# Refactor Actions

Refactor menu commands:

| Name | Description |
|---|---|
| Rename | Renames the selected element and (if enabled) corrects all references to the elements (also in other files).<br>Available: Methods, method parameters, fields, local variables, types, type parameters, enum constants, compilation units, packages, source folders, projects and on a text selection resolving to one of these element types<br>Shortcut: Alt + Shift + R<br>Options: Renaming a type does allow to rename similarly named variables and methods. Enable 'Update similarly named variables and methods' in the Rename Type dialog. Select 'Configure...' to configure the strategy for matching type names.<br>Renaming a package does allow to rename its subpackages. Enable 'Rename subpackages' in the Rename Package dialog.<br>Enable 'Keep original method as delegate to changed method' to keep the original method. Optionally you can deprecate the old method. |
| Move | Moves the selected elements and (if enabled) corrects all references to the elements (also in other files).<br>Available: Instance method (which can be moved to a component), one or more static methods, static fields, types, compilation units, packages, source folders and projects and on a text selection resolving to one of these element types<br>Shortcut: Alt + Shift + V<br>Options: You can use Drag & Drop in the Package Explorer to start this refactoring. |
| Change Method Signature | Changes parameter names, parameter types, parameter order and updates all references to the corresponding method. Additionally, parameters and thrown exceptions can be removed or added and method return type and method visibility can be changed.<br>Available: Methods or on text selection resolving to a method<br>Shortcut: Alt + Shift + C<br>Options: Enable 'Keep original method as delegate to changed method' in the Change Method Signature dialog to keep the original method. |
| 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.<br>Available: You can use *Expand Selection to* from the Edit menu to get a valid selection range. This refactoring is also available as quick assist on statements and expressions selected in the editor.<br>Shortcut: Alt + Shift + M |
| Extract Local Variable | Creates a new variable assigned to the expression currently selected and replaces the selection with a reference to the new variable.<br>Available: Text selections that resolve to local variables. You can use *Expand Selection to* from the Edit menu to get a valid selection range.<br>This refactoring is also available as quick assist on expressions selected in the editor.<br>Shortcut: Alt + Shift + L |
| 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.<br>Available: Constant expressions or text selections which resolve to constant expressions |

| | |
|---|---|
| | This refactoring is also available as quick assist on expressions selected in the editor. |
| Inline | Inline local variables, methods or constants.<br>Available: Methods, static final fields and text selections that resolve to methods, static final fields or local variables<br>This refactoring is also available as quick assist on local variables selected in the editor.<br>Shortcut: Alt + Shift + I |
| Convert Anonymous Class to Nested | Converts an anonymous inner class to a member class.<br>Available: Anonymous inner classes |
| Move Type to New File | Creates a new Java compilation unit for the selected member type or the selected secondary 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.<br>Available: Member types, secondary types, or text resolving to a member type or a secondary type. |
| 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.<br>Available: Text selections that resolve to local variables.<br>This refactoring is also available as quick assist on local variables selected in the editor. |
| 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.<br>Available: Types<br>Options: Enable 'Use the extracted class where possible' to use the newly created class wherever possible. See *Use Supertype Where Possible*. |
| Extract Interface | Creates a new interface with a set of methods and makes the selected class implement the interface.<br>Available: Types<br>Options: Enable 'Use the extracted interface type where possible' to use the newly created interface wherever possible. See *Use Supertype Where Possible*. |
| Use Supertype Where Possible | Replaces occurrences of a type with one of its supertypes after identifying all places where this replacement is possible.<br>Available: Types |
| Push Down | Moves a set of methods and fields from a class to its subclasses.<br>Available: One or more methods and fields declared in the same type or on a text selection inside a field or method |
| 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.<br>Available: One or more methods, fields and member types declared in the same type or on a text selection inside a field, method or member type |
| Extract Class | Replaces a set of fields with new container object. All references to the fields are updated to access the new container object.<br>Available: The set of fields or a type containing fields<br>Options: Enable 'Create Getter and Setters' to add accessor methods to the new type |
| Introduce Parameter Object | Replaces a set of parameters with a new class, and updates all callers of the method to pass an instance of the new class as the value to the introduce parameter.<br>Available: Methods or on text selection resolving to a method<br>Options: Enable 'Keep original method as delegate to changed method' in the Introduce Parameter Object dialog to keep the original method. |

| | |
|---|---|
| Introduce Indirection | Creates a static indirection method delegating to the selected method.<br>Available: Methods or on text selection resolving to a method<br>Options: Enable 'Redirect all method invocations' to replace all calls to the original method by calls to the indirection 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.<br>Available: Constructor declarations |
| 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.<br>Available: Text selections that resolve to expressions |
| Encapsulate Field | Replaces all references to a field with getter and setter methods.<br>Available: Field or a text selection resolving to a field.<br>This refactoring is also available as quick assist on field declarations and references selected in the editor. |
| 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.<br>Available: Type references and declarations of fields, local variables, and parameters with reference types |
| Infer Generic Type Arguments | Replaces raw type occurrences of generic types by parameterized types after identifying all places where this replacement is possible.<br>Available: Projects, packages, and types<br>Options: 'Assume clone() returns an instance of the receiver type'. Well-behaved classes generally respect this rule, but if you know that your code violates it, uncheck the box.<br>'Leave unconstrained type arguments raw (rather than inferring <?>)'. If there are no constraints on the elements of e.g. ArrayList a, uncheck this box will cause Eclipse to still provide a wildcard parameter, replacing the reference with ArrayList<?>. |
| 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.<br>Available: JAR Files on build path |
| 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*.<br>Available: Always |
| 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*.<br>Available: Always |
| History | Browses the workspace refactoring history and offers the option to delete refactorings from the refactoring history.<br>Available: Always |

Refactoring commands are also available from the context menus in many views and the Java editor.

### Related concepts

Refactoring support

### Related references

Refactoring dialogs
Java preferences