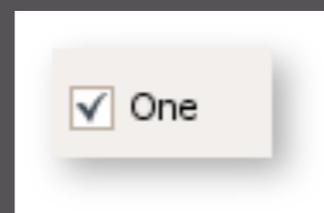
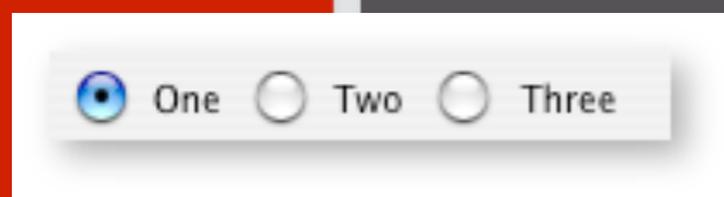
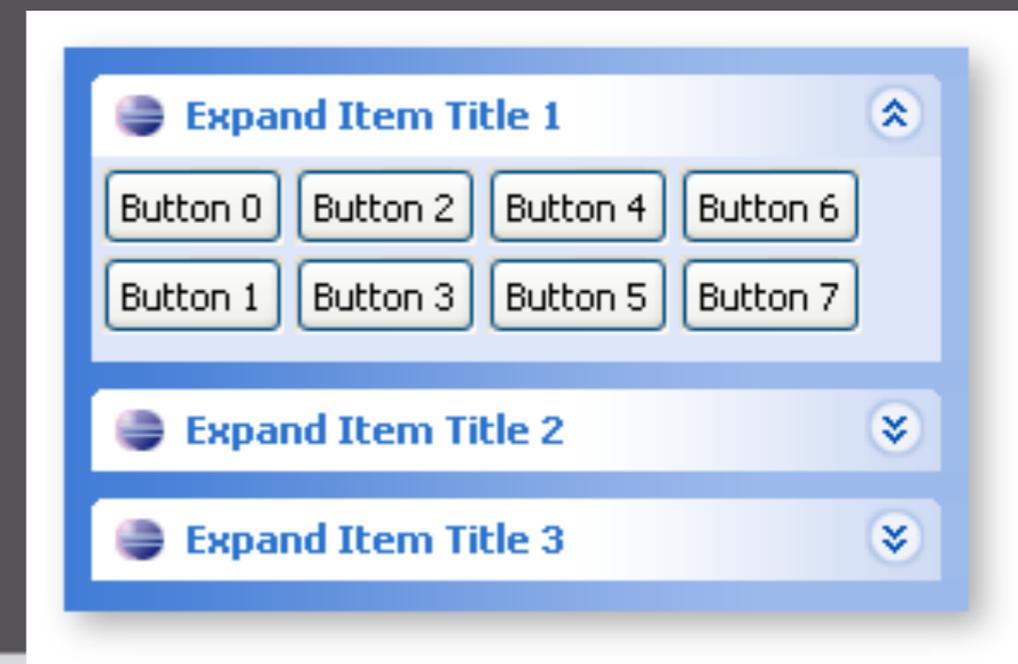
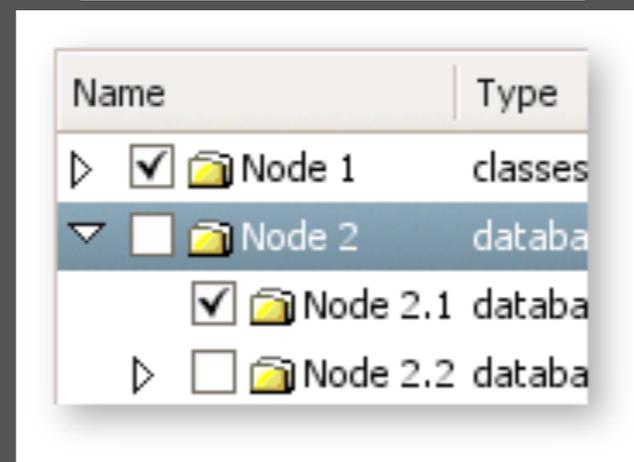
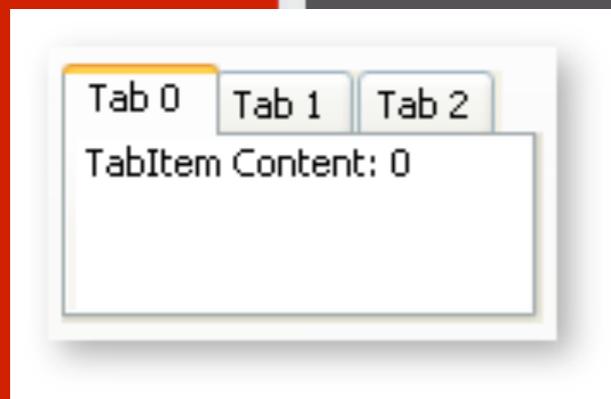
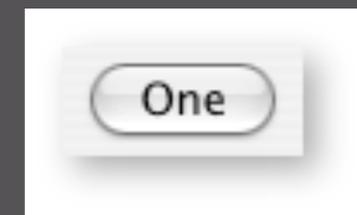
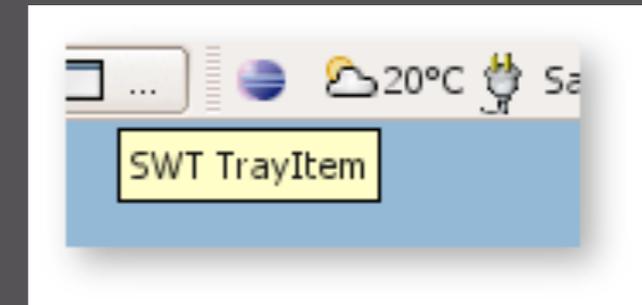
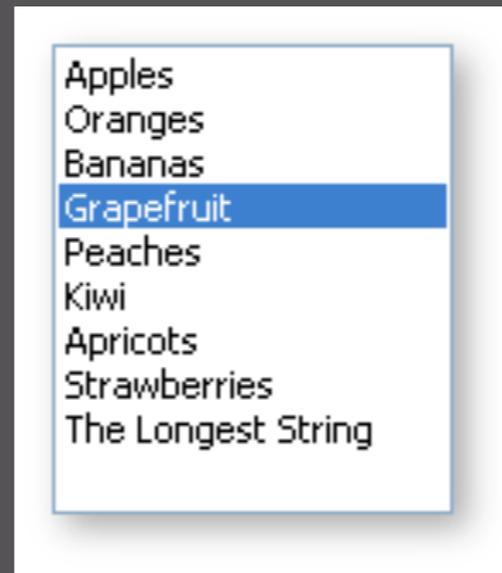
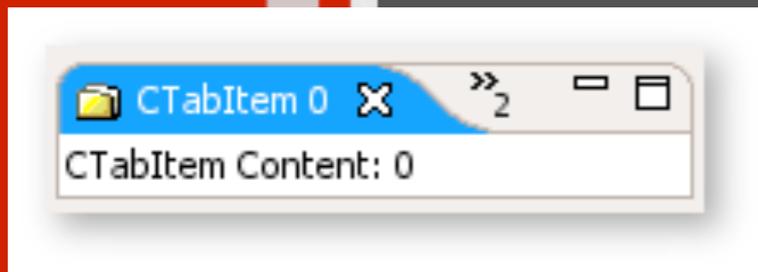


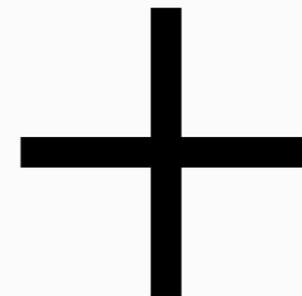


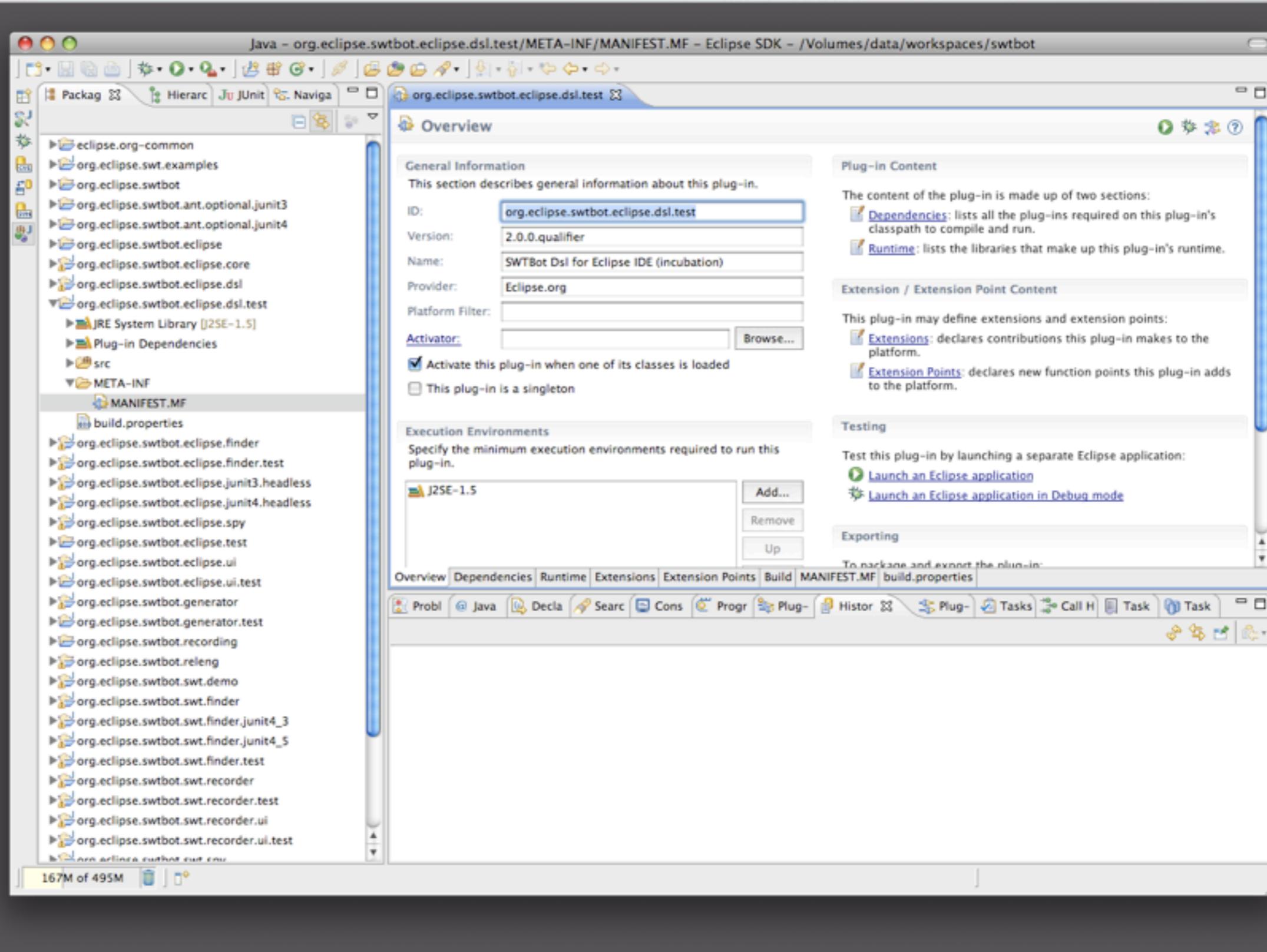
UI Test Automation with SWTBot

Ketan Padegaonkar, Code Monkey

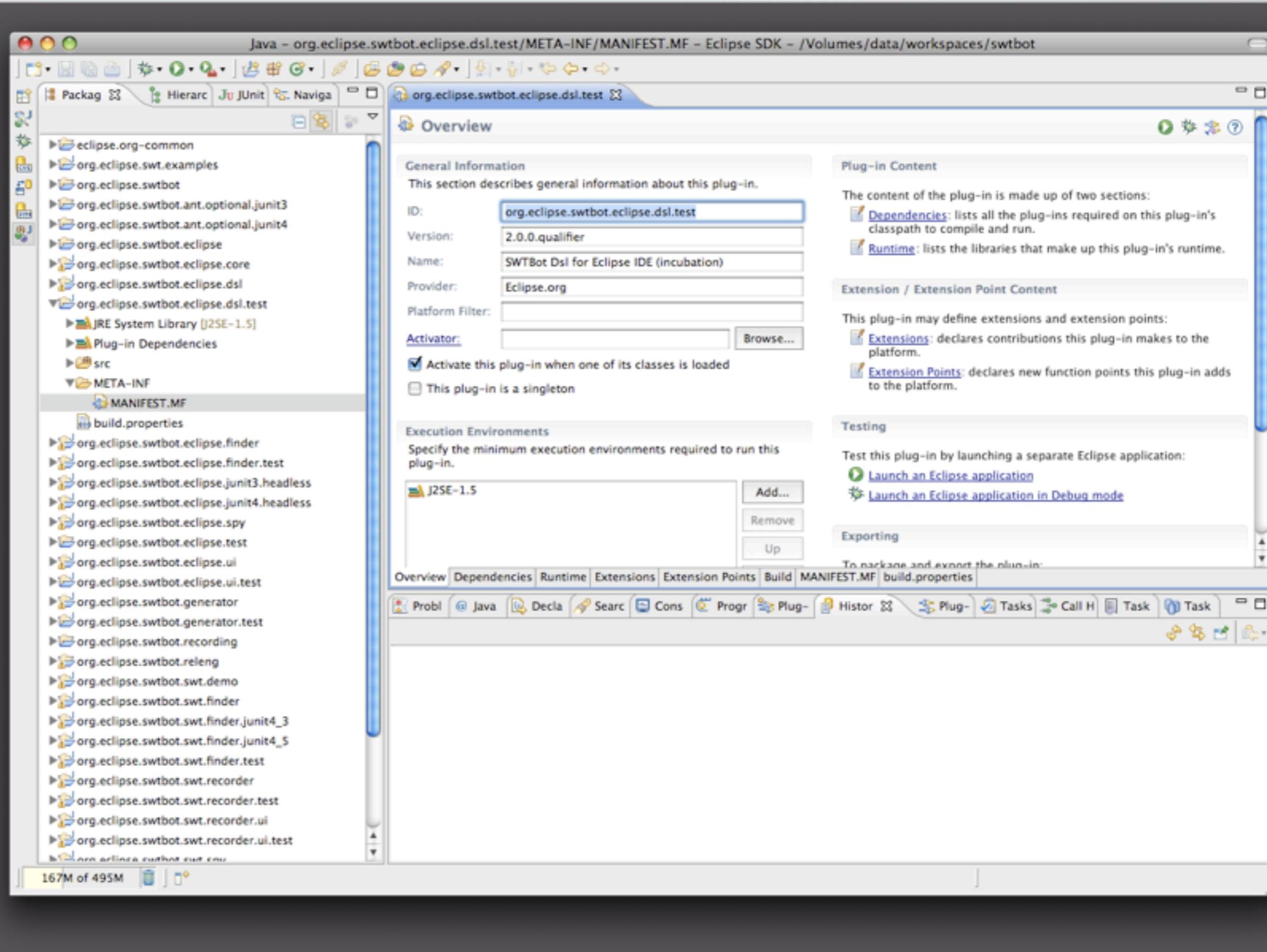
ThoughtWorks Studios.











The screenshot displays the LDAP Studio application window. The interface is divided into several panes:

- LDAP Browser:** Shows a directory tree with a hierarchy including `dc=example,dc=com`, `ou=Special Users`, `ou=Groups`, `ou=Netscape Servers`, `ou=People`, and `ou=system`. The `ou=People` entry is selected.
- Entry Editor:** Displays the details for the entry `uid=pcruse,ou=People,dc=example,dc=com`. It shows a table of attributes and their values:

Attribute Description	Value
objectclass	<i>inetOrgPerson (structural)</i>
objectclass	<i>organizationalPerson (structural)</i>
objectclass	<i>person (structural)</i>
objectclass	<i>top (abstract)</i>
cn	Patricia Cruse
sn	Cruse
facsimiletelephonenumber	+1 408 555 9751
givenname	Patricia
l	Santa Clara
mail	pcruse@example.com
ou	People
ou	Product Testing
roomnumber	3967
telephonenumber	+1 408 555 8641
uid	pcruse
userpassword	Plain text password
- Outline:** Shows a tree view of the entry's structure, including `objectclass`, `cn`, `sn`, `facsimiletelephonenumber`, `givenname`, `l`, `mail`, `ou`, `roomnumber`, `telephonenumber`, `uid`, and `userpassword`.
- Connections:** Shows a connection to `Apache DS 1.0`.
- Modification Logs:** Displays the following log entry:


```

                #!RESULT OK
                #!CONNECTION ldap://localhost:10389
                #!DATE 2007-02-12T14:37:32.810
                dn: uid=ahall,ou=People,dc=example,dc=com
                changetype: modify
                replace: l
                l: Santa Clara
            
```
- Progress:** Shows "No operations to display at this time".

EclipseCon 2009



The screenshot displays the Nomad PIM application interface with several panes:

- Event Details (Top Left):** Shows details for an event on 25.10.05 at 15:00. Description: "a finished event from today". A "finished" checkbox is checked.
- Event Details (Bottom Left):** Shows details for an event on 24.09.05 at 15:00. Description: "an event on another day". A "finished" checkbox is unchecked.
- Contact Details (Middle):** Shows details for "A. Friend". Name: "A. Friend". Description: "some additional information like the telephone number or the email address can be written down here". Birthday: "23.09.1964".
- Calendar (Top Right):** Shows a calendar for October 2005. The 25th is highlighted.
- Schedule (Bottom):** A table listing events:

date	type	description
22.09.05 16:00	event	another event
24.09.05 15:00	event	an event on another day
23.09.06	birthday	A. Friend (42 years)

The screenshot displays the Eclipse IDE interface with the following components:

- Navigator:** Shows a project structure for 'docs [qualicum.cs.ubc.ca]' with subfolders 'marketing', 'awards', 'media', and 'graphics'. The 'awards' folder contains '2008-01 Eclipse Awards.doc'. A 'Tasktop Logo.ai' file is also visible.
- Web Browser:** Displays the Eclipse Awards Guidelines page at http://www.eclipse.org/org/foundation/eclipseawards/technology_awards_guidelines.php. The page includes a navigation menu (Bugs, Articles, About Us, Foundation, Governance, Legal Resources, Contact Us), a 'Guidelines' section, and a 'Judging Criteria' section.
- Task List:** Lists several tasks with IDs and descriptions, such as '1778: make Tasktop nomi...', '1811: sketch business cases', and '1817: clarify key value prop...'. It also shows folders like '@Email - Tasktop' and 'All Mine - Marketing'.
- Calendar:** Shows a calendar for January 2008. The 25th is highlighted. Below the calendar, there are links for 'Outlook Calendar', 'Google Calendar', 'Refresh Events', and 'Configure...'. A task for 'Tasks: DUE - 1529: submit talks to Eclipse Foru... (All Day)' is also visible.



Challenges



Challenges

Identifying Controls



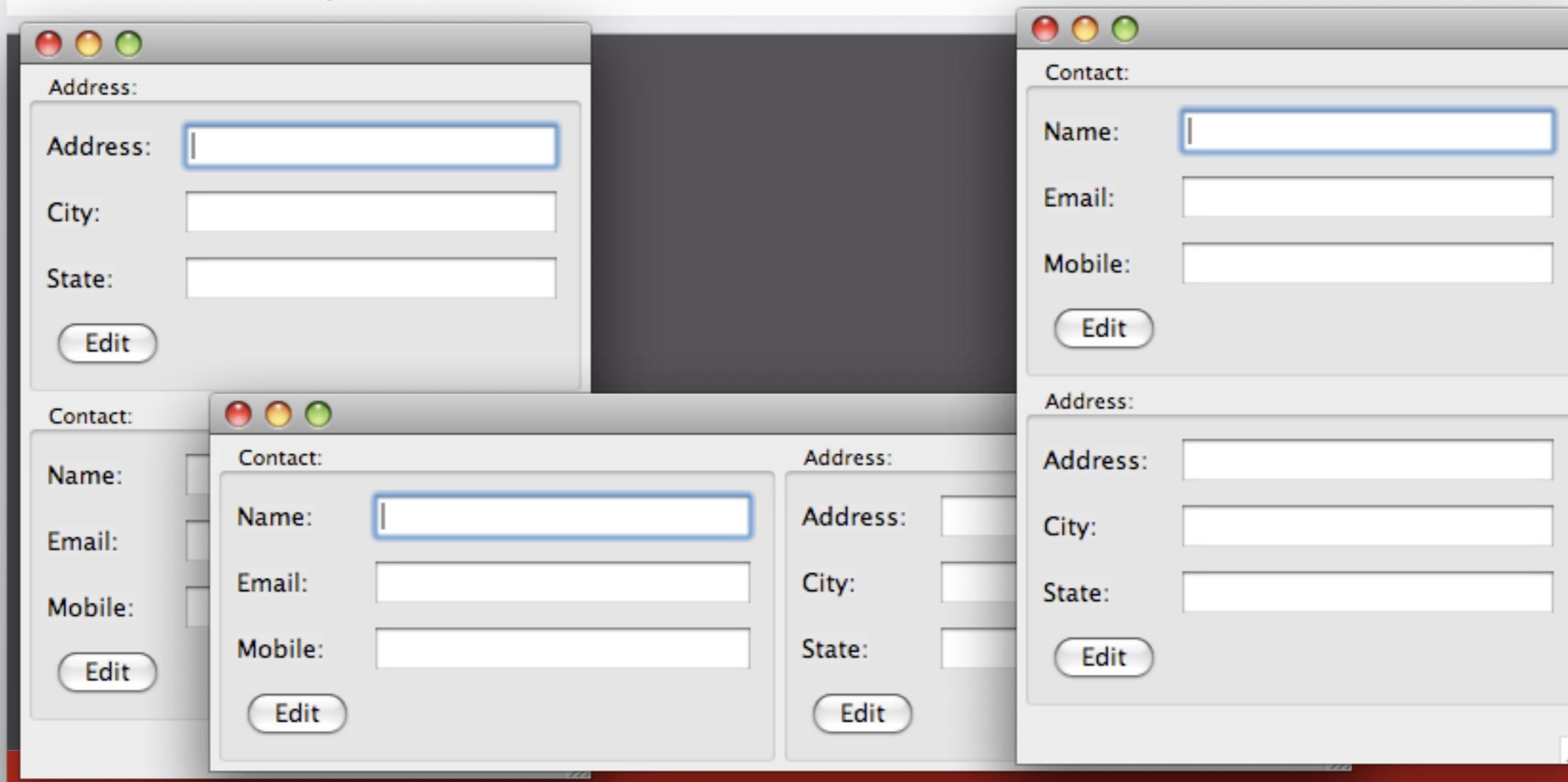
Challenges

Similar looking controls

Similar looking controls

The image shows a Java Swing dialog box with a standard Mac OS-style title bar (red, yellow, green buttons). The dialog is divided into two sections. The first section is titled "Address:" and contains three text input fields labeled "Address:", "City:", and "State:". Below these fields is an "Edit" button. The second section is titled "Contact:" and contains three text input fields labeled "Name:", "Email:", and "Mobile:". Below these fields is another "Edit" button. The dialog box is centered on a dark gray background.

Moving controls





Challenges

Sending “events” to controls



Challenges

Manage SWT Threading



Challenges

Tests to be non-blocking



Challenges

Run in a separate thread



Challenges

Run in a separate thread

Still manage synchronization between threads



Challenges

Multi threaded applications, background jobs



Challenges

Multi threaded applications, background jobs
“Non-deterministic” in amount of time required



Challenges

Internationalization (i18n) and Localization (L10n)



Challenges

Readability



Challenges

- ❑ Identifying controls
- ❑ Similar looking controls
- ❑ Moving controls
- ❑ Sending “events to controls”
- ❑ Manage SWT Threading
- ❑ Tests to be non-blocking
- ❑ Run in a separate thread, and manage synchronization
- ❑ Multi threaded applications, background jobs
- ❑ Internationalization (i18n) and Localization (L10n)
- ❑ Readability



Testing ?







Unit Testing





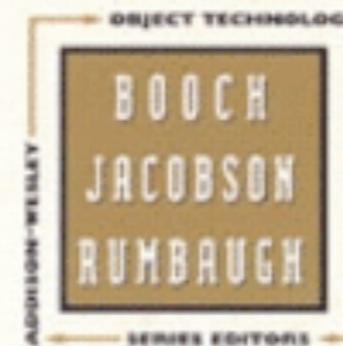
REFACTORING

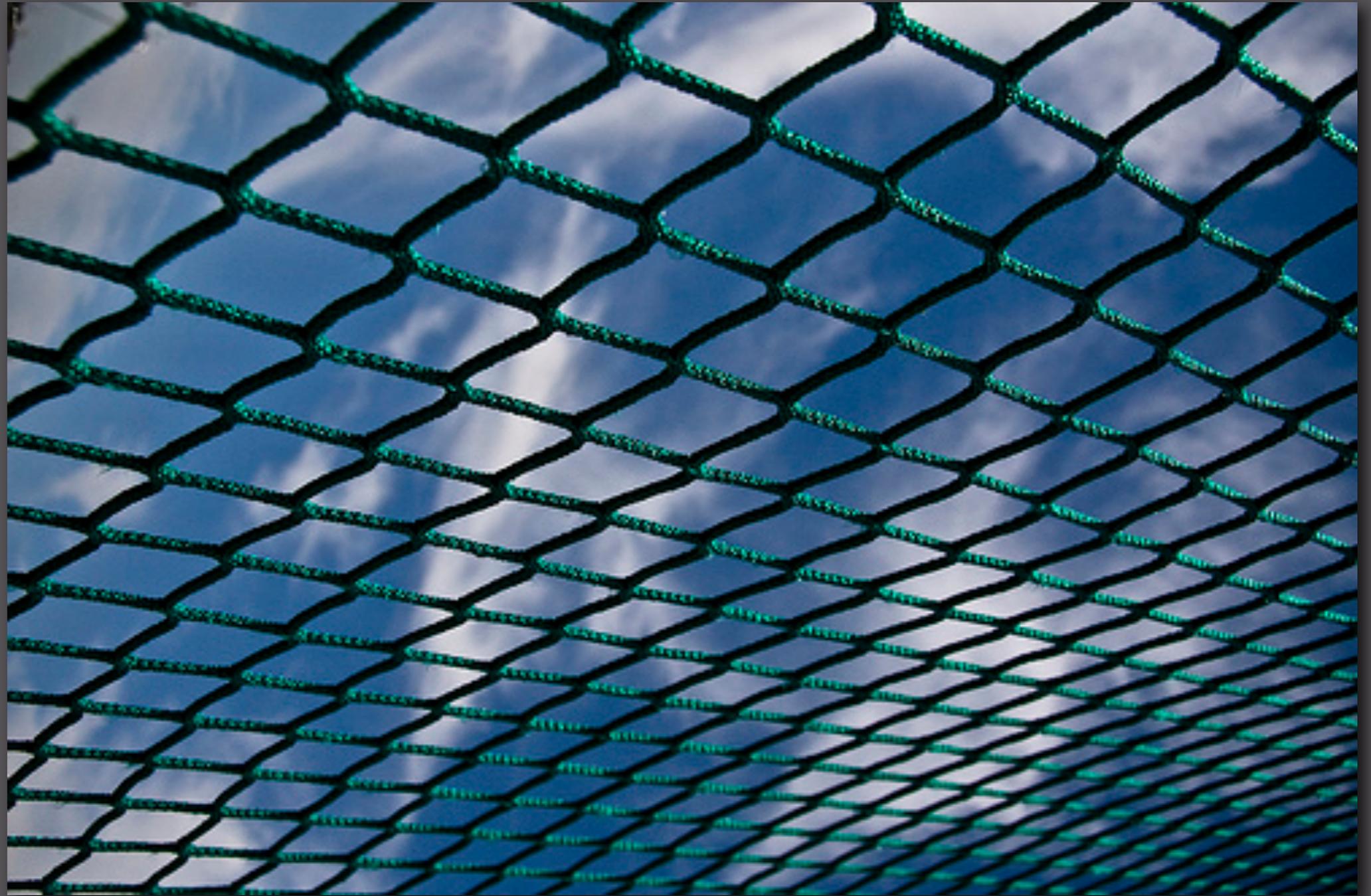
IMPROVING THE DESIGN
OF EXISTING CODE

MARTIN FOWLER

With Contributions by **Kent Beck, John Brant,
William Opdyke, and Don Roberts**

Foreword by **Erich Gamma**
Object Technology International Inc.





"red green refactor"

The screenshot displays the JUnit test runner interface within an IDE. At the top, a toolbar includes icons for 'Probl', '@ Java', 'Decla', 'Error', 'Search', 'Progr', 'Histor', 'Cons', 'Plug-', 'Book', 'Break', 'Tasks', 'Task', 'Plug-', 'JUnit', and 'ASTVI'. Below the toolbar, a status bar indicates 'Finished after 31.999 seconds'. The main area shows test results: 'Runs: 850/850', 'Errors: 0', and 'Failures: 0'. A green progress bar is visible above the test list. The test list contains the following entries:

- com.thoughtworks.twist.core.ParagraphTest [Runner: JUnit 3] (0.004 s)
- com.thoughtworks.twist.core.TagTest [Runner: JUnit 3] (0.002 s)
- com.thoughtworks.twist.core.CommentTest [Runner: JUnit 3] (0.540 s)
- com.thoughtworks.twist.recorder.core.NullTwistDriverTest [Runner: JUnit 3] (0.022 s)
- com.thoughtworks.twist.core.execution.java.JavaTest [Runner: JUnit 3] (0.678 s)
- com.thoughtworks.tide.core.utils.MathHelperTest [Runner: JUnit 3] (0.025 s)
- com.thoughtworks.twist.recorder.core.DefaultMultilineScriptGenerationStrategyTest [Runner: JUnit 3] (0.052 s)
- com.thoughtworks.twist.core.ScenarioSplitAndMergeTest [Runner: JUnit 3] (0.052 s)
- com.thoughtworks.twist.core.FixtureTest [Runner: JUnit 3] (0.001 s)
- com.thoughtworks.twist.core.ParameterTypeFactoryTest [Runner: JUnit 3] (0.011 s)
- com.thoughtworks.twist.core.parser.ListItemParserTest [Runner: JUnit 3] (0.004 s)
- com.thoughtworks.twist.core.execution.ant.ExecuteScenariosTaskTest [Runner: JUnit 3] (0.004 s)

On the right side, there is a 'Failure Trace' panel which is currently empty.



Functional Tests



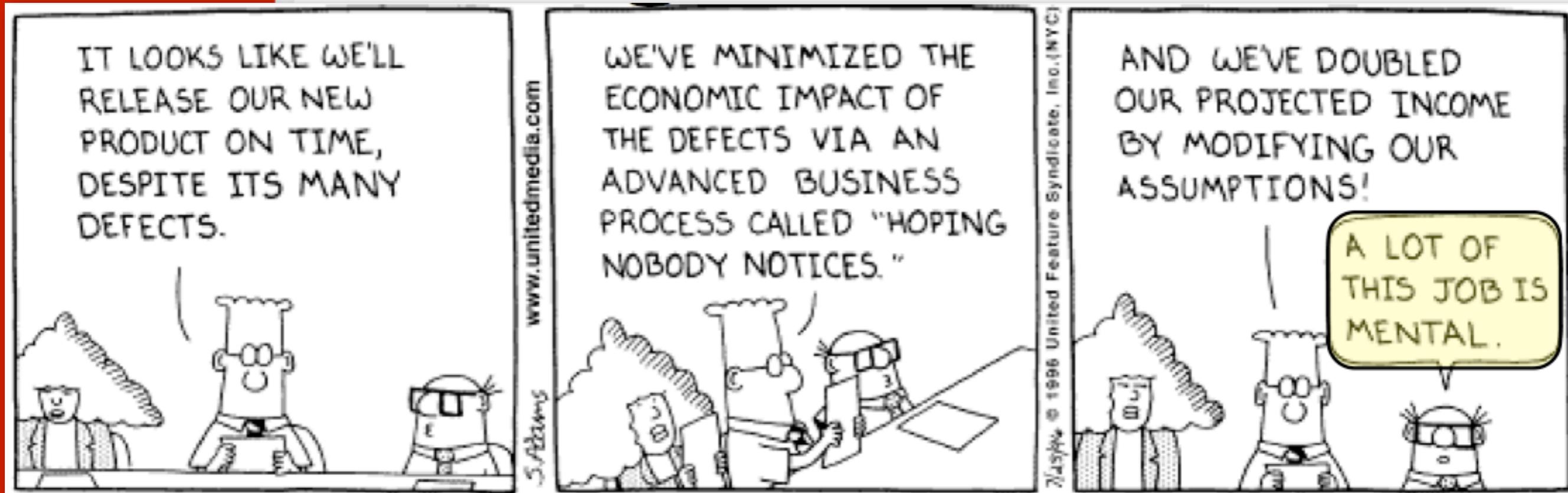








Writing SWT Tests





Understand SWT



Understand Threading





Quality Analysts/Testers



Quality Analysts/Testers









SWTBot



Agenda



- ■ setting up the environment
- ■ basic SWTBot API
 - ■ custom assertions
 - ■ analyze failures
- ■ how does it work?
 - ■ handling background jobs and long running operations
 - ■ thread safety
 - ■ improve performance
- ■ FluentAPI for common eclipse operations (DSL-ish)



Setting up the Environment



Setting up the Environment

- Eclipse 3.4
- SWTBot update site
 - <http://www.eclipse.org/swtbot/downloads.php>



Create a plugin project

- “org.eclipsecon.swtbot.example”

New Plug-in Project

Plug-in Project
Create a new plug-in project

Project name:

Use default location

Location:

Project Settings

Create a Java project

Source folder:

Output folder:

Target Platform

This plug-in is targeted to run with:

Eclipse version:

an OSGi framework:

Working sets

Add project to working sets

Working sets:



Setup Dependencies

- org.eclipse.ui
- org.eclipse.core.runtime
- org.eclipse.swtbot.eclipse.finder
- org.eclipse.swtbot.junit4_x
- org.eclipse.swtbot.swt.finder
- org.junit4
- org.hamcrest



org.eclipsecon.swtbot.example

Dependencies

Required Plug-ins ↓ a z

Specify the list of plug-ins required for the operation of this plug-in.

- org.eclipse.ui
- org.eclipse.core.runtime
- org.eclipse.swtbot.eclipse.finder
- org.eclipse.swtbot.junit4_x
- org.eclipse.swtbot.swt.finder
- org.junit4
- org.hamcrest

Total: 7

Imported Packages

Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.

Total: 0

Automated Management of Dependencies ↓ a z

Dependency Analysis

Overview | Dependencies | Runtime | Extensions | Extension Points | Build | MANIFEST.MF | build.properties



Basic SWTBot API



The first **red** and **green** bar

A “hello world” test!



Setup for the test

- close the “Welcome Page”



Create tests for

- creating a java project “MyFirstProject”
- creating a java class “org.eclipsecon.project.HelloWorld”
- type in a program that prints “Hello, World”
- execute the program
- verify that the program printed “Hello, World”

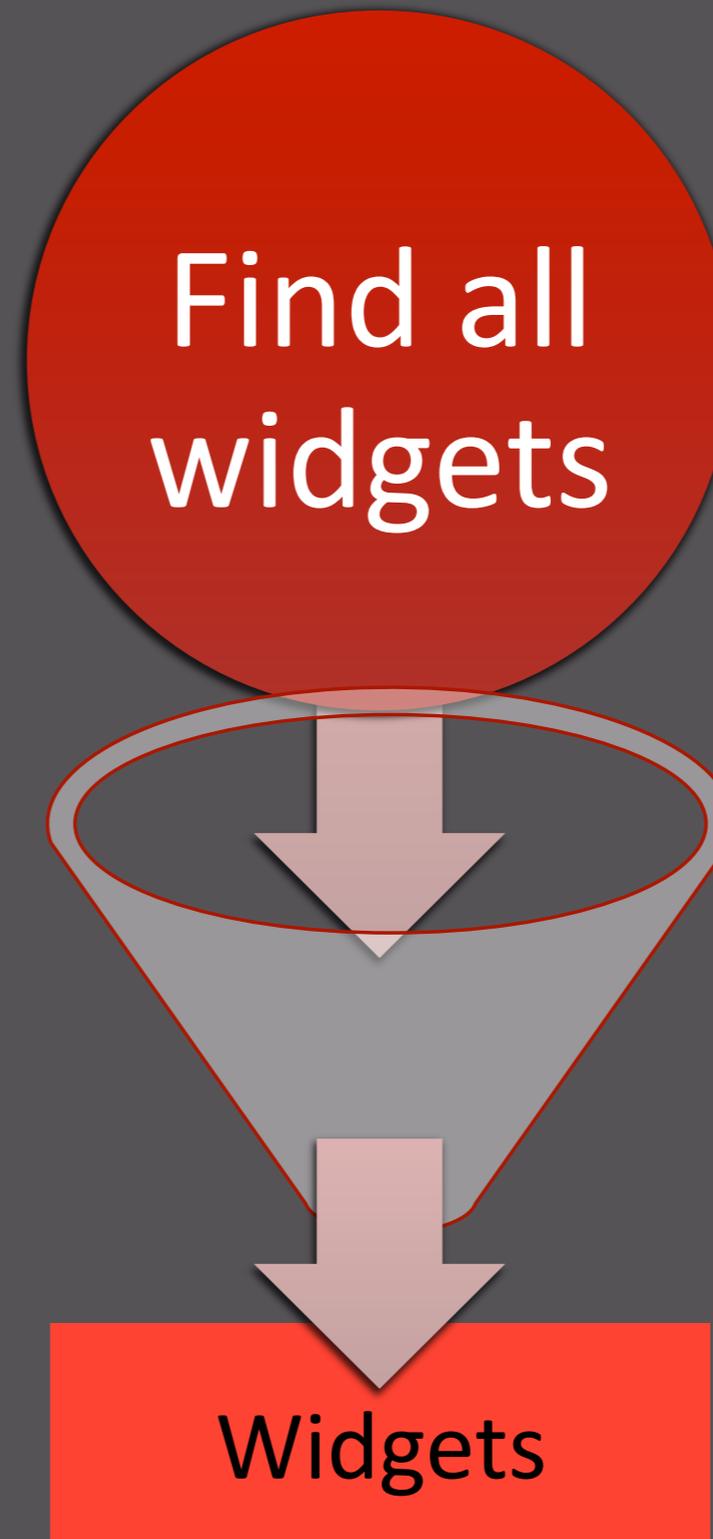


Teardown for the test

- delete the project



How does it work?





Redundancy and Failure Proofing





Find all widgets

- Depth first traversal of UI elements

1. Find top level widgets
 1. Find children of each widget
2. For each child do (1) and (2)



Creating matchers(simple)

- `withText("Finish")`
- `withLabel("Username:")`
- `withRegex("Proceed to step (.*)")`
- `widgetOfType(Button.class)`
- `withStyle(SWT.ARROW, "SWT.ARROW")`



...Creating matchers(combination)

- `allOf(matchers...)`
- `anyOf(matchers...)`
- `not(matcher)`
- `allOf(anyOf(matchers...), matchers...)`



Handling long running operations

- describe a condition
- poll for the condition at intervals
- wait for it to evaluate to **true** or **false**
- of course there's a timeout



Handling Waits(SWTBot.java)

```
private void waitUntil(ICondition condition, long timeout, long interval) {  
  
    long limit = System.currentTimeMillis() + timeout;  
    condition.init((SWTBot) this);  
    while (true) {  
        try {  
            if (condition.test())  
                return;  
        } catch (Throwable e) {  
            // do nothing  
        }  
        sleep(interval);  
        if (System.currentTimeMillis() > limit)  
            throw new TimeoutException("Timeout after: " + timeout);  
    }  
}
```



The whole thing put together

The End User API



Finding widgets (SWTBot.java)

```
public SWTBotTree treeWithLabelInGroup(String l, String g, int i) {  
  
    // create the matcher  
    Matcher matcher =  
        allOf(  
            widgetOfType(Tree.class), withLabel(l), inGroup(g)  
        );  
    // find the widget, with redundancy built in  
    Tree tree = (Tree) widget(matcher, index);  
    // create a wrapper for thread safety  
    // and convenience APIs  
    return new SWTBotTree(tree, matcher);  
}
```



Thread Safety

- Tests should run in non-ui thread
- query state of a widget
- change state of a widget



Thread Safety (Query state)

```
public class SWTBotCheckBox {
    public boolean isChecked() {
        // evaluate a result on the UI thread
        return syncExec(new BooleanResult() {
            public Boolean run() {
                return widget.getSelection();
            }
        });
    }
}
```



Thread Safety(change state)

```
public class SWTBotCheckBox {
    public void select() {
        asyncExec(new VoidResult() {
            public void run() {
                widget.setSelection(true);
            }
        });
        notifyListeners();
    }

    protected void notifyListeners() {
        notify(SWT.MouseDown);
        notify(SWT.MouseUp);
        notify(SWT.Selection);
    }
}
```



Building Abstractions



Features and capabilities of tests (Page Objects)

- Project Explorer
- The Editor
- The Console View
- The main menu bar, tool bar



Features and capabilities of tests (Domain Objects)

- Create a project
- Delete a project
- Create a class
- Execute a class
- more...



Page Objects

<http://code.google.com/p/webdriver/wiki/PageObjects>



Page Objects... should

- Represent the services offered by the page to the test developer
- Internally knows the details about how these services are offered and the details of UI elements that offer them
- Return other page objects to model the user's journey through the application
- Different results of the same operation modeled differently



Page Objects... should not

- Expose details about user interface elements
- Make assertions about the state of the UI



Page Objects (implementation)

```
public class LoginPage {  
  
    public HomePage loginAs(String user, String pass) {  
        // ... clever magic happens here  
    }  
  
    public LoginPage loginAsExpectingError(String user, String pass) {  
        // ... failed login here, maybe because one or both of  
        // username and password are wrong  
    }  
  
    public String getErrorMessage() {  
        // So we can verify that the correct error is shown  
    }  
}
```



Page Objects (usage)

```
// the bad test
public void testMessagesAreReadOrUnread() {
    Inbox inbox = new Inbox(driver);
    inbox.assertMessageWithSubjectIsUnread("I like cheese");
    inbox.assertMessageWithSubjectIsNotUndread("I'm not fond of tofu");
}

// the good test
public void testMessagesAreReadOrUnread() {
    Inbox inbox = new Inbox(driver);
    assertTrue(inbox.isMessageWithSubjectIsUnread("I like cheese"));
    assertFalse(inbox.isMessageWithSubjectIsUnread("I'm not fond of tofu"));
}
```



```
LoginPage login = new LoginPage();
HomePage home = login.loginAs("username", "secret");
SearchPage search = home.searchFor("swtbot");
assertTrue(search.containsResult("http://eclipse.org/swtbot"));
```



Exercise: Page Objects

- Refactor the tests in the form of a PageObject



Domain Objects



Domain Objects... should

- Represent the operations that can be performed on concepts



Domain Objects

```
public class JavaProject {
    public JavaProject create(String projectName){
        // create a project and return it
    }
    public JavaProject delete(){
        // delete the project and return it
    }
    public JavaClass createClass(String className){
        // create a class and return it
    }
}
```



Going ahead

- Commonly used functionality bundled as convenience API
- Eclipse Forms
- GEF!
- Use “real” events instead of “mocks”



Questions ?

newsgroup: news://news.eclipse.org/eclipse.swtbot

web: eclipse.org/swtbot

gmail: KetanPadegaonkar



Resources

- ❑ <http://flickr.com/photos/stuart100/288880576/>
- ❑ <http://flickr.com/photos/60373916@N00/229233928/>
- ❑ <http://www.flickr.com/photos/jfravel/1001472806/>
- ❑ <http://www.flickr.com/photos/54323936@N00/245650981/>
- ❑ <http://www.flickr.com/photos/aknacer/3196381450/>
- ❑ <http://www.flickr.com/photos/buttersweet/33684613/>
- ❑ <http://www.flickr.com/photos/pintuck/283795079/>
- ❑ <http://www.flickr.com/photos/o3bor/209925927/>
- ❑ <http://www.flickr.com/photos/leoniewise/3369871669/>