SOLID & Clean Code принципи

Обектно-ориентирано програмиране 2020/2021
27.05.2021г.
Любослав Карев

Съдържание

- Single-responsibility principle
- Open-closed principle
- Liskov substitution principle
- Interface segregation principle
- Dependency inversion principle
- Naming variables/functions
- Functions
- Formating
- C++ style guide

Single-responsibility principle

"There should never be more than one reason for a class to change."

Един клас, трябва да има една причина, поради която да бъде променян. На по-прост език, това означава, че един клас, трябва да прави точно едно нещо.

Open-closed principle

"Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification"

Вместо да променяме съществуващ код, трябва да разширяваме съществуващия такъв - това може да се реализира чрез използването на интерфейси или абстрактни класове

Liskov substitution principle

“If S is a subtype of T, then objects of type T may be replaced with objects of type S (i.e., an object of type T may be substituted with any object of a subtype S) without altering any of the desirable properties of the program (correctness, task performed, etc.).”

Съществува и “по-математическа” дефиниция...

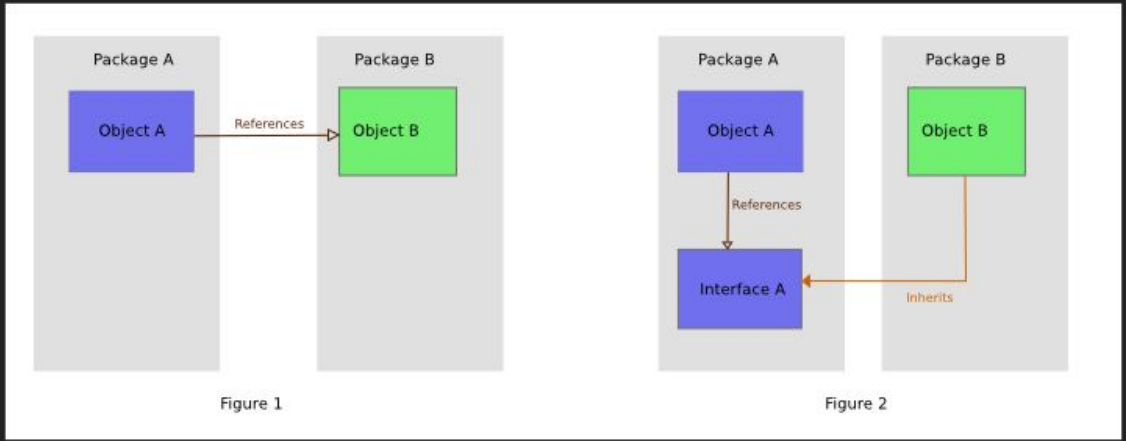
Interface segregation principle

“No client should be forced to depend on methods it does not use”

“По-добре няколко специфични класа/метода, вместо един, който да прави всичко”

Dependency inversion principle

- High-level modules should not depend on low-level modules. Both should depend on abstractions (e.g., interfaces).
- Abstractions should not depend on details. Details (concrete implementations) should depend on abstractions.



Naming

- Имената трябва да дават цялата необходима информация за една променлива - защо съществува, какво прави и как се използва
- Имената на променливи трябва да са съществителни имена
- Ако променливата е от тип bool - името и трябва да е под формата на въпрос, на който може да се отговори с true/false
- Имената трябва да са лесни за произнасяне/търсене
- Използвайте имена, от сферата за която е софтуера
- Не бъдете “хитри” и не прекалявайте

Functions

- Функциите трябва да извършват само едно нещо, на едно ниво на абстракция
- Имената на функциите трябва да са глаголи
- Функцията не трябва да е по-дълга от 20-30 реда
- Стремете се функцията да приема минимален брой аргументи*
- Избягвайте странични ефекти*

Formating

- “Code must read like a newspaper”
- “Headlines first, details later”
- Коментари ?

C++ style guide

- https://isocpp.org/wiki/faq/coding-standards
- https://google.github.io/styleguide/cppguide.html

Повече инфо

- Clean Code: A Handbook of Agile Software Craftsmanship (Amazon link)
- Design Patterns in Python (Udemy link)