Test 2 - CTS 2024

Calitate si testare Software (Academia de Studii Economice din Bucure□ti)

Observații

1. Implementările de Design Patterns trebuie să fie conform definiției GoF discutată în cadrul cursului și laboratoarelor. Implementările incomplete sau variațiile non-GoF ale patternurilor nu sunt acceptate și nu vor fi punctate.
2. Sunt luate în considerare doar implementările complete și corecte, implementările parțiale nu se punctează.
3. Soluțiile ce conțin erori de compilare nu vor fi evaluate.
4. Patternurile pot fi implementate separat sau utilizând același set de clase.
5. Implementările generale de design patterns, care nu sunt adaptate cerinței și nu rezolvă problema cerută nu vor fi luate în considerare.
6. Clasele/interfețele primite nu pot fi modificate.
7. Soluțiile vor fi verificate cu software antiplagiat. Partajarea de cod sursă între studenți nu este permisă. Soluțiile cu un nivel de similitudine peste 30% vor fi anulate.
8. Se acordă 1 punct din oficiu.

Cerințe Clean Code (nerespectarea lor va duce la depunctarea cu 2 puncte pentru fiecare cerință) - se pot pierde 8 puncte în total

1. Clasele, funcțiile, atributele și variabilele vor fi denumite conform convenției Java Mix CamelCase.
2. Fiecare pattern precum și clasa ce conține metoda `main()` vor fi definite în pachete diferite de forma `cts.nume.prenume.g<numarul_grupei>.pattern.<denumire_pattern>`, respectiv `cts.nume.prenume.g<numarul_grupei>.main` (studenții de la recuperare vor utiliza „recuperare” în loc de numărul grupei).
3. Clasele și metodele nu trebuie să încalce principiile KISS, DRY, YAGNI sau SOLID.
4. Numele claselor, metodelor, variabilelor și mesajele afișate la consolă trebuie să fie strict legate de subiectul curent (numele generice nu sunt acceptate). Mesajele afișate trebuie să simuleze scenariul cerut.
5. Nu sunt permise „magic numbers” sau valori hardcodate. Formatarea codului sursă trebuie să fie cea standard.

Realizați o aplicație software pentru o companie ce produce un analizator static de cod sursă.

3 p. Realizați o aplicație ce permite identificarea de apeluri unsafe în codul sursă C++ prin parsarea statică a codului. Orice unitate de cod va fi verificată trebuie să deriveze interfața `Code`. Utilizați un design pattern ce permite parsarea recursivă a tuturor claselor și/sau funcțiilor/metodelor dintr-o anumită clasă specificată inițial.

1.5 p. Testați implementarea prin crearea unei clase ce conține la rândul ei o funcție și alte 2 clase cu câte o funcție fiecare. Arătați că toate acestea sunt parsate. Demonstrați faptul că utilizarea unei funcții ca input va duce doar la parsarea funcției respective și nu a întregii clase.

3 p. Aplicația de tip analizator static trebuie notificată asincron de fiecare dată când un obiect de tip `Code` este modificat astfel încât să reia parsarea acestuia. Utilizați un design pattern ce implementează acest mecanism prin derivarea interfeței `Code`.

1.5 p. Testați în main implementarea prin crearea unei clase ce conține mai multe funcții și a unei funcții de sine stătătoare. Demonstrați faptul că modificarea oricăreia dintre ele va notifica aplicația ce va porni astfel parsarea.