10-Singleton Pattern

segunda-feira, 14 de abril de 2025 18:22

- Bad reputation because of its the global nature
  - Solution to represent the few global aspects in our code
- Use as implementation pattern rather than design pattern
  - Will be discussed
  - It's not a design patter because it doesn't have the propertied of a design pattern
- Usual problems,
  - global state
  - many, strong and artificial dependencies
  - impeded changeablity and testbility

This intent is visualized by the Gang of Four with the UML diagram in Figure 10-1, which introduces the `instance()` function as the global point of access to the unique instance.

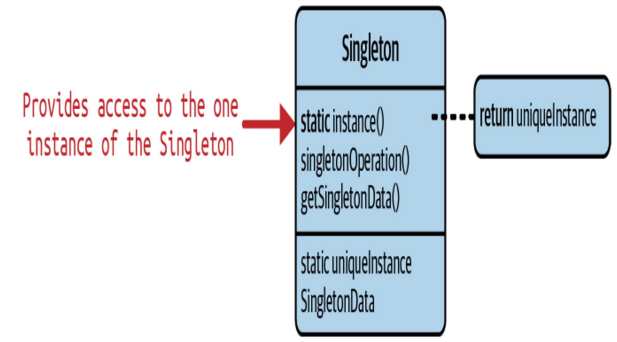

Figure 10-1. The UML representation of the Singleton pattern

The Singleton Pattern Explained

- garantee that only existiss one instance of a particular class
- might make sense for database access, logger clocks, configurations or any class that should not be instantiated multiple times because it represents something that exists only once
- Ensure a class has only one instance, and provide a global point of access to it

Meyers' Singleton

- a way to reresent and limit the number of objects of a class
- Defin the constructor and all assigned operations in as private methods and delete the operators. Not allowing overloads
- The only way to access the single instance is throug the public static instance() function
  The instance() function is implemented in terms of a static local variable. This means that the first time control passes through the declaration, the variable is initialized in a thread-safe way, and on all further calls the initialization is skipped. the function returns a reference to the static local variable

Singleton is not a design pattern.

- Design pattern has name; carries na intent; Introduces na abstraction; has been proven
  - Most of the things are true, but there is no kind of abstraction: no base class, no template parameters, nothing!!!
  - Singleton is focused on restricting the number of instantiations to exactly one. Thus, Singleton is not a design pattern but merely na implementation pattern.
- Reasons it's listed as design patter
  - There are languages that every class can be considered na abstracton, it's very commonly used, and they are in the process of understanding software design and desgn patterns
- So... we don't use singletons to decouple software entities, it's inly dealing with implementaion detals. Aas such, we should treat it as impementaton pattern

GUIDELINE 37: TREAT SINGLETON AN IMPLEMENTATION PATTERN, NOT A DESIGN PATTERN

- The goal of Singleton is not to decouple or manage dependencies, and thus it does not fulfill the expectations of a design pattern.
- Apply the Singleton pattern with the intent to restrict the number of instances of a particular class to exactly one.

Guideline 38: Design Singletons for Change and Testability