

Principii - Clean Code

1. KISS - Keep It Simple and Stupid

◦ codul săt mai simplu

2. DRY - Don't repeat yourself

↳ o funcție să facă un singur lucru

3. YAGNI - You Ain't Gonna Need It

↳ un programator trebuie să introducă funcționalități noi doar atunci cind sunt necesare

4. S - Single Responsibility principle

O - Open - Closed principle

L - Liskov Substitution principle

I - Interface Segregation principle

D - Dependency Inversion principle

Convenții de nume:

- numberofBigNumbers - lower Camel Case
- expectedResult - Snake case
- ObtainedTemperature - Upper Camel Case
- saveBtm - Hungarian Notation

System - Hungarian:

- encodarea tipul de date și variabilei
 - ↪ mCount; bEncoded
 - ^t ^d
 - int bool

Apps - Hungarian:

- encodarea remarcării / legătură a variabilei

în aplicație

→ saveBtm

→ rgbBackground

Design Patterns

• Creational : Singleton, Builder, Factory, Factory Method, Prototype

• Structural : Adapter, Composite, Decorator, Façade, Flyweight, Proxy

• Comportamental : Chain of Responsibility, Command, Memento, Observer, State, Strategy, Template

DP Creational:

- Singleton - se poate crea o singură instanță pt. clasa, care conține stocul privat
- Factory - crează obiecte dintr-o familie de clase.
- Builder - ajută la crearea obiectelor complexe cu foarte multe atribută

• Prototype - folosit atunci cand crearea unui obiect consumă foarte multe resurse. Se crează un prototip și este folosit pentru altele.

DP Structure:

- Adapter - adaptează un framework, astfel încât lucru să se întâmple cu un alt framework (ne adaugă funcționalitate);
- Facade - simplifică lucru cu framework-uri complexe;
- Decorator - adaugă noi funcționalități la run-time;
- Composite - folosit pentru creare și gestiunea ierarhiei;
- Flyweight - gestionarea eficientă obiectelor pentru a utiliza optimă memoriei.

- Proxy - controlarea comportamentului și impunerea de condiții

DP Comportamente

- Strategy - schimbă la run-time metoda apelată
- Observer - când un obiect își schimbă starea - în observatorii sunt notificati
- Chain of Responsibility - se formează un lanț pentru rezolvarea unei probleme. În cazul în care o persoană a întâlnește nu poate rezolva problema, apelașoare la persoana succesoare;
- State - schimbă comportamentul pe baza stării în care se află obiectul;
- Command - gestiunea de comenzi aplicate asupra obiectelor

◦ Template Method - pentru gesturi
unei pattern de posă

◦ Memento - salvarea și gestionare stării
anterioare ale obiectului

Diferență Prototype și Flyweight

◦ Prototype te ajută să creezi obiecte
noi prin clonarea uneia existentă

◦ Flyweight te ajută să poarte obiecte
ocelosi într-un mod care să nu
obiecte armonioase, pentru a
economisi memorie, reprezentând
stare unică fiecărui client.

Unit Testing

⇒ o cole de teste a codului de către programator încă din etape de dezvoltare a produsului software.

Tipuri de teste software

1) Acceptance testing

↳ teste realizate de client; testea că design-ul corespunde cu ceea ce s-a dobt.

2) System testing

↳ teste la întregului sistem (teste componente), testea că sistemul funcționează conform design-ului

3) Integration Testing

↳ teste la mai multor componente care sunt combinate sau integrate

4) Unit Testing - testarea celor mai mici părți din cod (close sau metode)

Def: Unit Test - securitate de cod folosită pentru testarea unei unități bine definite din codul aplicației software. De obicei, unitatea este reprezentată de o metodă.

JUnit - este un framework ce permite realizarea și găsirea de teste pentru diverse metode din codul proiectelor dezvoltate.

→ funcționează conform a 2 design-pattern:
Composite și Command

Concepție JUnit

- Fixture - set de obiecte utilizate în test
- Test Case - clasa ce definește setul de obiecte (fixture) pentru o singură testă
- Setup - o etapă / metodă de definire a setului de obiecte (fixture) pentru o singură testă
- Tear down - o metodă / etapă de distrugere a obiectelor (fixture) după terminarea testelor
- Test Suite - selecție de cozi de teste (test cases)
- Test Runner - instrument de rulare a testelor și de oferire a rezultatelor

Unit 4

@BeforeClass

@AfterClass

@Before

@After

Unit 5

@BeforeAll

@AfterAll

@BeforeEach

@AfterEach

Cel mai cunoscut principiu de testare:

Right - BICEP
men
↓

→ dă rezultatele furnizate de către metoda

Nunt corecte

B - trebuie verificate toate limitele (Boundary)

Ri - dă rezultatele furnizate sunt
de sine mereu corecte

i - trebuie verificate relațiile inverse (inverse)

C - trebuie verificate performanțe

Corectitudinea printre verificare

încrișata, folosind metode de calcul
numerice

E - trebuie simulată și făcută obținerea
erorilor pentru verificarea comportamentului
metodelui în cazul unui erori

P - trebuie verificată părtinerea performanței (Performance) între limitele acceptanței pentru procedură software final

→ Pentru JUnit și pentru a testa timpul în care rulează o anumită metodă, este fol. un solutie: @Test(timeout = 100)

DUBLURI DE TESTARE (Mocks)

→ Dummy object - un obiect care respectă interfața dar metodele nu fac nimic sau null

→ Stub - une describere de Dummies, met. dintr-un Stub vor întocce răspunsuri conservate

→ Spy - este un Stub care gestionează și contorizarea numărului de apeluri

→ Fake - este un obiect care se comportă asemănător cu unul real, dar are ca versiune simplificată.

→ Mock - diferit de fake cele bolte.

→ Mock Object - un obiect care imita
comportamentul unui obiect real,
înse într-un mod controlat.

Metode:

- doReturn()
- doAnswer()
- when()
- thenReturn()
- thenAnswer()
- thenThrow()
- doThrow()
- useCallRealMethod()

◦ Correct Boundary Conditions

- Conformance - valoarea are formatul corect?
- Ordering - setul de valori trebuie să fie ordonat sau nu
- Range - este valoarea între limitele (maxim și minim) acceptate
- Reference - valoarea referință componentă extensă care nu sunt controlate direct
- Existence - Valoare există (non-null, non-zero, parte dintr-un set)
- Consistency - setul de date conține suficiente valori
- Time - absolut și relativ - totul se întâmplă în ordine, la momentul potrivit, într-un timp finit?

Caracteristici ale testelor unitare

- Fast
- Isolated / Independent
- Repeatable
- Self - Validating
- ~~Through~~ Timely

Principii

B - Boundary

C - Conformance - pt. orice intrare si pt. orice ieșire, trebuie să se verifice conform. cu un format / standart

O - Ordering - pt. liste, verificăm dacă ordinea ast. este cea dorită

R - Range - orice valoare din interval, trebuie

R - References - anumite metode definite de lucru externe
pe de obiecte externe → verificare și control

E - Existence - ce se întâmplă cu metoda dacă nu există

C - Consistency - metodele sunt coherente între ele

T - Time - timpul

i - inverse relationship

C - Cross Check

E - Error conditions

P - Performance

Cardinality = 2 \rightarrow exact două valori distincte
în \mathbb{N}

Ordering \rightarrow ordeneuri distincte, cu dupăloc

Gurile - Unit Tests

1) Testarea White Box mai este cunoscută și sub formele:

- Clear Box Testing
- Open Box Testing
- Glass Box Testing
- Transparent Box Testing
- Code-Based Testing
- Structural Testing

2) Ce este Test Runner?

- c. Instrument de urcare a testelor

3) Teardowns:

- a. este o metodă de distrugere a obiectelor folosite după terminarea testelor

4) assert Equals - testăm egalitatea a două obiecte

5) Ce reprezintă fixtures?

- c. set de obiecte utilizate în test

6) Testarea Black Box:

- se mai numește și testare comportamentală
- persoanele care testează nu cunosc arhitectura internă a aplicației testate
- centrul cunoașterii din datele de intrare și datele de ieșire ale aplicației

7) Test case:

- o clasa care definește setul de obiecte (fixtures) pentru a rula mai multe teste

8) Care dintre urm. reprezintă o calitate utilizată în jUnit pt. metodele automate din JUnit?

→ Before

9) Pentru ce metodă este utilizată adnotarea @BeforeClass?

→ metodă SetUpBeforeClass()

- O suite de teste ("test suite") îți permite să controlezi:

1. Excluderea anumitor categorii

⇒ poti spune „rulează-le pe toate, cu excepția testelor marcate cu tag -ul \times^4

2. Includerea unor categorii

⇒ poti filtra doar testele care au un anumit tag sau atribut de categorie

3. Selectarea unor coșuri de teste individuale

(test cases)

⇒ în multe framework-uri poti lăbi implicit clasele de test și metodele de test pe care vei să le selecții

4. Selectarea unor metode de test individuale