2.3 Refactoring techniques

Move Method	Move method means moving a method to another class when classes have too many functionalities to do.
Extract Method	Extract method means extract a piece of code that is appear at many places in the source code and make a new method or class.
Replace Temp with query	Replace temporary variables with the method calls.It is same as extract method.
Rename method	Rename the method according to the functionality of the method.
Replace array with object	An array has certain elements with different things.Replace that array with an object in which there is a field for each element.
Inline Class	When a class does have much work to do then it is better to move its entire feature to another class and remove it.
Push Down	Some functionalities of the super class are valid for some subclasses. Move this functionality to those subclasses.

3. Literature Survey

It presents about the previous studies of evaluating what the other researchers have done regarding code smells detection.

Author	Description
Karnam Sreenu	Software refactoring is a technique that transforms the various types of software artifacts to improve the software internal structure without affecting the external behavior. Various types of object oriented metrics can be calculated to detect the bad smells.
Almas Hamid, Muhammad Ilyas, Muhammad Hummayun and Asad Nawaz	Refactoring is a technique to make a computer program more readable and maintainable. A bad smell is an indication of some setback in the code, which requires refactoring to deal with. Many tools are available for detection and removal of these code smells.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.
Francesca Arcelli,Fontana Pietro Braione,Marco Zanonia	Code smells are structural characteristics of software that may indicate a code or design problem that makes software hard to evolve and maintain, and may trigger refactoring of code. Recent research is active in defining automatic detection tools to help humans in finding smells when code size becomes unmanageable for manual review
Martin Fowler	Bad smells are signs of potential problems in code. Detecting bad smells, however, remains time consuming for software engineers. Large Class is a kind of bad smells caused by large scale, and the detection is hard to achieve automatically.
Marija Katic	The main definitions and terms concerning software redesign is closely connected with the testing. This paper briefly presents the software redesign process and methods that are used in that process. Although one can say that for example a source code redesign belongs to the implementation phase, tests are needed to ensure that the behavior is not changed.
Wei Liu	This paper discusses how to detect and eliminate the lazy class. An automatic syntax tree (AST) is proposed in this work. Firstly source code file is converted into ASTs and then three types of relationship are considered between the classes and extracted syntax tree. After carrying out several operations on these ASTs lazy class is obtained and removes it automatically. This approach has good efficiency, and its execution time has a linear relationship to the size of a system.
Raju M. Tugnayat	This paper discusses refactoring which is one of the techniques to keep software maintainable. However, refactoring itself will not bring the full benefits, if we do not understand when refactoring needs to be applied. To make it easier for a software developer to decide whether certain software needs refactoring or not, Fowler & Beck's idea was that bad code smells are a more concrete indication for the refactoring need than some vague idea of programming aesthetics.
Mohamed Eladawyl	In this paper a novel assessment criterion based on including the inherited attributes and methods has been proposed. Additionally, the effect of including the inherited attributes and methods in measuring class cohesion has been extensively discussed.

4. Proposed Work

A Window based GUI application has been developed to detect bad smells. It detects the bad smells according to the Object Oriented Metrics. Large Class, Switch Statements, Long Parameter List, Dead code, Conditional Statement, Duplicate Code, Comments are the bad smells that are detected by the GUI. This application detects the bad smells from the source code of java. Also refactor the detected bad smells by using appropriate refactoring techniques. It focuses on improving the quality or performance by decreasing the complexity of the source code.

4.1 Experimentation

The experimentation is done as follow:

The GUI interface is creates in VB.Net language and support detection and refactoring of bad smells. This application is created in Visual Studio 2010. It detects the bad smells according to the object oriented metrics and refactors them from the source code. To detect these bad smells different types of object oriented metrics are measured. For different kind of bad smell the object oriented metrics is also different.

4.2 Metrics tocalculate the bad smells:

1) **Lines of source code** Lines of code usually refer to non-commentary lines meaning pure white spaces and lines containing only comments are not included in the metric.
2) **Lines of comment** Lines of comment are used to describe the meaning of the statement in the code.

4.3 Detection of bad smells by using Window Based GUI Application

1. Conditional statements
Firstly we calculate the metrics to detect the conditional statements.