SOFTWARE QUALITY METRICS

1. *Variables*: The variables that are declared in the function.
2. *Closure*: The variables and parameters that are declared in the function that are used by its inner functions.
3. *Exceptions*: The variables that are declared by catch clauses of try statements.
4. *Outer*: Variables used by this function that are declared in other functions.
5. *Global*: Global variables that are used by this function. Keep these to a minimum.
6. *Label*: Statement labels that are used by this function.
7. *Number of lines of code*: The number of lines in the text of the program's source code.

Objective : Minimize the above scores for each function

Evaluation Method : Using static analysis tools such as JLint and JArchitect to measure the above metrics.

Other Metrics

1. *Lack of Cohesion of Methods* : Cohesion is an important concept in OO programming. It indicates whether a class represents a single abstraction or multiple abstractions. A connected component is a set of related methods (and class-level variables). There should be only one such a component in each class. If there are 2 or more components, the class should be split into so many smaller classes.
2. *Association Between Class (ABC)* : The Association Between Classes metric for a particular class or structure is the number of members of others types it directly uses in the body of its methods.
3. *Depth of Inheritance Tree (DIT)* : The Depth of Inheritance Tree for a class or a structure is its number of base classes. Types where Depth of Inheritance is higher or equal than 6 might be hard to maintain. However it is not a rule.
4. *Cyclomatic Complexity (CC)* : Defined for types and methods. Cyclomatic complexity is a popular procedural software metric equal to the number of decisions that can be taken in a procedure. Methods where CC is higher than 15 are hard to understand and maintain. Methods where CC is higher than 30 are extremely complex and should be split in smaller methods