REVIEW ON IDENTIFICATION AND REFACTORING OF BAD SMELLS USING ECLIPSE

Sandeep Kaur[1], Er Harpreet Kaur[2]
[1]Research Scholar, [2]Assi. Prof.
Department of Computer Engineering, UCOE, Punjabi University, Patiala

ABSTRACT: Bad smells indicate that there is something wrong in the code that we have to refactor. Bad smells are design flaws in the code. There are many tools that are available to identify the bad smells and remove these bad smells by using refactoring tools and by using refactoring technique. Refactoring is a technique that restruct our source code in a more readable and maintainable form by removing the bad smells from the code. Refactoring does not change the external behavior of software. In this paper we discussed about tools and techniques to refactor the source code.

KEYWORD: Refactoring, Bad smells, refactoring techniques, refactoring tools etc.

I. INTRODUCTION

A. BAD SMELLS:- Bad smells are potential problems that can be remove through refactoring. There are various kind of bad smells that make our source code difficult to understand and modify. Bad in a code is not any problem but may lead to any mistake in future. We can detect and refactor this bad smells through various tools.PMD and jdeodorant are such kind of tools that are used to detect the bad smells. Bad smells are of following types:

- LARGE CLASS means a class that is too large. Size of the large class is too much large. This type of class is difficult to understand and it is too much hard to understand that which functioning is performed by this class.
- LONG METHOD is a method that is too long. Long methods are same as large classes. Long methods also leads to confusion for the new developer and these are also very much difficult to understand.
- DUPLICATE CODE is a code that is repeated at too many places in a same source code. The problem arises when we try to update this code. We have updated the duplicate code at all the places in which this code is placed. If we forget to update the code on single place, it may create problem. Hence duplicate code is difficult to maintain.
- FEATURE ENVY is a bad smell that violates the principle of its class in a source code. From many discussion we found that this a bad smell that is not interested to use its own source class but interested to use any another source class.
- LONG PARAMETER LIST is list of parameter in which we declare so many parameters which are not actually neede. Long parameter list is hard understood because they are inconsistent and difficult to use.
- DEAD CODE is a code that is never run or does not perform any functionality in the source code.

All these bad smells can be clean up by using the refactoring.

II. REFACTORING

Refactoring has become popular technique in software engineering because in software engineering maintenance of the software take 50% development cost of creating software. One of the main reasons behind that is poorly designed code that is hard to understand for the new developer. So refactoring helps the developer to understand the code, find and fix the errors, adding new feature. Refactoring is a technique in which we apply changes to the internal source code of the software without changing its external behavior. The term refactoring was originally coined by Kent Beck. After Kent Beck a lot of work on refactoring is done by Martin Fowler. Refactoring is a way to clean up the code and minimize the chances of errors.

III. REFACTORING ACTIVITIES

There are various steps to carry out the refactoring in a successful way. These are as follow:

- Upload project.
- Identify where the software have to refactor.
- Identifying the bad smells by using the specified tool.
- Determine which refactoring technique we should apply to the code to refactor it or to remove the bad smells.
- Be assured that after applying the refactoring technique there should be no any change occur in the external behavior of the software.
- Apply refactoring.
- Run test to ensure things still working correctly.
- Repeat the cycle until the bad smells are not gone.

IV. REFACTORING TOOL

A. ECLIPSE:-

Eclipse is an IDE(Integrated Development Environment).In it there is a workspace and an extensible plug-in for customizing the environment. Written in java language it is used to develop application. Eclispe may also helpful to create the application in C,C++,PHP and COBOL and many more languages. Eclipse also provide an option to refactor the source code. There are many plug-in for eclipse that can used to detect the bad smells in the code. To refactor a the bad smells it provides various refactoring techniques.

Some these techniques are as follow:-

B. REFACTORING TECHNIQUES

- EXTRACT METHOD:- Extract method means extract a method that appear at many places in the source code and place it in a different method.
- INLINE METHOD:-Its working is totally opposite to the extract method. A method of body is as clear as its name. Put the body of the method into the body of its caller. Then remove the method.
- MOVE METHOD:-Move method means moving the method from one class to another when a class has too many functions to do.
- INLINE CLASS:-This method is applicable to those classes which are not doing too much work. With the help of inline class we can move the functionality of this class to another class and remove the class.
- RENAME METHOD:-Rename method simply means renames any method. We can rename the method according to the functionality of the method. It makes our code easier to understand.
- REPLACE ARRAY WITH OBJECT:-If we have an that have many or different elements. We can replace this array with an object, in this object each element have specific field.

C. LITERATURE REVIEW:- It present about the previous of studies detection of evaluating Refactoring, Bad Smells, Tools and what the researcher have done regarding to detect and refactor the bad smells.

Martin Fowler discusses in his paper that how refactoring improve the existing design. He introduce deeply about refactoring in his paper. This paper also discuss more about the bad smells. How to detect these smells and tells that which refactoring technique is applicable to remove this bad smell. Martin also introduce that a code that have bad smells are hard to maintain and hard to modify.[1]

Almas Hamid,Muhammad Ilyas, Muhammad Hummayun and Asad Nawaz discuss that Refactoring is a technique to make a program's source code more readable, efficient and maintainable. A bad smell is an indication of some problem in the code, which requires refactoring to deal with. Many tools are available here for detection and refactoring of these code smells. These tools vary greatly in detection methodologies and acquire different competencies. In this work, we studied different code smell detection tools minutely and try to comprehend our analysis by forming a comparative of their features and working scenario. We also extracted some suggestions on the bases of variations found in the results of both detection tools.This helps us to select the tool to rafactor the code.

Francesca Arcelli Fontana discuss in his paper that recent research is active in automatic detection tools to help the developer to detect the bad smells when the code becomes difficult to manage and understand. This paper present a comparison of 4 types of bad smells detection tools.

David Jönsson dt05dj3 discuss the tools about jdeodorant and PMD. He discuss about the benefits of this two tools and make a comparison between them. He also introduce that what type of bad smells can be detected through this tools.w hich tools is more beneficial among the both.

D. SELECTION OF BAD SMELL DETECTION TOOL:-

Jdeodorant is an Eclipse Plug-in that identifies the problems in the software, known as bad smells and helps to resolve them by applying appropriate refactoring. Jdeodorant provides a variety of methods and techniques in order to identify the bad smells and suggest the appropriate refactoring techniques that resolve them. Currently, j deodorant tool is used to detect four types of bad smells, namely "Feature Envy", "State Checking", "Long Method" and "God Class".

PMD is an eclipse plug-in that provides quick fix for bad smells. It can detect 5 types of bad smells. The bad smells that are detected by PMD are "large class"," long method"," large parameter list"," duplicate code" and "dead code". PMD tool takes following steps to detect the bad smells.

Step 1: Analyze

Every time we save our work, eclipse-pmd scans our source code and looks for problems like possible bugs, duplicate and dead code.

Step 2: Highlight the problem

When eclipse-pmd finds problems then it highlights those in our editor so we can fix them in a right way. Discovered problems are also viewed in the Eclipse Problems View.

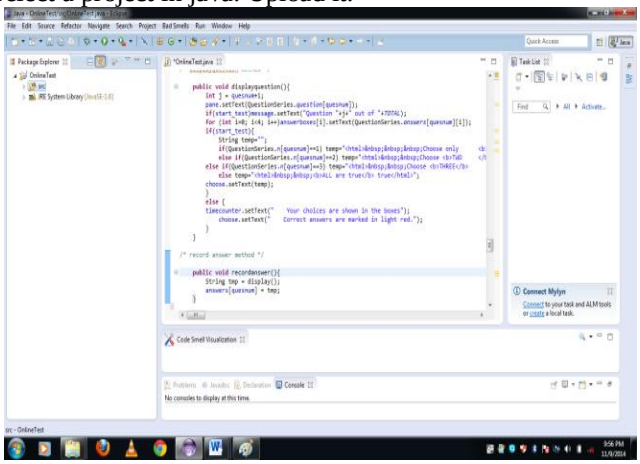
Step 3:Fix

When possible, eclipse-pmd offers quick fixes that automatically fix the problems. These quick fixes can be used to fix a single problem or all occurrences in our entire code.

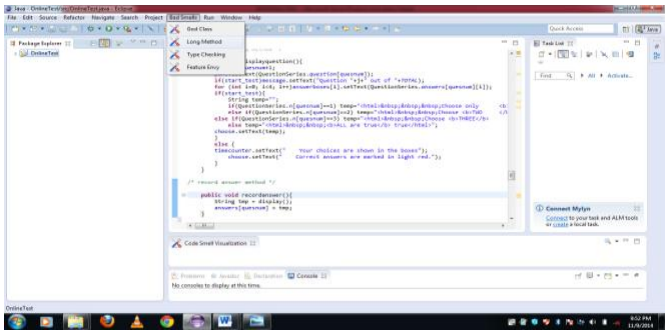
V. EXPERIMENT

The experiment is done by using the eclipse tool and jdeodrant as bad smell detection tool. We take an source code of online exam software which is written in Java. Steps to identify the bad smells and refactor them are as follow:

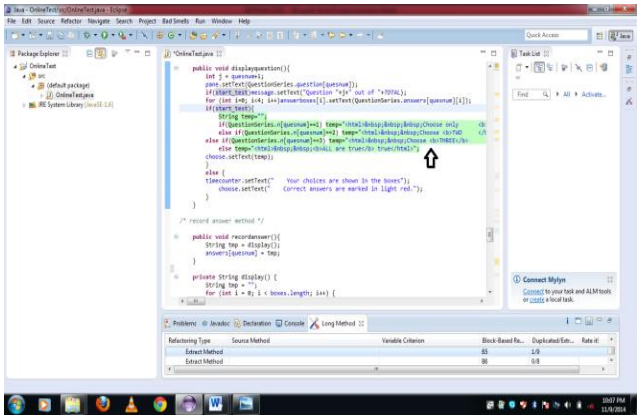
Select a project in java. Upload it.



Select a smell that we want to identify.

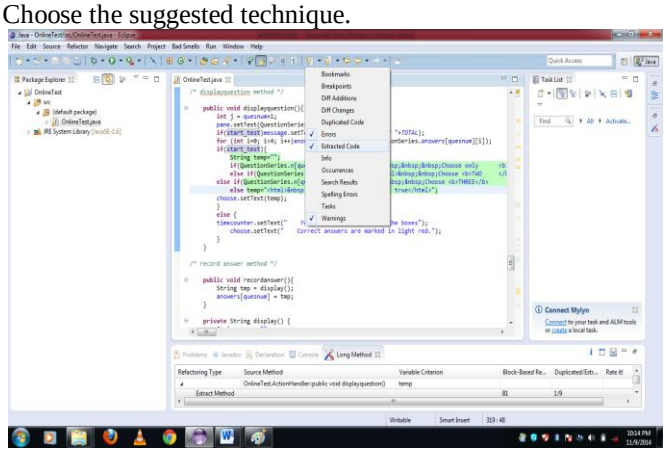


Suppose we choose long method bad smell. The identification result is shown below:

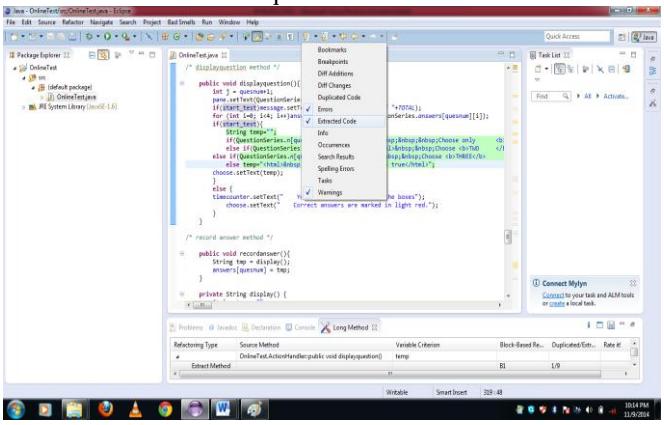


The result of the long method bad smell is highlighted by green color and also a suitable refactoring technique is shown below.

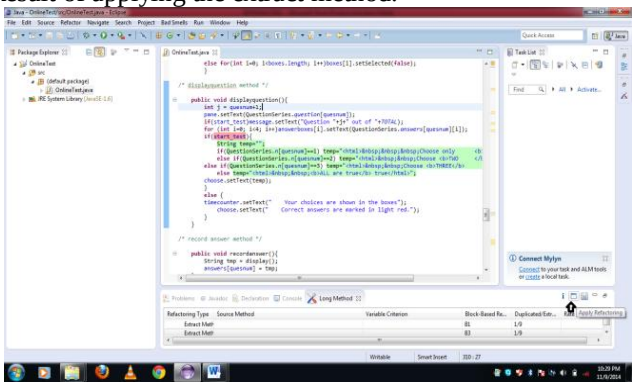
Choose the suggested technique.



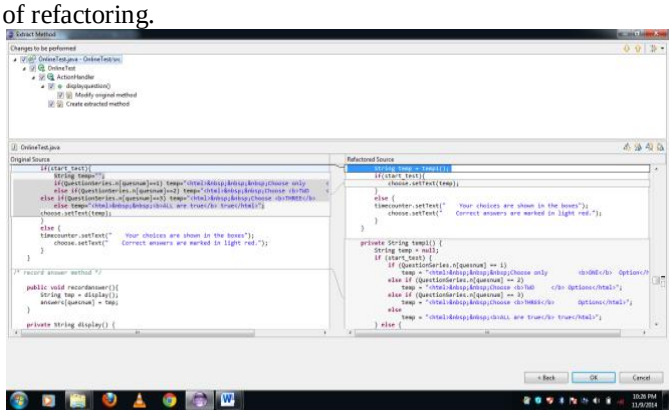
Select the refactor technique to refactor the code.



Result of applying the extract method.



We can also view a preview so that we can analyze the effect of refactoring.



COMPARISON BETWEEN PMD AND JDEODORANT

Jdeodrant is easy to use as comparison to the jdeodorant.PMD is much more complex tool even though it supports 5 types of bad smells. The complexity of using PMD stems from the high number of configuration choices available. Jdeodorant not only detecting bad smells it is one of the few tools that also gives the user an advice on refactoring. JDeodorant was simplistic in both design and usage while PMD was a bit more complex. Both tools can be use as plug-in in the eclipse.

Plugin/bad smells	Jdeodorant	PMD	Refactoring technique that can be applied
Feature envy	Yes	No	Move method
Type checking	Yes	No	Replace Conditional with PolymorphismOr replace type code with state
Large class	Yes	Yes	Move Method
Long Method	Yes	Yes	Extract Method
Dead code	No	Yes	Remove Method
Large parameter list	No	Yes	Extract class
Dulpicate code	No	Yes	Extract method

VI. CONCLUSION

Refactoring is an important and easy activity to refactor the source code in a well manner. Refactoring makes a code easier to understand. Bad smells make our source code more difficult to manage. We can detect these bad smells by jdeodorant. Jdeodorant also give solution that which refactoring technique can be applied to refactor this bad smells. By applying refactoring using Eclipse we can refactor theses bad smells and make our source essay to understand and modify.

VII. FUTURE WORK

In future we will study about more types of bad smells and develop an GUI that can automatically refactor these bad smells by applying various refactoring techniques.

REFRENCES

[1] M. Fowler, K. Beck, J. Brant, W. Opdyke and D. Roberts, “Refactoring: Improving the Design of Existing Code”, Addison-Wesley, (1999).
[2] Simon, F., Steinbr, F., Lewerentz, C., 2001. Metrics based refactoring. In: Proceedings of the 5th European Conference on Software Maintenance and Reengineering. IEEE CS Press,Lisbon,Portugal,pp:30-38.International Journal of Software Engineering & Applications (IJSEA), Vol.5, No.2, March 2014
[3] William C. Wake. Refactoring Work book. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003 .
[4] "Automatic detection of bad smells in code: An experimental assessment" Francesca Fontana, Pietro Braione, Marco Zanoni, University of Milano-Bicocca 2011
[5] A Survey of Software Refactoring, Tom Mens, Tom Tourw´
[6] IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. XX, NO. Y, MONTH 2004
[7] A. Abadi, R. Ettinger, and Y. A. Feldman, "Re-Approaching the Refactoring Rubicon," 2nd ACM Workshop on Refactoring Tools (WRT'08), 2008
[8] Fowler, M. Crossing Refactoring's Rubicon. http://www.martinfowler.com/articles/refactoringRubicon.html, 2001.

