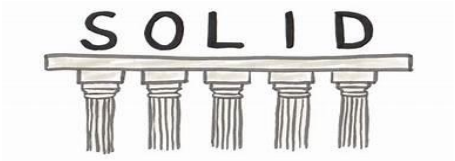

Tema 1 PSSC
Principii de Proiectare Orientată pe Obiecte

Principiile de proiectare și programare orientată pe obiect s-au dezvoltat în timp, pe baza problemelor practice şi a experienței dobândite de către alți programatori.

Dintre acestea, cele mai aplicate şi cunoscute sunt cele cinci care au fost rezumate sub acronimul **SOLID** (Single responsibility, Open-closed, Liskov substitution, Interface segregation şi Dependency inversion).

- **Principiul responsabilității unice (Single responsibility principle)**

Orice context (clasă, funcție, variabilă etc.) trebuie să aibă o unică responsabilitate, iar această responsibilitate trebuie să fie în întregime încapsulată în context. Toate serviciile sale trebuie să fie orientate pentru a servi această unică responsabilitate.

O "responsabilitate" poate fi definită și ca "motiv de schimbare". Ca exemplu, putem considera un program simplu care generează şi tipărește un raport. Un astfel de modul ar putea fi modificat din două motive: fie se schimbă conținutul raportului, fie se schimbă formatul raportului. Principiul responsabilității unice afirmă că aceste două aspecte ale problemei sunt două responsabilități separate şi trebuie tratate în clase sau module diferite.

- **Principiul deschis/închis (Open-closed principle)**

Clasele pe care le utilizăm ar trebui să fie deschise pentru extensie, dar închise pentru modificare. Acest lucru poate fi obținut prin moștenire. Nu trebuie să modificăm clasa pe care dorim să o extindem dacă creăm o subclasă a acesteia. Clasa originală este închisă pentru modificare, însă putem adăuga un cod personalizat în subclasă pentru a adăuga un comportament nou.

- **Principiul de substituție Liskov (Liskov substitution principle)**

Dacă S este un subtip al lui T, atunci obiectele de tip T pot fi substituite cu obiecte de tip S fără a afecta niciuna din proprietățile programului.

- **Principiul de segregare a interfețelor (Interface segregation principle)**

Acest principiu afirmă că niciun client nu trebuie să fie forțat să depindă de metode pe care nu le utilizează. Interfețele trebuie separate în alte interfețe mai mici şi mai specifice.

- **Principiul de inversare a dependențelor (Dependency Inversion principle)**

Modulele de pe nivelurile ierarhice superioare nu trebuie să depindă de modulele de pe nivelurile ierarhice inferioare. Toate ar trebui să depindă doar de module abstract.

De asemenea, abstractizările nu trebuie să depindă de detalii. Detaliile trebuie să depindă de abstractizări .