Bortica Dumitru-Daniel
An IV, Grupa 1.1

Tema 1 PSSC
Principiile SOLID

SOLID este un acronim pentru primele 5 principii enunțate de Robert C. Martin, în legătură cu proiectarea orientată pe obiecte. Aceste principii sunt: Principiul Singurei Responsabilități, Principiul Deschis-Închis, Principiul Substituției Liskov, Principiul Segregării Interfeței, Principiul Inversării Dependenței.

Principiul Singurei Responsabilități

Niciodată nu trebuie să existe mai mult de un motiv pentru a modifica o clasă, însemnând că o clasă trebuie să aibă un singur job. Când o clasă are mai multe responsabilități, șansele ca ea să trebuiască să fie modificată în timp cresc și de fiecare dată când va fi modificată, riscul de a introduce noi buguri crește. Atunci când se lucrează într-un proiect, care are un număr foarte mare de linii de cod, aceste riscuri vor fi limitate, dacă o clasa va avea o singură responsabilitate.

Principiul Deschis-Închis

Entitățile software (clasele, modulele, funcțiile) trebuie să fie deschise pentru extensie, dar închise pentru modificări. Ideea din spatele regulii este că atunci când un modul a fost dezvoltat si testat, codul nu mai trebuie să fie modificat, doar ajustat în cazul corecției eventualelor buguri și deschis pentru extensie, în sensul de a adăuga noi funcționalități. Acest principiu reduce riscul de a introduce noi erori, limitând modificările la codul existent.

Principiul Substituției Liskov

Funcțiile care utilizează pointeri sau referințe la clase de bază trebuie să poată folosi instanțe ale claselor derivate fără să își dea seama de acest lucru. Obiectele dintr-un program trebuie să poată fi înlocuibile cu istanțele subtipurilor lor, fără să altereze corectitudinea acelui program. Dacă o clasă are o dependență pentru un anumit tip dat, ar trebui să poată furniza un obiect, care să aibă acel tip sau pentru oricare din subclasele sale, fără să apară rezultate neașteptate, iar clasa dependentă să nu știe tipul actual al dependenței.

Principiul Segregării Interfeței

Clienții nu trebuie să fie forțați să depindă de interfețele pe care nu le folosesc. Când o clasă depinde de altă clasă, numărul membrilor din interfață care e vizibil către clasa dependentă trebuie să fie minimizat.

Principiul Inversării Dependenței

Modulele de nivel ierarhic inferior au de a face cu funcții detailate, iar modulele de nivel ierarhic superior folosesc clase de nivel inferior, atingând taskuri mai mari. Principiul specifică faptul că unde există dependență între clase, ele trebuie să fie definite utilizând abstractizări, cum ar fi interfețe. O a doua enunțare ar fi că abstractizările nu trebuie să depindă de detalii. Detaliile trebuie să depindă de abstractizări.