Advances in Science, Technology and Engineering Systems Journal Vol. 3, No. 6, 140-143 (2018)

www.astesj.com

First International Virtual Conference on Multidisciplinary Research 2018

ASTESJ

ISSN: 2415-6698

Analysis Refactoring with Tools

Zhala Sarkawt Othman[*]

Department of Software Engineering, Firat University, 23000, Elazig, Turkey

A R T I C L E I N F O

Article history:
Received: 30 September, 2018
Accepted: 14 October, 2018
Online: 14 November, 2018

Keywords:
Refactoring tools
Reverse engineering
Refactoring in VB
Analysis

A B S T R A C T

The drive for this report is to inaugurate the innumerable techniques espoused by the refactoring tools in coding development. The software product is a very complex and time-consuming process of development. Difficulty understanding and maintaining poorly designed software systems Software maintenance can take up to 50% of total development costs for software production. As a modus operando, the refactoring tools purpose ultimately to amend the basis codes into an easier and more comprehensible way.
Moreover, refactoring succors to check the trifle of the coding procedure. This is apparent through having deliberation on the program catalog, precision and the use of the deconstruct trees. Refactoring tools are convenient for innumerable observes done by the human beings. Software refactoring has a direct impact on reducing the cost of software maintenance by changing the internal structure of the code without changing its external behavior. So the time taken to process as well as doing a critical analysis of complex codes is reduced.
This report proposes to have a precarious scrutiny on the various use including the pluses of using refactoring tools.

1. Introduction

Refactoring is a technique of fluctuating the now prevailing basis codes using an unconventional as well as the incremental system. Moreover, the universal comportment of the code is not reformed through refactoring. Refactoring is advantageous as its reprocess results in either the refactored code being definitely employed in a more advanced way or for an additional tenacity.

Principally, refactoring is deliberated to be a nonspecific software design system which is neither tied to any defined solicitation language. For more accomplishment, the refactoring web site by Martin Fowler makes available a podium for use of a number of refactoring practices. Having a refactoring tool arranges for a much indispensable sustenance in scrutinizing a code rather than doing it manually. This is because manual refactoring consumes a lot of time.

Sanctioning the program writer to refactor his or her code without having them too personally retest the database is the significant driver for a refactoring tool [1]. This eradicates period lost when the process is either done physically or computerized. Moreover, the program catalog is critical and must be conserved repetitively in a securely cohesive atmosphere. This consents the computer operator to scan and detect cross-references. Furthermore, it disregards the actuality of vibrant collation arising from the codes.

Moreover, refactoring encompasses guidance of the portion of the arrangement that is present below the average level. This comprises references to database elements which are being altered. This leads to update prepared on the references, therefore, management of the structure of the method.

The origination of a construe tree will portray the existing interior structure for the method itself. This includes:

```
Void hello ()
 {
        System.out. printIn (“Hello World/n”);
}
```

Nonetheless, the refactoring process must be in route with the common comportment of the programs set in place. This is because the whole activities preservation for a program is unbearable to accomplish. This makes refactoring tools to be implemented to support improve the executions by most of the programs. This brands the programmers to either commend the usage of the refactoring techniques or fix glitches arising in their

[*]Zhala Sarkawt OTHMAN, Elazig 23119 Turkey, Phone: 05366836869;
E-mail: zhala.sarkawt@gmail.com

databases which cannot be done by the refactoring tools physically.

Moreover, the refactoring tool is eligible for backing the conduct of human. This involves, promptness: the breakdown, as well as the renovation necessary to complete refactoring, is time-consuming in cases where the manner is depicted to be erudite [2]. Several considerations are taken. This includes the cost of time plus the level of accuracy. In the case where refactoring consumes a lot of time, a programmer is restricted to the convention of the computerized refactoring but is accountable for manual refactoring as well as bear the consequences. This makes the speed of refactoring to be an important aspect.

The program writer is tasked with the choice of providing an eminence work at a cognizant time interval. Much of the statistics is already known to the programmer. However, this scheme can be perilously leading to the programmer making a lot of blunders while producing the prerequisite information. This fact countenances for most people to use refactoring tools as a foundation to avoid making errors plus a search platform for finding important information.

2. Method and Materials

A project analyzer is a refactoring tool that is essential to refactoring already in place Visual basic code. This tool has substantiated to be useful in various ways. The project analyzer will succor in the documentation of codes that which are openly connected resulting in its aiding from the refactoring techniques used.

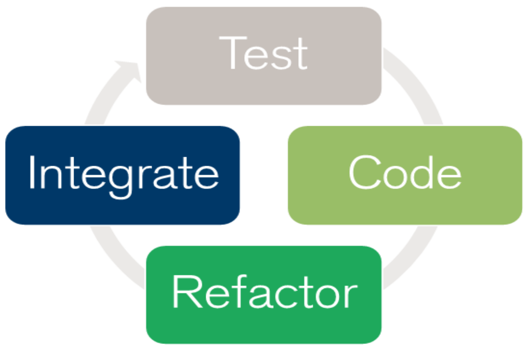
As a refactoring tool, the scheme analyzer also completes robotically a splinter of the refactoring. Structures on the project analyzer which take account of the auto-fix assists in the computerized amendment of the basic code to be in line with the now predefined guidelines. This is also trailed by the programmed encapsulation of session variables as property Get/Set [3].

A number of vagaries in the codes entail mortal consideration, therefore, making a refactoring to be completed by hand. However, on monotonous tasks, mechanical changes are required. Visiting is mostly used in the refactoring process. This is done by the displaying a flow chart for a given piece of executable code. The codes can be in various forms including Visual Basic, Java, Pascal or Ruby. However, codes containing nested conditions, loops as well as jumps have logic errors [4].

There are some existing refactoring tools for the most widely used modern languages, such as Java, C# and C++, mostly as refactoring browser plug-ins to the most mainstream IDEs (e.g., Microsoft Visual Studio or Eclipse) [5].

The construction of flowchart assists to extant program logic which is easy to apprehend. In addition, the database logic is not related to the fundamental code that it is initially written [6]. It is easy to read the source code with the aid of the flowchart. This sorts the user categorizes logic defects which may be solid to ascertain by a graphical apparatus on the code.

Moreover, the flowchart assists to refactor the codes to systems that are easily decipherable plus increased. The (flowchart 1) about Refactoring Process. This makes a graphic construal of the code to be more favored since it divulges designs which can be distinguished in the source code.



Flowchart 1, Refactoring Process

The refactoring techniques streamline methods, remove code duplication, and pave the way for future improvements.

In (Table 1) the following are examples of simple refactoring in Visual Studio.

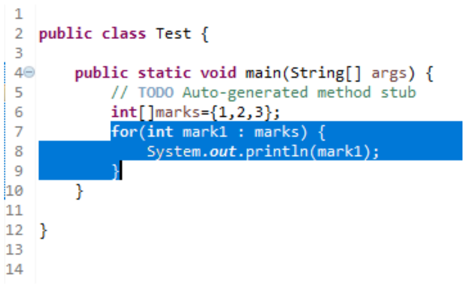
Table 1, Visual Studio Refactoring

Refactoring Technique	**Meaning in Life**
Extract Method	This allows you to define a new method based on a selection of code statements.
Encapsulate Field	Turns a public field into a private field encapsulated by a .NET property.
Extract Interface	Defines a new interface type based on a set of existing type members.
Reorder Parameters	Provides a way to reorder member arguments.
Remove Parameters	As you would expect, this refactoring removes a given argument from the current list of parameters.
Rename	This allows you to rename a code token (method name, field, local variable, and so on) throughout a project.
Promote Local Variable to Parameter	Moves a local variable to the parameter set of the defining method.

This example of the "Extract Method" Refactoring. We use one of the tools in programming to improve the code in order to make the intent of the code clearer; you will extract the code that collects test cases from base classes into a new method called “printmarks”.

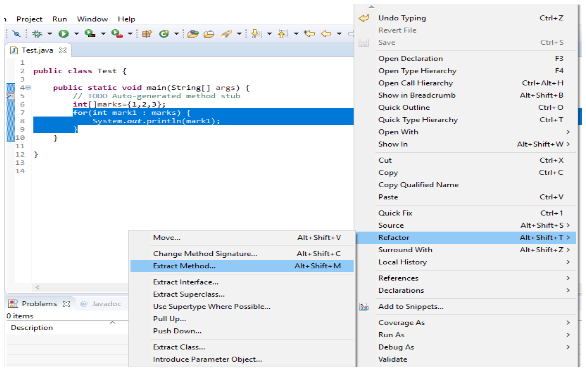
1. We will select the following range of code inside the Test (Class):

2. After we select three lines of the code from the selection's context menu in the editor; select Refactor > Extract Method or using (Alt+Shift+M) for the refactoring Extract Method:



```java

public class Test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int[]marks={1,2,3};
        for(int mark1 : marks) {
            System.out.println(mark1);
        }
    }

}


```

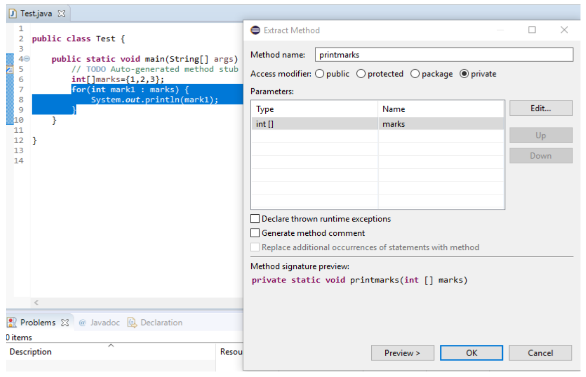
Figure 1, Select the following



```java

public class Test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int[]marks={1,2,3};
        for(int mark1 : marks) {
            System.out.println(mark1);
        }
    }

}
```

Figure 2, Refactoring Extract Method

3. In the **Method Name** field, type *printmarks*:



```java

public class Test {

    public static void main(String[] args)
        // TODO Auto-generated method stub
        int[]marks={1,2,3};
        for(int mark1 : marks) {
            System.out.println(mark1);
        }
    }

}
```

Extract Method

Method name: printmarks

Access modifier: ( ) public ( ) protected ( ) package [✓ private]

Parameters:

Type	Name
int []	marks

[ ] Declare thrown runtime exceptions

[ ] Generate method comment

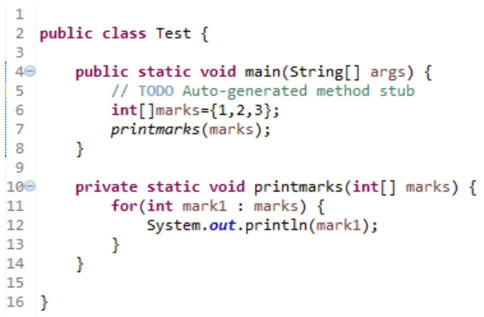
[ ] Replace additional occurrences of statements with method

Method signature preview:

```java
private static void printmarks(int [] marks)
```

Figure 3, Method Name

4. After refactoring, we get a cleaner and more readable code:



```java

public class Test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int[]marks={1,2,3};
        printmarks(marks);
    }

    private static void printmarks(int[] marks) {
        for(int mark1 : marks) {
            System.out.println(mark1);
        }
    }

}
```

Figure 4, After Refactoring

3. Results

Various changes in the codes require human consideration, therefore, making a refactoring to be done manually. However, on routine tasks, automated changes are required.

The important part of the test is to make sure that your application performs efficiently and responsibly. This is where code analysis and profiling tools and techniques are evaluated: allows you to evaluate your code for errors, bottlenecks, and efficient use of processing and memory resources. Modern code recognizers can direct you to the exact lines of code that need to be resold.

The re-export tools change between IDEs and software programs. Visual Studio contains embedded built-in analysis tools. In addition, there are excellent tools to help you get deeper into your application for performance and optimization testing, project templates that rely on effective accreditation and embedded testing frameworks, and solid tools to integrate automated system analysis and testing into your business structure and workflow [7].

4. Discussion and Conclusion

The use of the hide method by the project analyzer allows the detection of excess method scope. It also suggests how to make the procedure private where it is convenient. Variables which exist outside of their respective classes or module are hidden. This procedure can also be done by making the variable to be local. Moreover, the cases for substitute nested conditional with guard clauses, the project analyzer is able to identify unwarranted conditional nesting. This is essential to understanding the steps taken in its execution. Furthermore, there is a high probability to replace the nesting with guard clauses. This procedure includes a sequential on the non-nested if statement, making it easier to read [8].

The procedure of using the manual as a refactoring tool is long. This is with the inclusion of a similar line of code in various locations. However, with the use of the visiting, there is a high probability for the user to alter the logic available. This enables elimination of the any existing copied lines. In the case where a logical structure exists in more than two procedures which may be as a result of duplication and pasting of the codes. Nonetheless, detections allow for the logic to be restructured into a new function and its access is from other functions [9].

Multifaceted algorithms are divided into various functions. The formations of the flowchart allow easy identification of every block from which the new functions were formed from. In cases where intricate conditional expression exists, it is removed using decompose conditional which also rewrites it into a new function. The multifaceted expressions formed are easily detected in a flow chart. Moreover, it provides a suitable platform for the logic to be rewritten in a simpler manner.

The use of the converse conditional reverses the logic of a conditional statement through the use of the not operator. However, it is not convenient to remove the Not from the code. The flow charts assist to display the usage of the reverse logic which is convenient for refactoring [10].

References

[1] Murphy-Hill, E. and Black, A.P., 2008. Refactoring tools: Fitness for purpose. *IEEE software*, *25*(5).

[2] Campbell, D. and Miller, M., 2008, October. Designing refactoring tools for developers. In *Proceedings of the 2nd Workshop on Refactoring Tools* (p. 9). ACM.

[3] Garrido, A. and Johnson, R., 2003, October. Refactoring C with conditional compilation. In *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on* (pp. 323-326). IEEE.

[4] Murphy-Hill, E., 2006, October. Improving usability of refactoring tools. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications* (pp. 746-747). ACM.

[5] designs, S., 2016. *Semantic designs.* [Online] Available at: https://www.semanticdesigns.com

[6] Fleming, Scott D., et al. "An information foraging theory perspective on tools for debugging, refactoring, and reuse tasks." ACM. Transactions on Software Engineering and Methodology (TOSEM) 22.2 (2013): 14.

[7] Dorsey, T., 2017. *visual studio magazine.* [Online] Available at: https://visualstudiomagazine.com[Accessed 26 10 2017].

[8] Liebig, Jörg, et al. "Morpheus: Variability-aware refactoring in the wild." Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on. Vol. 1. IEEE, 2015.

[9] Murphy-Hill, E. and Black, A.P., "Breaking the barriers to successful refactoring: observations and tools for extract method" In Proceedings of the 30th international conference on Software engineering (pp. 421-430). ACM, 2008, May.

[10] Fontana, F.A., Mangiacavalli, M., Pochiero, D. and Zanoni, M. "On experimenting refactoring tools to remove code smells". In Scientific Workshop Proceedings of the XP2015 (p. 7). ACM. 201