Guide with the Design Problems and Code Smells

Design problems

A design problem is a part of the software design (a design fragment) that negatively impacts one or more quality attributes, such as comprehensibility, maintainability and performance. Following we have a brief description of some examples of design problems. We highlight that the types of design problems are not limited to the following ones.

- **Ambiguous Interface**: Interface of a design component that offers only a general, ambiguous entry-point that provides non-cohesive services, thereby complicating the clients' logic.

- **Component Overload:** Design components that fulfill too many responsibilities.

- **Cyclic Dependency**: Two or more design components that directly or indirectly depend on each other.

- **Delegating Abstraction:** An abstraction that exists only for passing messages from one abstraction to another.

- **Fat Interface:** Interface that is overloaded with many clients accessing it. That is, an interface with "too many clients".

- **Scattered Concern:** Multiple components that are responsible for realizing a crosscutting concern.

- **Unused Abstraction:** Design abstraction that is either unreachable or never used in the system.

- **Unwanted Dependency:** Dependency that violates an intended design rule.

Code Smells

A code smell is a structure in the implementation of the program that possibly indicates a deep problem. Following we present a brief description of 15 types of code smells. We highlight that the types of code smells are not limited to the following ones.

- **Blob Class:** Long and complex class that centralizes the intelligence of the system.

- **Class Data Should Be Private:** A class exposing its fields, violating the principle of data hiding.

- **Complex Class:** A class having at least one method having a high cyclomatic complexity.

- **Data Class:** A class has this code smell when it has only fields, getting and setting methods for the fields, and nothing else. The aim is to hold data and, consequently, be manipulated by other classes.

- **Divergent Change:** This code smell occurs when one class is commonly changed in different ways for different reasons.

- **Feature Envy:** Feature envy occurs in a class when the method of that class seems more interested in a class other than the one it actually belongs.

- **God Class:** This code smell occurs when the class centralizes more the one functionality, realizing more than one purpose.

- **Lazy Class:** A class having very small dimension, few methods and with low complexity.

- **Long Method:** This code smell occurs when a method has grown too large, containing too many lines of code.

- **Long Parameter List:** This code smell occurs in a method that has a long list of parameters, in which makes readability and code quality worse.

- **Message Chain:** A long chain of method invocations is performed to implement a class functionality.

- **Refused Bequest:** A subclass that uses only some of the methods and properties inherited from its parents.

- **Shotgun Surgery:** This code smell is the opposite of Divergent Change. It happens when every time a kind of change is made in the code, that change triggers many little changes in many different classes.

- **Spaghetti Code:** A class implementing complex methods interacting with them, with no parameters, using global variables.

- **Speculative Generality:** A class declared as abstract having very few children classes that use its methods.