Running Eclipse

After installing the Eclipse SDK in a directory, you can start the Workbench by running the Eclipse executable included with the release (you also need a Java SE 5 JRE, not included with the Eclipse SDK). On Windows, the executable file is calledeclipse.exe, and is located in the `eclipse` sub-directory of the install. If installed at `c:\eclipse-SDK-4.1-win32`, the executable is `c:\eclipse-SDK-4.1-win32\eclipse\eclipse.exe`. **Note:** Set-up on most other operating environments is analogous. Special instructions for Mac OS X are listed below.

Allocating enough memory and solving OutOfMemoryErrors

By default, Eclipse will allocate up to 384 megabytes of Java heap memory. This should be ample for all typical development tasks. However, depending on the JRE that you are running, the number of additional plug-ins you are using, and the number of files you will be working with, you could conceivably have to increase this amount. Eclipse allows you to pass arguments directly to the Java VM using the `-vmargs` command line argument, which must follow all other Eclipse specific arguments. Thus, to increase the available heap memory, you would typically use:

```
eclipse -vmargs -Xmx<memory size>
```

with the `<memory size>` value set to greater than "384M" (384 megabytes -- the default).

When using an Oracle (Sun) VM, you may also need to increase the size of the permanent generation memory. The default maximum is 64 megabytes, but more may be needed depending on your plug-in configuration and use. When the VM runs out of permanent generation memory, it may crash or hang during class loading. This failure is less common when using Sun JRE version 1.5.0_07 or greater. The maximum permanent generation size is increased using the -XX:MaxPermSize=<memory size> argument:

```
eclipse -vmargs -XX:MaxPermSize=<memory size>
```

This argument may not be available for all VM versions and platforms; consult your VM documentation for more details.

Note that setting memory sizes to be larger than the amount of available physical memory on your machine will cause Java to "thrash" as it copies objects back and forth to virtual memory, which will severely degrade your performance.

Selecting a workspace

When the Workbench is launched, the first thing you see is a dialog that allows you to select where the workspace will be located. The workspace is the directory where your work will be stored. If you do not specify otherwise, Eclipse creates the workspace in your user directory. This workspace directory is used as the default content area for your projects as well as for holding any required metadata. For shared or multi-workspace installs you must explicitly specify the location for your workspace using the dialog (or via the "`-data`" command line argument).

Specifying the Java virtual machine

Here is a typical Eclipse command line:

```
eclipse -vm c:\jdk5u22\jre\bin\javaw
```

Tip: It's generally a good idea to explicitly specify which Java VM to use when running Eclipse. This is achieved with the "`-vm`" command line argument as illustrated above. If you don't use "`-vm`", Eclipse will look on the O/S path. When you install other Java-based products, they may change your path and could result in a different Java VM being used when you next launch Eclipse.

To create a Windows shortcut to an installed Eclipse:

1. Navigate to `eclipse.exe` in Windows Explorer and use Create Shortcut on the content menu.
2. Select the shortcut and edit its Properties. In the Target: field append the command line arguments.

Opening this shortcut launches Eclipse. (You can drag the shortcut to the Windows Desktop if you want to keep it in easy reach.)

Mac OS X

On Mac OS X, you start Eclipse by double clicking the Eclipse application. If you need to pass arguments to Eclipse, you'll have to edit the `eclipse.ini` file inside the Eclipse application bundle: select the Eclipse application bundle icon while holding down the Control Key. This will present you with a popup menu. Select "Show Package Contents" in the popup menu. Locate `eclipse.ini`file in the `Contents/MacOS` sub-folder and open it with your favorite text editor to edit the command line options.

On MacOS X you can only launch a UI program more than once if you have separate copies of the program on disk. The reason for this behavior is that every UI application on Mac can open multiple documents, so typically there is no need to open a program twice. Since Eclipse cannot open more than one workspace, this means you have to make a copy of the Eclipse install if you want to open more then one workspace at the same time (bug 139319).

If you need to launch Eclipse from the command line, you can use the symbolic link "eclipse" in the top-level eclipse folder. It refers to the eclipse executable inside the application bundle and takes the same arguments as "eclipse.exe" on other platforms.

On Mac OS X 10.4 and later, you may notice a slow down when working with significant numbers of resources if you allow Spotlight to index your workspace. To prevent this, start System Preferences, select the Spotlight icon, then the Privacy tab, then click the Add button ("+") and find your workspace directory in the dialog that appears.

Shared Install

The startup speed of a shared install can be improved if proper cache information is stored in the shared install area. To achieve this, after unzipping Eclipse distribution, run Eclipse once with the "-initialize" option from an account that has a write access to the install directory.

Advanced Topics in Running Eclipse

The Eclipse executable and the platform itself offer a number of execution options of interest to people developing or debugging parts of Eclipse. This is a list of the commonly used options, for a full list see the Eclipse runtime options page in the Platform Plug-in Developer Guide. The general form of running the Eclipse executable is:

```
eclipse [platform options] [-vmargs [Java VM arguments]]
```

Command	Description	Since
`-arch` `architecture`	Defines the processor architecture on which the Eclipse platform is running. The Eclipse platform ordinarily computes the optimal setting using the prevailing value of Java `os.arch` property. If specified here, this is the value that the Eclipse platform uses. The value specified here is available to plug-ins as Platform.getOSArch(). Example values: "x86", "sparc", "PA-RISC", "ppc".	2.0
`-application` `applicationId`	The application to run. Applications are declared by plug-ins supplying extensions to the org.eclipse.core.runtime.applications extension point. This argument is typically not needed. If specified, the value overrides the value supplied by the configuration. If not specified, the Eclipse Workbench is run.	1.0
`-clean`	Cleans cached data used by the OSGi framework and Eclipse runtime. Try to run Eclipse once with this option if you observe startup errors after install, update, or using a shared configuration.	3.0

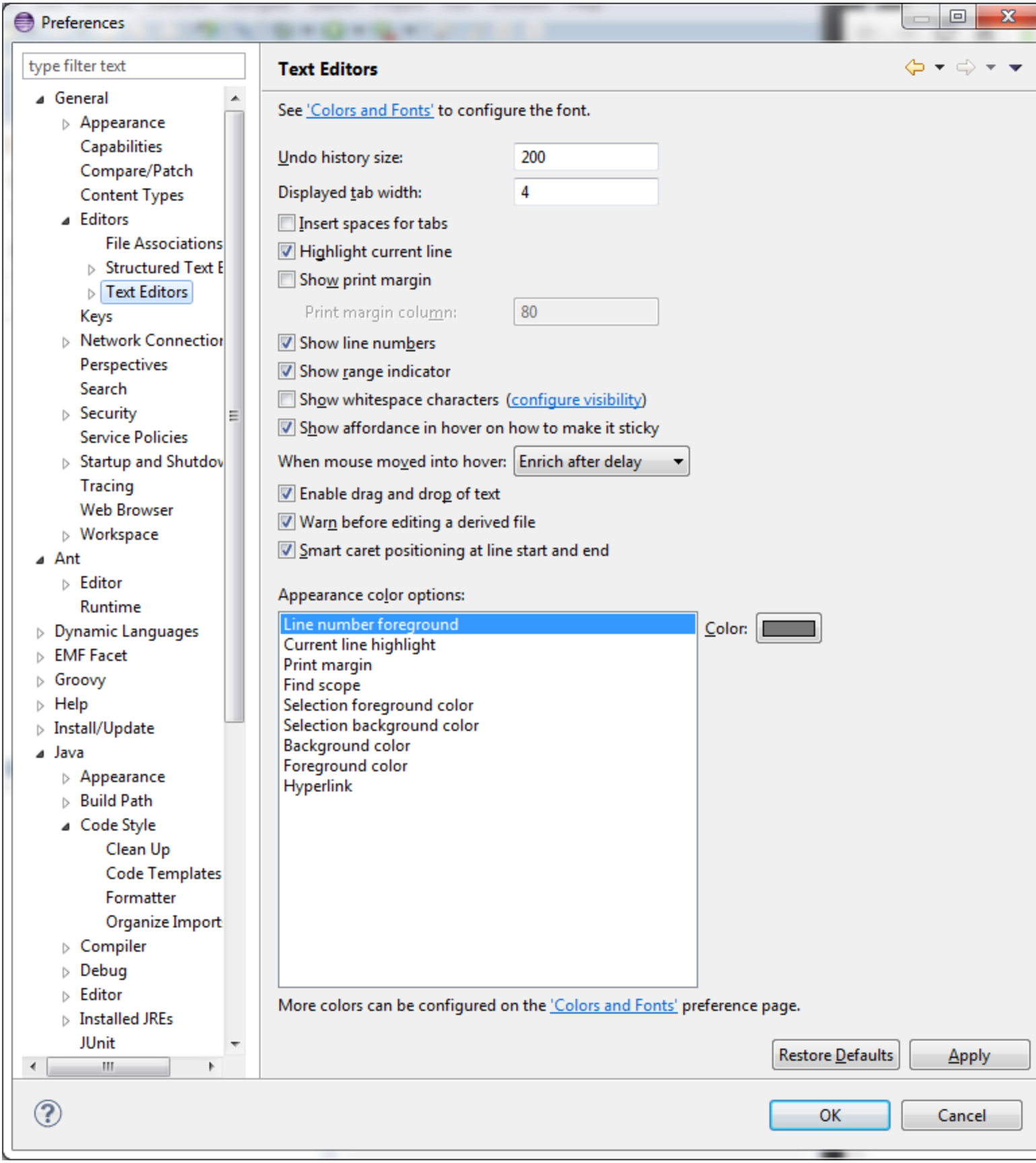
-configuration configURL	The location for the Eclipse Platform configuration file, expressed as a URL. The configuration file determines the location of the Eclipse platform, the set of available plug-ins, and the primary feature. Note that relative URLs are not allowed. The configuration file is written to this location when the Eclipse platform is installed or updated.	2.0
-consolelog	Mirrors the Eclipse platform's error log to the console used to run Eclipse. Handy when combined with `-debug`.	1.0
-data workspacePath	The path of the workspace on which to run the Eclipse platform. The workspace location is also the default location for projects. Relative paths are interpreted relative to the directory that Eclipse was started from.	1.0
-debug [optionsFile]	Puts the platform in debug mode and loads the debug options from the file at the given location, if specified. This file indicates which debug points are available for a plug-in and whether or not they are enabled. If a file location is not given, the platform looks in the directory that eclipse was started from for a file called ".options". Both URLs and file system paths are allowed as file locations.	1.0
-dev [classpathEntries]	Puts the platform in development mode. The optional classpath entries (a comma separated list) are added to the runtime classpath of each plug-in. For example, when the workspace contains plug-ins being developed, specifying `-dev bin` adds a classpath entry for each plug-in project's directory named`bin`, allowing freshly generated class files to be found there. Redundant or non-existent classpath entries are eliminated.	1.0
-initialize	Initializes the configuration being run. All runtime related data structures and caches are refreshed. Handy with shared installs: running Eclipse once with this option from an account with write privileges will improve startup performance.	3.0
-keyring keyringFilePath	The location of the authorization database (or "key ring" file) on disk. This argument must be used in conjunction with the `-password` option. Relative paths are interpreted relative to the directory that Eclipse was started from.	1.0
-nl locale	Defines the name of the locale on which the Eclipse platform is running. The Eclipse	2.0

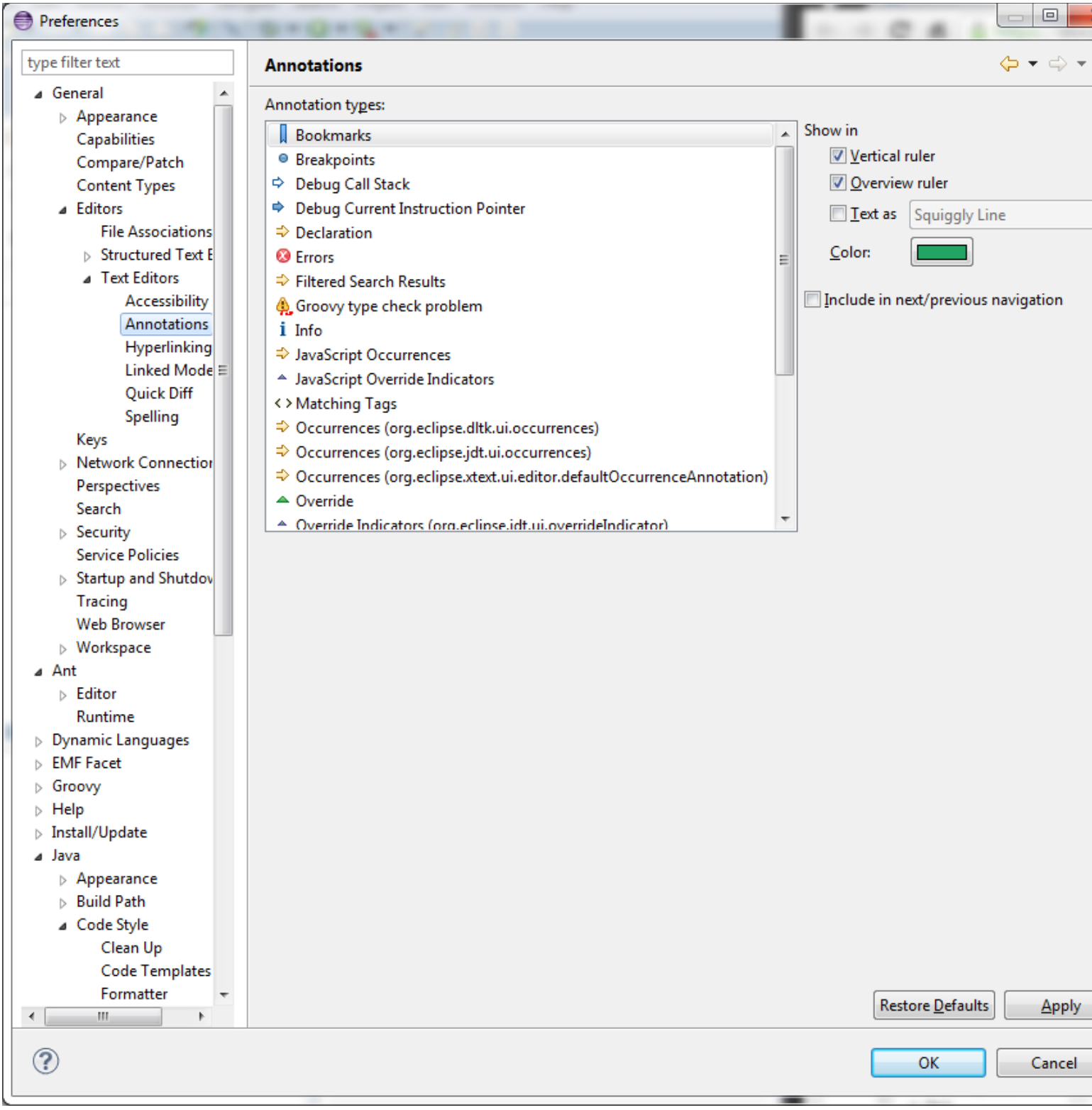
	platform ordinarily computes the optimal setting automatically. If specified here, this is the value that the Eclipse platform uses. The value specified here is available to plug-ins as Platform.getNL(). Example values: "en_US" and "fr_FR_EURO".	
-nosplash	Runs the platform without putting up the splash screen.	1.0
-os `operatingSystem`	Defines the operating system on which the Eclipse platform is running. The Eclipse platform ordinarily computes the optimal setting using the prevailing value of Java `os.name` property. If specified here, this is the value that the Eclipse platform uses. The value specified here is available to plug-ins as Platform.getOS(), and used to resolve occurrences of the `$os$` variable in paths mentioned in the plug-in manifest file. Example values: "win32", "linux", "hpux", "solaris", "aix".	1.0
-password `password`	The password for the authorization database. Used in conjunction with the `-keyring` option.	1.0
-perspective `perspectiveId`	The perspective to open in the active workbench window on startup. If this parameter is not specified, the perspective that was active on shutdown will be opened.	1.0
-plugincustomization `propertiesFile`	The location of a properties file containing default settings for plug-in preferences. These default settings override default settings specified in the primary feature. Relative paths are interpreted relative to the directory that eclipse was started from.	2.0
-product *productId*	The ID of the product to run. The product gives the launched instance of Eclipse its personality, and determines the product customization information used. This replaces -feature, which is still supported for compatibility.	3.0
-refresh	Option for performing a global refresh of the workspace on startup. This will reconcile any changes that were made in the file system since the platform was last run.	1.0
-showlocation `[workspaceName]`	Option for displaying the location of the workspace in the window title bar. In release 2.0 this option only worked in conjunction with the -data command line argument. In 3.2, an optional workspace name argument was added that displays the provided name in the window title bar instead of the location of the	2.0

	workspace.	
-vm vmPath	The location of Java Runtime Environment (JRE) to use to run the Eclipse platform. If not specified, the launcher will attempt to find a JRE. It will first look for a directory called `jre` as a sibling of the Eclipse executable, and then look on the operating system path. Relative paths are interpreted relative to the directory that eclipse was started from.	1.0
-vmargs *args*	When passed to the Eclipse, this option is used to customize the operation of the Java VM used to run Eclipse. If specified, this option must come at the end of the command line. The given arguments are dependent on VM that is being run.	1.0

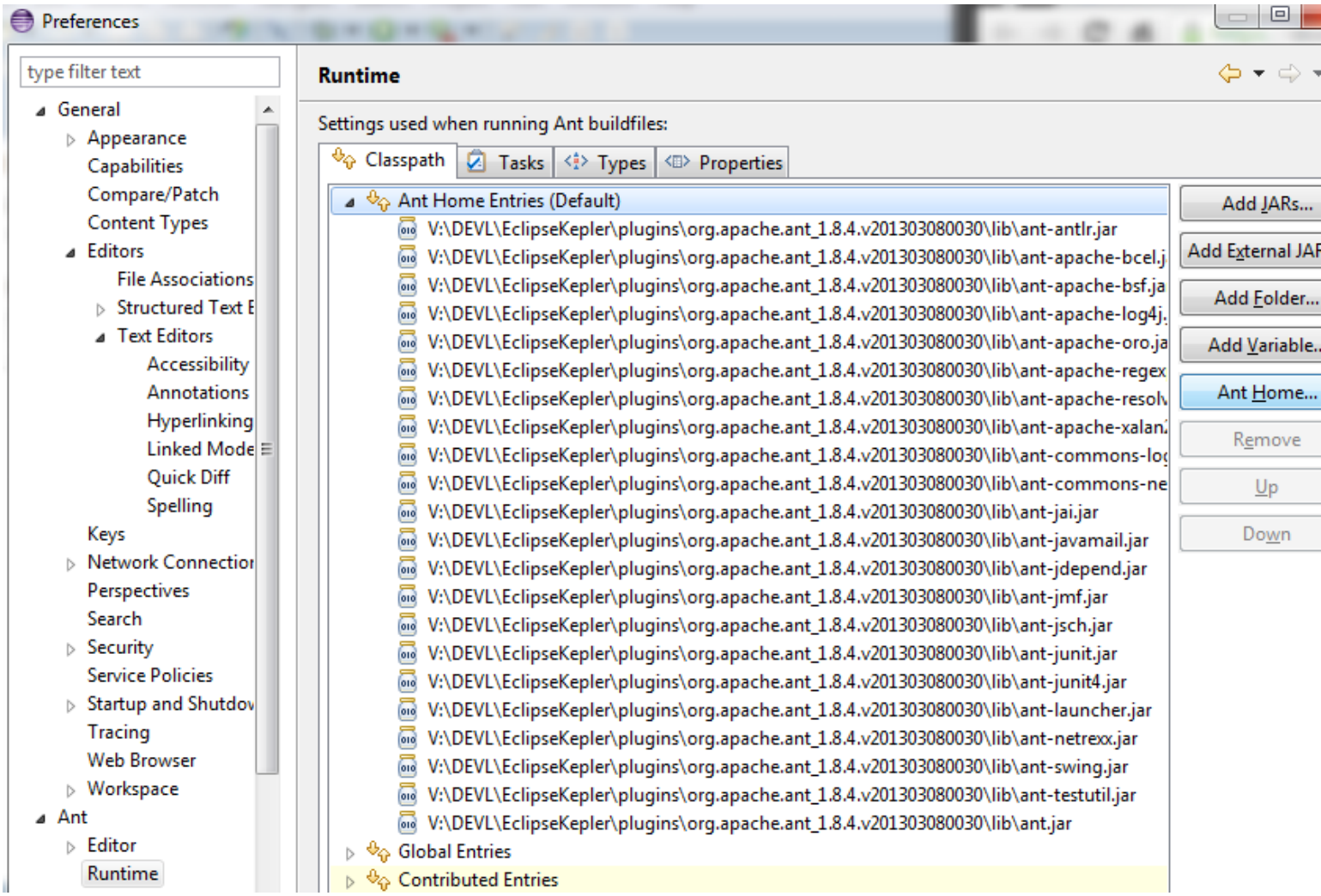
All arguments following (but not including) the -vmargs entry are passed directly through to the indicated Java VM as virtual machine arguments (that is, before the class to run). **Note:** If an Eclipse startup argument, such as -data, is provided after the Java vm arguments (-vmargs), Eclipse will not start and you will receive a "JVM terminated. Exit code=1" error.

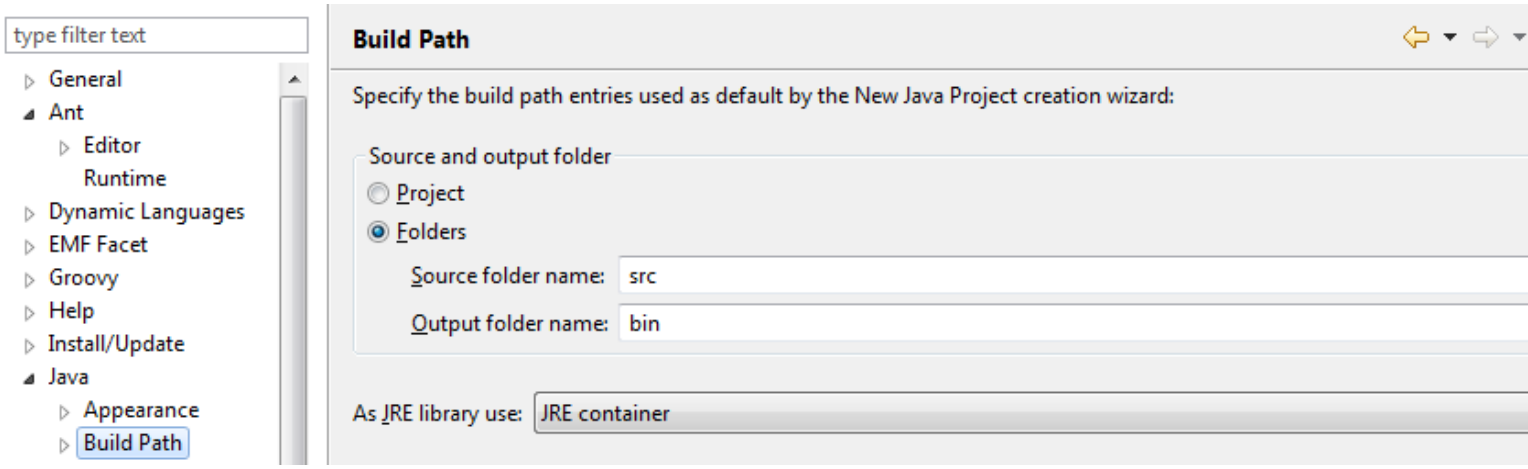
General, Editors, Text Editors

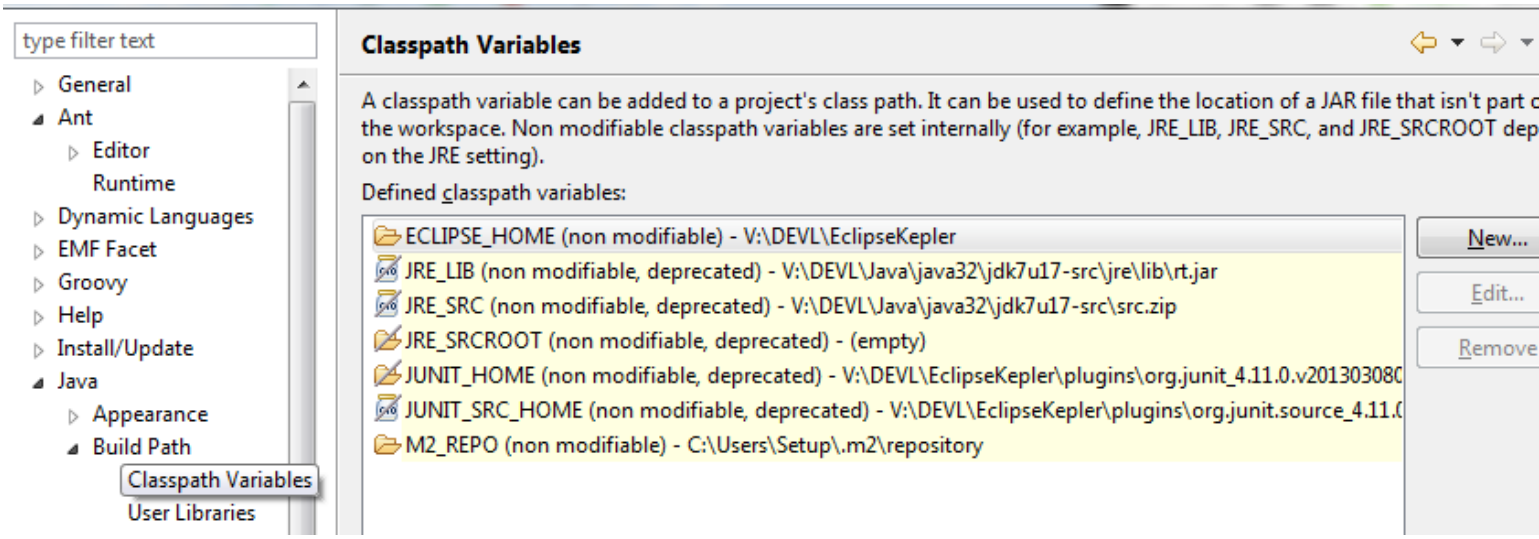

Preferences
type filter text
General
Appearance
Capabilities
Compare/Patch
Content Types
Editors
File Associations
Structured Text E
Text Editors
Keys
Network Connectior
Perspectives
Search
Security
Service Policies
Startup and Shutdov
Tracing
Web Browser
Workspace
Ant
Editor
Runtime
Dynamic Languages
EMF Facet
Groovy
Help
Install/Update
Java
Appearance
Build Path
Code Style
Clean Up
Code Templates
Formatter
Organize Import
Compiler
Debug
Editor
Installed JREs
JUnit
Text Editors
See 'Colors and Fonts' to configure the font.
Undo history size:
200
Displayed tab width:
4
Insert spaces for tabs
Highlight current line
Show print margin
Print margin column:
80
Show line numbers
Show range indicator
Show whitespace characters (configure visibility)
Show affordance in hover on how to make it sticky
When mouse moved into hover:
Enrich after delay
Enable drag and drop of text
Warn before editing a derived file
Smart caret positioning at line start and end
Appearance color options:
Line number foreground
Current line highlight
Print margin
Find scope
Selection foreground color
Selection background color
Background color
Foreground color
Hyperlink
Color:
More colors can be configured on the 'Colors and Fonts' preference page.
Restore Defaults
Apply
OK
Cancel

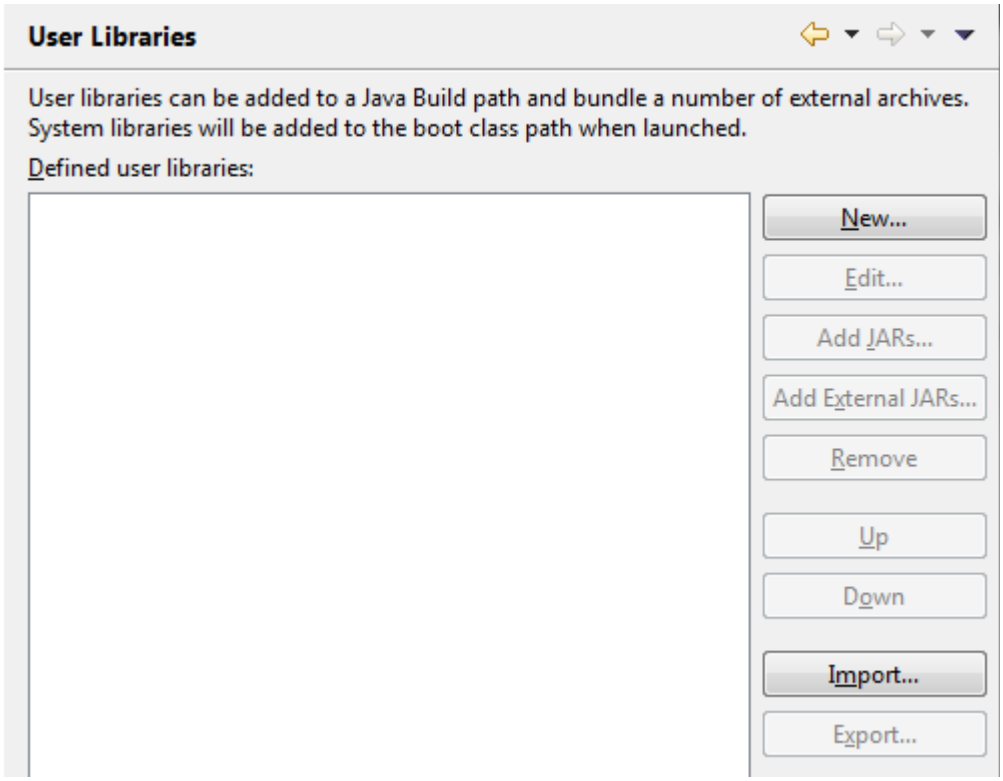


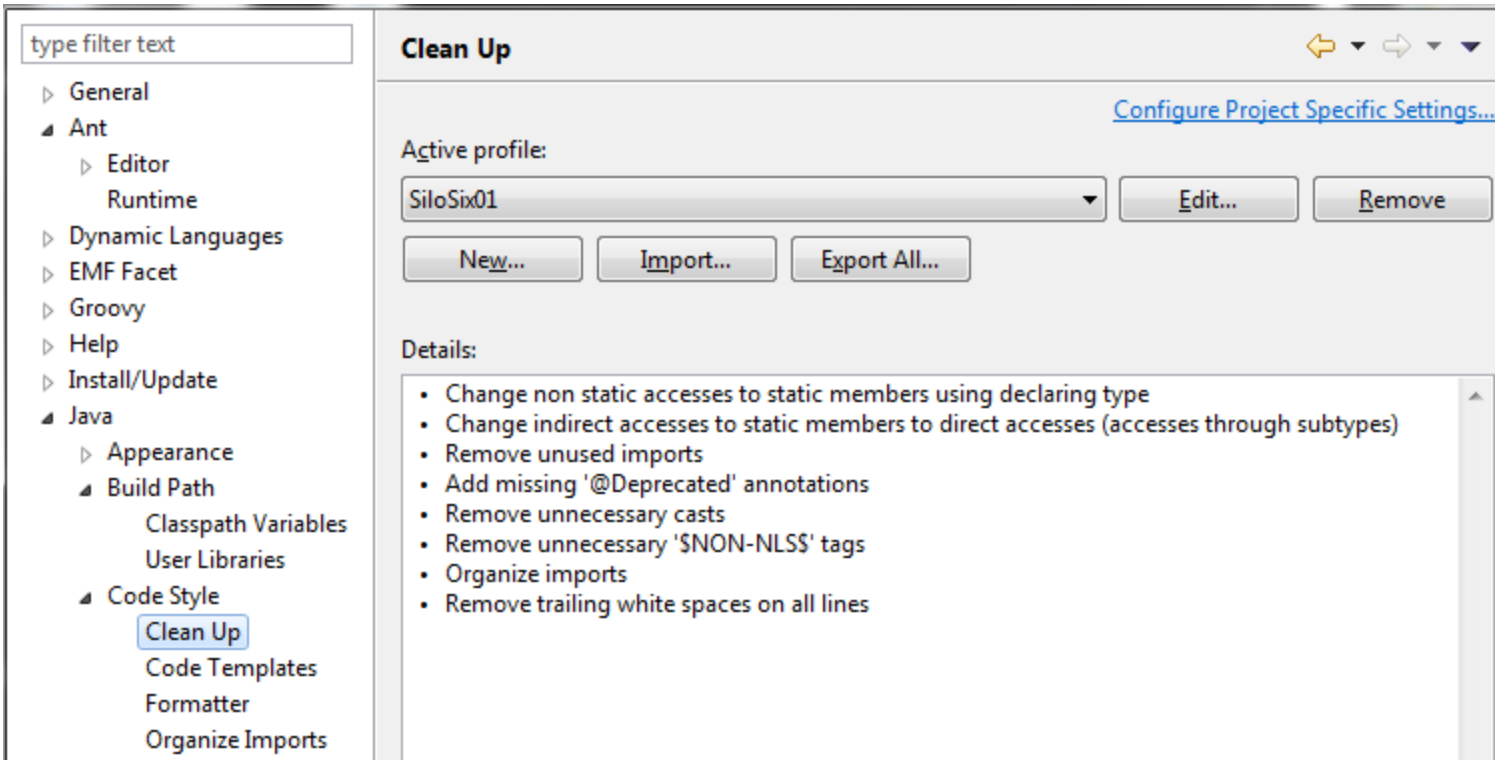
Ant Runtime Location


Preferences
type filter text
General
Appearance
Capabilities
Compare/Patch
Content Types
Editors
File Associations
Structured Text E
Text Editors
Accessibility
Annotations
Hyperlinking
Linked Mode
Quick Diff
Spelling
Keys
Network Connectior
Perspectives
Search
Security
Service Policies
Startup and Shutdov
Tracing
Web Browser
Workspace
Ant
Editor
Runtime
Runtime
Settings used when running Ant buildfiles:
Classpath
Tasks
Types
Properties
Ant Home Entries (Default)
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-antlr.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-apache-bcel.j
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-apache-bsf.ja
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-apache-log4j.
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-apache-oro.ja
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-apache-regex
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-apache-resolv
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-apache-xalan2
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-commons-log
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-commons-ne
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-jai.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-javamail.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-jdepend.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-jmf.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-jsch.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-junit.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-junit4.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-launcher.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-netrexx.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-swing.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant-testutil.jar
V:\DEVL\EclipseKepler\plugins\org.apache.ant_1.8.4.v201303080030\lib\ant.jar
Global Entries
Contributed Entries
Add JARs...
Add External JAR
Add Folder...
Add Variable...
Ant Home...
Remove
Up
Down

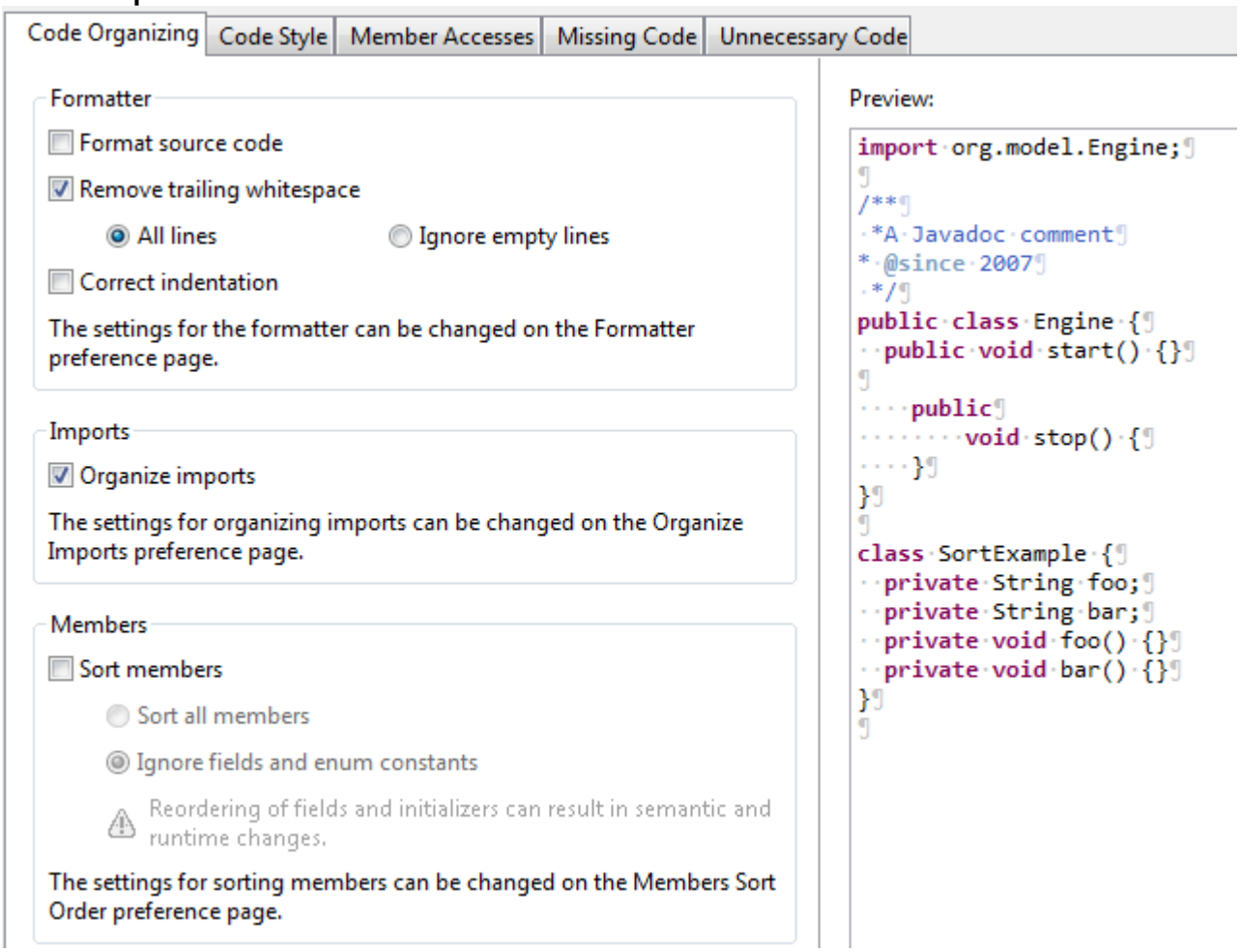

type filter text
General
Ant
Editor
Runtime
Dynamic Languages
EMF Facet
Groovy
Help
Install/Update
Java
Appearance
Build Path
Build Path
Specify the build path entries used as default by the New Java Project creation wizard:
Source and output folder
Project
Folders
Source folder name:
src
Output folder name:
bin
As JRE library use:
JRE container


type filter text
General
Ant
Editor
Runtime
Dynamic Languages
EMF Facet
Groovy
Help
Install/Update
Java
Appearance
Build Path
Classpath Variables
User Libraries
Classpath Variables
A classpath variable can be added to a project's class path. It can be used to define the location of a JAR file that isn't part c
the workspace. Non modifiable classpath variables are set internally (for example, JRE_LIB, JRE_SRC, and JRE_SRCROOT dep
on the JRE setting).
Defined classpath variables:
ECLIPSE_HOME (non modifiable) - V:\DEVL\EclipseKepler
JRE_LIB (non modifiable, deprecated) - V:\DEVL\Java\java32\jdk7u17-src\jre\lib\rt.jar
JRE_SRC (non modifiable, deprecated) - V:\DEVL\Java\java32\jdk7u17-src\src.zip
JRE_SRCROOT (non modifiable, deprecated) - (empty)
JUNIT_HOME (non modifiable, deprecated) - V:\DEVL\EclipseKepler\plugins\org.junit_4.11.0.v201303080
JUNIT_SRC_HOME (non modifiable, deprecated) - V:\DEVL\EclipseKepler\plugins\org.junit.source_4.11.0
M2_REPO (non modifiable) - C:\Users\Setup\.m2\repository
New...
Edit...
Remove


User Libraries
User libraries can be added to a Java Build path and bundle a number of external archives.
System libraries will be added to the boot class path when launched.
Defined user libraries:
New...
Edit...
Add JARs...
Add External JARs...
Remove
Up
Down
Import...
Export...



CleanUp Profile



Code Organizing | Code Style | Member Accesses | Missing Code | Unnecessary Code

Control statements

- [ ] Use blocks in if/while/for/do statements
  - (•) Always
  - ( ) Always except for single 'return' or 'throw' statements
  - ( ) Only if necessary
- [ ] Convert 'for' loops to enhanced

Expressions

- [ ] Use parentheses in expressions
  - ( ) Always
  - (•) Only if necessary

Variable declarations

- [ ] Use modifier 'final' where possible
  - [x] Private fields
  - [ ] Parameter
  - [x] Local variables

Preview:

```
if (obj == null) {
    throw new IllegalArgumentException();
}
if (ids.length > 0) {
    System.out.println(ids[0]);
} else
    return;


for (int i = 0; i < ids.length; i++) {
    double value= ids[i] / 2;
    System.out.println(value);
}

boolean b= (i > 0 && i < 10 || i == 50);

private int i= 0;
public void foo(int j) {
    int k, h;
    h= 0;
}
```

Code Organizing | Code Style | Member Accesses | Missing Code | Unnecessary Code

Non static accesses

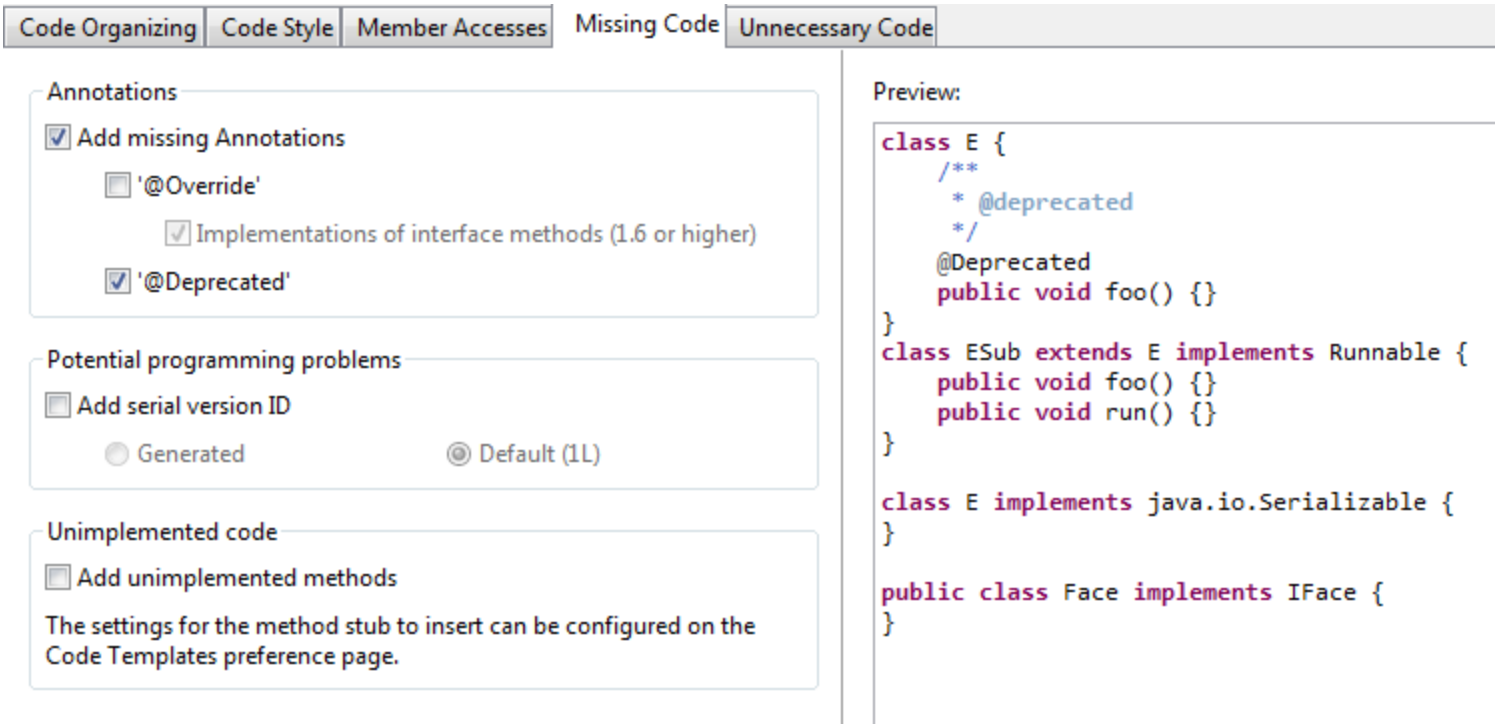
- [ ] Use 'this' qualifier for field accesses
  - ( ) Always
  - (•) Only if necessary
- [ ] Use 'this' qualifier for method accesses
  - ( ) Always
  - (•) Only if necessary

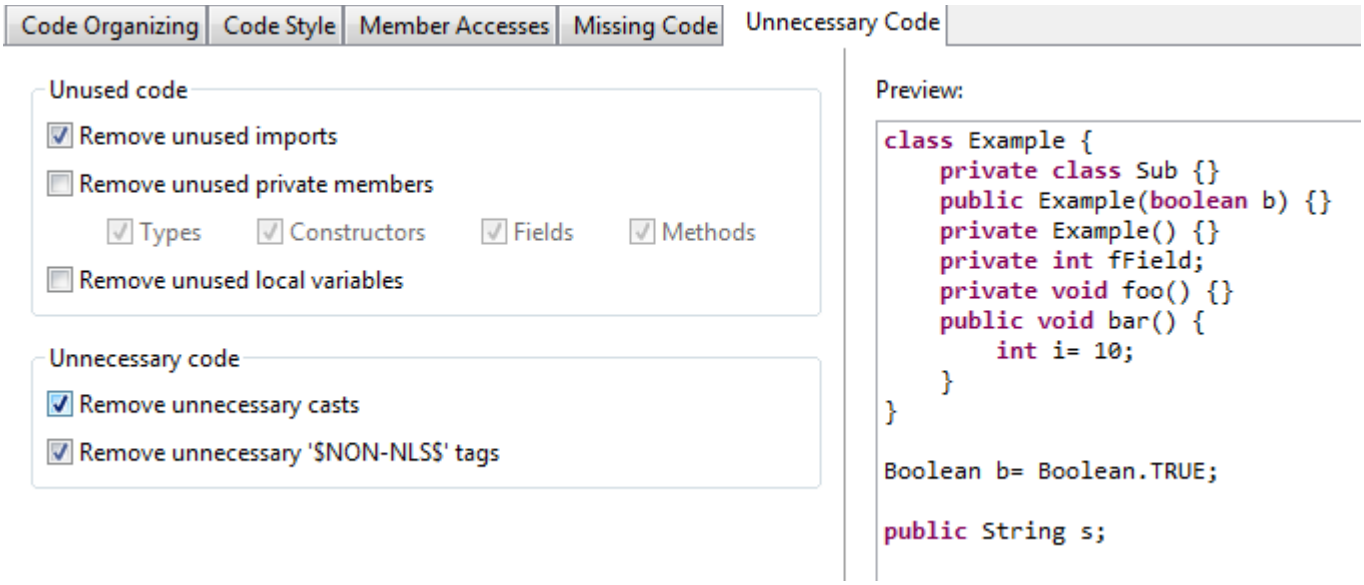
Static accesses

- [x] Use declaring class as qualifier
  - [ ] Qualify field accesses
  - [ ] Qualify method accesses
  - [x] Change all accesses through subtypes
  - [x] Change all accesses through instances

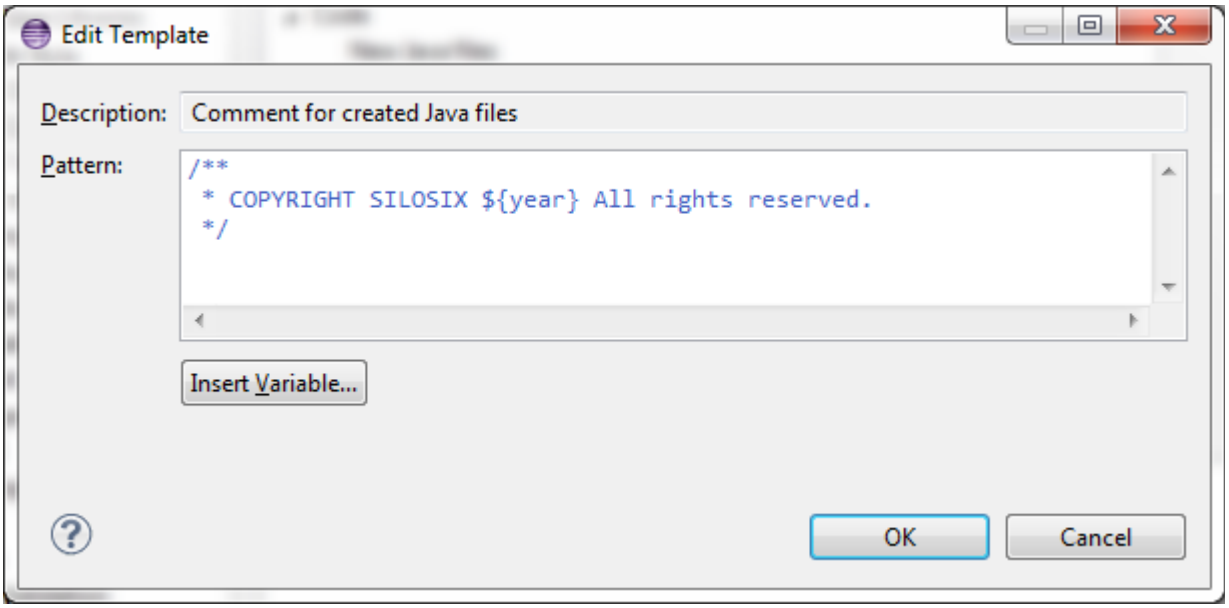
Preview:

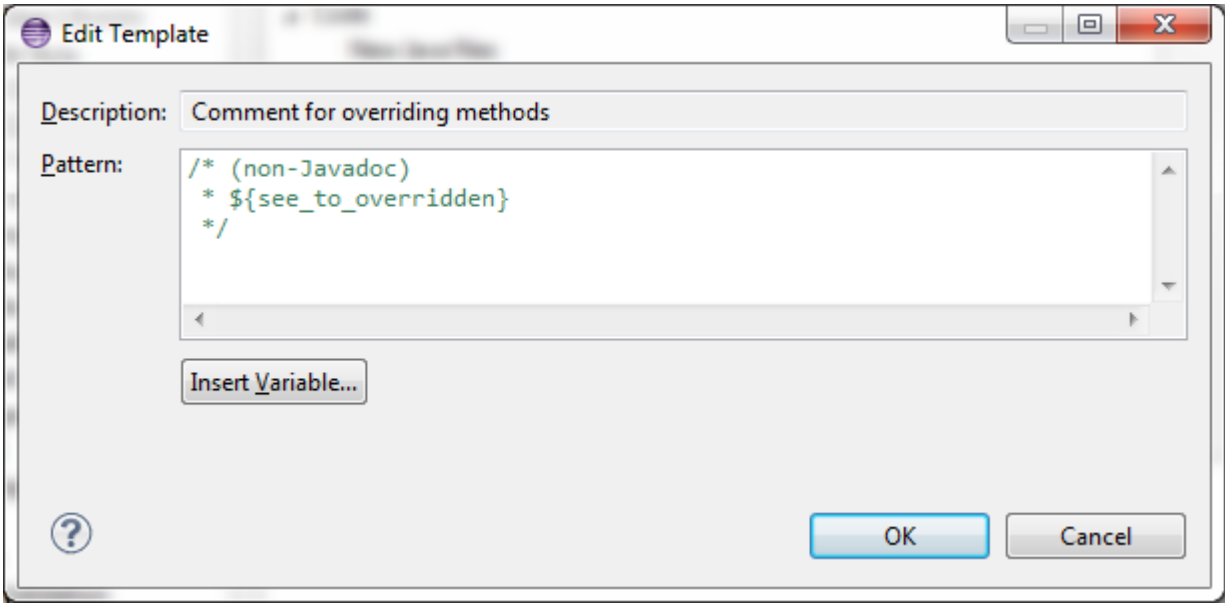
```
private int value;
public int get() {
    return this.value + value;
}

public int getZero() {
    return this.get() - get();
}

class E {
    public static int NUMBER;
    public static void set(int i) {
        NUMBER= i;
    }

    public void reset() {
        set(0);
    }
}

class ESub extends E {
    public void reset() {
        E.NUMBER= 0;
    }
}

public void dec() {
    E.NUMBER--;
}
```

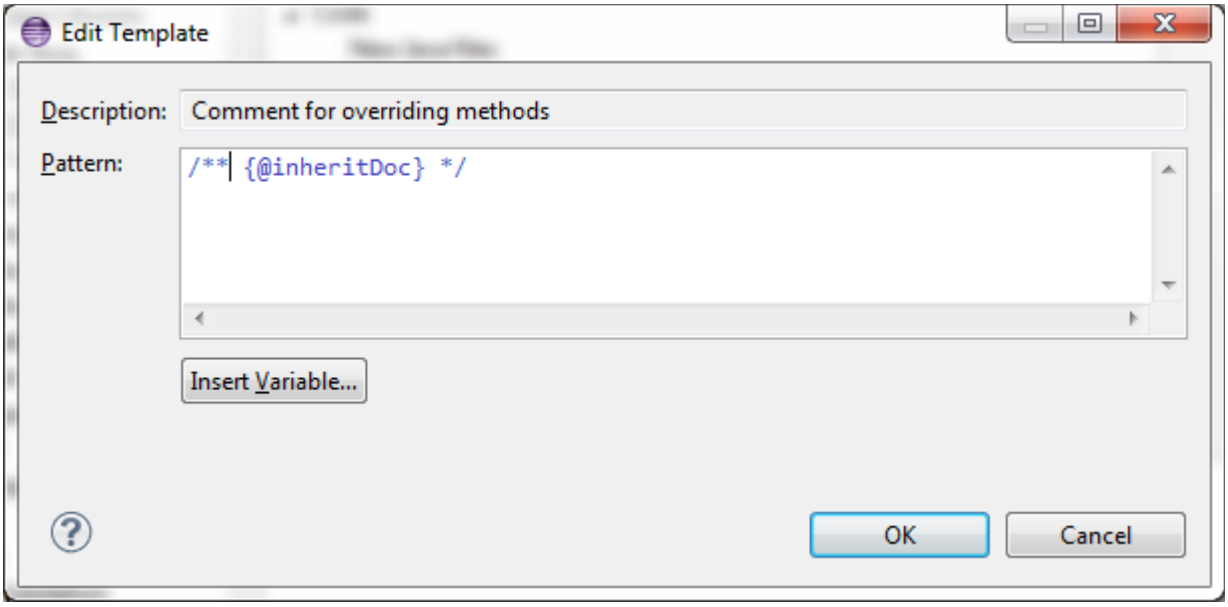


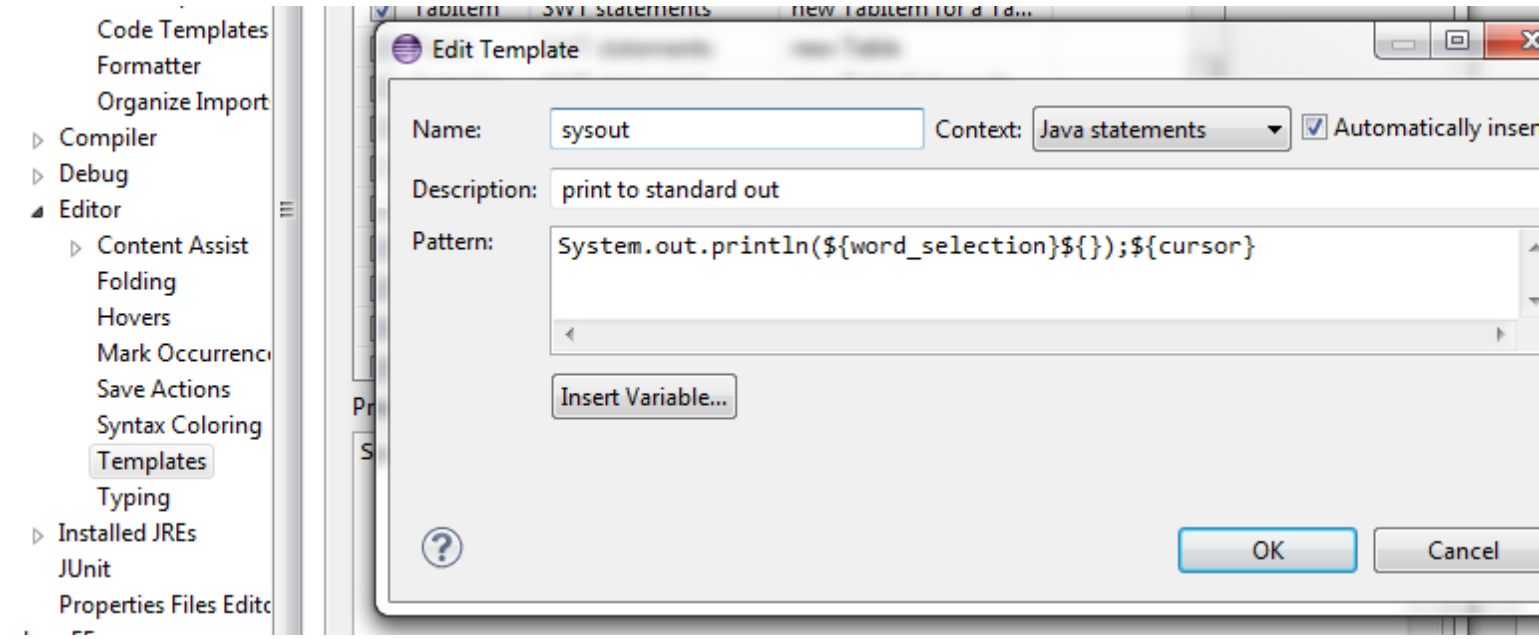


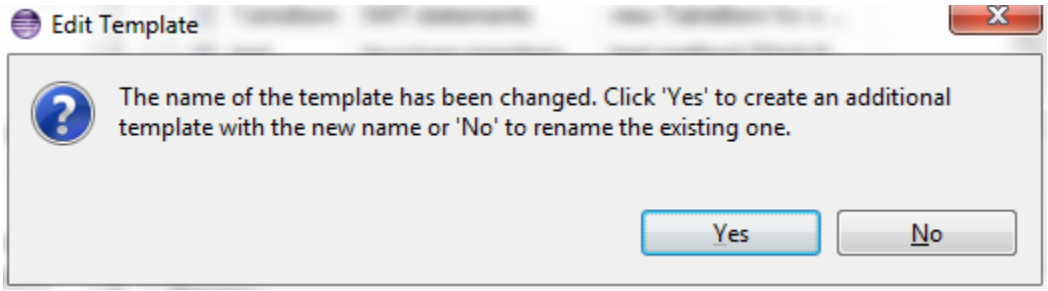
Code Templates

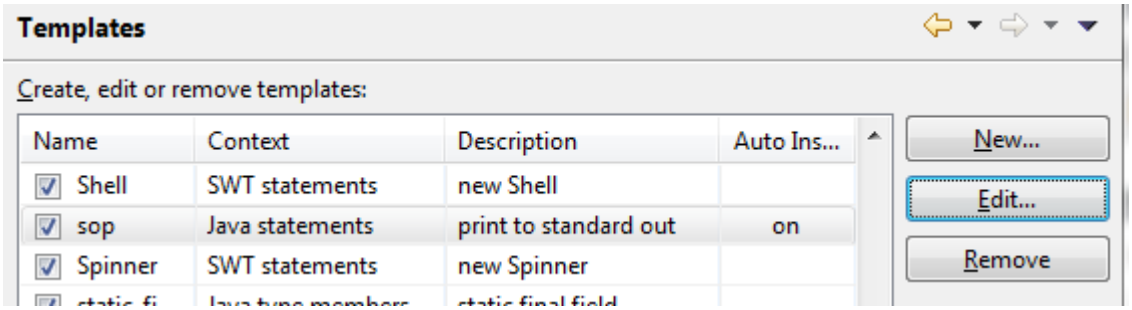

Edit Template
Description:
Comment for created Java files
Pattern:
/**
* COPYRIGHT SILOSIX ${year} All rights reserved.
*/
Insert Variable...
OK
Cancel

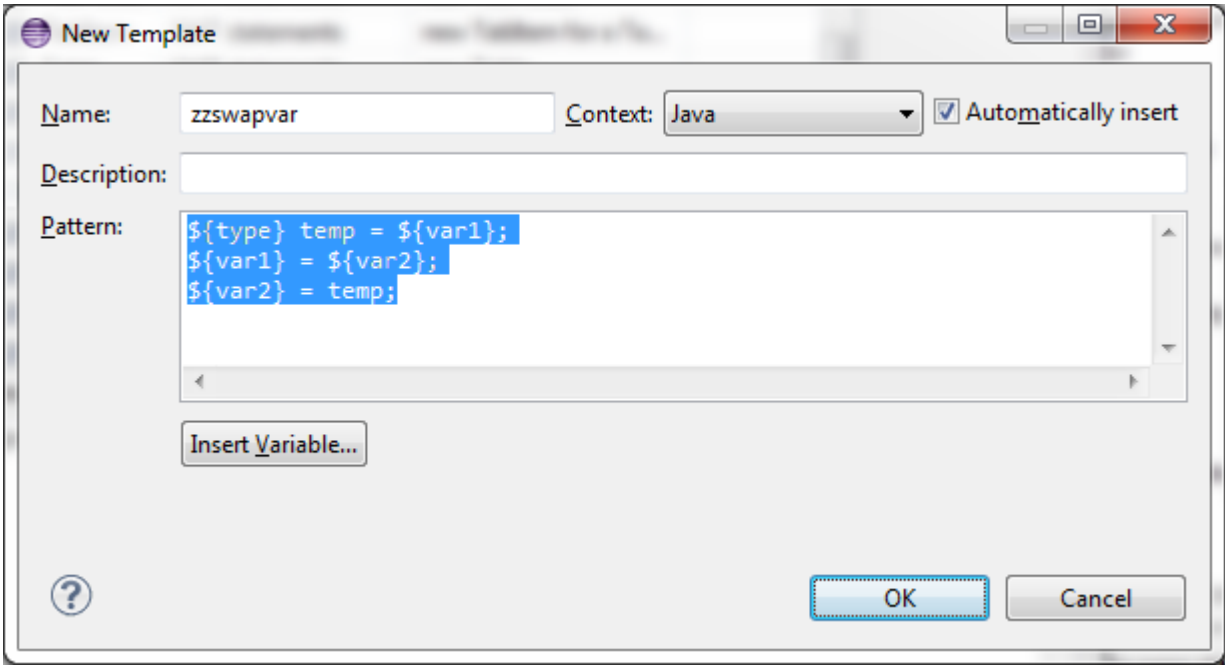

Edit Template
Description:
Comment for overriding methods
Pattern:
/* (non-Javadoc)
* ${see_to_overridden}
*/
Insert Variable...
OK
Cancel

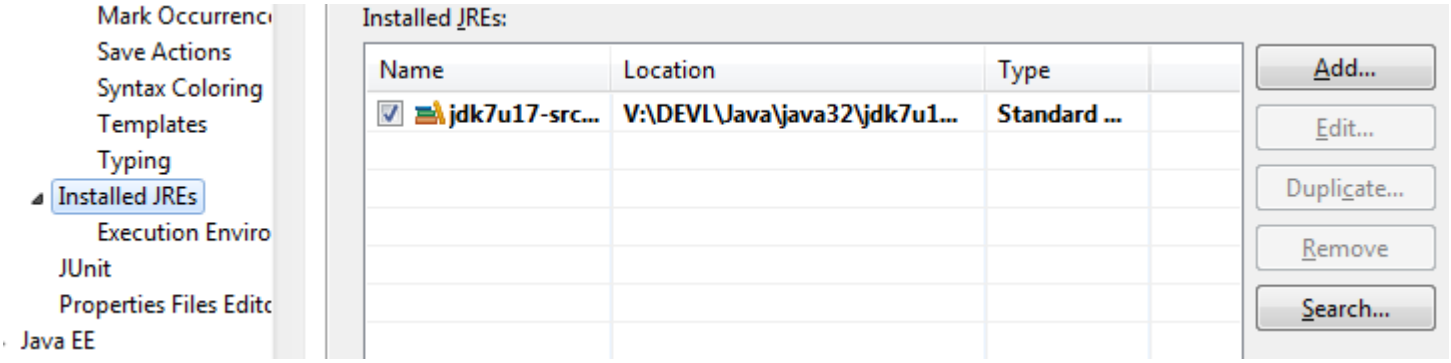

Edit Template
Description:
Comment for overriding methods
Pattern:
/** {@inheritDoc} */
Insert Variable...
OK
Cancel

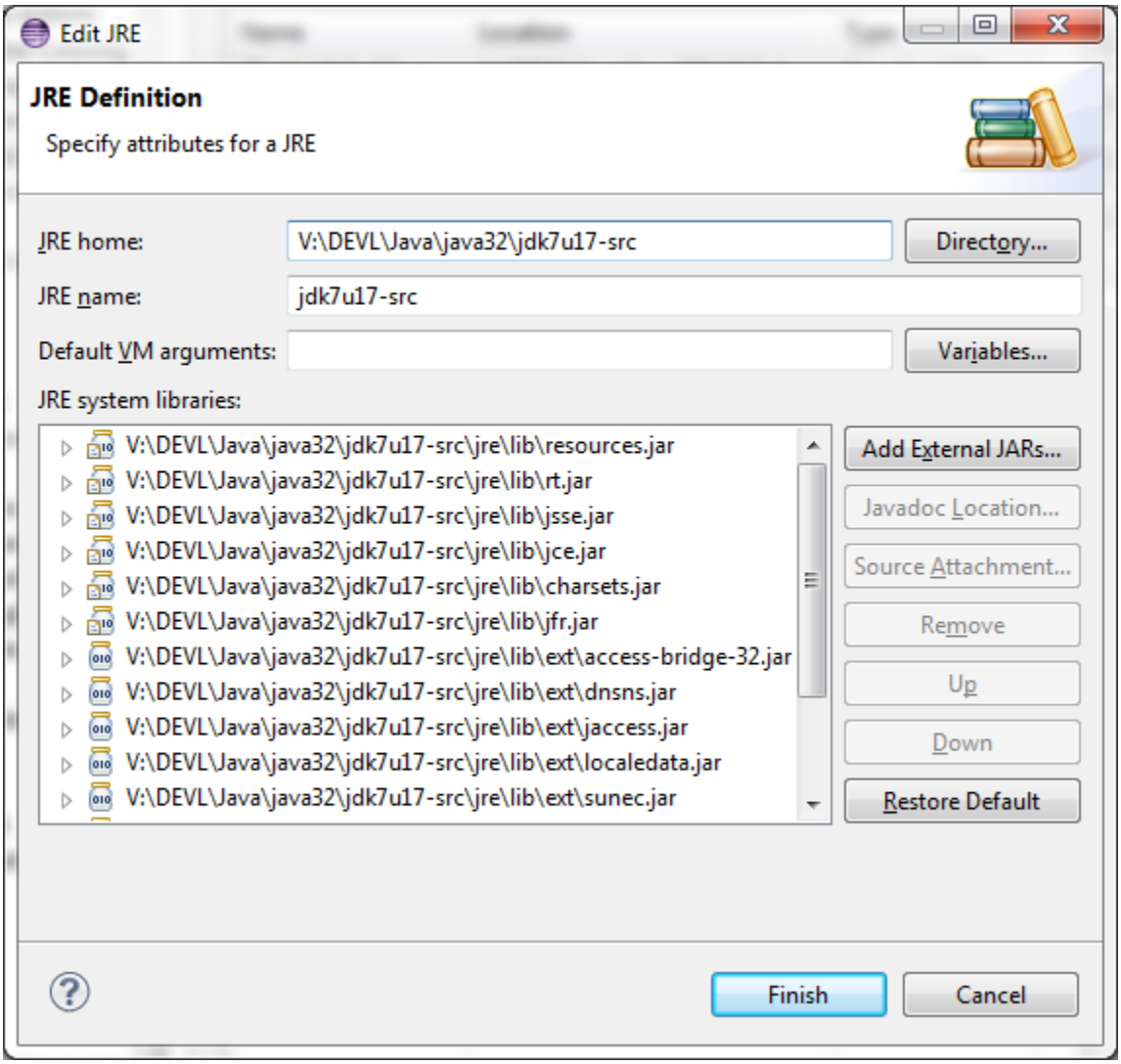

Code Templates
Formatter
Organize Import
Compiler
Debug
Editor
Content Assist
Folding
Hovers
Mark Occurrenc
Save Actions
Syntax Coloring
Templates
Typing
Installed JREs
JUnit
Properties Files Editc
Edit Template
Name:
sysout
Context:
Java statements
Automatically inser
Description:
print to standard out
Pattern:
System.out.println(${word_selection}${});${cursor}
Insert Variable...
OK
Cancel


Edit Template
The name of the template has been changed. Click 'Yes' to create an additional template with the new name or 'No' to rename the existing one.
Yes
No


Templates
Create, edit or remove templates:
Name
Context
Description
Auto Ins...
Shell
SWT statements
new Shell
sop
Java statements
print to standard out
on
Spinner
SWT statements
new Spinner
New...
Edit...
Remove


New Template
Name:
zzswapvar
Context:
Java
Automatically insert
Description:
Pattern:
${type} temp = ${var1};
${var1} = ${var2};
${var2} = temp;
Insert Variable...
OK
Cancel


Mark Occurrenc
Save Actions
Syntax Coloring
Templates
Typing
Installed JREs
Execution Enviro
JUnit
Properties Files Editc
Java EE
Installed JREs:
Name
Location
Type
jdk7u17-src...
V:\DEVL\Java\java32\jdk7u1...
Standard ...
Add...
Edit...
Duplicate...
Remove
Search...



Eclipse Natures (.project)

Metatag for Eclipse
Two natures

.project

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
	<name>JavaScriptLib</name>
	<comment></comment>
	<projects>
	</projects>
	<buildSpec>
		<buildCommand>
			<name>org.eclipse.wst.jsdt.core.javascriptValidator</name>
			<arguments>
			</arguments>
```

```
		</buildCommand>
	</buildSpec>
	<natures>
		<nature>org.eclipse.wst.jsdt.core.jsNature</nature>
	</natures>
</projectDescription>
```