Bota Adrian
AN 4, IS, GR 1.1

Tema PSSC
Principiile SOLID

Principiile SOLID sunt un grup de cinci principii aplicate limbajelor de programare care au la baza programarea orientată pe obiecte. Rolul acestora este acela de face codul mai curat, flexibil, ușor de menținut dar și ușor de modificat deci fără relații puternice de dependență în acesta. Aceste principii au fost puse laolaltă, nu inventate, de Robert C. Martin în anul 1995 dar majoritatea erau folosite deja de-a lungul anilor de programatori.

Numele de SOLID reprezintă un acronim unde:
S = Principiul singurei responsabilități(SRP) - O clasă ar trebui să aibă o singură responsabilitate. Iar prin responsabilitate Martin dorea să zică „motiv de a se schimba”. Lăsând fiecare clasa cu doar o responsabilitate face clasele mult mai ușor de întreținut și menținut. Dacă o clasa combină două sau mai multe entități diferit
e, care se modifică din motive diferite aceste entități își vor strica logica și bunul mers una celeilalte.

O = Principiul deschis/închis(OCP) - O clasă este deschisă extinderii dar închisă modificării. Adică dacă dorim să adăugăm noi funcționalități unei clase deja existente ar trebui să putem face asta fără să modificăm din codul deja existent în clasă ci doar să adăugăm cod care să extindă clasa adăugând astfel funcționalitățile noi dorite. Dacă nu se respectă acest principiu o regulă derivată din acest principiu ar fi crearea unei alte clase care să împlinească nouă funcționalitate decât modificarea clasei deja existente.

L = Principiul Substitutiei Liskov(LSP) - Acest principiu sugerează că clasele de baza ar putea să fie cu ușurință înlocuite cu clasele lor derivate și aplicația încă să ruleze fără probleme. Deci instanțele clasei de baza pot fi înlocuite cu cele ale clasei derivate fără că aplicația să genereze rezultate incorecte sau neașteptate.

I = Principiul Segregării Interfeței(ISP) - Principiul este destul de clar și spune că este mult mai bine să ai un număr mai mare de interfețe specifice pentru fiecare cerință a unui client decât o interfață mai mare și care să cuprindă toate cerințele unui client. Deci clienții nu ar trebui să depinda de interfețe pe care nu le folosesc.

D = Principiul Inversării Dependetei(DIP) - Acest principiu vrea să zică, că este mult mai bine să lucrezi cu interfețe pentru a realiza comunicarea între două clase decât acele clase să realizeze comunicarea în mod direct una cu cealaltă. În concluzie ambele clase trebuie să depindă de abstractizări iar abstractizările să nu depindă de detalii ci detaliile să depindă de abstractizări.