Laboratorul 4. TMPS

Principiile SOLID

Să se implementeze la nivel de cod principiile SOLID.

Atenție:

- Elaborați cîte o aplicație pentru fiecare principiu ales în parte. Aplicațiile elaboratre trebuie **să posede un sens**, să nu fie doar structura principiului.
- Limbajul de programare în care veți efectua laboratorul trebuie să fie orientat pe obiecte.
- Pentru fiecare laboratorul trebuie să faceți **raport**
- În codul sursă **să nu aveți comentarii**
- La apărarea laboratorului **nu se permite cu conspect,** doar cu o foaie/caiet curat pentru UML
- Studenții la care **laboratoarele sunt asemănatoare** vor fi nevoiți să răspundă din nou cu altă versiune a aplicației și raportului
- Pe baza baremului de notare **se formează nota finală**
- Lucrarea de laborator nu este susținută dacă studentul **nu corespunde cerințelor** pentru Nota 5 din baremul de notare (codul nu-i aparține, nu știe răspunsul corect la întrebările date de profesor, nu poate face modificări în codul elaborat)

Barem de notare:

Nota 5

1) Codul sursa să fie scris în totalmente de voi (se verifica la plagiat prin intermediul la Github, Bitbucket, Google, Yandex, Yahoo, Rapoartele studenților din anii anteriori)
2) Să puteți explica corect orice rînd scris de cod
3) Să puteți explica corect pentru fiecare principiu SOLID (clasele, interfețele, relațiile dintre clase și sensul lor)
4) Să puteți face schimbări corecte în codul elaborat (adăugare de atribute, metode, schimbare de nume a unor clase, utilizarea clasei abstracte în loc de interfață și invers)

Nota 6

1) Să puteți adăuga participați noi în aplicație la nivel de cod (în contextul aplicației create)

2) Să puteți șterge participați din aplicație la nivel de cod (în contextul aplicației create)

Nota 7

1) Să puteți adăuga participați noi în aplicație la nivel de diagramă UML (în contextul aplicației create)

2) Să puteți șterge participați din aplicație la nivel de diagramă UML (în contextul aplicației create)

Nota 8

1) Să puteți identifica corect participanții principiilor SOLID pe baza codului elaborat la nivel de UML

Nota 9

1) Să puteți explica relațiile principiului SOLID pe care-l răspundeți cu șabloanele de proiectare creaționale, structurale și comportamentale.

Nota 10

1) Pentru principiile SOLID elaborate să puteți argumenta cînd utilizarea acestora va fi drept un antipattern și dacă în cadrul aplicației elaborate utilizarea a fost una corectă

Întrebări la apărarea laboratorului:

- **SRP(Single Responsibility Principle)** - scopul, problema care o rezolvă, structura(diagrama de clase), aplicabilitatea, modul de implementare, argumente pro şi contra, relațiile cu șabloane de proiectare
- **OCP(Open-Closed Principle)** - scopul, problema care o rezolvă, structura(diagrama de clase), aplicabilitatea, modul de implementare, argumente pro şi contra, relațiile cu alte șabloane de proiectare
- **LSP(Liskov Substitution Principle)** - scopul, problema care o rezolvă, structura(diagrama de clase), aplicabilitatea, modul de implementare, argumente pro şi contra, relațiile cu alte șabloane de proiectare
- **ISP(Interface Segregation Principle)** - scopul, problema care o rezolvă, structura(diagrama de clase), aplicabilitatea, modul de implementare, argumente pro şi contra, relațiile cu alte șabloane de proiectare
- **DIP(Dependency Inversion Principle)** - scopul, problema care o rezolvă, structura(diagrama de clase), aplicabilitatea, modul de implementare, argumente pro şi contra, relațiile cu alte șabloane de proiectare

- Cum reduceți cuplarea și creșteți coeziunea în cod ?
- Care sunt cele mai bune practici pentru evitarea codului duplicat ?
- Care sunt câteva modalități de a evita utilizarea moștenirii multiple în programare?
- Ce caracteristici ale limbajului de programare vă ajută să vă asigurați că principiile SOLID sunt respectate în timpul dezvoltării aplicației ?
- Care este Legea lui Demeter? De ce este importantă?

Link-uri utile:

- https://climbtheladder.com/solid-design-principles-interview-questions/
- https://www.baeldung.com/solid-principles
- https://medium.com/mindorks/solid-principles-explained-with-examples-79d1ce114ace
- https://reflectoring.io/lsp-explained/
- https://www.freecodecamp.org/news/solid-principles-single-responsibility-principle-explained/
- https://stackify.com/solid-design-principles/
- https://www.toptal.com/software/single-responsibility-principle
- https://stackify.com/solid-design-liskov-substitution-principle/
- https://stackify.com/solid-design-open-closed-principle/
- https://stackify.com/dependency-inversion-principle/
- https://stackify.com/interface-segregation-principle/