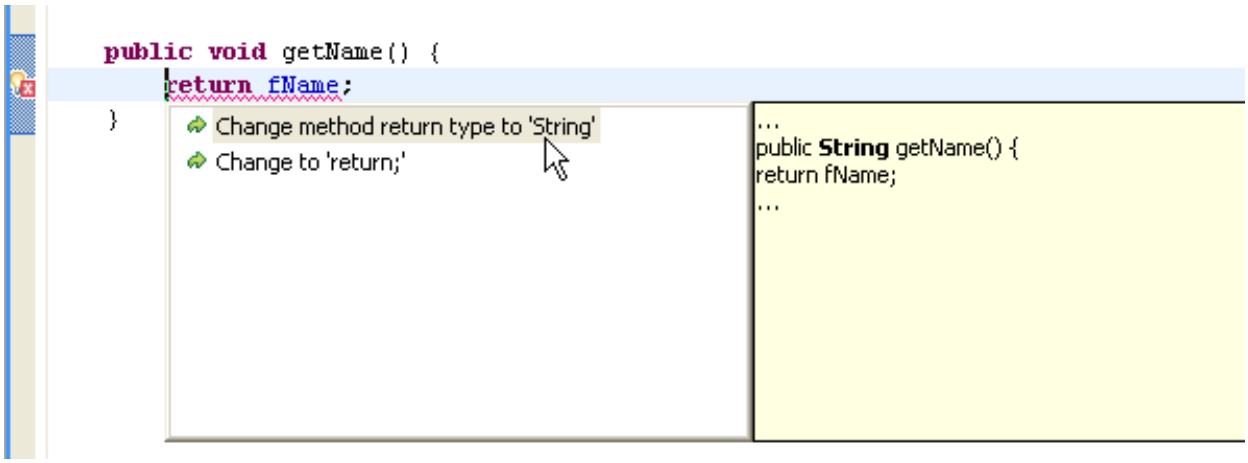
Java development user guide > Reference

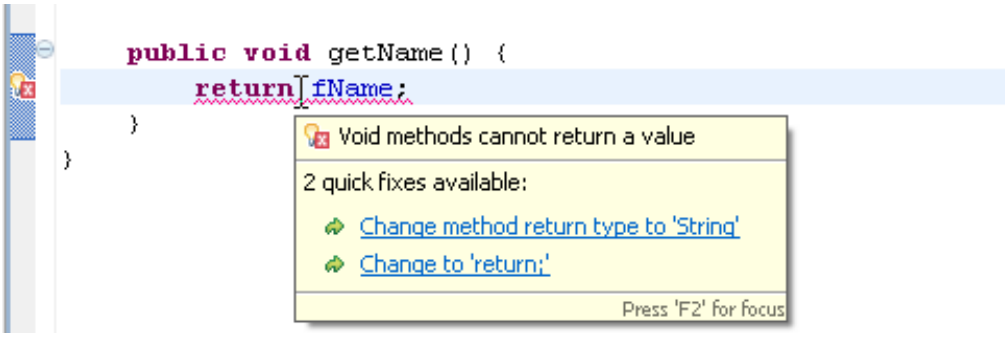
Quick Fix

The Java editor offers corrections to problems found while typing and after compiling. To show that correction proposals are available for a problem or warning, a 'light bulb' is visible on the editor's annotation bar.

Left click on the light bulb or invoking **Ctrl+1 (Edit > Quick Fix)** brings up the proposals for the problem at the cursor position.
Each quick fix shows a preview when selected in the proposal window.



Quick fixes are also shown directly in problem hovers (but there, no preview is available).



Usage hint: Quick fixes are not only useful to fix errors that accidentally occurred. An other common usage pattern is to inentionally write "incorrect" code, for example by referring to a local variable that is not declared yet. Then, the **Create local variable** Quick Fix can generate the declaration in no time, and it can even infer the variable type.

Some selected quick fixes can also be assigned with direct shortcuts. You can configure these shortcuts on the **General > Keys** preference page (in the 'Source' category).

Some quick fixes offer to fix all problems of the same kind in the current file at once. The information text in the proposal window contains this information for all applicable proposals. To fix all problems of the same kind, press **Ctrl+Enter**.

Here's a selection of available quick fixes:

Package Declaration	• Add missing package declaration or correct package declaration • Move compilation unit to package that corresponds to the package declaration
Imports	• Remove unused, unresolvable or non-visible import • Invoke 'Organize imports' on problems in imports

Types

- Create new class, interface, enum, annotation or type variable for references to types that can not be resolved
- Change visibility for types that are accessed but not visible
- Rename to a similar type for references to types that can not be resolved
- Add import statement for types that can not be resolved but exist in the project
- Add explicit import statement for ambiguous type references (two import-on-demands for the same type)
- If the type name is not matching with the compilation unit name either rename the type or rename the compilation unit
- Remove unused private types
- Add missing type annotation attributes

Constructors

- Create new constructor for references to constructors that can not be resolved (this, super or new class creation)
- Reorder, add or remove arguments for constructor references that mismatch parameters
- Change method with constructor name to constructor (remove return type)
- Change visibility for constructors that are accessed but not visible
- Remove unused private constructor
- Create constructor when super call of the implicit default constructor is undefined, not visible or throws an exception
- If type contains unimplemented methods, change type modifier to 'abstract' or add the method to implement

Methods

- Create new method for references to methods that can not be resolved
- Rename to a similar method for references to methods that can not be resolved
- Reorder or remove arguments for method references that mismatch parameters
- Correct access (visibility, static) of referenced methods
- Remove unused private methods
- Correct return type for methods that have a missing return type or where the return type does not match the return statement
- Add return statement if missing
- For non-abstract methods with no body change to 'abstract' or add body
- For an abstract method in a non-abstract type remove abstract modifier of the method or make type abstract
- For an abstract/native method with body remove the abstract or native modifier or remove body
- Change method access to 'static' if method is invoked inside a constructor invocation (super, this)
- Change method access to default access to avoid emulated method access
- Add 'synchronized' modifier
- Override hashCode()
- Open the 'Generate hashCode() and equals()' wizard

Fields and variables

- Correct access (visibility, static) of referenced fields
- Create new fields, parameters, local variables or constants for references to variables that can not be resolved

- Rename to a variable with similar name for references that can not be resolved
- Remove unused private fields
- Correct non-static access of static fields
- Add 'final' modifier to local variables accessed in outer types
- Change field access to default access to avoid emulated method access
- Change local variable type to fix a type mismatch
- Initialize a variable that has not been initialized
- Create getter and setters for invisible or unused fields
- Create loop variable to correct an incomplete enhanced 'for' loop by adding the type of the loop variable

Exception Handling

- Remove unneeded catch block
- Remove unneeded exceptions from a multi-catch clause (1.7 or higher)
- Handle uncaught exception by surrounding with try/catch or adding catch block to a surrounding try block
- Handle uncaught exceptions by surrounding with try/multi-catch or adding exceptions to existing catch clause (1.7 or higher)
- Handle uncaught exception by adding a throw declaration to the parent method or by generalize an existing throw declaration

Build Path Problems

- Add a missing JAR or library for an unresolvable type
- Open the build path dialog for access restriction problems or missing binary classes.
- Change project compliance and JRE to 1.5
- Change workspace compliance and JRE to 1.5

Others

- Add cast or change cast to fix type mismatches
- Let a type implement an interface to fix type mismatches
- Add type arguments to raw references
- Complete switch statements over enums
- Remove dead code
- Insert '//$FALL-THROUGH$'
- Insert null check
- For non-NLS strings open the NLS wizard or mark as non-NLS
- Add missing @Override, @Deprecated annotations
- Add missing Javadoc comments
- Add missing Javadoc tags
- Suppress a warning using @SuppressWarnings
- Throw the allocated object
- Return the allocated object
- Add @SafeVarargs annotation for heap pollution warnings (1.7 or higher)
- Remove invalid @SafeVarargs annotations (1.7 or higher)
- Remove redundant type arguments (1.7 or higher)
- Add inferred type arguments (1.5 and 1.6)
- Remove unused type parameter (1.5 or higher)

Quick Assists are proposals available even if there is no problem or warning. See the Quick Assist page for more information.

Related concepts

Java editor
Quick fix and assist

Related reference

Quick Assist
JDT actions