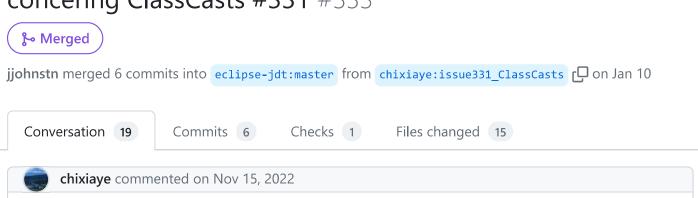


# Improve the Safety of Extract Local Variable Refactorings concering ClassCasts #331 #333



This is a patch based on the solution explained in the bug report.

The main changed file is Checker.java, the description of the modification of this file is as follows:

### 1. Implement Void ClassCasts Exception

Compared with the solution proposed in the original issue, we apply a more generalized solution. First, we traverse the selected expression and create a mapping from expression to type for each CastExpression . Second, we validate the code snippet between the variable declaration and the first expression matching the selected expression. If we detect the InstanceofExpression and value corresponding to the key value, leftOperand of InstanceofExpression, is equal to the rightOperand of InstanceofExpression or its supertype, we would assert that such extracting is unsafe.

## 2. Refactoring

Considering this class not only aims at avoiding NullPointerException but also ClassCastExecption, we renamed it to Check and renamed the inner class NullMiddleCodeVisitor to MiddleCodeVisitor. Besides, we removed some private fields of the inner classes to make the code more concise.



[2] Iiuhuigmail mentioned this pull request on Nov 25, 2022

Incorrect "Extract Variable" refactoring, resulting in NullPointerException #39



jjohnstn requested changes on Dec 1, 2022

View reviewed changes



jjohnstn left a comment

Looks good but a few comments to address.



```
...clipse.jdt.ui/core refactoring/org/eclipse/jdt/internal/corext/refactoring/
util/Checker.java

233 + this.castFlag= true;
234 + return false:
```

```
233 + this.castFlag= true;
234 + return false;
235 + }
236 + }
```



jjohnstn on Dec 1, 2022

If we have an InstanceOfExpression, we can't match the InfixExpression nor set the target variable so just return here or move line 237 up to the top and then make the InfixExpression node check an else if.





Reply...

...clipse.jdt.ui/core refactoring/org/eclipse/jdt/internal/corext/refactoring/
util/Checker.java

Outdated

```
42 46
43 47 import org.eclipse.jdt.internal.corext.dom.fragments.ASTFragmentFactory;
44 48 import org.eclipse.jdt.internal.corext.dom.fragments.IASTFragment;
45 49

46 - public class NullChecker {
50 + public class Checker {
```



jjohnstn on Dec 1, 2022

IMO, Checker is a very generic name. I might propose UnsafeCheckTester but regardless of a name change a class javadoc comment would be helpful.



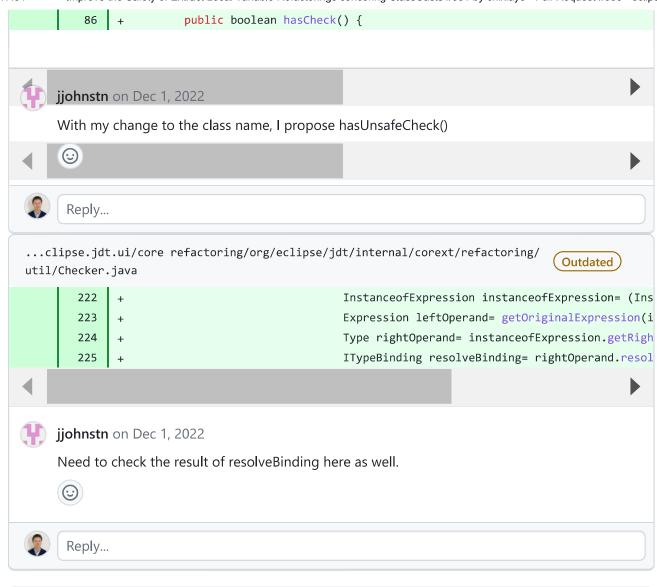


Reply...

...clipse.jdt.ui/core refactoring/org/eclipse/jdt/internal/corext/refactoring/
util/Checker.java

(Outdated)

utii/	Cnecker	ava
77		NullMiddleCodeVisitor nullMiddleCodeVisitor= new NullMiddleCode
78		this.fCommonNode.accept(nullMiddleCodeVisitor);
79		<pre>return nullMiddleCodeVisitor.hasNullCheck();</pre>



```
...clipse.jdt.ui/core refactoring/org/eclipse/jdt/internal/corext/refactoring/
                                                                                      Outdated
util/Checker.java
222
        259
                                                           this.nullFlag= true;
223
        260
                                                           return false;
224
        261
                                                   }
225
        262
226
        263
                                          return super.preVisit2(node);
227
        264
                                  }
        265
        266
                                  private boolean hasInheritanceRelationship(ITypeBinding itb1, I
        267
    jjohnstn on Dec 1, 2022
     There are also interfaces to check which getSuperClass() does not return.
     \odot
      Reply...
```



chixiaye commented on Dec 6, 2022 • edited ▼

@jjohnstn Hi, as you suggested, we improved the relevant code. I think the check of resolveBinding can be merged into the check that <code>getSuperClass()</code> does not return, so I did not add an extra resolveBinding check.



[2] Iiuhuigmail mentioned this pull request on Dec 7, 2022

Improve the Safety of "Extract Local Variable" Refactorings concering ClassCasts #331



liuhuigmail commented on Dec 10, 2022

@jjohnstn Please let us known if there is anything deserving further improvement. Thank you.



jjohnstn commented on Dec 13, 2022

**@liuhuigmail** Will look at it today or tomorrow. I was on vacation last week and will be on vacation from Dec 19th onward.



#### liuhuigmail commented on Dec 14, 2022

@jjohnstn Thank you. Wish you happy holidays!



jjohnstn commented on Dec 15, 2022 • edited ▼

Hi, everything looks good except you haven't fixed hasInheritanceRelationship() to handle interfaces. For example, you could have:

```
public interface A {
    public int getInt();
}

public class B implements A {
    public int getInt() {
        return 7;
    }
}

public class C {
    public void foo(Object obj) {
        if ((obj instanceof A) && ((A)obj).getInt() > 3) {
            ....
        }
    }
}
```

If an instance of B is passed to C.foo() and the user asks to put the 2nd part of the if statement in a local variable,

your logic for hasInheritanceRelationship() won't catch it because getSuperClass() only finds those types specified

in an "extends" clause, but not an "implements" clause so you will never find a match with interface A. There is a separate call for getting interfaces getInterfaces() that needs to be called in addition to getSuperClass() and you should make the method recursive. Essentially you can check the passed in class and all of its interfaces, then recursively call for each of the interfaces and with the super class if it exists. This will ensure you check the whole inheritance tree. Each recursive call can check if you get a match and stop the recursion.



**jjohnstn** commented on Dec 15, 2022

**@chixiaye** BTW: I added a N&N entry for your previous null check fix in 2022-12 and it made the release video:

https://youtu.be/jJau4kUoLrA



#### liuhuigmail commented on Dec 15, 2022

@jjohnstn Thank you for the comments. We will fix it quickly and resubmit the patch.

By the way, we have submitted another issue concering the Extract Local Varialbe refactoring (#348).

Would you please kindly have a look and let us known whether it is ok for us to prepare a patch (pull request) as we specified in the issue report. Thanks.



#### chixiaye commented on Dec 17, 2022

Hi, the method hasInheritanceRelationship has been improved to identify the inheritance concerning the interface. And the implementation of hasInheritanceRelationship has also been changed to a recursive form. Besides, Test-128 and Test-129 were added to verify the inheritance relationship between classes and interfaces. Test-130 and Test-131 were added to test the inheritance relationship between interfaces.





**n chixiaye** mentioned this pull request on Jan 6

Improve the Safety of "Extract Local Variable" Refactorings by Identifying the Side Effect of Selected Expression #348





jjohnstn approved these changes on Jan 7

View reviewed changes



jjohnstn approved these changes on Jan 7

View reviewed changes

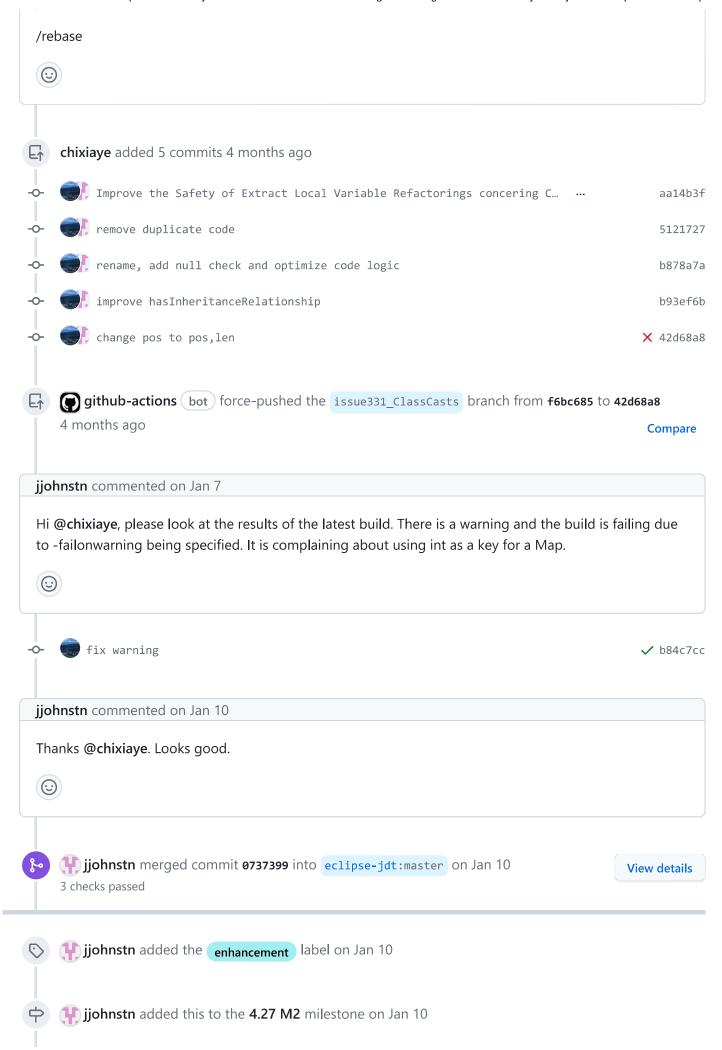


jjohnstn left a comment

Looks good. Just need to get build working.



jjohnstn commented on Jan 7



Reviewers

ijohnstn

Assignees

No one assigned

Labels
enhancement

Projects

None yet

Milestone

4.27 M2

Development

Successfully merging this pull request may close these issues.

None yet

### 3 participants

