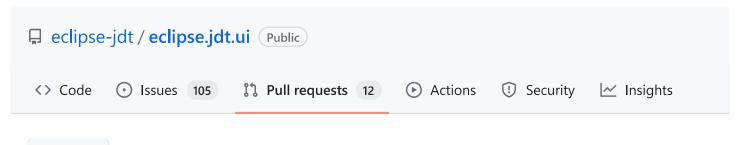
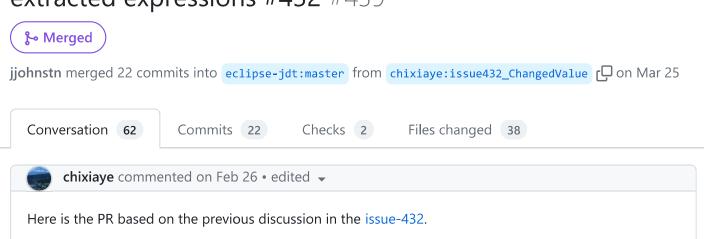
<> Code ▼



Improve the Safety of "Extract Local Variable" refactorings by identifying statements that may change the value of the extracted expressions #432 #439



What it does

changedValueChecker is the core part that is used to identify statements that may change the value of the extracted expressions. It first records the values that the selected expression depends on during the initialization phase. Then it calculates the path between the temp declaration and the last candidate extracted expression. Lastly, it validates intermediate statements in parallel to detect the intersection of the value updated by the intermediate code and the value read by the expression without duplication.

For <code>ExtractTempRefactoring.java</code>, in addition to the call of the <code>ChangedValueChecker</code> class, we have changed the original refactoring strategy when replacing the selected expression with temp declaration. Considering such example:

```
1 {
2  f();
3  int x=f();
4 }
5 {
6  int y=f();
7 }
```

If we select f() at Line 2, the refactored code shown as follows has obviously syntax errors:

```
1 {
2  int f=f();
3  int x=f;
```

Jump to bottom

```
4 }
5 {
6 int y=f;
7 }
```

I think it is better to extract only once for the selected expression used as a statement. Because such an expression tends to modify the system state (otherwise it would be useless).

How to test

Tests have been appended in ExtractTempTests to verify the correctness of the patch. Test-140 is added to test for a variable the selected expression depends on has been changed within the method invocation. Test-141/Test-142 is to test the case of loop/condition statement to ensure that the path is not just calculated on the basis of offsets. Test-143, Test-144, and Test-145 were designed to test complex scenarios to ensure the robustness of the code concerning calculating the path. Test-146 and Test-147 were added to verify the identification of changed values under different cases.

Notably, we disabled Test-46, Test-110, and Test-111 because of the changed refactoring strategy. Besides, Test-73 was also excluded due to the added more precise flow analysis.

Author checklist

- I have thoroughly tested my changes
- ☑ The change is following the coding conventions
- ✓ I have signed the Eclipse Contributor Agreement (ECA)





[2] **(a) liuhuigmail** mentioned this pull request on Feb 28

Improve the Safety of "Extract Local Variable" refactorings by identifying statements that may change the value of the extracted expressions #432





jjohnstn requested changes on Mar 2

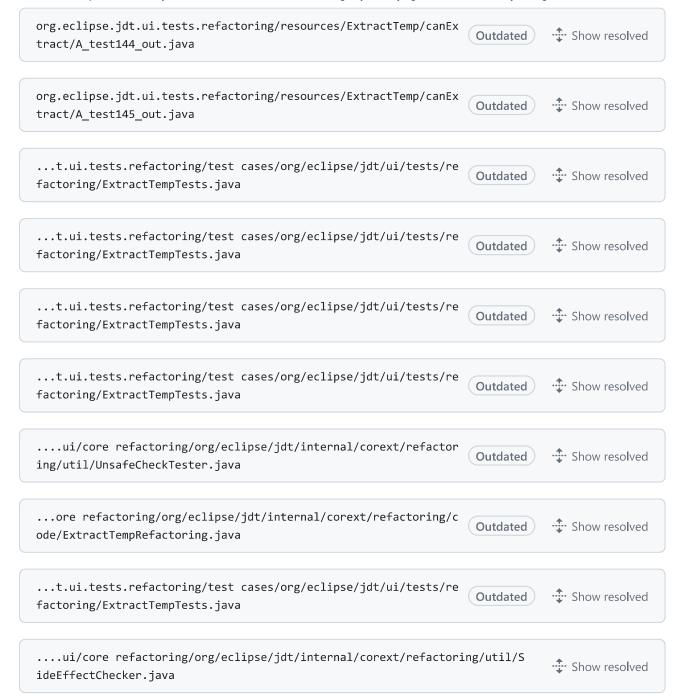
View reviewed changes



jjohnstn left a comment

There is a question of logic brought up by a few tests that you have chosen to ignore that I believe should work as there are no state changes involved. I will wait until those are answered to do more indepth reviewing as the code may change. There are other minor coding issues.





chixiaye commented on Mar 5

@jjohnstn Thanks for your kind comments. In the latest version, I revised the following parts:

- 1. I updated the naming rules for introducing multiple new variables in one refactoring. This rule is that the smaller the offset value of the variable declaration position, the smaller the additional number, and the variable without an extra number will always come first.
- 2. For the disabled tests. f method in Test-46 will have StackOverflowError at runtime. So I replaced f method with a more reasonable method. Besides, as you suggested, I have modified the Test-73 to be valid. As for Test-110 and Test-111, calculateCount(); is a statement that does nothing. Because its return-value does not read by any variable/object and calculateCount() method does not change the state of the program. I think that most of the time, only one method call in a single-line statement indicates that such a method has side effects, I suggest only replacing this expression and keeping other occurrences. So I modified the implementation of calculateCount() to make the tested code look closer to actual development.

3. For the change descriptions, I updated all the descriptions(including previous PRs) to make sure they are consistent with the patches.





jjohnstn requested changes on Mar 9

View reviewed changes



jjohnstn left a comment

Thanks. Looks much better. Good new tests of the new functionality. Just a few old tests to put back and a new recursive test to add. Please reply to each review comment either by stating it is done and then we can resolve the comment or add some information if you don't agree with the comment or there is further information required.







jjohnstn requested changes on Mar 11

View reviewed changes

org.eclipse.jdt.ui.tests.refactoring/resources/ExtractTemp/canEx tract/A_test110_out.java

Outdated

Outdated

iloveeclipse commented on Mar 17

@chixiaye: unfortunately I saw this a bit late.

I see you are always doing merge of master into your branch. Please avoid doing that, instead please rebase your branch on top of master. This is needed to have a clean git history state.



chixiaye commented on Mar 17

@chixiaye: unfortunately I saw this a bit late. I see you are always doing merge of master into your branch. Please avoid doing that, instead please rebase your branch on top of master. This is needed to have a clean git history state.

Okay. Thanks for the reminder. I am sorry and promise this won't happen again.





chixiaye force-pushed the issue432_ChangedValue branch from be15ebe to a2c939c

Compare

2 months ago

- chixiaye requested a review from jjohnstn 2 months ago
- jjohnstn requested changes on Mar 22

View reviewed changes

...t.ui.tests.refactoring/test cases/org/eclipse/jdt/ui/tests/re Outdated Show resolved factoring/ExtractTempTests.javaui/core refactoring/org/eclipse/jdt/internal/corext/refactoring/util/U • Show resolved nsafeCheckTester.java ...t.ui.tests.refactoring/test cases/org/eclipse/jdt/ui/tests/re Outdated Show resolved factoring/ExtractTempTests.java org.eclipse.jdt.ui.tests.refactoring/resources/ExtractTemp/canEx Outdated Show resolved tract/A_test110_out.java

jjohnstn approved these changes on Mar 22

View reviewed changes

jjohnstn requested changes on Mar 22

View reviewed changes



jjohnstn left a comment

Thanks for the changes. The code is working as expected and tests are working without disablement. I have added some review comments/questions regarding the ChangedValueChecker. I also have one test below that isn't working as expected:

```
public void foo(int k) {
        {
                 f();
                 switch (k) {
                 case 3:
                          f();
                          int x=f();
                          break;
                 default:
                          int y=f();
                 }
        }
}
```

The test works fine if I choose the top f() call. If I choose the one in case 3: it does not change the following statement declaring x for some reason.



...i/core refactoring/org/eclipse/jdt/internal/corext/refactoring/util/Changed ValueChecker.java



```
137
                                   });
138
                          }
139
                          try {
140
                                   threadPool.shutdown();
```



👭 jjohnstn on Mar 22

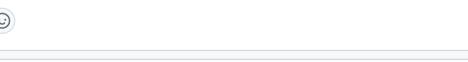
The orderly shutdown could take time. A check should be made for fConflict being true and then just do a shutdownNow(). Have you done any performance checks on a large routine being checked?





chixiaye on Mar 23

I added the new check for fconflict . awaitTermination can ensure the time cost is limited to less than 5s. shutdownNow is used in finally block to avoid other exceptions. As for time performance, about 97% of refactorings can be finished within 5 seconds according to the evaluation in real projects in the previous experiment. Considering the submitted patch is much more simple than our original implementation, the actual time cost would be less.





 $... i/core\ refactoring/org/eclipse/jdt/internal/corext/refactorin\ g/util/ChangedValueChecker.java$

Outdated •• Show resolved

...i/core refactoring/org/eclipse/jdt/internal/corext/refactorin
g/util/ChangedValueChecker.java

Outdated \$\frac{1}{4}\$ Show resolved

...i/core refactoring/org/eclipse/jdt/internal/corext/refactoring/util/ChangedValueChecker.java

\$\tag{\tag{\tag{theta}}}\$ Show resolved

...i/core refactoring/org/eclipse/jdt/internal/corext/refactoring/util/Cha
ngedValueChecker.java

• Show resolved

...i/core refactoring/org/eclipse/jdt/internal/corext/refactoring/util/Cha
ngedValueChecker.java

\$\displays \text{Show resolved}



chixiaye force-pushed the issue432_ChangedValue branch from 72b5934 to 0b94333

Compare

2 months ago

chixiaye commented on Mar 23

Thanks for the changes. The code is working as expected and tests are working without disablement. I have added some review comments/questions regarding the ChangedValueChecker. I also have one test below that isn't working as expected:

The test works fine if I choose the top f() call. If I choose the one in case 3: it does not change the following statement declaring x for some reason.

Hi, I re-checked the code and fixed this problem and attached Test-151 and Test-152 to validate.





chixiaye requested a review from jjohnstn 2 months ago

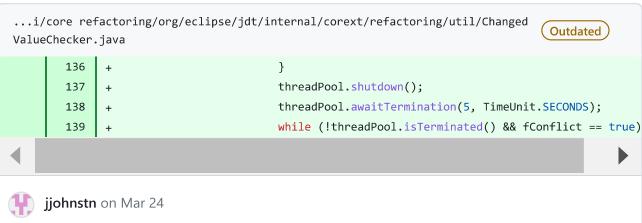


jjohnstn approved these changes on Mar 24

View reviewed changes

jjohnstn requested changes on Mar 24

View reviewed changes



Last comment. You only want to loop if fConflict is false which means that nothing has found a conflict yet. Once a conflict is found, there is no need to do any more processing or waiting for termination because nothing more will be found. That is why I suggested checking if fConflict is true before even asking for termination or shutdown since the finally clause will shutdownNow() regardless.





chixiaye on Mar 25

I think I understand what you mean. I modified the code to only call shutdown when fconflict is false, and the detection would be interrupted if fconlict is true.





jjohnstn on Mar 25

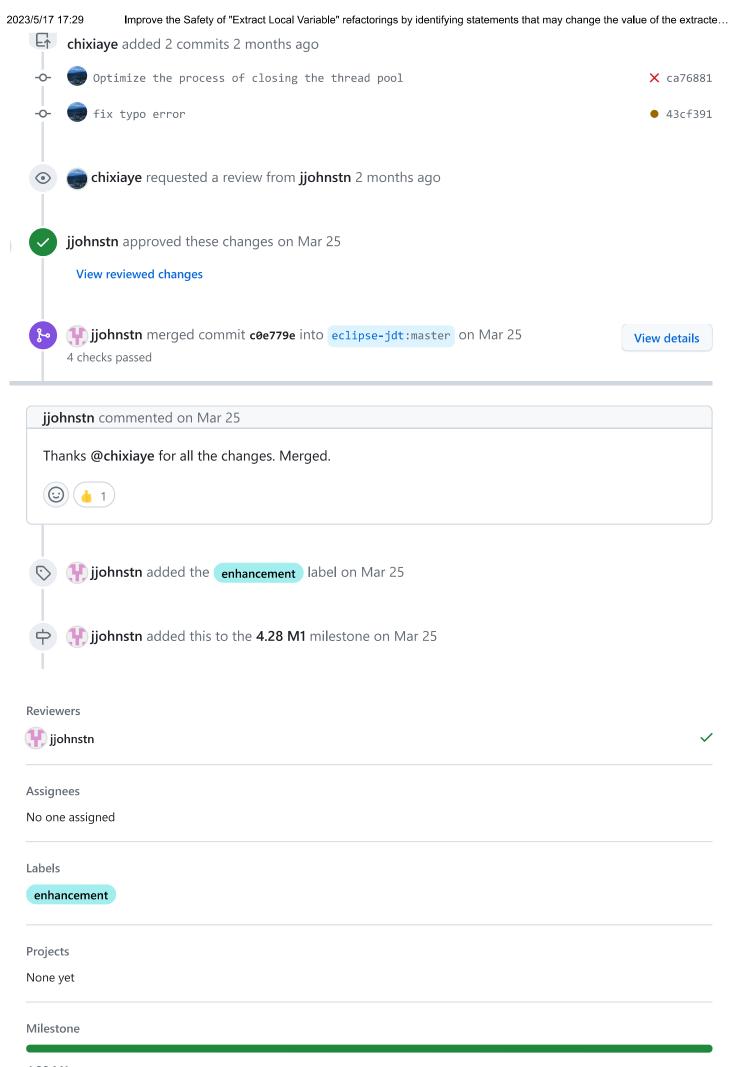
Yes, change looks good.





Reply...

	chixiaye force-pushed the issue432_ChangedValue branch from b6d7f5e to 0b94333 2 months ago	Compare
	chixiaye and others added 12 commits 2 months ago	
-	update Copyright	2443932
-	■ Improve the Safety of Extract Local Variable refactorings by identify	0294ee6
-0-	Discard two test cases	fc261e7
-0-	<pre>Update ExtractTempTests1d7.java</pre>	b1ead8b
-0-	Update ExtractTempTests1d8.java	78b0fb7
-0-	<pre>Update ExtractTempTests1d8.java</pre>	cfeea97
-0-	revision about test cases and coding issues	8da0166
-0-	solve the casuse of failed tests	4d1d6f5
-	eplace tab with space	e095a04
-0-	change test-47	7247bbb
-0-	reformat	7449095
-0-	restore tests	ba87335
	chixiaye and others added 8 commits 2 months ago	
- O -	restore the original strategy and add a new test	dc3ce7e
- O -	fix problems for AssistQuickFixTest and AdvancedQuickAssistTest	6338ee2
-0-	replace thread list with threadPool	57db006
- O -	enhanced closing threadpool's part	03b5f04
- 0-	fixed new variable scope problem while replacing	c85d9d5
- 0-	Minor corrections.	2c97b91
- 0-	Some revisions based on comments	ea32ac5
-	fix bug while extracting in switchstmts	X 569efc1
	chixiaye force-pushed the issue432_ChangedValue branch from 0b94333 to 569efc1 2 months ago	Compare



Development

Successfully merging this pull request may close these issues.

None yet

3 participants





