**A Survey : Code Optimization using Refactoring**

Refactoring in common language could be define as modifying the code without changing its external behaviour. The changed code is optimized code in terms of oriented feature such as Encapsulation, Polymorphism, Inheritance or in terms of performance system such as response time, and execution time. In terms of implementation, refactoring is divided into six main groups depending upon its functionality:

1. Composing method : it deals with problems that related to methods and parameters. It included methods such as extract method, inline method, and replace parameter with method. These type of refactoring solved problems such as duplicate code in a single class, lengthy methods and long parameter lists.

2. Moving features between objects : used for designing the software design. It included methods such as move method and extract class. It used to solve related to large classes and multi-level classes

3. Organizing data : it included methods such as replace data value with object

4. Simplifying conditional expression : it deals with optimization and simplification of conditional expressions.

5. Making method calls simpler : included methods such as preserve whole object and introduce parameter object

6. Dealing with generalization : It included methods such as pulled up field, pulled up method and push down field and method

**Extract Method**

This method turns a group of lines into function or method. It helps in modularizing the code. It helps in reusability of code and method overriding.

**Original Code**

void printOwing(double amount) {

printBanner();  
//print details  
System.out.println ("name:" + \_name);

System.out.println ("amount" + amount); }

**Refactored code after applying Extract Method**

void printOwing(double amount) {  
printBanner();

printDetails(amount);---------------------🡪**Refactored line** }

void printDetails (double amount) {

System.out.println ("name:" + \_name);

System.out.println ("amount" + amount);

}