ]

Provider: [ ]

Execution Environment: [ ]

Options

Generate an activator class

Activator: [ ]

This plug-in will be a rich client application

Enable API annotations

Rich Client Application

Would you like to generate a rich client application?

**Overview**

General Information

This section describes general information about the plug-in project.

ID: GeeConApplication1

Version: 1.0.0.qualifier

Name: GeeConApplication1

Dependencies

GeeConApplication1/META-INF/MANIFEST

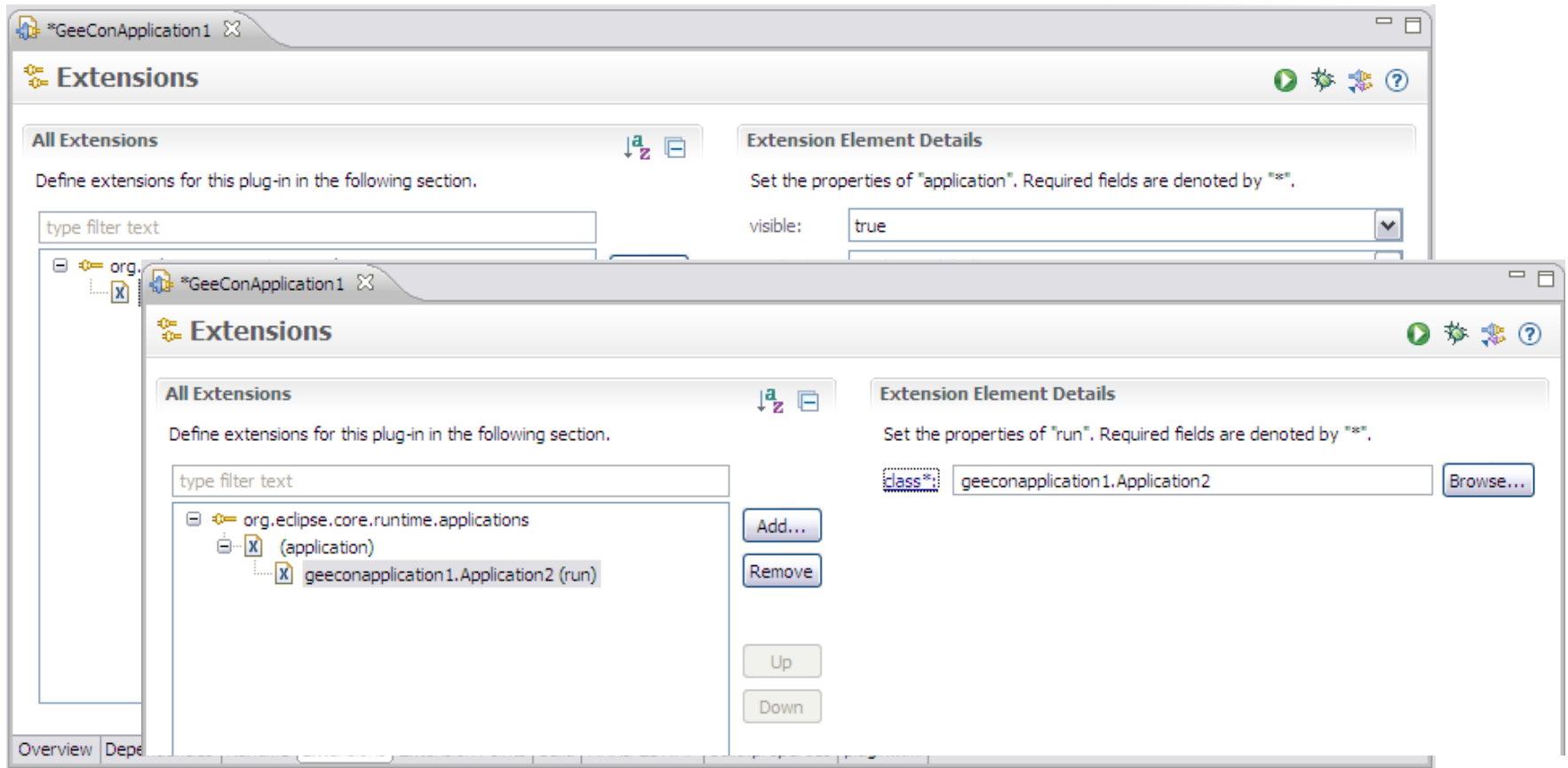
**Extensions**

Define extensions for this plug-in in the following sections.

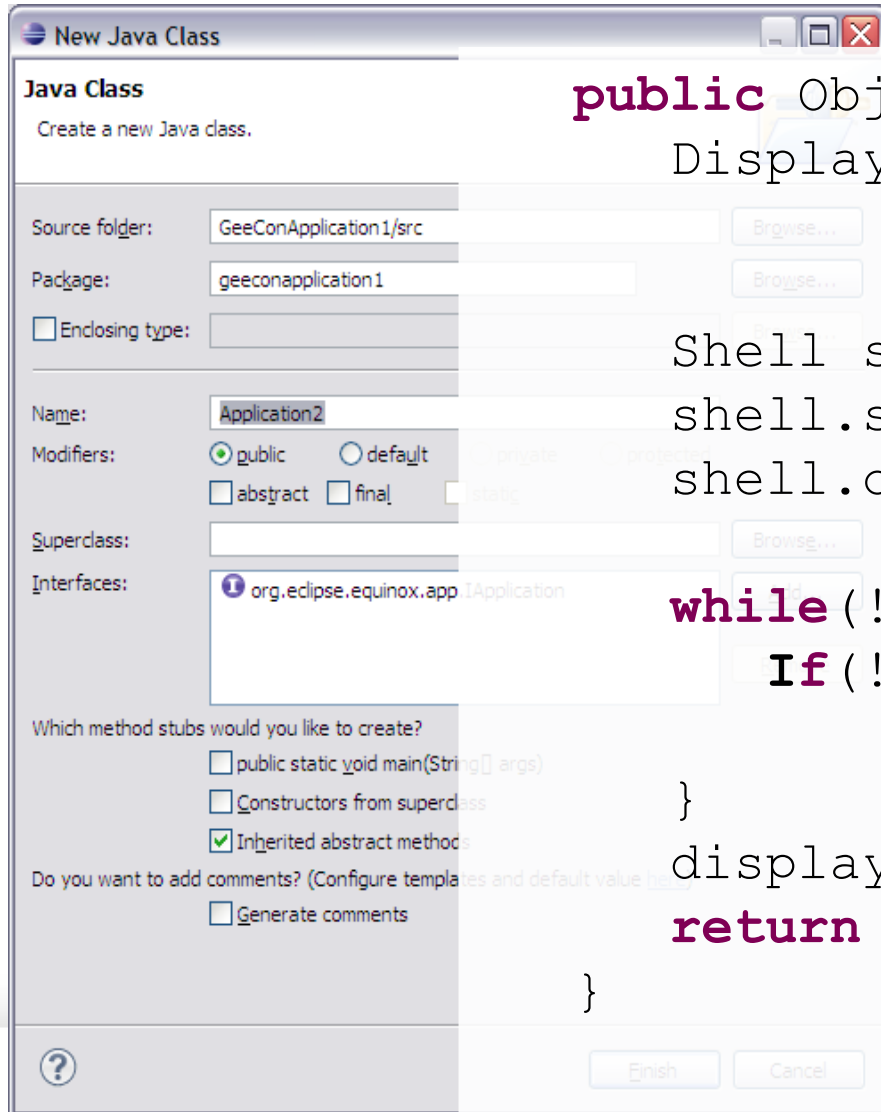
type filter text

Overview Dependencies Runtime Extensions Extensions

# SWT – Hello World

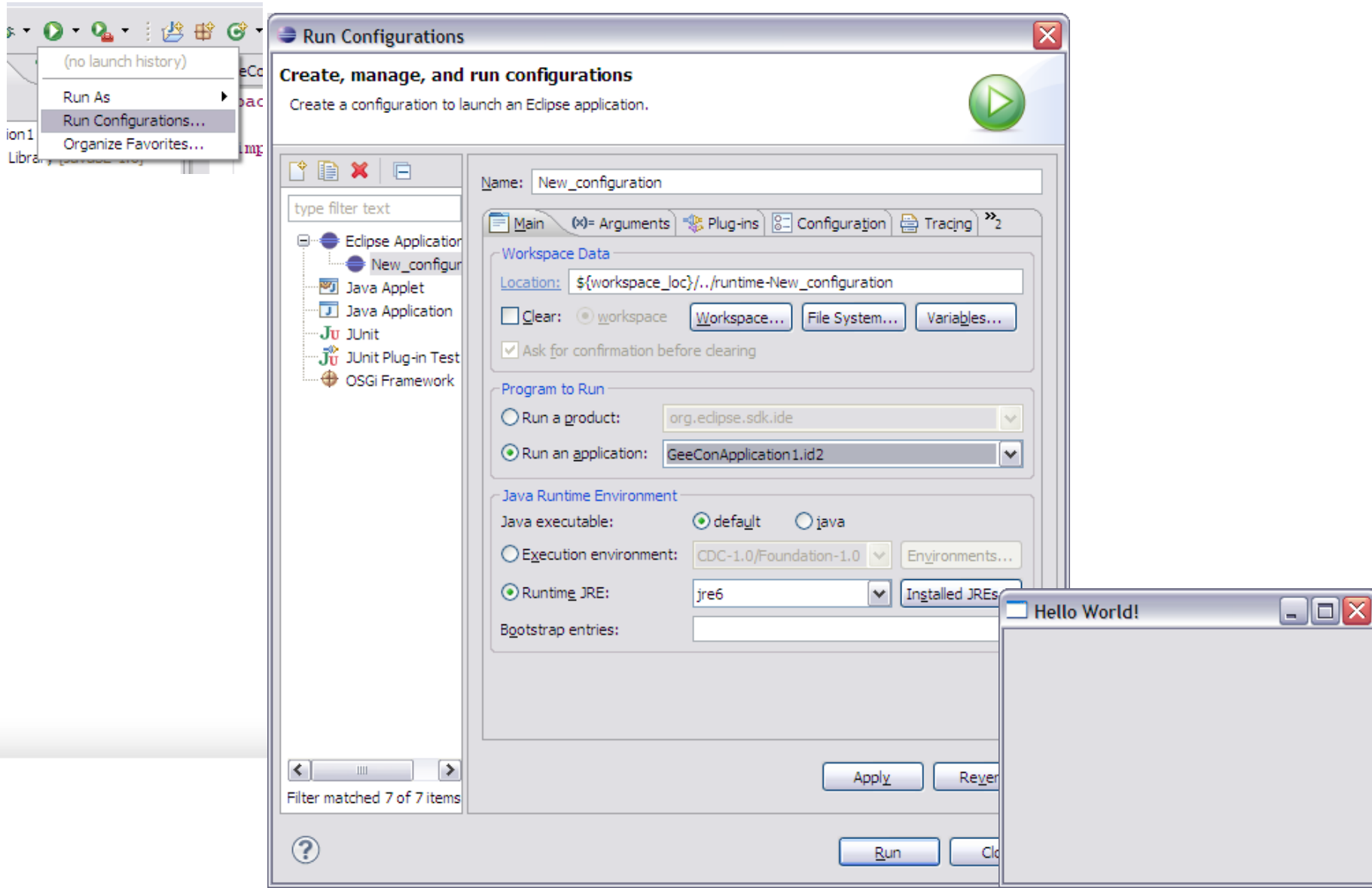


# SWT – Hello World



```
public Object start [...] {  
    Display display =  
        Display.getDefault();  
  
    Shell shell = new Shell(display);  
    shell.setText("Hello World!");  
    shell.open();  
  
    while (!shell.isDisposed()) {  
        if (!display.readAndDispatch())  
            display.sleep();  
    }  
    display.dispose();  
    return null;  
}
```

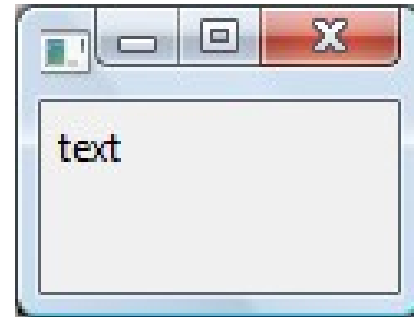
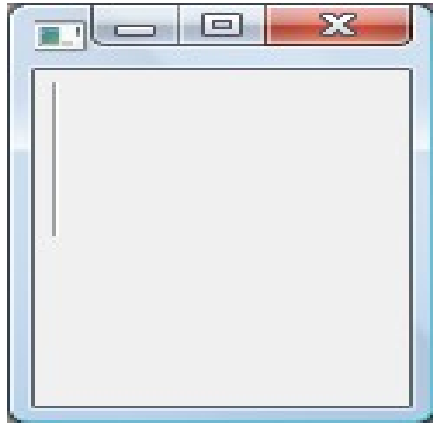
# SWT – Hello World



# Basic Components - Label

```
Label label = new Label(shell, SWT.NONE);  
label.setText("&Don't talk to me about life");
```

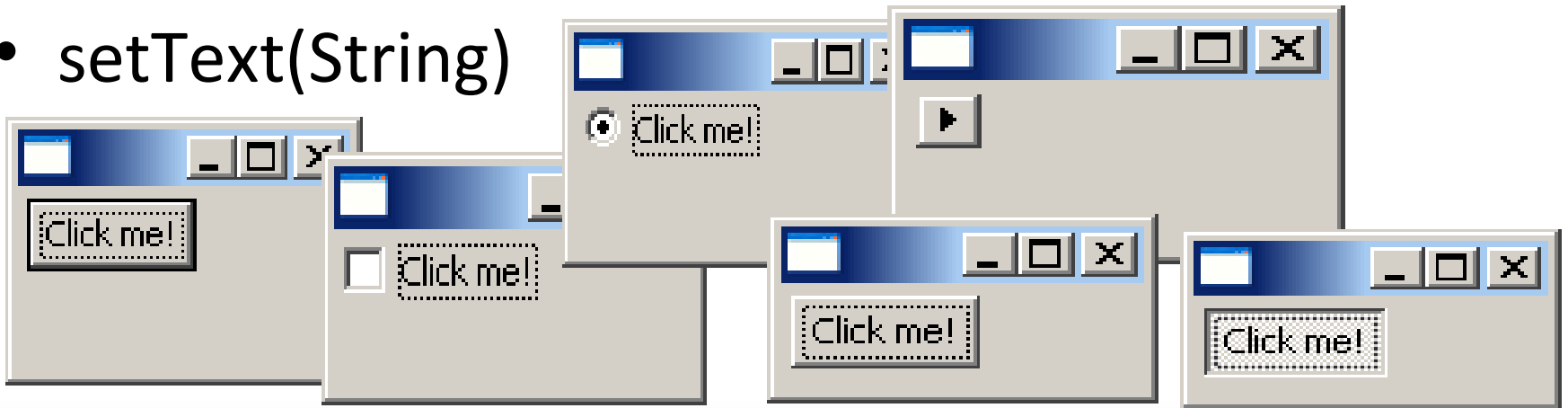
- Different styles, f.e. SWT.SEPARATOR
- setText(String), setImage(Image)



# Basic Components - Button

```
Button button = new Button(shell, SWT.NONE);  
button.setText("&Don't!");  
button.addSelectionListener([...]);
```

- PUSH, CHECK, ARROW|RIGHT, RADIO, TOGGLE
- `setText(String)`



# Let's add some widgets

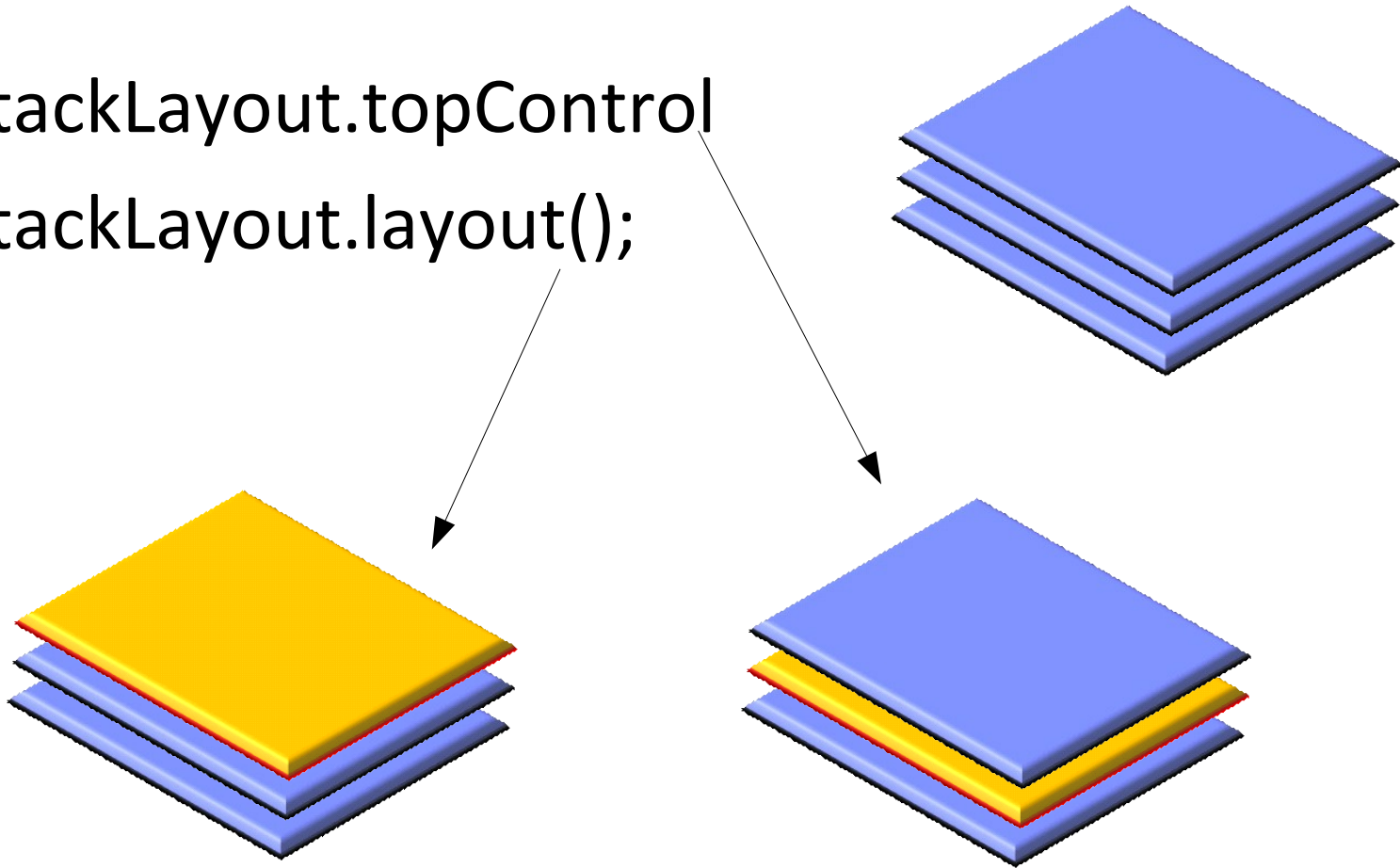
## It does not work

# Layout Manager

- Layout manager needed
  - Extends Layout
  - Size and position under control
  - FillLayout, RowLayout, FormLayout, StackLayout, GridLayout
- LayoutData

# StackLayout

- `StackLayout.topControl`
- `StackLayout.layout();`



# GridLayout

- numColumns



- makeColumnsEqualWidth



# GridData

- `horizontalAlignment`, `verticalAlignment`
- `SWT.BEGINNING`, `CENTER`, `END`, `FILL`
- `horizontalSpan`, `verticalSpan`
- `grabExcessVerticalSpace`,  
`grabExcessHorizontalSpace`

# The first serious Task

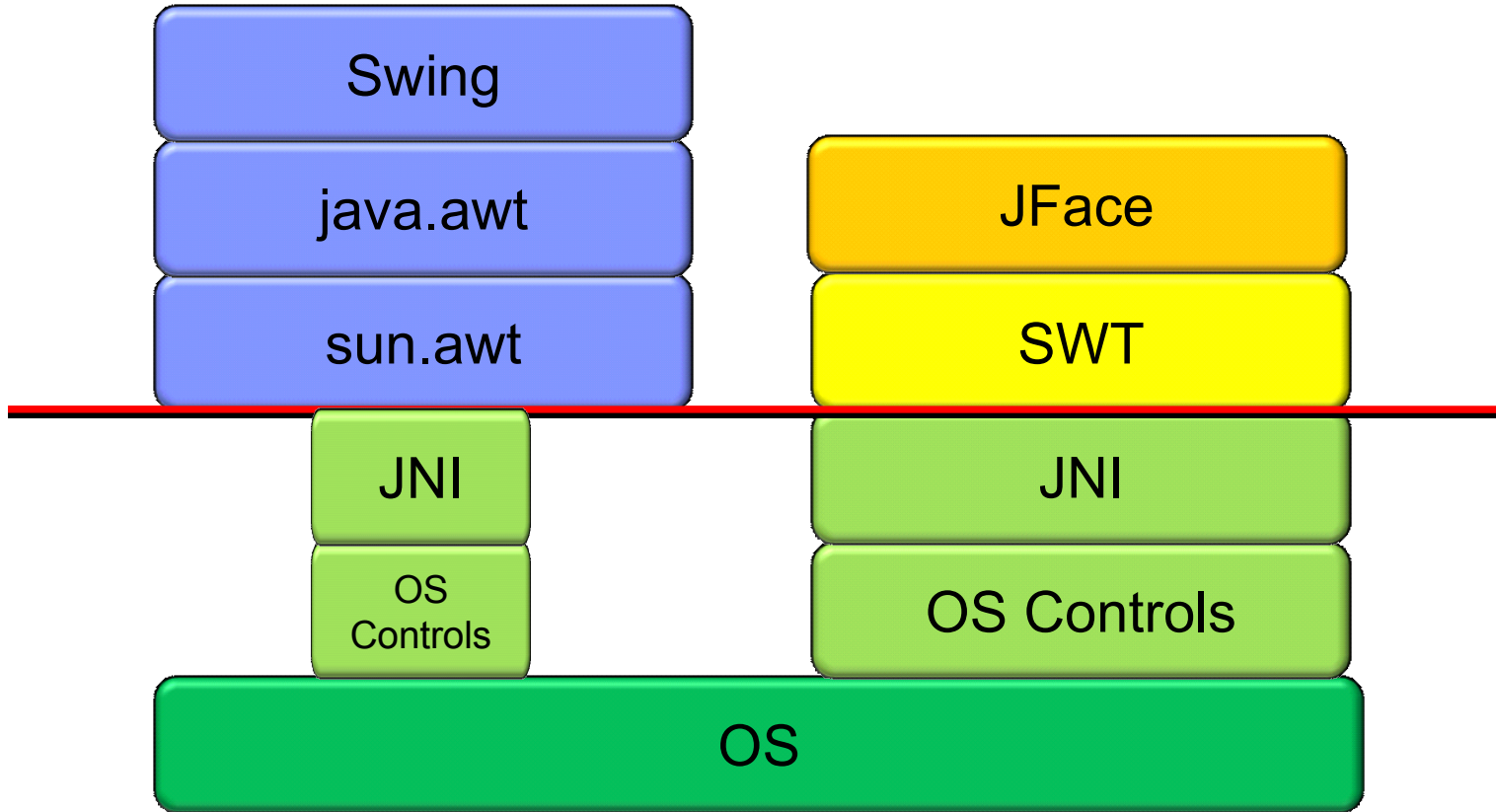


- Text
- Button
- Browser
- GridLayout
- ProgressBar

# Further reading

- The true story of SWT
- <http://www.mail-archive.com/jug-discussion@tucson-jug.org/msg00355.html>
- [www.eclipse.org/swt](http://www.eclipse.org/swt)
- <http://www.eclipse.org/swt/snippets/>

# JFace



**Hint:** add jface to required plugins

# Usability improvements - Registries

- FontRegistry
- ColorRegistry
- JFaceResources
- JFaceColors



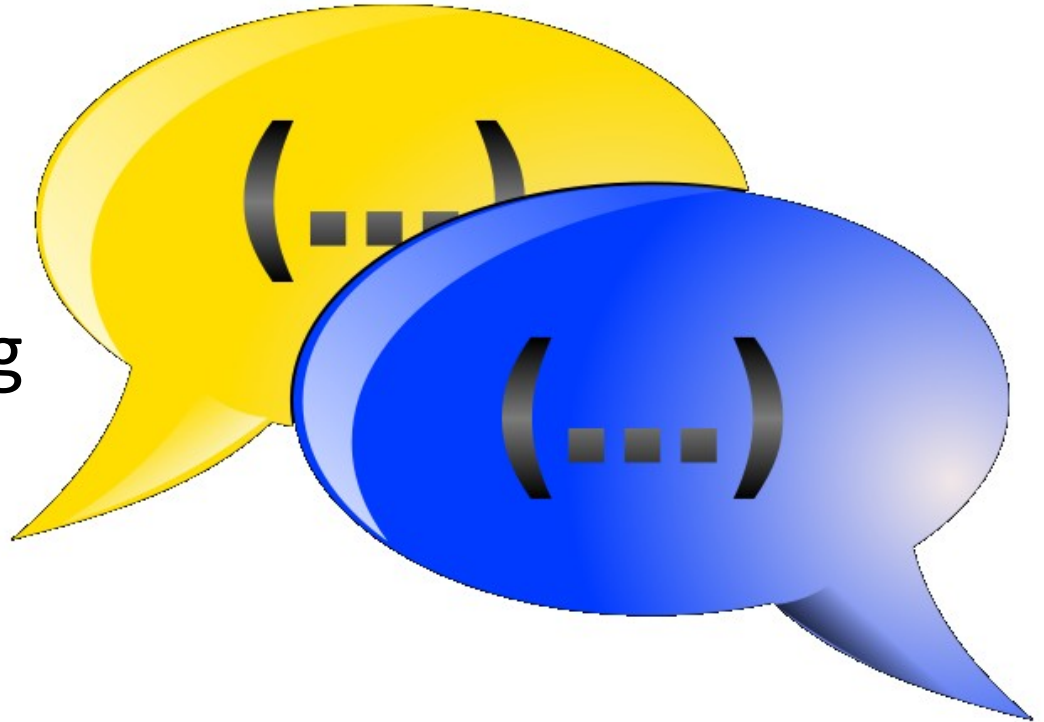
# Registry example

```
FontRegistry fontRegistry =  
    JFaceResources.getFontRegistry();  
  
FontData mainFont = new FontData("Arial", 18,  
    SWT.NORMAL);  
  
fontRegistry.put("mainFont",  
    new FontData[]{mainFont});  
  
Label label = new Label(shell, SWT.NONE);  
label.setText("This is main font");  
  
label.setFont(fontRegistry.get("mainFont"));
```



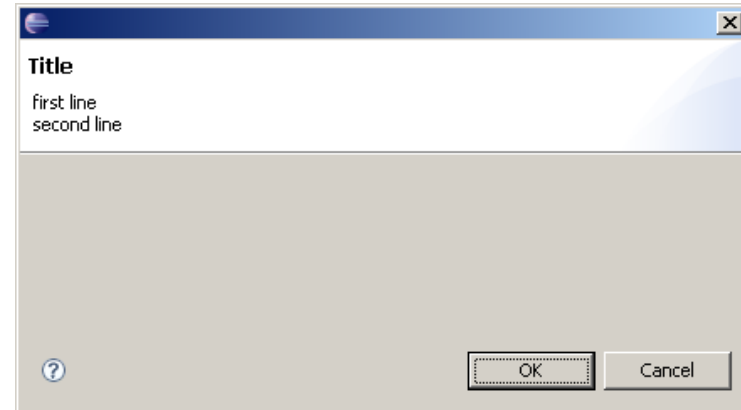
# Usability Improvements - Dialogs

- `ErrorDialog`
- `InputDialog`
- `MessageDialog`
- `ListSelectionDialog`
- ...



# TitleAreaDialog

```
final
TitleAreaDialog titleAreaDialog
    = new TitleAreaDialog(shell);
titleAreaDialog.create();
titleAreaDialog.setTitle("Title");
titleAreaDialog.setMessage(
    "first line\nsecond line");
```

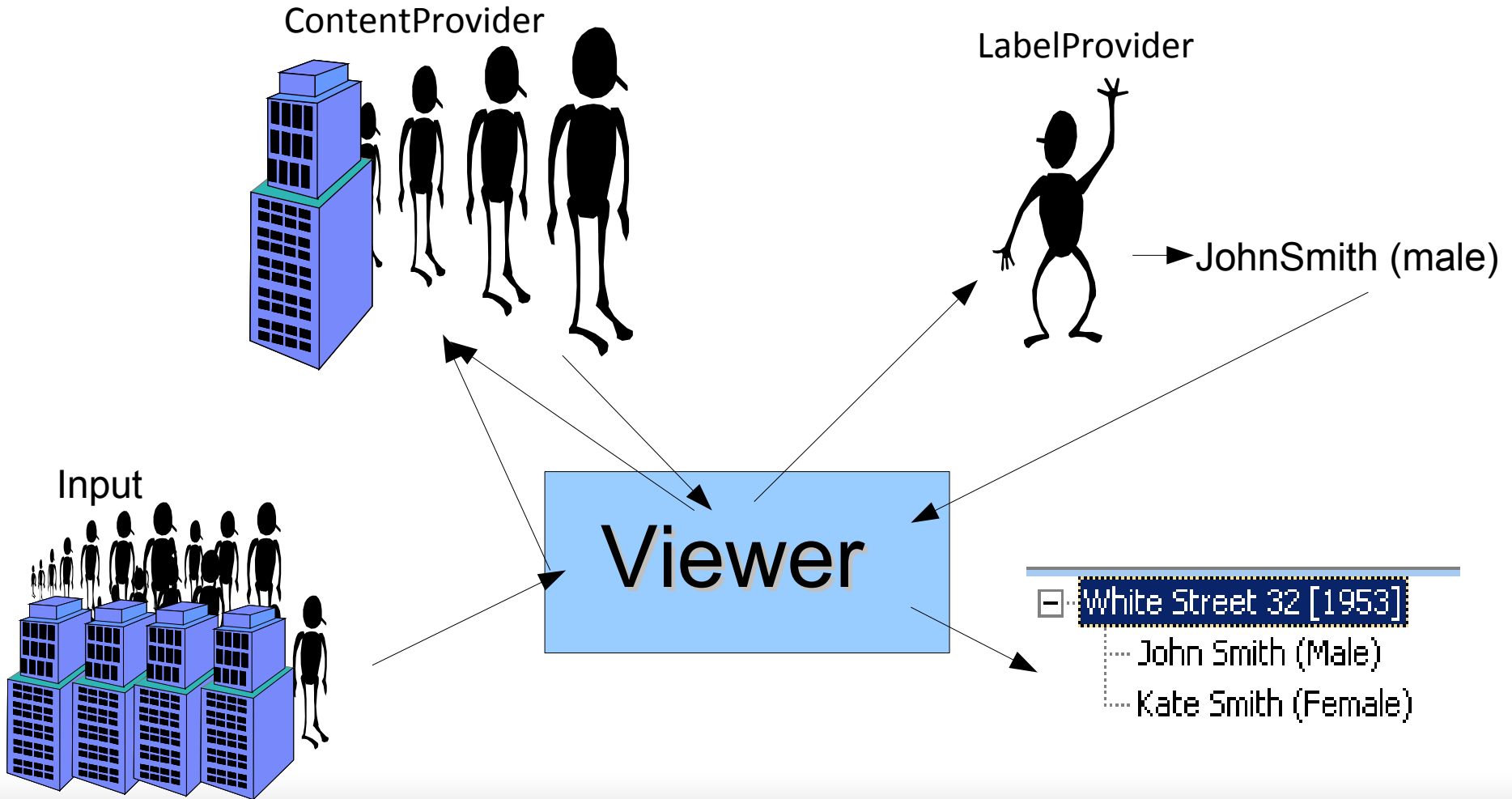


```
final Button button = new Button(shell, SWT.PUSH);
button.addSelectionListener(new SelectionAdapter() {
public void widgetSelected(SelectionEvent e) {
    if (titleAreaDialog.open() == Dialog.OK) {
        button.setText("OK was pressed!");
    }
}});
```

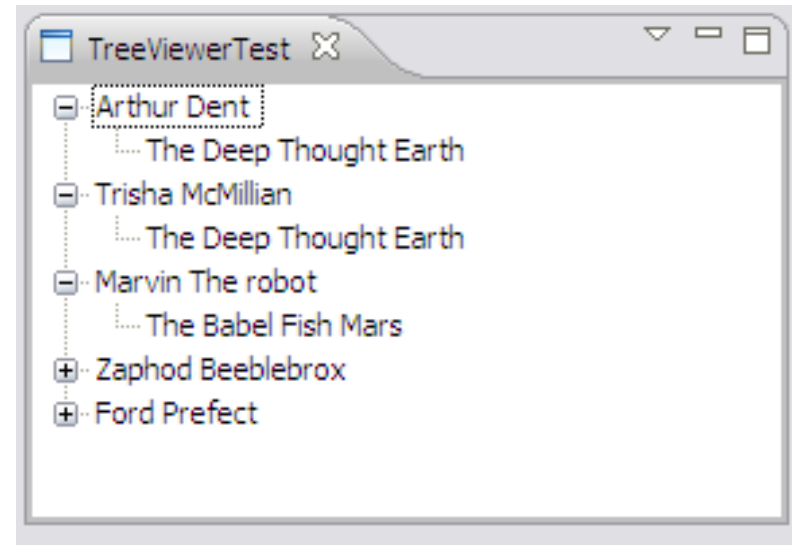
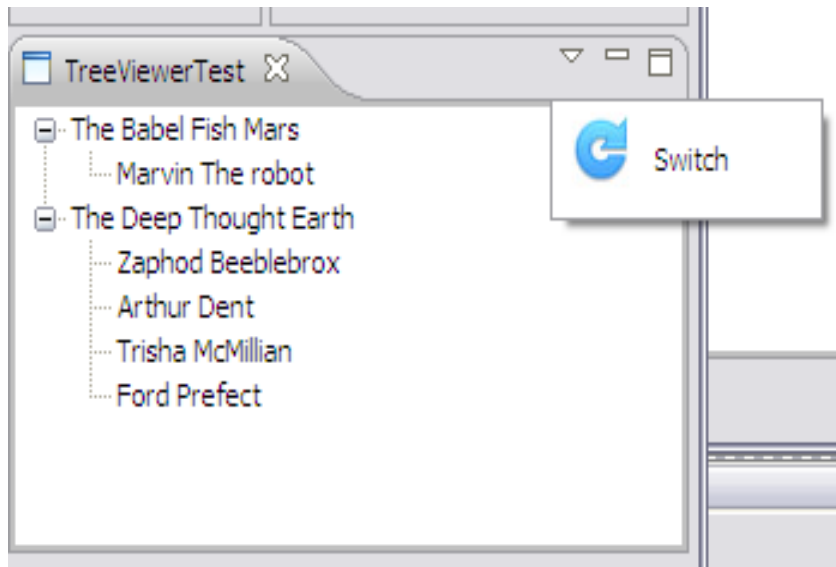
# Creating your own dialog

- Extend one of existing jface Dialog
- Override any of following methods:
  - createDialogArea(Composite c)
  - createButtonBar(Composite c)
  - createButtonsForButtonBar(Composite c)
  - createButton(Composite c, int id, String label, boolean default)
  - setReturnCode(int code)

# Usability Improvements - Viewers



# The second serious Task



# The second serious task

- org.eclipse.ui + model
  - <http://libra.cs.put.poznan.pl/kdaniel/geecon/>
- org.eclipse.ui.views
- org.eclipse.ui.viewActions
- TreeViewer, ITreeContentProvider, LabelProvider

# The third serious task

- Add „configure” dialog where one can specify which details should be displayed

# Q & A