Tutorial 02: Writing Source Code

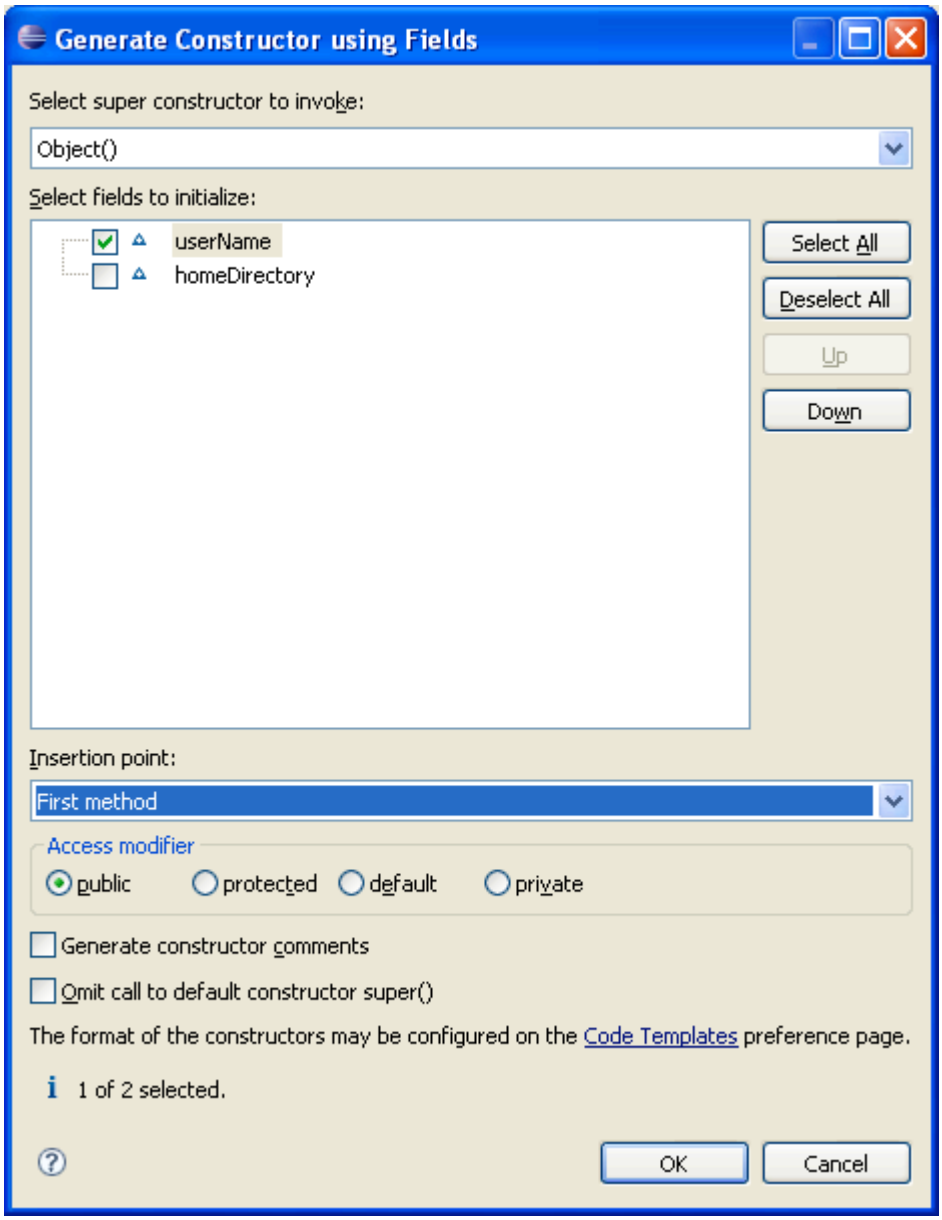
Contents:

Generating a constructor

1. Create a new Java class, Programmer.
2. Type the following code in Programmer.java.

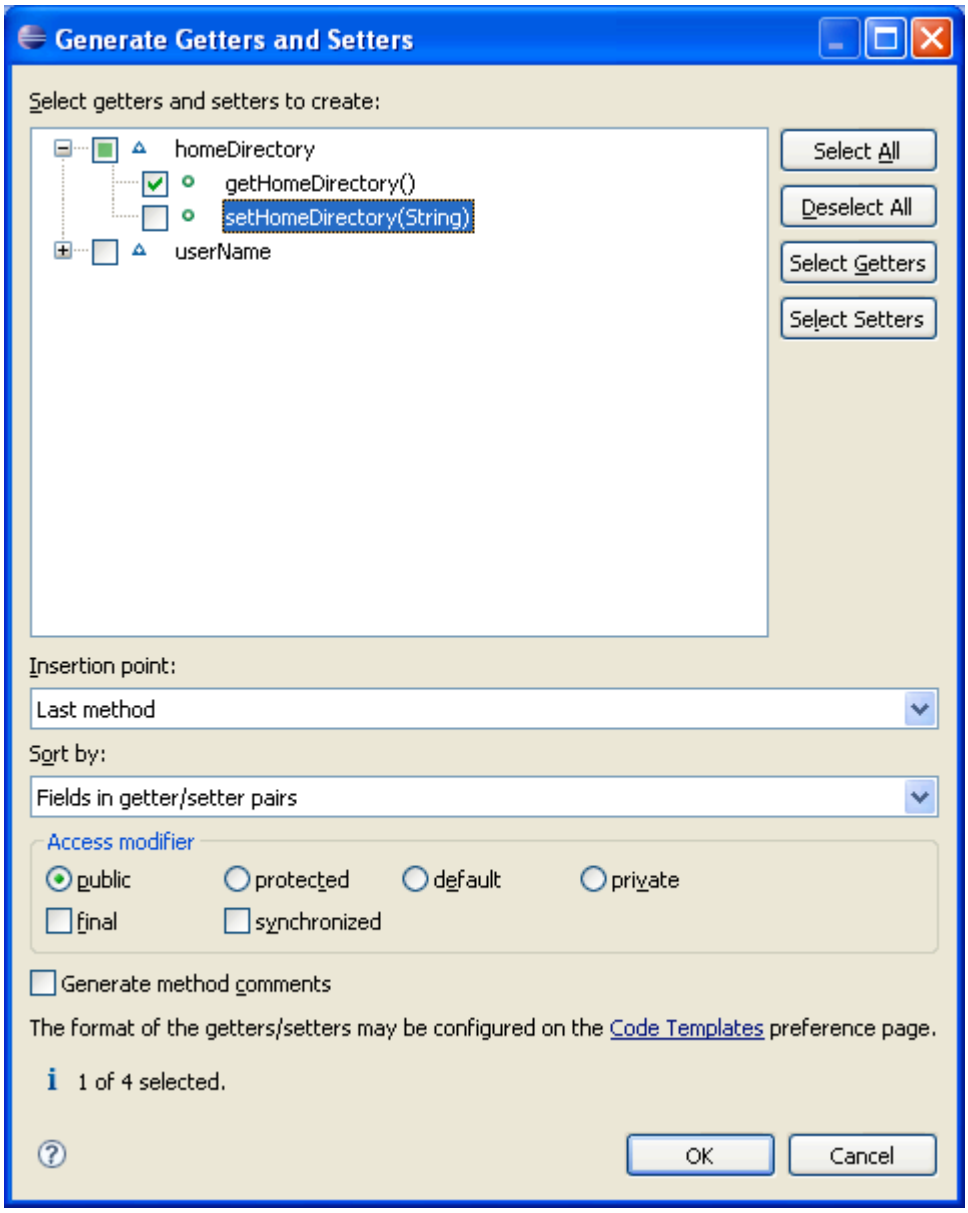
```
package org.eclipse.course.hellojava;

public class Programmer {
     String userName; // the user name
     String homeDirectory; // the home directory
}
```

3. Select the word Programmer and right-click it, and select Source > Generate Constructor using Fields…



Generating getters and setters

1. Select the word homeDirectroy and right-click it, and select Source > Generate Getters and Setters…



2. Edit the code

```
package org.eclipse.course.hellojava;

public class Programmer {
	String userName; // the user name
	String homeDirectory; // the home directory

	public Programmer(String userName) {
		super();
		this.userName = userName;
		printProgrammer();
	}
```

```
	public String getHomeDirectory() {
		return homeDirectory;
	}

	public void printProgrammer() {
		System.out.println("A new Programmer");
		System.out.println("Name : " + userName);
	}
}
```

Renaming a method

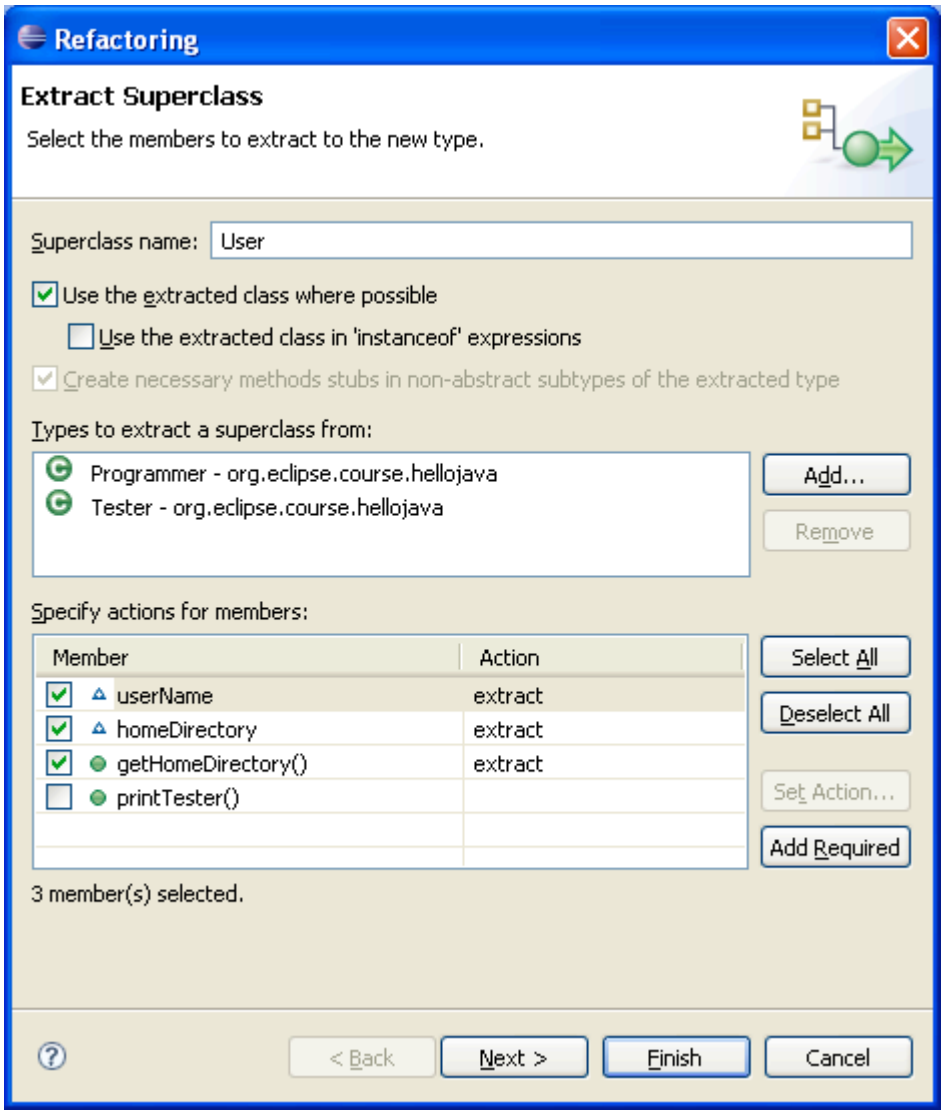
1. Select Programmer.java in the Package Explorer view.
2. Click Edit > Copy and Edit > Paste.
3. Put Tester as the name of the new class.
4. Select the word printProgrammer in the printProgrammer method and right-click it, and select Refactor > Rename…
5. Put printTester as the new name and notice that Eclipse changes the method call in the constructor as well (compare this with renaming printProgrammer() directly by editing its name).

> Refactoring is the process of changing a software system to improve its internal structure and reusability, without altering the external behavior of the program.

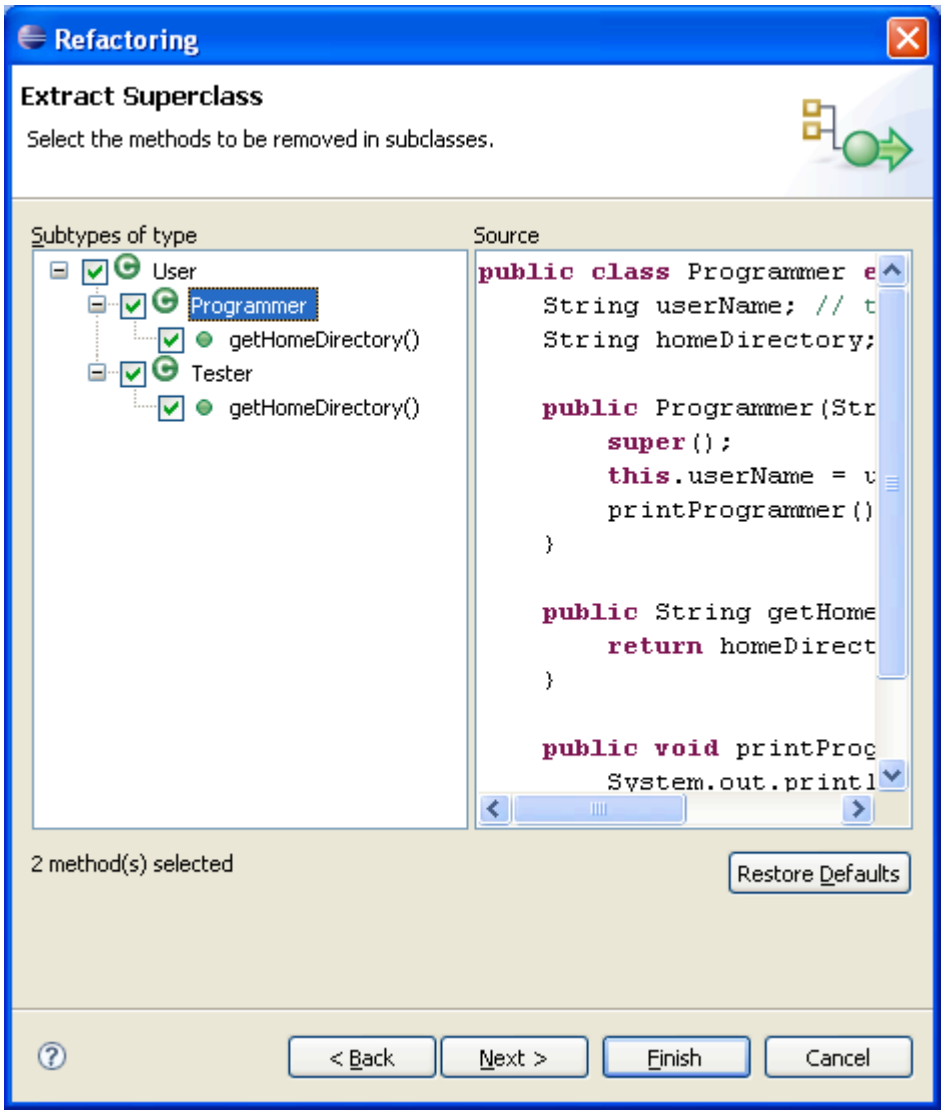
6. Change printTester() to print “A new Tester”.

Extracting a superclass

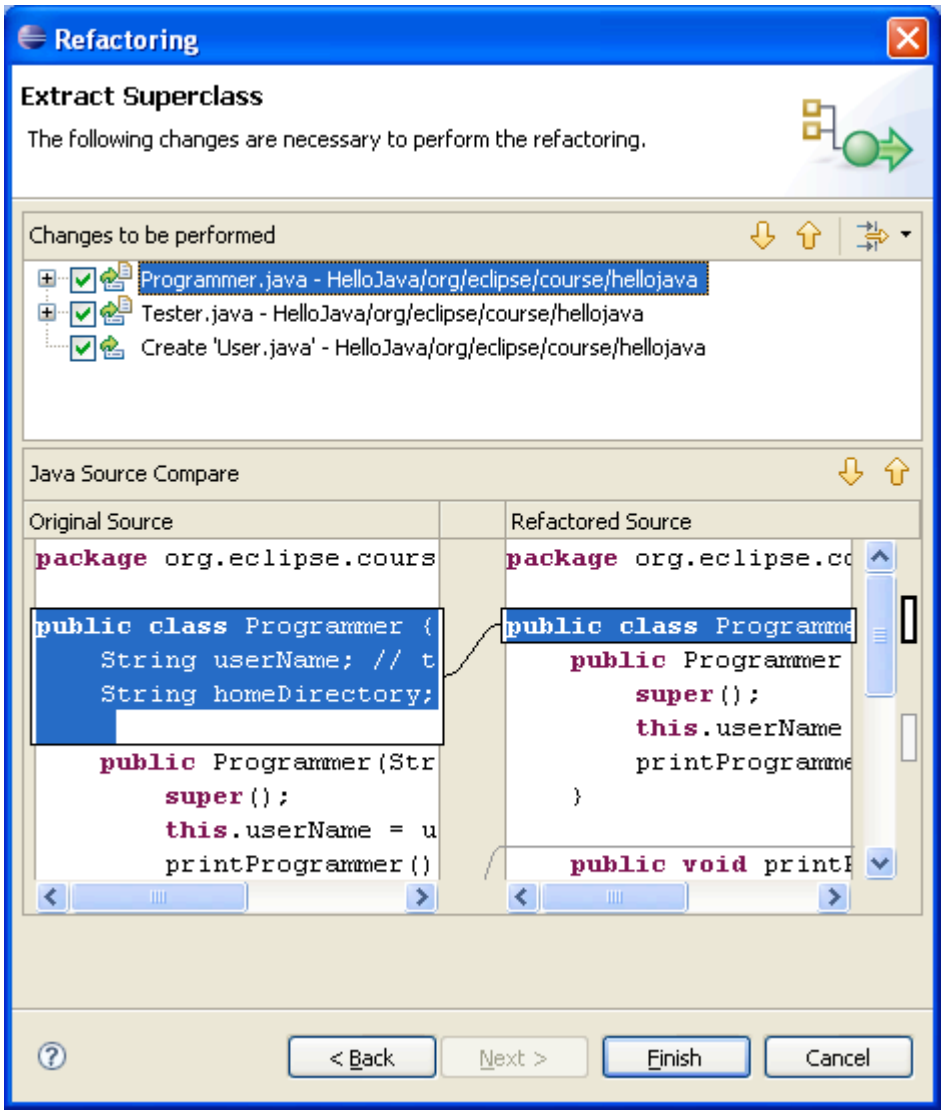
1. Select the word Tester in the class declaration and right-click it, and select Refactor > Extract Superclass…
2. Put User as the superclass name.
3. Click Add… to add Programmer type.
4. Click the checkboxes of userName, homeDirectory, and getHomeDirectory().



5. Click the checkboxes of Programmer and getHomeDirectory().



6. You can review the changes to be performed. Make sure that all checkboxes are clicked.



7. Inspect Programmer.java, Tester.java, and User.java.

Using other refactor menu items

Rename: Renames the selected element and (if enabled) corrects all references to the elements (also in other files).

Move: Moves the selected elements and (if enabled) corrects all references to the elements (also in other files).

Change Method Signature: Changes parameter names, parameter types, parameter order and updates all references to the corresponding method. Additionally, parameters can be removed or added and method return type as well as its visibility can be changed.

Extract Method: Creates a new method containing the statements or expression currently selected and replaces the selection with a reference to the new method. This feature is useful for cleaning up lengthy, cluttered, or overly-complicated methods.

Extract Local Variable: Creates a new variable assigned to the expression currently selected and replaces the selection with a reference to the new variable.

Extract Constant: Creates a static final field from the selected expression and substitutes a field reference, and optionally rewrites other places where the same expression occurs.

Inline: Inlines local variables, methods or constants.

Convert Anonymous Class to Nested: Converts an anonymous inner class to a member class.

Convert Member Type to Top Level: Creates a new Java compilation unit for the selected member type, updating all references as needed. For non-static member types, a field is added to allow access to the former enclosing instance, if necessary.

Convert Local Variable to Field: Turn a local variable into a field. If the variable is initialized on creation, then the operation moves the initialization to the new field's declaration or to the class's constructors.

Extract Superclass: Extracts a common superclass from a set of sibling types. The selected sibling types become direct subclasses of the extracted superclass after applying the refactoring.

Extract Interface: Creates a new interface with a set of methods and makes the selected class implement the interface.

Use Supertype: Where Possible Replaces occurrences of a type with one of its supertypes after identifying all places where this replacement is possible.

Push Down: Moves a set of methods and fields from a class to its subclasses.

Pull Up: Moves a field or method to a superclass of its declaring class or (in the case of methods) declares the method as abstract in the superclass.

Introduce Indirection: Creates a static indirection method delegating to the selected method.

Introduce Factory: Creates a new factory method, which will call a selected constructor and return the created object. All references to the constructor will be replaced by calls to the new factory method.

Introduce Parameter: Replaces an expression with a reference to a new method parameter, and updates all callers of the method to pass the expression as the value of that parameter.

Encapsulate Field: Replaces all references to a field with getting and setting methods.

Generalize Declared Type: Allows the user to choose a supertype of the reference's current type. If the reference can be safely changed to the new type, it is.

Infer Generic Type Arguments: Replaces raw type occurrences of generic types by parameterized types after identifying all places where this replacement is possible.

Migrate JAR File: Migrates a JAR File on the build path of a project in your workspace to a newer version, possibly using refactoring information stored in the new JAR File to avoid breaking changes.

Create Script: Creates a script of the refactorings that have been applied in the workspace. Refactoring scripts can either be saved to a file or copied to the clipboard. See Apply Script.

Apply Script: Applies a refactoring script to projects in your workspace. Refactoring scripts can either be loaded from a file or from the clipboard. See Create Script.

History: Browses the workspace refactoring history and offers the option to delete refactorings from the refactoring history.

Using other source menu items

Toggle Comment: Comments or uncomments all lines containing the current selection.

Add Block Comment: Adds a block comment around all lines containing the current selection.

Remove Block Comment: Removes a block comment from all lines containing the current selection.

Generate Element Comment: Adds a comment to the selected element. See the Code templates preference page to specify the format of the generated comments. Available on types, fields, constructors, and methods.

Shift Right: Increments the level of indentation of the currently select lines. Only activated when the selection covers multiple lines or a single whole line.

Shift Left: Decrements the level of indentation of the currently select lines. Only activated when the selection covers multiple lines or a single whole line.

Correct Indentation: Corrects the indentation of the lines denoted by the current text selection.

Format: Uses the code formatter to format the current text selection. The formatting options are configured on the Code Formatter preference page (Window > Preferences > Java > Code Style > Formatter)

Format Element: Uses the code formatter to format the Java element comprising the current text selection. The Format Element action works on method and type level. The formatting options are configured on the Code Formatter preference page (Window > Preferences > Java > Code Style > Formatter)

Add Import: Creates an import declaration for a type reference currently selected. If the type reference if qualified, the qualification will be removed if possible. If the referenced type name can not be mapped uniquely to a type of the current project you will be prompted to specify the correct type. Add Import tries to follow the import order as specified in the Organize Import preference page.

Organize Imports: Organizes the import declarations in the compilation unit currently open or selected. Unnecessary import declarations are removed, and required import declarations are ordered as specified in the Organize Imports preference page (Window > Preferences > Java > Organize Imports). Organize imports can be executed on incomplete source and will prompt you when a referenced type name can not be mapped uniquely to a type in the current project.
You can also organize multiple compilation units by invoking the action on a package or selecting a set of compilation units.

Sort Members: Sorts the members of a type according to the sorting order specified in (Window > Preferences > Java > Appearance > Members Sort Order)

Clean up: Shows a dialog which enables you to perform various changes in order to clean up your code.

Override/Implement Methods: Opens the Override Method dialog that allows you to override or implement a method in the current type. Available on types or on a text selection inside a type.

Generate Getter and Setter: Opens the Generate Getters and Setters dialog that allows you to create Getters and Setters for fields in the current type. Available on fields and types or on a text selection inside a type.

Generate Delegate Methods: Opens the Generate Delegate Methods dialog that allows you to create method delegates for fields in the current type. Available on fields and types with fields.

Generate hashCode() and equals(): Opens the Generate HashCode and Equals dialog that allows you to create and control the generation of hashCode and equals methods in the current type.

Generate Constructor using Fields: Adds constructors which initialize fields for the currently selected types. Available on types, fields or on a text selection inside a type.

Add Constructor from Superclass: Adds constructors as defined in the super class for the currently selected types. Available on types or on a text selection inside a type.

Surround With: Surround the selected statements with a code template. Create your own templates on the template preference page. Further, you can use Expand Selection to from the Edit menu to get a valid selection range. Alt + Shift + Z

Externalize Strings: Opens the Externalize strings wizard. This wizards allows you to replace all strings in the code by statements accessing a property file.

Find Broken Externalized Strings: Find broken externalized strings.