

# Eclipse Basics

(with Eclipse Juno for Java)

Ming-Hsien Tsai

# Eclipse

- <http://www.eclipse.org>
- Integrated development environment (IDE)
- Extensible with plugins

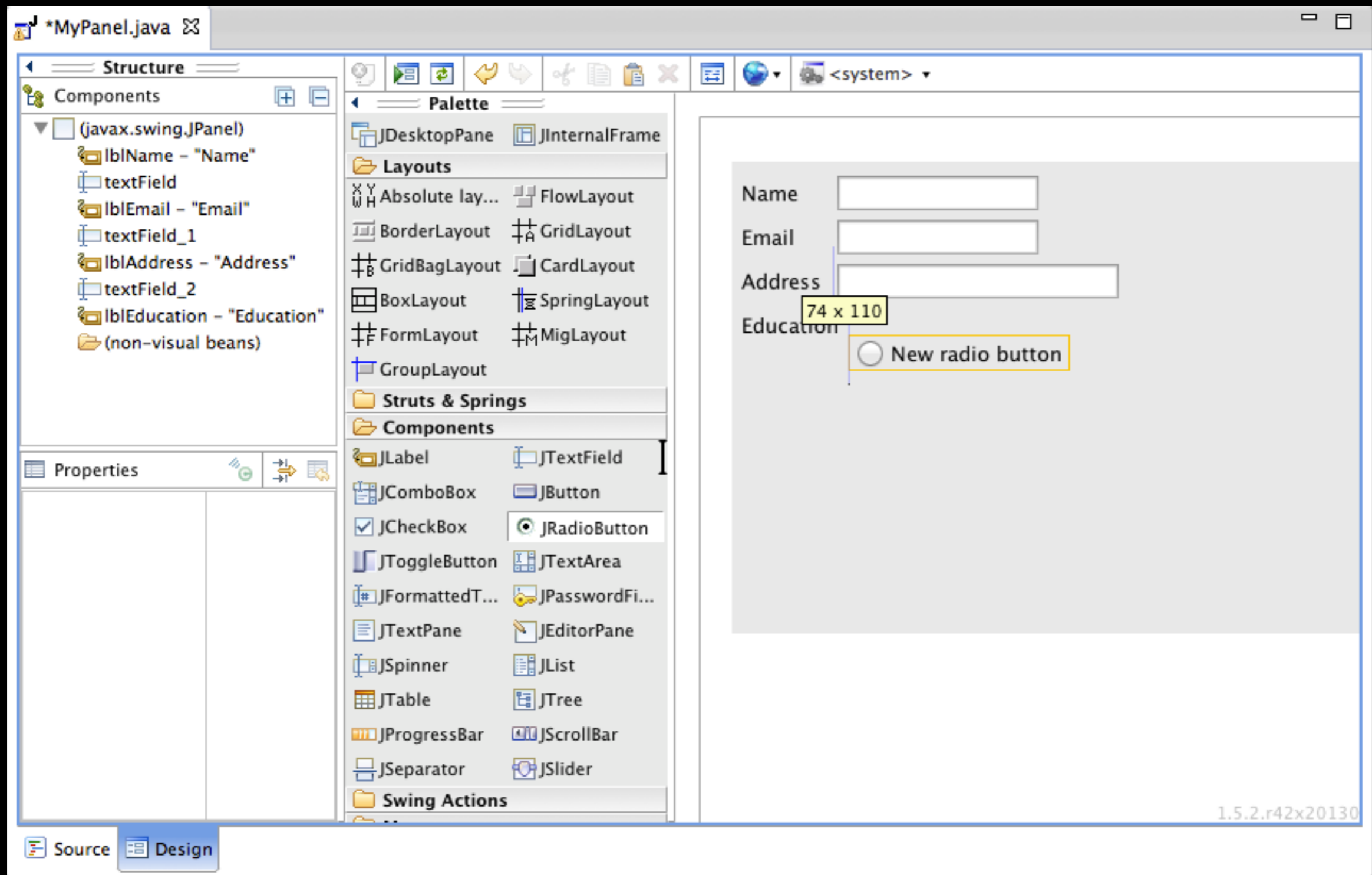
# Without IDE

obj.???

obj.func(???)

```
add(comp1, BorderLayout.NORTH);  
add(comp2, BorderLayout.CENTER);  
cs.weightx = 1;  
comp2.add(comp3, cs);  
cs.weightx = 2;  
comp2.add(comp4, cs);
```

# With IDE

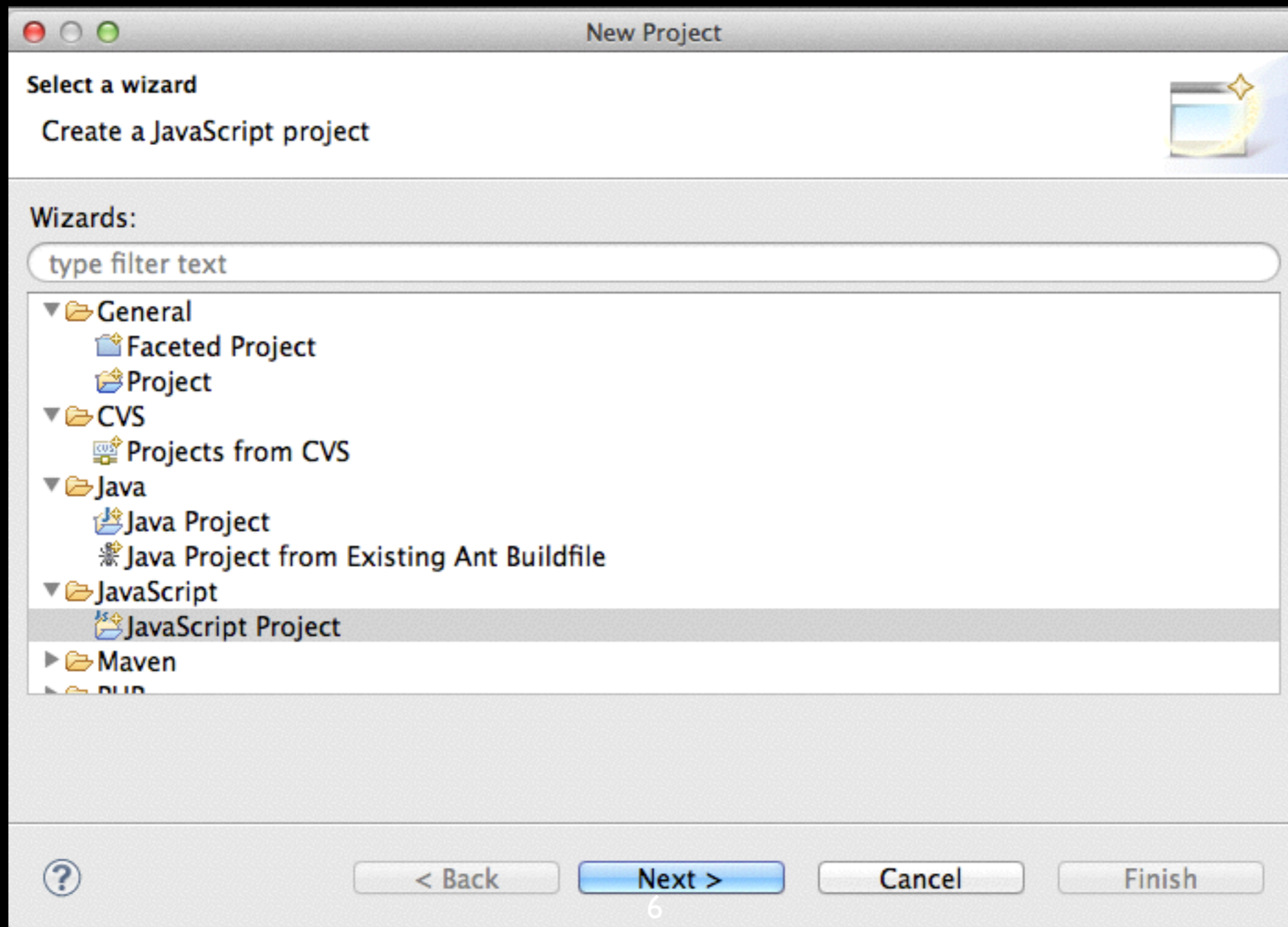


# Workspace

- Where your projects are stored
- Multiple workspaces are allowed

# Create New Project

*File / New / Project...*



# Perspective - Java

The screenshot displays the Eclipse IDE interface in Perspective view. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The title bar shows the current file: Java - GOAL2/plugins/org.svrl.goal.core/source/org.svrl.goal.core/aut/Automaton.java - Eclipse - /Users/mht208/Documents/workspace. The main workspace is divided into three panes:

- Package Explorer (Left):** Shows the project structure. The current project is GOAL2 [GOAL2 develop], with a sub-project plugins/org.svrl.goal.core. The file Automaton.java is selected under the org.svrl.goal.core.aut package.
- Code Editor (Center):** Displays the source code for Automaton.java. The method `renamePropositions` is visible, with line numbers 2455 to 2480. The code includes a Javadoc comment and the implementation of the method.
- Outline (Right):** Lists the methods and classes in the current file. The `renamePropositions` method is highlighted.

The bottom status bar shows various toolbars and indicators, including Problems, Javadoc, Declaration, Search, Console, Progress, History, Git Repositories, Tasks, Call Hierarchy, and LogCat. The status bar also displays the current line and column: 2462 - 18.

```
2455     }
2456   }
2457
2458   /**
2459    * Renames simultaneously the propositions/symbols according to a specified
2460    * substitution mapping.
2461    *
2462    * @param sub
2463    *       a mapping from old proposition names to new literals
2464    * @throws IllegalArgumentException
2465    *       if this automaton does not have the old propositions, or a
2466    *       proposition is renamed to an existing proposition
2467    */
2468   public void renamePropositions(Map<String, String> sub) {
2469     /* Format the literals first. */
2470     Map<String, String> m = new HashMap<String, String>();
2471     for (String key : sub.keySet()) {
2472       try {
2473         key = atype.formatLabel(key);
2474         String value = atype.formatLabel(sub.get(key));
2475         /* Skip the substitution for literals that map to themselves. */
2476         if (!key.equals(value))
2477           m.put(key, value);
2478       } catch (IllegalArgumentException e) {
2479       }
2480     }
```

# Perspective - Browsing

The screenshot displays the Eclipse IDE in the Java Browsing perspective. The main window shows the source code for `CPALib.java`. The interface includes several panels: **Projects** (left), **Packages** (top-left), **Types** (top-right), and **Members** (right). The **Members** panel lists fields like `ERROR_OUTPUT`, `ERROR_EXIT_CODE`, and `cpachecker`, and a method `initialize(String[])`. The code editor shows the `initialize` method implementation, which handles command-line arguments and configuration.

```
62 static ShutdownRequestListener forcedExitOnShutdown = null;
63
64 @SuppressWarnings("resource")
65 // We don't close LogManager
66 public static void initialize(String[] args) {
67     // initialize various components
68     Configuration cpaConfig = null;
69     LogManager logManager = null;
70     try {
71         try {
72             Pair<Configuration, String> p = createConfiguration(args);
73             cpaConfig = p.getFirst();
74         } catch (InvalidCmdlineArgumentException e) {
75             ERROR_OUTPUT.println("Could not process command line arguments: " + e.getMessage());
76             System.exit(ERROR_EXIT_CODE);
77         } catch (IOException e) {
78             ERROR_OUTPUT.println("Could not read config file " + e.getMessage());
79             System.exit(ERROR_EXIT_CODE);
80         }
81
82         logManager = new BasicLogManager(cpaConfig);
83
84     } catch (InvalidConfigurationException e) {
85         ERROR_OUTPUT.println("Invalid configuration: " + e.getMessage());
86         System.exit(ERROR_EXIT_CODE);
87         return;
88     }
89     cpaConfig.enableLogging(logManager);
```

At the bottom of the IDE, the status bar shows "Writable", "Smart Insert", and "69 : 13".



# Perspective - Debug

The screenshot shows the Eclipse IDE in the Debug perspective. The main editor displays the source code for `CPALib.java`. The code includes a static field `forcedExitOnShutdown` and a static method `initialize` that sets up configuration and logging. The `initialize` method uses `createConfiguration` to parse command-line arguments. The `Outline` view on the right shows the class hierarchy for `org.sosy_lab.cpagechecker.cmdline`, including `CPALib`, `BootstrapOptions`, and `MainOptions`, along with various static fields and methods.

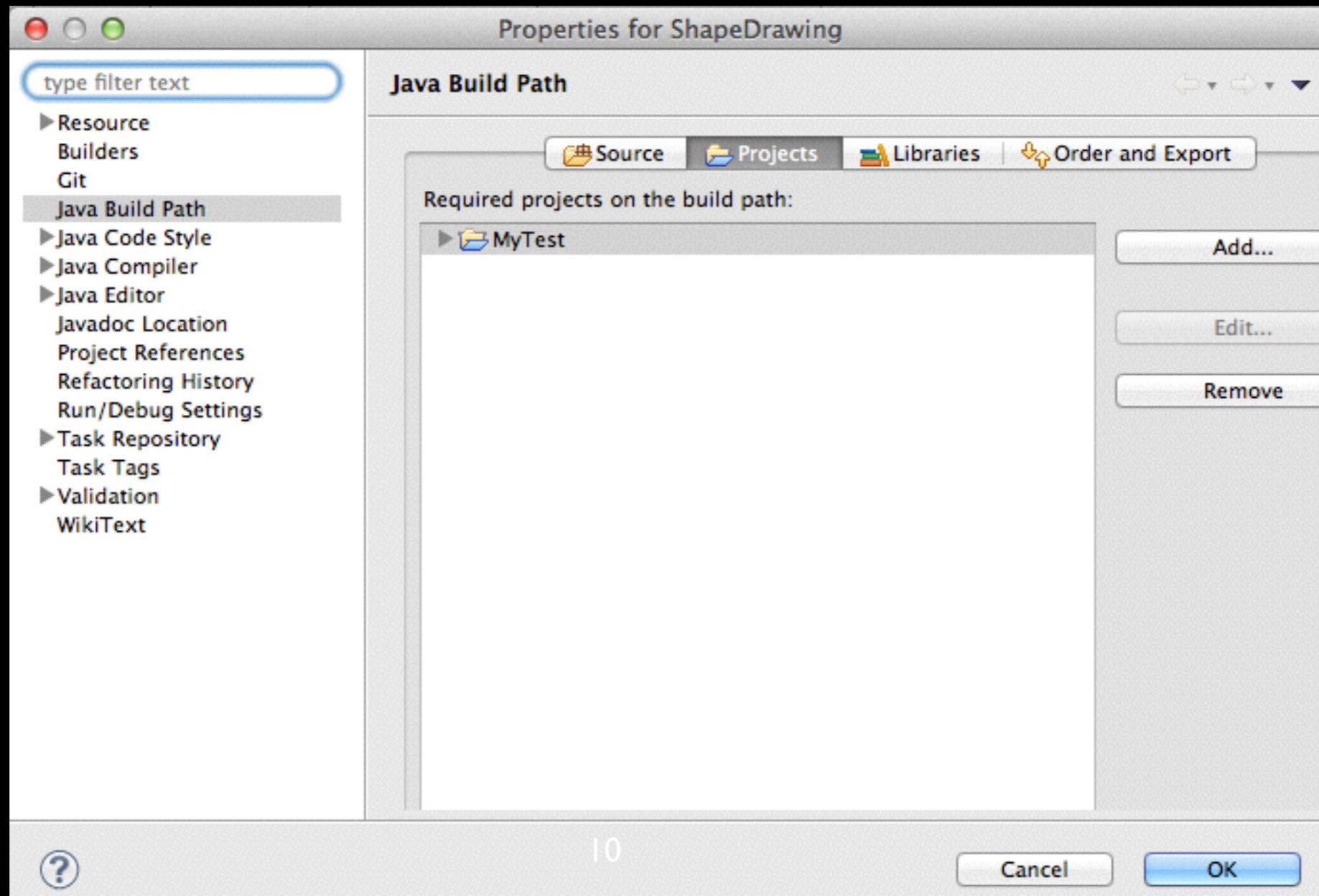
```
62 static ShutdownRequestListener forcedExitOnShutdown = null;
63
64 @SuppressWarnings("resource")
65 // We don't close LogManager
66 public static void initialize(String[] args) {
67     // initialize various components
68     Configuration cpaConfig = null;
69     LogManager logManager = null;
70     try {
71         try {
72             Pair<Configuration, String> p = createConfiguration(args);
73             cpaConfig = p.getFirst();
74         } catch (InvalidCmdlineArgumentException e) {
75             ERROR_OUTPUT.println("Could not process command line arguments: " + e.getMessage());
76         }
77     }
78 }
```

org.sosy\_lab.cpagechecker.cmdline

- CPALib
  - BootstrapOptions
  - MainOptions
    - cpachecker : CPAchecker
    - ERROR\_EXIT\_CODE : int
    - ERROR\_OUTPUT : PrintStream
    - forcedExitOnShutdown : ShutdownRequestListener
    - shutdownHook : ShutdownHook
    - shutdownNotifier : ShutdownNotifier
    - createConfiguration(String[]): Pair<Configuration, String>

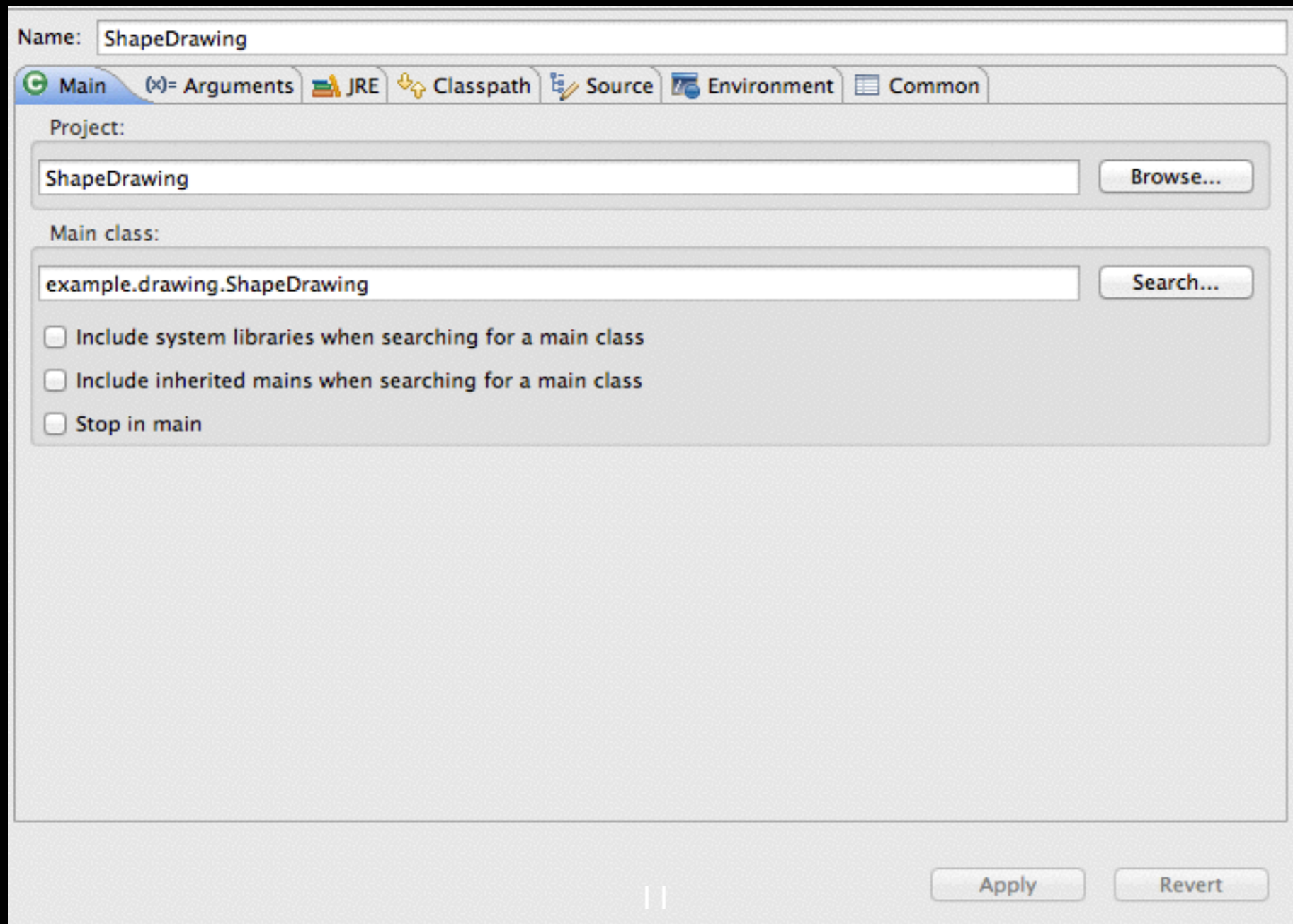
# Project Reference

*Project popup menu / Properties / Java Build Path / Projects / Add...*



# Run Your Application

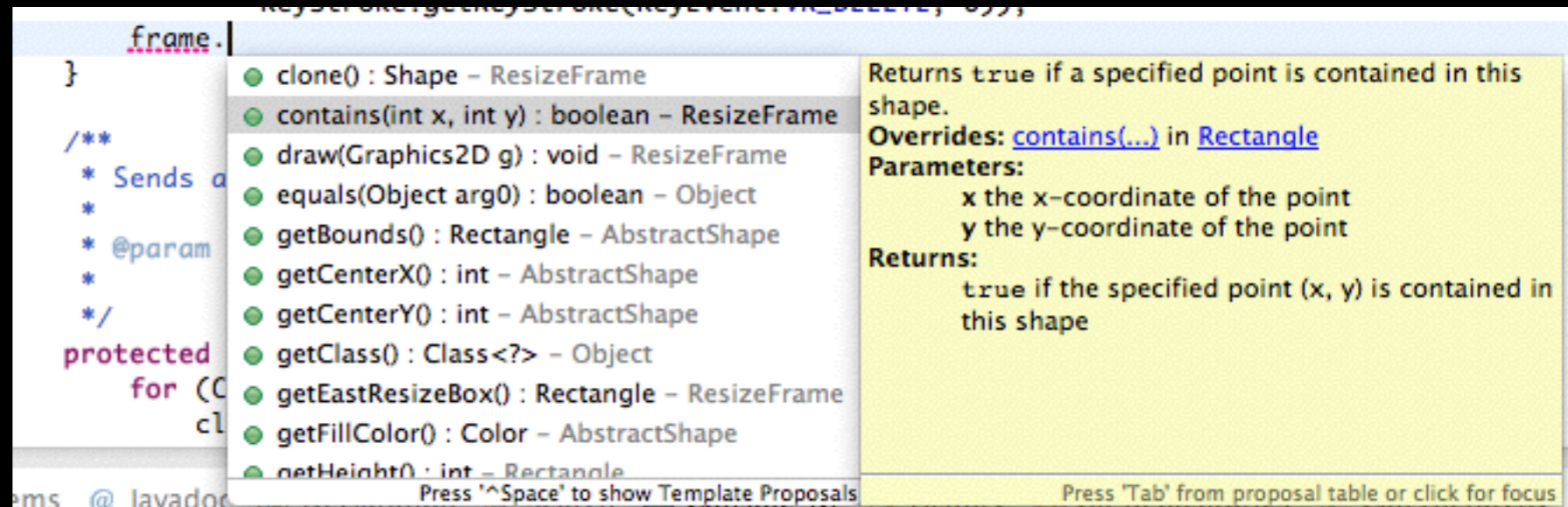
*Run / Run Configurations...*



# Java Doc

⌘⇧J Alt-Shift-J

/\*\*



/\*\*

\* Returns {@code true} if a specified point is contained in this shape.

\*

\* @param x

\*       the x-coordinate of the point

\* @param y

\*       the y-coordinate of the point

\* @return {@code true} if the specified point (x, y) is contained in this

\*       shape

\*/

public boolean contains(int x, int y) {

# Java Doc - Tags

@author <NAME>

@version <VERSION>

@param <VARIABLE> <DESCRIPTION>

@return <DESCRIPTION>

@deprecated <DESCRIPTION>

@since <VERSION>

@throws <EXCEPTION> <DESCRIPTION>

@exception <EXCEPTION> <DESCRIPTION>

@see <CLASSPATH>

...

# Java Doc - Export

*File / Export / Java / Javadoc*

[All Classes](#)

Packages

- [example.drawing](#)
- [example.drawing.action](#)
- [example.drawing.io](#)
- [example.drawing.menu](#)
- [example.drawing.preference](#)
- [example.drawing.shape](#)
- [undo](#)

---

[PreferenceDialog](#)

[PreviousWindowAction](#)

[Properties](#)

[Rectangle](#)

[RectangleBrush](#)

[RectangleCodec](#)

[RedoAction](#)

[ResizeFrame](#)

[ResizeShapeEdit](#)

[SaveAction](#)

[SelectTool](#)

[Shape](#)

[ShapeDrawing](#)

[Star](#)

[StarBrush](#)

[StarCodec](#)

[StarPolygon](#)

[StarPolygonBrush](#)

[StarPolygonCodec](#)

[StarPolygonOptionsPanel](#)

[ToolBar](#)

[UIDialog](#)

[UndoAction](#)

[Util](#)

[Window](#)

[WindowMenu](#)

[XMLUtil](#)

[Overview](#) [Package](#) **[Class](#)** [Use Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

---

example.drawing.shape

## Interface Shape

**All Superinterfaces:**

- [java.lang.Cloneable](#)

**All Known Implementing Classes:**

- [AbstractShape](#), [Ellipse](#), [Rectangle](#), [ResizeFrame](#), [Star](#), [StarPolygon](#)

---

```
public interface Shape
extends java.lang.Cloneable
```

A shape is an object that can be drawn on a canvas. Every shape must be enclosed by a minimal rectangle, called frame. The location and the size of a shape may be adjusted by changing its frame. The following additional properties are defined for all shapes: line width, line color, and fill color. Note that not all the predefined properties are used by all shapes.

**Author:**

"Ming-Hsien Tsai"

---

### Method Summary

<a href="#">Shape</a>	<a href="#">clone()</a> Makes a clone of this shape.
boolean	<a href="#">contains(int x, int y)</a> Returns true if a specified point is contained in this shape.

# Other Documentation Generators

- Oxygen
  - C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl
- Sphinx
  - Python, C/C++
- ScalaDoc
- ocamlDoc

More generators can be found in [https://en.wikipedia.org/wiki/Comparison\\_of\\_documentation\\_generators](https://en.wikipedia.org/wiki/Comparison_of_documentation_generators)

# Code Generation

Getters/Setters:

*Source / Generate Getters and Setters...*

Override/Implement:

*Source / Overwrite/Implement Methods...*

...



# Navigation

*Navigate / Open Declaration*

**F3**

*Navigate / Open Type Hierarchy*

**F4**

*Navigate / Open Call Hierarchy*

**^⌘H    Ctrl-Alt-H**

# Search

*Search / References / Workspace*

⬆️ ⌘G    **Ctrl-Shift-G**

# Source

*Source / Format*

⌘⇧F    **Ctrl-Shift-F**

*Source / Organize Imports*

⌘⇧O    **Ctrl-Shift-O**

*Source / Toggle Comment*

⌘/    **Ctrl-/**

# Refactor

*Refactor / Rename...*

**⌘⇧R**    **Alt-Shift-R**

*Refactor / Move...*

**⌘⇧V**    **Alt-Shift-V**

*Source / Toggle Comment*

**⌘/**    **Ctrl-/**

# Others

Quick Fix:

⌘1 **Ctrl-1**

Shortcuts reference:

⇧⌘L **Shift-Ctrl-L**

# Build Tools

- GNU Make
- Apache Ant with Ivy
- Apache Maven
- Gradle

# Other Languages

- Eclipse CDT for C/C++
  - <http://www.eclipse.org/cdt/>
- Eclipse PDT for PHP
  - <http://projects.eclipse.org/projects/tools.pdt>
- Eclipse JSDT for Javascript
  - <http://www.eclipse.org/webtools/jsdt/>
- PyDev for Python
  - <http://marketplace.eclipse.org/content/pydev-python-ide-eclipse/metrics#.UkJQuxY5SfQ>
- Scala IDE for Scala
  - <http://scala-ide.org>

# Other Features

(may need third-party plugins)

- Debugging
- UML diagrams and code generation
  - UML Designer, UML to Java code generator
- Task management
  - Mylyn
- Issue tracking
  - Bugzilla, JIRA, Redmine, ...



# Other Features

(may need third-party plugins)

- Continuous integration
  - Eclipse Hudson
- Program verification
  - Java PathFinder, Leon, EpiSpin