Quick Assist

Quick assists perform local code transformations. They are invoked on a selection or a single cursor in the Java editor and use the same shortcut as quick fixes (**Ctrl+1**), but quick assist are usually hidden when an error is around. To show them even with errors present on the same line, press **Ctrl+1** a second time.

A selection of quick assists can be assigned to a direct shortcut. By default, these are:

- Rename in file: **Ctrl+2, R**
- Assign to local: **Ctrl+2, L**
- Assign to field: **Ctrl+2, F**

Assign more shortcuts or change the default shortcuts on the **General > Keys** preference page (in the 'Source' category).

A quick assist light bulb can be turned on on the **Java > Editor** preference page.

Name	Code example			Invocation location
Inverse if statement	`if (x) a(); else b();`	>	`if (!x) b(); else a();`	On 'if' statements with 'else' block
Convert to if-!-return	`if (x == 1) a();`	>	`if (x != 1) return;` `a();`	On an 'if' statement
Inverse boolean expression	`a && !b`	>	`!a || b`	On a boolean expression
Invert local variable	`boolean a = false;` `if (a) {}`	>	`boolean notA = true;` `if (!notA) {}`	On a boolean variable
Invert equals	`a.equals(b)`	>	`b.equals(a)`	On a invocation of 'equals'
Inverse conditional expression	`x ? b : c`	>	`!x ? c : b`	On a conditional expression
Pull negation up	`b && c`	>	`!(!b || !c)`	On a boolean expression
Push negation down	`!(b && c)`	>	`!b || !c`	On a negated boolean expression
Remove extra parentheses	`if ((a == b) && (c != d) {}`	>	`if (a == b && c != d) {}`	On selected expressions
Put expression in parentheses	`return a > 10 ? 1 : 2;`	>	`return (a > 10) ? 1 : 2;`	On selected expression
Put expressions in parentheses	`if (a == b && c != d) {}`	>	`if ((a == b) && (c != d)) {}`	On selected expressions

Join nested if statements	`if (a) { if (b) {} }`	>	`if (a && b) {}`	On a nested if statement
Swap nested if statements	`if (a) { if (b) {} }`	>	`if (b) { if (a) {} }`	On a nested if statement
Split if statement with and'ed expression	`if (a && b) {}`	>	`if (a) { if (b) {} }`	On an and'ed expression in a 'if'
Join selected 'if' statements with ||	`if (a) x();  if (b) x();`	>	`if (a || b) x();`	On selected 'if' statements
Join 'if' sequence in if-else-if	`if (a) x();  if (b) y();`	>	`if (a) x();  else if (b) y();`	On selected 'if' statements
Split if statement with or'd expression	`if (a || b) x();`	>	`if (a) x();  if (b) x();`	On an or'd expression in a 'if'
If-else assignment to conditional expression	`if (a) x= 1; else x= 2;`	>	`x= a ? 1 : 2;`	On an 'if' statement
If-else return to conditional expression	`if (a) return 1;` `else return 2;`	>	`return a ? 1 : 2;`	On an 'if' statement
Conditional expression assignment to If-else	`x= a ? 1 : 2;`	>	`if (a) x= 1; else x= 2;`	On a conditional expression
Conditional expression return to If-else	`return  a ? 1 : 2;`	>	`if (a) return 1; else return 2;`	On a conditional expression

Switch to If-else	`switch (kind) {`
	`case 1: return -1;`
	`case 2: return -2;`
	`}`
		>	`if (kind == 1) {`
			`  return -1;`
			`} else if (kind == 2) {`
			`  return -2;`
			`}`
				On a switch statement

Convert if-else to switch	`if (kind == 1) {`
	`  return -1;`
	`} else if (kind == 2) {`
	`  return -2;`
	`}`
		>	`switch (kind) {`
			`case 1: return -1;`
			`case 2: return -2;`
			`}`
				On an 'if' statement

Add missing case statements on enums	`switch (e){`
	`}`
		>	`switch (e){`
			`case E1: break;`
			`case E2: break;`
			`}`
				On a switch statement

Exchange operands	`a + b`	>	`b + a`	On an infix operation
Cast and	`if (obj instanceof Vector) {`	>	`if (obj instanceof Vector) {`	On an

assign	`}`		`Vector vec= (Vector)obj;` `}`	instanceof expression in an 'if' or 'while' statement

Use separate catch blocks	`try {`
	`} catch (FileNotFoundException | InterruptedIOException e) {`
	`}`
		>	`try {`
			`} catch (FileNotFoundException e) {`
			`} catch (InterruptedIOException e) {`
			`}`
				On a multi-catch block (1.7 or higher)

Move exceptions to separate catch blocks	`try {`
	`} catch (FileNotFoundException | InterruptedIOException | IllegalArgumentException e) {`
	`}`
		>	`try {`
			`} catch (FileNotFoundException e) {`
			`} catch (InterruptedIOException | IllegalArgumentException e) {`
			`}`
				On exceptions in a multi-catch clause (1.7 or higher)

Combine catch blocks	`try {`
	`} catch (FileNotFoundException e) {`
	`} catch (InterruptedIOException e) {`
	`}`
		>	`try {`
			`} catch (FileNotFoundException | InterruptedIOException e) {`
			`}`
				On a catch block (1.7 or higher)

Add finally block	`try {`
	`} catch (Exception e) {`
	`}`
		>	`try {`
			`} catch (Exception e) {`
			`} finally {}`
				On a try/catch statement

Add else block	`if (a) b();`	>	`if (a) b(); else { }`	On a if statement
Replace statement with block	`if (a) b();`	>	`if (a) { b(); }`	On a if statement
Unwrap blocks	`{ a() }`	>	`a()`	On blocks, if/while/for statements
Combine to single String	`String phrase= "one" + " two " + "three";`	>	`String phrase= "one two three";`	On a string concatenation expression
Pick out string	`"abcdefgh"`	>	`"abc" + "de" + "fgh"`	select a part of a string literal

Convert string concatenation to StringBuilder (J2SE 5.0) or StringBuffer	`"Hello " + name`	>	`StringBuilder builder= new StringBuilder();`
			`builder.append("Hello ");`
			`builder.append(name);`
				select a string literal

Convert string concatenation to MessageFormat	`"Hello " + name`	>	`MessageFormat.format("Hello {0}", name);`	select a string literal

Split variable	`int i= 0;`	>	`int i; i= 0;`	On a variable with initialization
Join variable	`int i; i= 0;`	>	`int i= 0`	On a variable without initialization
Assign to variable	`foo()`	>	`X x= foo();`	On an expression statement
Extract to local	`foo(getColor());`	>	`Color color= getColor();` `foo(color);`	On an expression

Assign parameter to field	`public A(int color) {}`	>	`Color fColor;`
			`public A(int color) {`
			`  fColor= color;`
			`}`
				On a parameter

Array initializer to Array creation	`int[] i=  { 1, 2, 3 }`	>	`int[] i= new int[] { 1, 2, 3 }`	On an array initializer

Create 'for' loops	`void foo(Map<String, Integer> map) {`
	`  map.keySet();`
	`}`
		>	`void foo(Map<String, Integer> map) {`
			`  for (String string : map.keySet()) {`
			`  }`
			`}`
				On arrays, `Collection`s and `List`s

Convert to 'enhanced for loop' (J2SE 5.0)	`for (Iterator i= c.iterator();i.hasNext();) {` `}`	>	`for (x : c) {` `}`	On a for loop

Convert to indexed 'for' loop (J2SE 5.0)	`for (x : c) {`
	`}`
		>	`for (int i = 0; i < c.size(); i++) {`
			`  x = c[i];`
			`}`
				On an enhanced for loop

Convert to Iterator-based 'for' loop (J2SE 5.0)	`for (x : c) {` `}`	>	`for (Iterator i= c.iterator();i.hasNext();) {` `}`	On an enhanced for loop
Create method in super class				On a method declaration
Rename in file				On identifiers
Rename in workspace				On identifiers
Extract to local variable	`a= b*8;`	>	`int x= b*8;` `a= x;`	On expressions
Extract to constant	`a= 8;`	>	`final static int CONST= 8;` `a= CONST;`	On expressions
Extract method	`int x= p * 5;`	>	`int x= getFoo(p);`	On expressions

				and statements
Inline local variable	`int a= 8, b= a;`	>	`int b= 8;`	On local variables
Convert local variable to field	`void foo() { int a= 8; }`	>	`int a= 8; void foo() {}`	On local variables
Convert anonymous to nested class	`new Runnable() { };`	>	`class RunnableImplementation implements Runnable { }`	On anonymous classes

Convert to lambda expression	`Runnable r= new Runnable() {`
	`  public void run() {}`
	`};`
		>	`Runnable r= () -> {};`	On anonymous classes implementing a functional interface (1.8 or higher)

Convert to anonymous class creation	`Runnable r= () -> {};`	>	`Runnable r= new Runnable() {`
			`  public void run() {}`
			`};`
				On lambda expressions (1.8 or higher)

Convert to lambda expression	`Consumer<Integer> c= System.out::println;`	>	`Consumer<Integer> c= t -> System.out.println(t);`	On method references (1.8 or higher)
Convert to method reference	`Consumer<Integer> c= t -> System.out.println(t);`	>	`Consumer<Integer> c= System.out::println;`	On lambda expressions (1.8 or higher)

Change body expression to block	`Runnable r= () -> System.out.println();`	>	`Runnable r= () -> {`
			`  System.out.println();`
			`};`
				On lambda expressions with body as expression (1.8 or higher)

Change body block to expression	`Runnable r= () -> {`
	`  System.out.println();`
	`};`
		>	`Runnable r= () -> System.out.println();`	On lambda expressions with body as block (1.8 or higher)

Add inferred lambda parameter types	`Consumer<Integer> c= t -> System.out.println(t);`	>	`Consumer<Integer> c= (Integer t) -> System.out.println(t);`	On lambda expressions with inferred parameter types (1.8 or higher)
Add parentheses around lambda parameter	`Consumer<Integer> c= t -> System.out.println(t);`	>	`Consumer<Integer> c= (t) -> System.out.println(t);`	On lambda expressions (1.8 or higher)
Remove parentheses around lambda parameter	`Consumer<Integer> c= (t) -> System.out.println(t);`	>	`Consumer<Integer> c= t -> System.out.println(t);`	On lambda expressions (1.8 or higher)

Replace with getter and setter (Encapsulate Field)	`p.x;`	>	`p.getX();`	On fields
Insert inferred type arguments	`List<String> list = new ArrayList<>();`	>	`List<String> list = new ArrayList<String>();`	On generic instance creation expressions (1.7 or higher)

The following quick assists are available in the **Properties File Editor**:

- **Rename in workspace** - renames the key in the properties file and updates all references
- **Create field in '...'** - creates the corresponding field in the resource bundle accessor class
- **Remove property** - deletes the property from the properties file and the field from the resource bundle accessor class
- **Remove properties** - deletes the selected properties from the properties file and the fields from the resource bundle accessor class
- **Escape backslashes** - escape all backslashes in the selected text
- **Escape backslashes in original string** - escape all backslashes in the pasted text
- **Unescape backslashes** - unescape all backslashes in the selected text

Related concepts

Java Editor
Quick Fix and Quick Assist

Related reference

Quick Fix
JDT actions