Zestroiu Maria-Adriana

Anul IV - IS

PRINCIPIILE SOLID

În ingineria programării, unul dintre conceptele de bază care trebuie avut în vedere atunci când se face proiectarea unui sistem este reprezentat de principiile SOLID. Aceste principii au fost prezentate și promovate de Robert C. Martin, deși acronimul de SOLID a fost introdus mai târziu, de către Michael Feathers. Strâns legate de principiile GRASP și bază a conceptelor de AGILE sau ADAPTIVE DEVELOPMENT, pot fi aplicabile oricărui limbaj de programare orientat pe obiecte și stau la baza unui design bun de software, făcând codul mai ușor de înțeles, de modificat și de reutilizat.

Numele de SOLID provine de fapt de la un set de 5 principii, și anume:

- **S**ingle-Responsibility Principle – o clasă ar trebui să aibă un singur motiv să se schimbe. Cu alte cuvinte, o clasă nu ar trebui să îndeplinească mai multe funcții deodată, pentru că astfel ar fi mult mai greu de depanat sau reutilizat.
- **O**pen/Closed Principle – o clasă ar trebui să fie deschisă la extindere, dar închisă la modificări. Acest lucru se realizează prin utilizarea interfețelor, ajungând treptat la o ierarhie de clase, fiecare cu un comportament specific în funcție de nevoile clientului.
- **L**iskov Substitution Principle – o clasă derivată ar trebui să poată fi substitută cu clasa de bază, fără ca aceasta să își piardă vreo funcționalitate. Acest principiu se bazează pe polimorfism la runtime.
- **I**nterface Segregation Principle – un client nu ar trebui să fie forțat să implementeze interfețe pe care nu le folosește. Dacă se vrea adăugarea de funcționalități noi, acestea ar trebui implementate în interfețe noi, și nu în cea existentă.
- **D**ependency Inversion Principle – un modul high-level nu ar trebui să depindă de un modul low-level. Cu alte cuvinte, o abstractizare nu trebuie să depindă de o specializare, ci invers, pentru a putea avea un cod flexibil. În mod interesant, Robert C. Martin nu consideră acest principiu ca fiind unul complet separat, ci pur si simplu rezultatul respectării principiilor LIP și OCP.

Unii ingineri software sunt de părere că aplicarea SOLID duce la cod mult prea complex și inteligibil, cu multe interfețe granulare și clase de dimensiuni foarte mici și nu atât de adaptabil precum se așteaptă developerii, iar înțelegerea unui singur modul din sistem ar putea duce la navigarea între multe fișiere.

Este important totuși să reținem că principiile SOLID sunt doar atât – principii. Cu alte cuvinte, a le respecta nu este neapărat echivalent cu a scrie cod de calitate(clean code). Ceea ce contează cu adevărat este aplicarea lor într-un mod inteligent, prin înțelegerea cerințelor sistemului, proiectarea riguroasă și mult exercițiu, obținându-se astfel arhitecturi de sistem ușor de modificat, depanat și extins în timp.