Refactoring: Extract a local variable

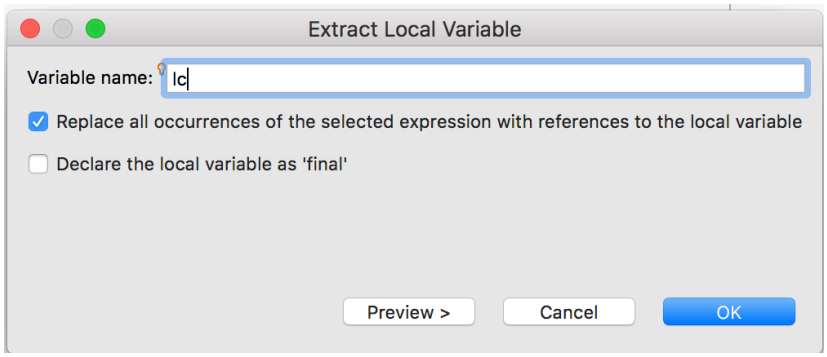
Sometimes, we notice that an expression is repeated several times, and in order to simplify the program and make it more efficient, we want to introduce a local variable to contain the value of the expression and evaluate the expression only once. Eclipse has a refactoring tool that makes this easy to do.

In nonsense method *m* to the right, the expression *b*c* appears three times. So, we:

1. Select or highlight one of them.

2. Choose menu item *Refactor -> Extract local variable*. This causes this window to pop up:

```
public void m() {
    double b= 5;
    double c=7;
    if (b*c < 8) {
        double d= b*c;
    }
    b= b*c;
}
```



We already typed in a name for the local variable: lc.

The default is to replace all occurrences of the selected expression. Uncheck the box if you want only the selected expression to be replaced.

You can also declare the local variable as *final* so that it cannot be changed.

When we hit button *OK*, method *m* is changed, as shown on the right. The assignment of *b*c* to *lc* was inserted and all occurrences of *b*c* were replaced by *lc*.

```
public void m() {
    double b= 5;
    double c=7;
    double lc = b*c;
    if (lc < 8) {
        double d= lc;
    }
    b= lc;
}
```

Be careful!

Procedure *m*, to the right, is the same as the one above except that the statement shown by the arrow has been added. Because the statement before it changes the value of *b*, the expressions *b*c* in these two statements have different values. When using the *Extract local variable* tool as above, all the occurrences of *b*c*, including the one marked by the arrow, will be replaced by variable *lc*. So, the final values of *b* before and after refactoring are different.

```
public void m() {
    double b= 5;
    double c=7;
    if (b*c < 8) {
        double d= b*c;
    }
    b= b*c;
    b= b*c;  <-----
}
```

The moral of the story is that the result of refactoring a method in any way must be checked carefully to make sure the intended effect was achieved. Hopefully, in a real setting, a Junit testing procedure was written to test the method, and retesting will show any errors.