Facultatea Calculatoare, Informatica si Microelectronica

Universitatea Tehnica a Moldovei

Analiza si principiile programarii orientate pe obiect

Lucrarea de laborator#2

Implementare principielor SOLID

Autor:
Ana Sutreac

lector asistent:
Mihail Pecari
lector superior:
Andrei Postaru

Laboratory work #2

1 Scopul lucrarii de laborator

In cadrul laboratorului e nevoie de realizat un sofware care o sa contina realizarea principielor S (Single responsibility) si I (Interface segregation).

2 Obiective

- Single responsibility
- Interface segregation

3 Laboratory work implementation

3.1 Analiza lucrarii de laborator

Single-responsibility principle

După cum puteți vedea, acest principiu afirmă că un obiect / clasă ar trebui să aibă doar o singură responsabilitate și că ar trebui să fie complet încapsulat de clasă. Aici, când vorbim despre o responsabilitate, ne referim la un motiv de schimbare. Acest principiu va conduce la o mai mare coeziune în clasă și la o legătură mai relaxantă între clasele de dependență, la o mai bună citire și la un cod cu o complexitate mai mică.

Este mult mai dificil să înțelegi și să editezi o clasă atunci când are diverse responsabilități. Deci, dacă avem mai multe motive să ne schimbăm, funcționalitatea va fi împărțită în două clase și fiecare își va ocupa propria responsabilitate.

Ne pasă de separarea funcționalităților, deoarece fiecare responsabilitate este un acces al schimbării. Atunci când o clasă are mai mult decât o singură responsabilitate, aceste responsabilități devin cuplate și această cuplare poate duce la o bază fragilă de coduri care este dificil de refăcut atunci când cerințele dvs. apar.

Interface segregation principle

Practic, odată ce o interfață devine prea mare este absolut necesar să o împărțim în interfețe mici care sunt mai specifice. Interfața va fi definită de clientul care o va utiliza, ceea ce înseamnă că clientul interfeței nu va cunoaște decât metodele legate de ele.

De fapt, dacă adăugați metode care nu ar trebui să fie acolo, clasele de implementare a interfeței vor trebui să implementeze și aceste metode. Acesta este motivul pentru care; clientul nu ar trebui să fie obligat să depindă de interfețele pe care nu le utilizează. Interface segregation principle are scopul de a menține un sistem decuplat și, prin urmare, mai ușor de refactor, de schimbare și de implementare.

Concluzie

Atunci când dezvoltăm orice software, există două concepte care sunt foarte importante: coeziunea (atunci când diferite părți ale unui sistem vor lucra împreună pentru a obține rezultate mai bune decât dacă fiecare parte ar lucra individual) și cuplajul (poate fi văzut ca un grad de dependență a o clasă, o metodă sau orice altă entitate software).

Cuplarea este prezentă, de obicei, într-o mulțime de cod și așa cum am menționat mai devreme, situația optimă ar fi să aibă o cuplare scăzută și o coeziune înaltă. Cu această scurtă introducere la cele 5 principii SOLID, trebuie să fi înțeles că ne ajută când vine vorba de asta.

References

1 Aldebran Robotics, *official page*, www.aldebaran.com/en

2 Timo Ojala, *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*, 2002

3 Biometric, www.biometricupdate.com/201501/history-of-biometrics