

Cand este folosit design pattern-ul Flyweight?

- A. Atunci cand este necesara crearea unei structure ierarhice prin compunerea de obiecte
- B. **Atunci cand trebuie sa construim foarte multe instanțe ale unei clase, obiecte ce au o parte comună**
- C. Atunci cand se dorește modificarea functionalitatii unui obiect la runtime
- D. Atunci cand se dorește adăugarea de noi functionalitati claselor existente

Ce tip de design este Proxy?

- A. **Structural**
- B. Niciuna dintre variante
- C. Comportamental
- D. Creational

Ce presupune implementarea unui Decorator?

- A. Toate variantele
- B. **Pentru metoda din interfață, se furnizează o implementare, dar nu se adaugă noi funcții abstracte**
- C. În cadrul clasei abstracte, se implementează interfața care definește familia de obiecte și se creează o interfață ... de tipul acelei interfețe
- D. În cadrul clasei decorator concret, nu se implementează noile metode din decoratorul abstract

Pentru ce tip de obiecte este recomandata folosirea design pattern-ului Prototype?

- a. obiecte a caror construire consumă foarte multe resurse
- b. **Ambele variante**
- c. obiecte a caror construire durează foarte mult

Care dintre urmatoarele afirmații despre Factory Method este adevarată?

- a. **Toate variantele**
- b. Poate fi găsit și sub denumirea de Virtual Constructor
- c. Pentru apeluri se folosesc abstractizări, nu obiecte concrete
- d. Nu folosește structuri switch sau if-else

Prin ce este reprezentat CareTaker al design pattern-ului Memento?

- a. Clasa care are obiecte pentru care se vor salva stări intermediare
- b. **Clasa care gestionează obiectele de tip Memento**
- c. Niciuna dintre variante
- d. Clasa care gestionează starea internă a obiectelor

Ce tip de design pattern este Chain of Responsibility?

- a. Structural

- b. Niciuna dintre variante
- c. Comportamental
- d. Creational

Participantii din cadrul State:

- a. Niciuna dintre variante
- b. O interfata + Clase concrete ce definesc starile obiectului + Clasa concreta ce defineste obiectul
- c. O interfata + Clase concrete ce definesc obiecte in diferite stari
- d. O interfata + Clase abstracte

Ce reprezinta numele Template Method?

- a. Este folosit atunci cand un algoritm este cunoscut si urmeaza anumiti pasi precisi
- b. Toate variantele
- c. Nu exista nicio metoda care sa implementeze algoritmul si sa apeleze toate celelalte metode
- d. Toti pasii sunt realizati de aceeasi metoda

Facade este util atunci cand:

- a. Se doreste adaugarea de functionalitati
- b. Niciuna dintre variante
- c. Se doreste crearea de obiecte cu multe proprietati optionale
- d. Se doreste simplificarea unui proces

Ce reprezinta Chain of Responsibility?

- a. Aceste obiecte posibile se ordoneaza intr-un lant, apoi cel care are problema de rezolvat apeleaza pentru prima zi din lant
- b. Este folosit atunci cand cel care are nevoie de rezolvarea unei probleme are o lista de posibile obiecte ce pot rezolva problema
- c. Toate variantele
- d. Este folosit atunci cand cel care are nevoie de rezolvarea unei probleme nu stie exact cine poate rezolva problema

Participantii din cadrul Proxy:

- a. Niciuna dintre variante
- b. O interfata + Clasa concretă Proxy
- c. O interfata + Clasa concreta Entitate + Clasa concreta Proxy
- d. O interfata +Clasa abstracta Entitate + Clasa abstracta Proxy

Care din urmatoarele variante reprezinta contextul in care se utilizeaza Observer?

- a. Baze de date
- b. DNS Resolver
- c. Niciuna dintre variante
- d. Model View Controller

Care este partea componentă care se ocupă de faptul că poate fi creat un singur obiect al clasei Singleton?

- a. Niciun răspuns
- b. Utilizatorul
- c. Apelatorul
- d. Clasa

Cum este implementat Adapter-ul de obiecte?

- a. Clasa Adapter moștenește clasa existentă și implementează interfața la care trebuie să facă adaptarea
- b. Clasa Adapter conține o instanță a clasei existente și implementează interfața la care trebuie să facă adaptarea
- c. Niciun răspuns
- d. Clasa Adapter conține instanțe ale celor două clase existente

Witch design pattern provides a single class which provides simplified methods required by client and delegates call to those methods?

- a. Adapter pattern
- b. Singleton
- c. Builder (nu asta)
- d. Facade
- e. Prototype
- f. State

Design patterns are divided into multiple fundamental groups. Select one or more categories.

- a. J2EE Patterns
- b. Behavioral
- c. Structural
- d. Creational
- e. MVC

Which design pattern defines one-to-many dependency among objects?

- a. Facade
- b. Observer
- c. Singleton
- d. Factory method

Which of the following SOLID principles is based on “Program to interfaces, not implementations”:

- a. Liskov substitutions
- b. It's not part of SOLID
- c. Interface Segregation
- d. Single Responsibility

e. Dependency Inversion

Which dp ensures that only one object of particular class gets created?

- a. Filter
- b. Adapter
- c. State
- d. Singleton**
- e. Bridge
- f. Builder

Which dp suggests multiple classes through which request is passed and multiple but only relevant classes carry out operations on the request?

- a. Bridge
- b. CoR**
- c. Builder
- d. Adapter
- e. Singleton
- f. State

Dacă doriți să vă testați propria funcție Fibonacci (pe baza algoritmului $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$) folosind funcția următoare, ce strategie de test utilizați?

```
public int getFibonacciNumber(int n) {  
    double phi = 1.61803;  
    int val1 = (int) Math.pow(phi, n);  
    int val2 = (int) Math.pow(-1, n);  
    int val3 = (int) Math.pow(phi, n);  
  
    return ((int)(val1 - (val2 / val3)) / (int)Math.sqrt(5));  
}
```

Select one:

- a.
Inverse Relationship
- b. Conformance
- c. Cross-Check
- d.
Error condition

Implementați clasa Product care conține atributul cantitate. Atributul acceptă valori între 5 și 100 (inclusiv aceste 2 valori).

Pentru a permite clientilor să modifice cantitatea de produs, implementați și metoda setQuantity (int value). Metoda ar trebui să valideze valoarea de intrare.

Dacă implementați următorul test, ce valori ar trebui să folosiți pentru „X” dacă testul este unul de tip BOUNDARY? Alegeți una sau mai multe opțiuni

```
@Test  
public void testSetQuantity() {  
  
    Product product = new Product();  
  
    int testValue = X;  
    product.setQuantity(testValue);  
    int obtainedValue = product.getQuantity();  
  
    assertEquals(testValue, obtainedValue);  
}
```

Select one or more:

- a. 6
- b. 5
- c. 100
- d. 4
- e. 99
- f. 101

Implementați clasa Product care conține atributul cantitate. Atributul acceptă valori între 5 și 100 (inclusiv aceste 2 valori).

Pentru a permite clienților să modifice cantitatea de produs, implementați și metoda setQuantity (int valoare). Metoda ar trebui să valideze valoarea de intrare.

Dacă implementați următorul test, ce valori ar trebui să folosiți pentru „X” dacă testul este unul de tip RIGHT? Alegeti una sau mai multe opțiuni

```
@Test  
public void testSetQuantity() {  
  
    Product product = new Product();  
  
    int testValue = X;  
    product.setQuantity(testValue);  
    int obtainedValue = product.getQuantity();  
  
    assertEquals(testValue, obtainedValue);  
  
}
```

Select one or more:

- a. 4
- b. 101
- c. 5
- d. 6
- e. 100
- f. 99

Dacă testăm metoda isFibonacciNumber() cu valoarea 8 și obținem rezultatul TRUE, efectuăm un test de tip _____ (alegeti opțiunea corectă)

```
private boolean perfectSquare(int number) {  
    return ((int) Math.sqrt(number)) * ((int) Math.sqrt(number)) == number;  
}  
  
public boolean isFibonacciNumber(int number) {  
    int val = 5 * number * number;  
    return perfectSquare(val - 4) || perfectSquare(val + 4);  
}
```

Select one:

- a.
Error condition
- b.
Performance
- c.
Cross Check
- d.
Right

Pentru a efectua un test de condiții de eroare (Error conditions) pentru metoda isFibonacciNumber(), trebuie să utilizăm valoarea:

```
private boolean perfectSquare(int number) {  
    return ((int) Math.sqrt(number)) * ((int) Math.sqrt(number)) == number;  
}  
  
public boolean isFibonacciNumber(int number) {  
    int val = 5 * number * number;  
    return perfectSquare(val - 4) || perfectSquare(val + 4);  
}
```

Select one:

- a. 4
- b. 0
- c. 3
- d. 2
- e. 1

Pentru un test de tip RIGHT test, daca apelam metoda urmatoare cu valoarea 4 (unde 4 reprezinta un index), asteptam sa primim valoarea ?

```
public int getFibonacciNumber(int n) {  
    double phi = 1.61803;  
    int val1 = (int) Math.pow(phi, n);  
    int val2 = (int) Math.pow(-1, n);  
    int val3 = (int) Math.pow(phi, n);  
  
    return ((int)(val1 - (val2 / val3)) / (int) Math.sqrt(5));  
}
```

Select one:

- a. 3
- b. 5
- c. 1
- d. 2
- e. 4

Pentru a efectua un test de tip ORDERING pentru metoda următoare, trebuie să îl apelăm cu următoarele valori:

```
public int getFibonacciNumber(int n) {  
    double phi = 1.61803;  
    int val1 = (int) Math.pow(phi, n);  
    int val2 = (int) Math.pow(-1, n);  
    int val3 = (int) Math.pow(phi, n);  
  
    return ((int)(val1 - (val2 / val3))) / (int)Math.sqrt(5));  
}
```



Select one:

- a. 2, 3, 4
- b. 1, 1, 2
- c.
1, 1, 4
- d. 1, 1, 3

Implementați clasa Product care conține atributul cantitate. Atributul acceptă valori între 5 și 100 (inclusiv aceste 2 valori).

Pentru a permite clienților să modifice cantitatea de produs, implementați și metoda setQuantity (int valoare). Metoda ar trebui să valideze valoarea de intrare.

Dacă implementați un test unitar, ce valori ar trebui să utilizați ca intrare pentru setQuantity() pentru un test de tip RANGE? Alegeti una sau mai multe opțiuni



Select one or more:

- a. 5
- b. 100
- c. 6
- d. 4
- e. 99

1. Testul unitar de mai jos poate fi un test de tip? (testCalculeazaMedie())
Alegeti una sau mai multe optiuni:

Testul unitar de mai jos poate fi un test de tip?

```
@Test(timeout = 1000)
public void testCalculeazaMedie() {
    Student s = new Student();
    double result = s.calculeazaMedie();
    assertEquals(0, result, 0.01);
}
```

- a. Ordering
 - b. Performance
 - c. Cardinality
 - d. Error condition
 - e. Time
2. Dezvolti o aplicatie software destinata unei companii care realizeaza instalarea de geamuri Termopan pentru clienti. Pentru montarea ferestrelor termopan unei cladiri se urmeaza un set prestabilit de etape cunoscute de catre membrii echipei de instalare. Sa se implementeze un modul ce ofera posibilitatea de a stimula montarea de geamuri termopan mai multor tipuri de cladiri tinand cont de faptul ca modul de montare a acestora este acelasi indiferent de tipul de cladiri, iar pasii parcursi sunt aceiasi. **Template**

Managerul lucrarii furnizeaza in fiecare dimineata comenzi catre echipele formate. In timpul zilei acestia trebuie sa execute toate comenziile primite de dimineata. Sa se implementeze modul care permite trimiterea de comenzi de catre managerul lucrarii catre echipele de lucru. Comenziile sunt gestionate de seful de echipa.

Care sunt cele 2 design patternuri care rezolva optim aceste doua probleme?

Prototype, Observer, Adapter, Decorator, Flyweight, Strategy, Builder, Singleton, State, **Command**, Composite, Chain of Responsibility, **Template**

3. O institutie bancara realizeaza platile cu cardul prin intermediul unei interfete IViza. Banca doreste sa renunte la acest furnizor de carduri si sa foloseasca un altul ce expune o interfata de tip IManagerCard, fara a modifica foarte mult solutia existenta. In plus, deoarece crearea unui card bancar nou dureaza minim 1 saptamana, banca doreste o modalitate prin care sa ii furnizeze titularului o clona temporara digitala a cardului pana la furnizarea celui nou. Ce DP (unul sau mai multe) ati putea utiliza pentru a rezolva cele 2 probleme?

- a. Composite
- b. Flyweight
- c. Prototype
- d. Proxy
- e. Observer
- f. **Adapter**
- g. Decorator
- h. Command
- i. Chain of Responsibility

4. Ce pattern permite incapsularea unui obiect pentru a-i oferi un comportament specific unei alte interfete?

- a. Nici unul dintre cele enumerate
- b. Adapter - grile dubioase+Marius&Teo=<3
- c. Proxy
- d. Template
- e. Strategy
- f. Command?? --grile dubioase
- g. State
- h. Decorator
- i. Observer
- j. Memento

5. Ce DP este implementat corect si complet in clasa de mai jos? (DesignPattern field)

```
public class DesignPattern implements Cloneable {
    private static DesignPattern field;

    public static DesignPattern getField() {
        if(field == null) {
            field = new DesignPattern();
        }
        return field;
    }

    @Override
    public Object clone() throws CloneNotSupportedException {
        return field;
    }
}
```

- a. Singleton
- b. Flyweight
- c. Builder
- d. Factory Method
- e. Prototype
- f. niciunul
- g. Simple Factory

6. Incalcarea carui principiu poate fi sugerata de secevena de cod de mai jos? (unused)

```
@SuppressWarnings("unused")
private void method() {
    System.out.println("Method");
}
```

- a. Hollywood Principle
- b. DRY
- c. KISS

- d. YAGNI
- e. SOLID

7. Ce pattern permite incapsularea unui obiect in vederea controlarii accesului la acesta?
- a. Command
 - b. Niciunul
 - c. Adapter
 - d. Observer
 - e. Memento
 - f. Decorator - Irina
 - g. Proxy - Mete, Lavi, grile dubioase
 - h. State
 - i. Template
 - j. Strategy

8. Luati in considerare urmatorul UnitTest. Ce afiseaza dupa executarea sa?

```
public class TestMetode {  
  
    @BeforeClass  
    public void setUpBeforeClass() {  
        System.out.print("Before ");  
    }  
  
    @Test  
    public void test1() {  
        System.out.print("Test1 ");  
    }  
  
    @Test  
    public void test2() {  
        System.out.print("Test2 ");  
    }  
  
    @After  
    public void tearDown() {  
        System.out.print("After ");  
    }  
}
```

- a. Before Test1 After Before Test2 After
- b. Before Test1 Before Test2 After
- c. Before Test1 Test2 After
- d. Before Test1 After Test2 After

9. Daca realizam un test cu valorile 3 si 4 si asteptam rezultatul 5, pentru a metoda din figura urmatoare. Ce fel de test am realizat? Alegeti una sau mai multe optiuni.

```
public double teoremaLuiPitagora(int cateta1, int cateta2) {  
    return Math.sqrt(cateta1 * cateta1 + cateta2 * cateta2);  
}
```

- a. Boundary
- b. Range**
- c. Order
- d. Right - grile dubioase
- e. Time
- f. Performance
- g. Reference

10. O companie doreste realizarea unui editor de texte online asemanator Google Documents/Microsoft Word online. Ce DP sugerati sa fie implementate pentru a da utilizatorilor posibilitatea sa revina la versiunile anterioare ale documentelor si la fiecare modificare locala sa se transmita automat si asincron un mesaj catre server pentru a face salvarea documentului? Una sau mai multe optiuni:

- a. State
- b. Chain of Responsibility
- c. Observer - grile dubioase
- d. Proxy
- e. Command - Lavi ca zice ca e asincron
- f. Memento - Lavi si eu zic asta**
- g. Template
- h. Decorator
- i. Strategy

11. Testul unitar de mai jos poate fi un test de tip? (testSetNameForNull)

```
@Test(expected = InvalidNameException.class)  
public void testSetNameForNull() throws InvalidNameException {  
    Student s = new Student();  
    s.setName(null);  
}
```

- a. Conformance
- b. Error condition**
- c. Right
- d. Existence - grile dubioase
- e. Performance

12. Ce principii SOLID sunt incalcate de metoda de mai jos? (calcul , flag)

```

public long calcul(int flag, int value1, int value2) {
    if(flag == 1) {
        return value1 + value2;
    } else if(flag == 2) {
        return value1 * value2;
    } else {
        return value1 - value2;
    }
}

```

- a. LSP
- b. DIP
- c. ISP
- d. OCP - grile dubi**
- e. SRP

13. Dezvolti o aplicatie software pentru un restaurant. Restaurantul are mai multi sefi de sala care au in subordine ospatari. Ospatarii nu ai nicio persoana in subordine. Se doreste implementarea unui modul pentru reprezentarea diagramei organizationale a ospatarilor si a sefilor de sala.

In cadrul restaurantului exista un singur Chef care gateste toate felurile de mancare. Trebuie implementata clasa Chef, astfel incat sa existe unul singur in restaurant la un moment dat, si care sa primeasca toate felurile de mancare de gatit. In cazul in care sunt mai multi clienti la un moment dat in restaurant, toate felurile de mancare pe care trebuie sa le gateasca Chef vor fi ordonate intr-o coada de asteptare.

Care sunt cele 3 dp care rezolva optim aceste probleme?

- a. Template
- b. Strategy
- c. Builder
- d. Chain of Responsibility
- e. Prototype
- f. Adapter
- g. State
- h. Observer
- i. Memento
- j. Singleton**
- k. Composite -1**
- l. Decorator**
- m. Command - 2**

14. Cele cinci principii SOLID conduc la scrierea codului intr-un mod:

- a. usor de modificat**
- b. autonom
- c. util
- d. independent
- e. distinct

15.Cate teste unitare vor rula cu succes din test case-ul de mai jos? (TestCase)

```

public class TestCase {

    @Test
    public void test1() {
        fail();
    }

    @Test
    public void test2() {
    }

    @Test
    public void test3() {
        Object o1 = new Object();
        Object o2 = o1;
        assertEquals(o1, o2);
    }

}

```

- a. 2
- b. 1
- c. 0
- d. 3
- e. test case-ul nu poate fi executat

16. Ce principiu SOLID este incalcat in urmatoarea secventa de cod sursa? Service

```

class Service {
    private String name;

    private void method() {
        //...
    }
    //...
}

public class Model {
    public Model(Service service) {
        //...
    }
}

```

- a. LSP
- b. ISP-Irina**
- c. OCP
- d. DIP - Mete, Lavi, grile dubioase**
- e. SRP - Diana**

17. Dezvolti o solutie software bazata pe 2 microservicii ce au rolul de a gestiona repartizarea locurilor in camin pentru o universitate cu foarte multi studenti. Solutia trebuie sa proceseze un numar mare de solicitari intr-un timp scurt si din acest motiv se implementeaza un mecanism asincron de prelucrare a solicitarilor studentilor. Un

microserviciu are rolul de a valida si inregistra cererile studentilor, ele urmand a fi stocate si procesate printr-un mecanism de tip FIFO. Deoarece volumul mare de cereri nu permite procesarea lor pe acelasi microserviciu, detaliiile cererilor sunt incapsulate si transferate pe un alt microserviciu unde sunt si execute.

Care sunt cele 2 dp care rezolva optim aceste doua probleme? **COMMAND, ADAPTER?**

Flyweight, Command-Lavi zice de asincron-grile dubioase, Adapter, prototype si builder-grile dubi

18. Ce model permite subclaszelor sa decida cum sa implementeze pasii intr-un algoritm?

- a. Observer
- b. Niciunul
- c. Memento
- d. Proxy
- e. Adapter
- f. Command
- g. Decorator
- h. Template-grile dubi**
- i. State
- j. Strategy

19. Care dintre urmatoarele enunturi descrie modelul Facade corect?

- a. Acest model este utilizat acolo unde trebuie sa tratam un grup de obiecte in mod similar ca un singur obiect
- b. Acest model permite utilizatorului sa adauge functionalitati noi la un obiect existent fara a-i modifica structura
- c. Acest model permite definirea unei familii de algoritmi, fiecare fiind incapsulat intr-un obiect. Clientul poate folosi orice algoritm fara sa fie afectat de variația acestora.
- d. Acest model este utilizat in principal pentru a reduce numarul de obiecte create si pentru a reduce amprenta de memorie
- e. Niciuna din aceste variante
- f. Acest model ascunde complexitatea sistemului/modului si ofera clientului o interfata pe care o poate folosi pentru a accesa/utiliza sistemul**

20. Dezvolti o aplicatie software pentru o companie de inchirieri masini. Compania detine o flota de masini DACIA din diferite categorii: Break, Berlina sau SUV. Atunci cand un client doreste sa inchirieze o masina specifica tipul dorit, iar aplicatia furnizeaza un obiect masina de tipul dorit. Trebuie implementat modulul care creeaza acest obiect. **FACTORY**

Plata pentru inchiriere se face de catre client in momentul prezentarii la ghiseu. Modul de plata poate fi cash in lei, cash in valuta sau cu cardul. Modul de plata il decide clientul in momentul efectuarii platii. Trebuie implementat modulul care permite efectuarea platii.

STRATEGY

Care sunt cele 2 dp care rezolva optim aceste 2 probleme?

Decorator, Strategy, State, Observer, Proxy, Adapter, Memento, Chain of Responsibility, Builder, Composite, Singleton, Command, Prototype, Flyweight, Factory(simple sau method), Template

21. Pentru un test de tip RIGHT test, daca apelam metoda urmatoare cu valorile 9 si 40, asteptam sa primim valoarea: (teoremaLuiPitagora)

```
public double teoremaLuiPitagora(int cateta1, int cateta2) {  
    return Math.sqrt(cateta1 * cateta1 + cateta2 * cateta2);  
}
```

- a. 31
- b. 260
- c. 49
- d. 419
- e. 41

22. Dezvolti o solutie software pentru asociatiile de proprietari dintr-un cartier de blocuri care sa le permita sa gestioneze detaliiile legate de costurile de intretinere aferente unei luni pentru fiecare proprietar de apartament. Aceste date (numarul apartamentului, costul apei calde, costul caldurii, costul gazelor, etc.) sunt stocate in fiecare luna. Soluția trebuie să permită salvarea acestor date la finalul lunii astfel incat să poată oferi in orice moment informatii cu privire la istoricul acestor costuri pe fiecare apartament.

Independent de cerinta anterioara, blocul are la intrare un interfon prin care este solicitat/controlat accesul. Deoarece copiii din cartier abuzeaza de acest serviciu si deranjeaza locatarii, se doreste adaugarea in sistem a unei camere video care sa verifice dacă apelantul este adult sau copil. Aceasta camera va fi conectata la sistemul existent (nu se modifica interfonul) si va controla dacă se poate fi folosit interfonul sau nu. Dacă camera detecteaza un copil atunci intertonul nu va suna la apartamentul apelat. Care sunt cele 2 design patternuri care rezolva optim aceste 2 probleme ?

- a. Prototype
- b. Proxy si Memento - Teo&Marius
- c. Factory (simple sau method)
- d. Strategy
- e. Decorator
- f. Builder
- g. Flyweight
- h. Observer
- i. Singleton
- j. Adapter
- k. Composite

23. Pentru metoda cmmdc este realizat un test in care este folosita metoda "realizeazaCalcul". Ce fel de test este realizat?

- a. Cross-check - grile dubi
- b. Performance
- c. Order
- d. Inverse relationship
- e. Time

```

public int cmmdc(int param1, int param2){
    while(param1!=param2){
        if(param1 > param2)
            param1 -= param2;
        else
            param2 -= param1;
    }
    return param1;
}

public int realizeazaCalcul(int a, int b){
    while(b != 0)
    {
        int r = a % b;
        a = b;
        b = r;
    }
    return a;
}

```

24. Urmatoarea implementare de Singleton este gresita deoarece:

O optiune:

```

public class Singleton {
    private int atribut;
    private static Singleton instanta=null;

    private Singleton(int parametru) { this.atribut=parametru; }

    public static Singleton getInstanta(int parametru){
        if(instanta==null){
            return new Singleton(parametru);
        }else{
            return instanta;
        }
    }
}

```

tar

- instanta este statica
- constructorul este privat si trebuie sa fie public, astfel incat sa poata fi apelat
- functia getInstanta() trebuie sa fie privata
- instanta nu este initializata niciodata prin apelul constructorului
- instanta este initializata cu NULL atunci cand este declarata

25. Dezvoltă o soluție software pentru asociațiile de proprietari dintr-un cartier de blocuri. În cadrul cartierului sunt mai multe meseriași (instalaitori, electricieni, etc) care repară și îndreaptă lucrurile din tot cartierul. Atunci când apare o problemă, locatarii solicită intervenția acestora și solicitarea lor este salvată într-o coadă așteptare, iar atunci când unul dintre

meseriașii cartierului este liber, va prelua această problemă și o va rezolva. Să se implementeze modulul care permite gestionarea cozii de comenzi de probleme către meseriașii cartierului. **COMMAND**

De asemenea, se dorește implementarea unei soluții care să permită locatarilor unui bloc să aleagă modul în care sunt publicate anunțurile din bloc. Asociația proprietari poate decide în cadrul ședințelor anuale ca anunțurile să fie printate la avizier sau să fie trimise prin email sau prin sms, etc. Odată soluția aleasă ea va fi menținută până la ședința următoare.

STRATEGY

26. Ce principiu SOLID incalca fragmentul de cod din imagine?

```
public interface IStudent{
    void getVarsta();
    void eat();
    void move();
    void rentAStudio();
    void sustineExamen();
    void run();
    void purchaseHouse();
    void packForTrip();
}
```

- a. Interface segregation - grile dubi
- b. Single responsibility
- c. Open-Closed
- d. Liskov principle
- e. Dependency inversion trei teste

27. Pentru metoda din figura urmatoare sunt realizate trei teste

Pentru metoda din figura urmatoare sunt realizate trei teste.

```
public double teoremaLuiPitagora(int cateta1, int cateta2) {  
    return Math.sqrt(cateta1 * cateta1 + cateta2 * cateta2);  
}
```

```
@Test  
public void test1() {  
    int cat1=3;  
    int cat2=4;  
    double ipotenuza=obiect.teoremaLuiPitagora(cat1,cat2);  
    assertEquals(cat1, Math.sqrt(Math.pow(ipotenuza,2)-Math.pow(cat2,2)), delta: 0.01);  
}  
  
@Test  
public void test2() {  
    int cat1=3;  
    int cat2=4;  
    double ipotenuza=obiect.teoremaLuiPitagora(cat1,cat2);  
    assertEquals(ipotenuza, Math.sqrt(Math.pow(cat1,2)+Math.pow(cat2,2)), delta: 0.01);  
}  
  
@Test  
public void test3() {  
    int cat1=3;  
    int cat2=4;  
    double ipotenuza=obiect.teoremaLuiPitagora(cat1,cat2);  
    assertEquals( expected: 5, ipotenuza, delta: 0.01);  
}
```

Testele din figura de mai sus acoperă următoarele principii:

Alegeți una sau mai multe opțiuni:

- a. Cross-check
- b. Range - toate
- c. Time
- d. Right - 3
- e. Inverse Relationships - grile dubi-1,2
- f. Ordering
- g. Reference
- h. Performance
- i. Error Conditions

28. Realizați o aplicație folosind modelul MVC (Model-View-Controller). Din acest motiv doriti sa decuplati Modelul (clasele) de View (interfața utilizator). Selectați din lista de mai jos ce Design Patterns ati utiliza pentru a actualiza unul sau mai multe campuri din interfața utilizator la schimbarea unui atribut din clasa, respectiv apelul unei metode din clasa la click pe un buton din interfața utilizator.

Alegeți una sau mai multe opțiuni:

- a. Decorator
- b. Builder
- c. Command
- d. State
- e. Strategy
- f. Observer - Marius**
- g. Template
- h. Proxy

29. Ce combinatie de pattern-uri (unul sau mai multe) este prezenta in codul de mai jos?

```
interface IDP {
    default void execute() {
        String message = receive();
        send(message);
    }
    void send(String message);
    String receive();
}

class Class implements IDP {
    @Override
    public void send(String message) {
        System.out.println(message);
    }
    @Override
    public String receive() {
        String result = "";
        //...
        return result;
    }
}

public class DP implements IDP {
    private IDP idp;
    //...
    @Override
    public void send(String message) {
        System.out.println("Message: ");
        idp.send(message);
    }
    @Override
    public String receive() {
        return idp.receive();
    }
}
```

Alegeți una sau mai multe opțiuni:

- a. Template method
- b. Proxy
- c. State
- d. Composite
- e. Strategy
- f. Command**
- g. Facade
- h. Prototype
- i. Decorator

30. Pentru metoda onMetoda() ce este corect implementata, care dintre urmatoarele valori pot fi folosite pentru teste de tip Range

Alegeti una sau mai multe optiuni:

```
int oMetoda(int valoare) throws Exception {  
    int rez = 0;  
  
    switch (valoare) {  
  
        case 1:  
            rez = 10;  
            break;  
  
        case 5:  
            rez = 20;  
            break;  
        case 10:  
            rez = 30;  
            break;  
  
        default:  
            throw new Exception("Valoare gresita");  
    }  
  
    return rez;  
}
```

- a. Metoda nu poate fi testata prin teste Boundary, deoarece genereaza o exceptie
- b. 20
- c. 0
- d. 1- grile dubi gen 1,5,10 Mete, Alex
- e. 5
- f. 10
- g. 30

32. Prin metoda urmatoare, care este numarul minim de teste unitare (cu diferite combinatii de valori) ce trebuie scrise pentru a asigura un code coverage de 100%? 3

```
public int calcul(int a, int b){  
    int rez = 0;  
  
    if(a < 0)  
        return 10;  
  
    if(b < 0)  
        rez = 20;  
    else  
        rez = 30;  
  
    if(a >= 0 )  
        rez += 5;  
  
    return rez;  
}
```

33. Dezvolti o aplicație software pentru un restaurant. În cadrul restaurantului există o echipă de Chefi care gătesc toate felurile de mâncare. Trebuie implementat modulul care să permită trimitera tipurilor de preparate ce urmează a fi gătite. În cazul în care sunt mai mulți clienti la un moment dat în restaurant, toate felurile de mâncare pe care trebuie să le gătească echipa de Chefi, vor fi ordonate intr-o coadă de așteptare. **COMMAND**

Bucătăria trebuie să anunțe toți ospătarii în momentul în care livrează un fel de mâncare pregătit, astfel încât ospătarii să știe să se duca să il ia și să il ducă la masa de la care a fost comandat. Bucătăria anunță doar ospătarii care lucrează în acel moment și au date comenzi către bucătărie. **OBSERVER**

Care sunt cele 2 design patternuri care rezolvă optim aceste 2 probleme?

Strategy, Prototype, Chain of Responsibility, Proxy, Decorator, Command, Flyweight, Template, Builder, Memento,

34. O companie dorește să poată modifica la run-time modalitatea de aprobare a unor documente. **DECORATOR? STRATEGY????**

Dacă se utilizează o procedură de urgență atunci este nevoie de aprobarea managerului direct și apoi a CEO-ului. Dacă se utilizează o procedură normală atunci toți managerii ierarhic superiori trebuie să aprobă documentul în ordine. **ORI COMPOSITE ORI CHAIN?? DAR CRED CĂ MAI MULT COMPOSITE**

Ce combinatie de Design Patterns (unul sau mai multe) sugerati să fie utilizata pentru a rezolva aceasta problema?

- a. Composite-Marius
- b. Obsserver
- c. Proxy
- d. Command
- e. Decorator - Marius
- f. State
- g. Chain ????
- h. Strategy
- i. Adapter

35. Dezvolti o soluție software pentru asociația de proprietari dintr-un bloc care să permită gestiunea eficientă a evenimentelor ce pot apărea, evenimente care necesită luarea unor măsuri adecvate și care să protejeze viața locatarilor. Soluția procesează diferite mesaje/evenimente primite de la senzori sisteme din bloc și va lua următoarele decizii

- dacă s-a declanșat alarmă de incendiu atunci este notificat serviciul 112
- dacă este anunțată o inundație atunci este notificat administratorul
- dacă este doar o informare atunci sunt notificați locatarii

- (optional) pentru alarmele de incendiu și inundație sunt anunțați și locatarii **CHAIN**

Independent de implementarea scenariului anterior găsiți o soluție care să permite integrarea sistemului existent de procesare a alertelor (cu interfața IProcesareEveniment) în sistemul național de alertare ce este construit în jurul interfeței IAlertarePublică. **ADAPTER**

Chain of Responsibility, Memento, Adapter, State, Proxy, Strategy, Observer, Decorator, Composite, Prototype, Template, Builder, Factory, Singleton, Command, Flyweight

36. Ce principiu SOLID este încalcăt în urmatoarea secvență de cod sursă? NumarNatural

```
public class NumarNatural {
    private int valoare;

    public void setValoare(int valoare) {
        if(valoare > 0) {
            this.valoare = valoare;
        }
    }

    public int getValoare() {
        return valoare;
    }
}

class NumarIntreg extends NumarNatural {
    private boolean semn;
    //...
}
```

- ISP
- LSP grile dubi, mete, alex, nu irina
- SRP grile dubi
- DIP
- OCP

37. Doriti realizarea unei solutii software de gestionare a unor componente electronice smart ce isi schimba culoarea. Utilizatorii pot crea componente noi prin combinarea de componente existente, iar acestea isi vor schimba culoarea toate odata. Ce Design Patterns (unul sau mai multe) propuneti pentru a da utilizatorilor posibilitatea de a schimba culoarea tuturor componentelor detinute in acelasi timp avand in vedere urmatoarele: 1. se pot realiza modele utilizand orice combinatii de componente, iar acestea sunt interschimbabile; 2.interfata unei componente este complexa si doriti sa expuneti doar functia de schimbare a colorii catre utilizatorii finali. **FACADE, OBSERVER?**

State, **Facade-Lavi**, Composite, **Observer**, Decorator, Strategy, Proxy, chain of Responsability, Command

GRILE RESTANTA

1. O colectie de teste unitare se numeste:

Alegeti una sau mai multe optiuni:

- a. Test Case ??
- b. Unit Test
- c. Test
- d. JUnit
- e. Setup
- f. Fixture
- g. Suite

2. In cadrul unei suite de teste (test suite) pot fi specificate spre a fi executate:

Alegeti una sau mai multe optiuni:

- a. test cases
- b. alte suite
- c. doar anumite categorii de teste
- d. test methods
- e. toate testele cu exceptia unor categorii

3. Trebuie realizat un test de Right pentru metoda din imagine. Daca metoda este apelata pentru numerele 120 si 90, rezultatul este comparat cu:

```
public int cmmdc(int param1, int param2){  
    while(param1!=param2){  
        if(param1 > param2)  
            param1 -= param2;  
        else  
            param2 -= param1;  
    }  
    return param1;  
}
```

Alegeti o optiune:

- a. 120
- b. 10
- c. 30
- d. 40
- e. 90

4. Ce design pattern defineste o familie de algoritmi, incapsuleaza fiecare algoritm si permite interschimbarea acestora:

- a. Composite
- b. Template++
- c. Adapter

- d. Factory Method
 - e. Strategy-Alex--scrie in foi
 - f. Proxy
 - g. Abstract Factory
 - h. Decorator
5. Pentru functia urmatoare, selectati combinatia de valori ce pot fi utilizate pentru a testa metoda si pentru a asigura un code coverage de 100%.

```
public int oMetodaSimpla(int valoare) throws Exception
{
    int rezultat = 0;
    switch(valoare) {
        case 10:
            rezultat = 10;
            break;
        case 50:
            rezultat = 20;
            break;
        case 100:
            rezultat = 30;
            break;
        default:
            throw new Exception("Valoare gresita");
    }

    return rezultat;
}
```

- a. 10,50,100,101
 - b. 10,50,100
 - c. 10,50
 - d. 50,100,101
 - e. 10,100,101
6. O metoda ce realizeaza un back-up al bazei de date trebuie executata in fiecare noapte la ora 2:00. Doriti sa scrieti un test unitar ce verifica daca metoda chiar este apelata la acea ora. Acest test este de tip:
- a. Cardinality
 - b. Error-condition
 - c. Right
 - d. Performance
 - e. Time
7. Realizati o aplicatie ce permite completarea de chestionare online. Indiferent de tipul chestionarului (Google Forms, Microsoft Forms, etc) este nevoie ca datele sa fie colectate intr-o anumita ordine: mai intai se colecteaza datele personale, mai apoi datele legate de chestionarul efectiv si la final datele legate de platforma utilizata. Ce DP puteti utiliza pentru a va asigura ca indiferent de tipul chestionarului regulile de mai sus sunt respectate?
- a. Command
 - b. Facade
 - c. Decorator
 - d. Template method

- e. Composite
 - f. Strategy
 - g. State
 - h. Prototype
 - i. Proxy
8. Care dintre următoarele enunțuri sunt corecte (unul sau mai multe) despre pattern-ul de tip Factory?
- a. Pattern-ul decoupleaza clientul de detaliiile specifice crearii de obiecte - Albert
 - b. Acest tip de pattern face parte din categoria creationale
 - c. Pattern-ul se refera la obiectul nou creat folosind o interfata comuna
 - d. Creeaza obiecte fara a expune clientului logica si detaliiile procesului de creatie.-Albert
9. Pentru urmatoarea implementare a unui pattern de tip **Builder**, indicați una sau mai multe greșeli de implementare:

```

1  public class Tesla {
2
3      private int maxSpeedLimit;
4      private float price;
5      private String color;
6      private int batteryLevel;
7
8      public Tesla() { }
9
10     private Tesla(int maxSpeedLimit, float price, String color, int batteryLevel) {
11         super();
12         this.maxSpeedLimit = maxSpeedLimit;
13         this.price = price;
14         this.color = color;
15         this.batteryLevel = batteryLevel;
16     }
17
18     public static class TeslaConfigurator {
19         private Tesla tesla = new Tesla();
20         public TeslaConfigurator(float price) {
21             tesla.price = price;
22         }
23
24         public TeslaConfigurator setColor(String color) {
25             tesla.color = color;
26             return this;
27         }
28
29         public TeslaConfigurator setMaxSpeedLimit(int maxSpeedLimit) {
30             tesla.maxSpeedLimit = maxSpeedLimit;
31             return this;
32         }
33
34         public Tesla build() {
35             return new Tesla();
36         }
37     }
38 }
```

- a. Clasa builder, TeslaConfigurator, nu contine metode “set” pentru toate atributele
- b. Clasa Tesla nu contine metode “set” publice pentru atributele din clasa
- c. Clasa Tesla defineste atributele private si din acest motiv clasa Builder nu le poate accesa-Albert
- d. Clasa builder, TeslaConfigurator, este definita printre-o clasa statica interna clasei Tesla
- e. Constructorul default din clasa Tesla este definit public
- f. Metoda build() din clasa builder, TeslaConfigurator, este implementat gresit-Albert

10. Care din urmatoarele DP incalca Dependency Inversion Principle?

- a. Command (Alex)
- b. Decorator
- c. State
- d. Factory Method
- e. Simple Factory-Albert
- f. Strategy
- g. Observer - aLEX ZICE CA MAI MULT ASTA

11. Doresti sa reduci costurile de dezvoltare prin reutilizarea metodelor implementate in componente din alte proiecte. Ce DP iti va permite sa faci asta fara sa le modifici?

- a. Decorator
- b. Adapter
- c. State
- d. Strategy
- e. Delegation-Mete
- f. Command
- g. Facade
- h. Singleton

12. Actualizati (refactorizati) o aplicatie web. Realizati faptul ca de fiecare data cand un nou tip de utilizator este necesar trebuie sa mai adaugati un caz intr-o instructiune de selectie multipla (switch). Ce principiu SOLID este incalcat?

- a. LSP
- b. SRP
- c. ISP
- d. OCP-Albert
- e. DIP

13. Editati o aplicatie Windows Forms. Realizati fapul ca modificarea unui control definit de utilizator duce la modificarea formularului in care este definit controlul, fapt ce duce la modificarea formularului principal ce porneste formularul precedent. Ce principiu SOLID este incalcat?

- a. LSP
- b. ISP
- c. OCP-Lavi, ALEX
- d. DIP
- e. SRP

14. In clasa Student este implementata metoda setVarsta(int varsta). Stiind ca varsta acceptata este cuprinsa in intervalul (14,90] indicati valorile ce pot fi folosite pentru teste de tip Range (una sau mai multe valori).

- a. 0
- b. 15
- c. 1000
- d. 14
- e. 13 sau toate??

15. Patternul Facade promoveaza deciplarea ("weak coupling") intre subsistem si clientii sai

- a. Adevarat
- b. Fals
- c.

16. Care dintre urmatoarele enunturi descrie modelul Composite corect?

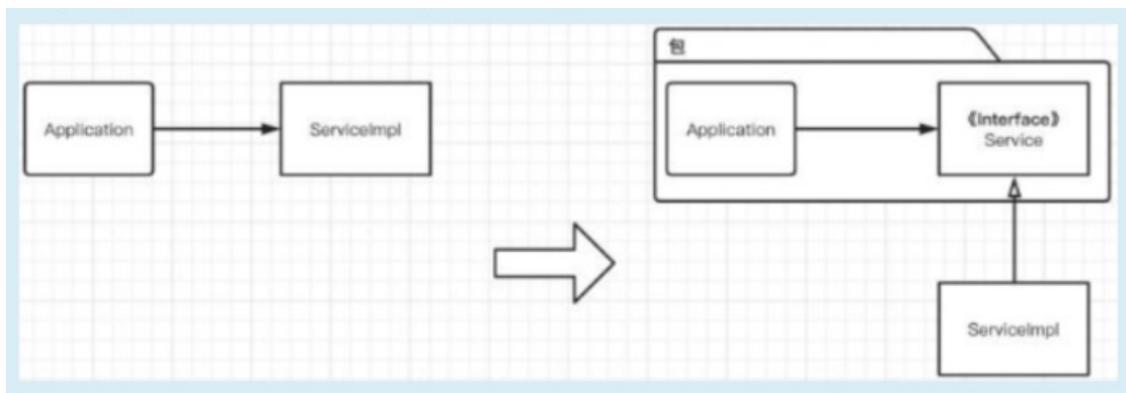
- a. Acest model este utilizat in principal pentru a reduce numarul de obiecte create si pentru a reduce amprenta de memorie
- b. Acest model permite utilizatorului sa adauge functionalitati noi la un obiect existent fara a-i modifica structura
- c. Acest model este utilizat acolo unde trebuie sa tratam un grup de obiecte in mod similar ca un singur obiect
- d. Acest model permite definirea unei familii de algoritmi, fiecare fiind encapsulat intr-un obiect. Clientul poate folosi orice program fara sa fie afectat de variația acestora.
- e. Niciuna dintre aceste afirmații - Lavi
- f. Acest model ascunde complexitatea sistemului/modului si ofera clientului o interfata pe care o poate folosi pentru a accesa/utiliza sistemul

17. Ce principiu SOLID a fost urmarit in implementarea următoare?

```
1 package ro.ase.acs.mesaje;
2 package ro.ase.acs.mesaje.service.LaRevedere
3
4
5 public class ControlMesaje {
6
7     private optiuneLaRevedere = new LaRevedere();
8
9     public String spuneLaRevedere() {
10        return "Mesaj: " + this.optiuneLaRevedere.getMessage();
11    }
12}
```

- a. Dependency Inversion
- b. Single Responsibility
- c. Open-Close
- d. Liskov Substitution
- e. Interface Segregation

18. Ce principiu de clean code a fost aplicat in imaginea următoare? O optiune:



- a. Interface Segregation
- b. Single Responsibility

- c. Liskov Principle
- d. Open-closed
- e. Dependency Inversion- Lavi**

19. Ce reprezinta dezvoltarea de tip test-driven (TDD)?

- a. Testele sunt scrise în mod normal înainte de implementarea codului
- b. TDD nu necesita scrierea testelor unitare
- c. Codul este implementat și apoi sunt scrise testele
- d. Testerii dezvoltă software folosind TDD
- e.

20. Pentru metoda din figura de mai jos sunt realizate trei teste.

Care dintre aceste teste sunt pentru verificarea corectitudinii rezultatului? Alegeti o optiune

```
public double teoremaLuiPitagora(int cateta1, int cateta2) {
    return Math.sqrt(cateta1 * cateta1 + cateta2 * cateta2);
}
```

```
@Test
public void test1() {
    int cat1=3;
    int cat2=4;
    double ipotenuza=obiect.teoremaLuiPitagora(cat1,cat2);
    assertEquals(cat1, Math.sqrt(Math.pow(ipotenuza,2)-Math.pow(cat2,2)), delta: 0.01);
}

@Test
public void test2() {
    int cat1=3;
    int cat2=4;
    double ipotenuza=obiect.teoremaLuiPitagora(cat1,cat2);
    assertEquals(ipotenuza, Math.sqrt(Math.pow(cat1,2)+Math.pow(cat2,2)), delta: 0.01);
}

@Test
public void test3() {
    int cat1=3;
    int cat2=4;
    double ipotenuza=obiect.teoremaLuiPitagora(cat1,cat2);
    assertEquals( expected: 5, ipotenuza, delta: 0.01);
}
```

- a. test 1 si test 2
- b. Niciun test
- c. Doar test 1 -- daca cere Inverse Relationship**
- d. Doar test 2**
- e. test 2 si test 3
- f. test 1 si test 3
- g. Doar test 3

h. Toate cele trei teste **Test 3 e sigur!!**

21. Testați o metoda ce depinde de o conexiune la o baza de date specificată prin interfață IConnection. Cum procedați?

Alegeti una sau mai multe opțiuni:

- a. realizati un Fake al interfeței IConnection
- b. utilizati conexiunea aferenta aplicatiei
- c. realizati un Stub al interfeței IConnection
- d. utilizati null în locul referinței respective
- e. realizati un Mock al interfeței IConnection

22. Care dintre următoarele enunțuri descrie modelul Adapter corect?

- a. Nici una dintre aceste afirmații
- b. Acest model ascunde complexitatea sistemului/modului și oferă clientului o interfață pe care o poate folosi pentru a accesa/utiliza sistemul
- c. Acest model permite definirea unei familii de algoritmi, fiecare fiind încapsulat într-un obiect. Clientul poate folosi orice algoritm fără să fie afectat de variația acestora
- d. Acest model este utilizat acolo unde trebuie să tratez un grup de obiecte în mod similar ca un singur obiect
- e. Acest model este utilizat în principal pentru a reduce numărul de obiecte create și pentru a reduce amprenta de memorie
- f. Acest model permite utilizatorului să adauge funcționalități noi la un obiect existent fără să-l modifice structura.

23. În principiile SOLID "S" înseamnă:

- a. Structural
- b. Single Responsibility
- c. Singleton
- d. Strength
- e. Standard Session

24. Care dintre următoarele variabile respectă convențiile de nume specifice Java?

- a. ceasCuBaterie, MonitorWide, intrebareexamen, limajdeProgramare
- b. documentFinal, clientFidel, limajulC#, limajdeProgramare
- c. variabilaLocala, ceasFaraBaterie, mailPrimit, formulaMatematica
- d. intrebareRetorica, dispositivMobil, peisajCuMunti, studentIntegralist
- e. ContractJuridic, PrezentarePPT, prezentareJURIDICA, fisier_editabil

25. Dezvolti o aplicatie pentru asociatiile de proprietari dintr-un cartier. Fiecare bloc are locatari, are un administrator și fiecare scara de bloc are un președinte de scara. Președintii de scara se ocupă doar de problemele locatarilor de pe scara sa, iar administratorii gestionează toate scările unui bloc. Administratorii la randul lor vor

raporta unui reprezentant al primariei aceste probleme. Sa se implementeze modulul care permite realizarea unei structuri pentru reprezentarea ierarhica a locatarilor, presedintilor de scara si a administratorilor din bloc.

In cadrul solutiei un administrator trebuie sa poata anunta toti locatarii din bloc atunci cand afiseaza la avizier costurile cu intretinerea. Sa se realizeze un modul care sa permita administratorului sa gestioneze eficient o lista de locatari pe care sa-i anunte automat atunci cand face acest lucru.

Care sunt cele 2 DP care rezolva optim aceste 2 probleme?

Factory, Chain of Responsibility, Decorator, Memento, Proxy, Prototype, Flyweight, Composite, Template, Command, State, Builder, Singleton, Adapter, Strategy, Observer

GRILE NEVERIFICATE

26. Dezvolti o aplicatie software pentru procesul de inchiriere de masini. Trebuie implementata clasa Masina. astfel incat masina sa poate fi intr-o dintre **stările**: liberă. inchiriată. necurătată. Dacă o mașină se află în starea inchiriată sau necurătată, nu poate fi inchiriată. Doar dacă este liberă poate fi inchiriată. in cazul în care este inchiriată nu poate fi trimisă la curățare. Dacă este liberă, se consideră că este deja și curătată. Se dorește implementarea unui modul care să anunțe compania de inchiriat mașini în momentul în care clientul ieșe dintr-o zonă limitată pentru mașina inchiriată. Astfel, în momentul în care mașina inchiriată doar pentru un oraș, părăsește orașul respectiv, compania este anunțată automat. Acest lucru este realizat automat, angajații companiei nefiind obligați să verifice ocazional unde se află mașinile din flotă. **State/Observer**
27. Dezvolti o solutie software care sa permita implementarea unui modul care sa poata gestiona o casa inteligenta. Modulul primește informații de la diferiti senzori instalati in casa si poate lua decizii intr-un mod independent privind incalzirea sau racirea casei, udarea plantelor, inchiderea geamurilor etc. Toate aceste actiuni sunt determinate de starea incintei. Utilizatorii nu controleaza direct modulul, acesta luand decizii in mod autonom.

In timp se doreste ca sa se permita utilizatorilor sa seteze un mod ambiental generic (cald, racoros, comfort). Acesteia vor putea seta modul generic de lucru al modulului dupa care acesta va monitoriza mediul si va lua decizii autonom, insa urmand directia data de utilizator.

Care sunt cele 2 DP ce rezolva aceste 2 probleme? **STATE, STRATEGY**

28. Dezvolti o solutie software care sa permita realizarea unei librarii pentru prelucrari grafice. Libraria contine foarte multe clase, pachete,etc care acopera o arie mare de prelucrari. Pentru a facilita integrarea librariei in alte proiecte software trebuie gasita o solutie care sa simplifice utilizarea ei pentru scenarii des intalniti. De asemenea, anumite prelucrari trebuie sa fie facute prin parcurgerea unui set predefinit de pasi ce trebuie realizata intr-o ordine fixa. Programatorii pot controla detaliile fiecarui pas

insa nu pot modifica ordinea lor de executie. Care sunt cele 2 DP? **TEMPLATE, FACADE**

29. Dezvolti o solutie software care sa permita profesorilor si studentilor sa comunice prin intermediul mesajelor in cadrul unor grupuri de discutii. In momentul inregistrarii, utilizatorii trebuie sa isi defineasca un profil definit de o serie de atribute: nume(obligatoriu), prenume (obligatoriu), tip (obligatoriu, profesor sau student) varsta (optional), gen (optional), grupa (optional), facultatea (optional), curs coordonat (optional). Gasiti o solutie care sa le permita programatorilor sa genereze eficient crearea acestor obiecte complexe. **BUILDER**

Solutia permite participantilor sa comunice prin mesaje text ce sunt transmise prin diferite grupuri. Mesajele sunt gestionate la nivel de grup (ex. WhatsApp) si transmiterea unui nou mesaj va genera afisarea acestuia prin interfata tuturor clientilor abonati la acel grup. Un client/utilizator abonat la un grup are interfața **ITransmitereMesaj**. Clientii au posibilitatea sa se aboneze la diferite grupuri si de asemenea sa iasa cand doresc dintr-un grup existent. care sunt cele 2 DP?

OBSERVER

30. NFC or bluetooth
31. Care dintre urmatoarele enunturi sunt corecte (unul sau mai multe) despre pattern-ul de tip Factory?
- Creaza
32. Pentru sistemul software al unei universitati trebuie adaugata o noua functionalitate de sustinere a unui examen de catre student la finalul unui stagiu in mod unic. Programatorii decid sa realizeze modificari in codul original. Modificările sunt semnificative pentru a accepta aceasta caracteristica. Ce principiu SOLID incalca acest lucru?
- Single Responsibility
 - Interface Segregation
 - Liskov principle
 - Dependency Inversion
 - Open-Closed
33. Dezvoltați o soluție software/plugin care să permită verificarea fișierelor descărcate de pe Internet și a link-urilor Web accesate de utilizatori prin intermediul browser-ului. Această soluție este bazată pe module ce vor verifica acțiunile utilizatorilor, module ce au interfața **IAntivirus** (enumerarea se poate muta într-o clasă separată). Modulele vor face o verificare completă a acțiunilor utilizatorului conform următoarelor condiții:
- dacă fișierul descărcat este de tip .exe (dacă numele fișierului conține .exe) se verifică dacă conține virusi, dacă da download-ul este întrerupt (se afișează mesaj), astfel se trece la următoarea verificare.
- dacă fișierul descărcat este de tip .pdf se verifică dacă conține malware, dacă da atunci downloadul este întrerupt (se afișează mesaj), astfel se trece la următoarea verificare

-dacă se accesează un link se verifică dacă este unul de tip https, dacă nu este atunci este alertat utilizatorul printr-un mesaj, în ambele situații link-ul este accesat

-ultima modul va accesa link-ul sau va descărca fișierul și va consemna această acțiune în istoricul acțiunilor (istoricul se poate simula printr-o colecție standard ce este gestionată de acest modul) **CHAIN**

În timp trebuie să se găsească o soluție prin care un modul al plugin-ului să poate fi modificat/customizat în timp real prin adăugarea de setări noi, dorite de utilizatori. Aceasta soluție trebuie să permită utilizatorului să modifice modulul în timpul execuției plugin-ului prin adăugarea de setări noi. Acestea vor afecta și modul în care funcționează modulul respectiv verificarea acțiunii utilizatorilor. Modulele pot fi modificate doar dacă utilizatorii doresc, astfel modulul va funcționa normal. Care sunt cele 2 design patternuri care rezolvă optim aceste 2 probleme? **DECORATOR**

Grile multe

34. Se dorește realizarea unei aplicații software aferentă unui automat de bilete. S-a decis faptul că persoana va selecta la început dacă dorește un bilet de tren sau unul de autocar. De asemenea, în funcție de selectii, nu toate detaliile biletului sunt obligatorii. Ce Design Patterns (unul sau mai multe) ati utiliza pentru a crește calitatea și menținabilitatea codului sursa având în vedere faptul că în acest moment crearea și tipărirea biletului pentru detaliile selectate se face în acest fel:

```
IBilet bilet = null;
if(optiune.equals("tren")) {
    bilet = new BiletTren("Popescu Ion", 100, 25, 190, false,1);
} else {
    bilet = new BiletAutocar("Popescu Ion", 100, 25, 190,false);
}
bilet.tipareste();}
```

Strategy?Builder

35. Care dintre următoarele enunțuri descrie modelul Flyweight corect?

Acest model este utilizat în principal pentru a reduce numărul de obiecte create și pentru a reduce amprenta de memorie.

36. Primăria municipiului București dorește implementarea unei aplicații software pentru gestiunea tuturor comerciantilor din cadrul orașului. Comercianții/magazinile sunt grupate la nivel de primărie de sector, urmând ca sectoarele să aparțină primăriei generale. Să se implementeze o modalitate de stocare a tuturor comerciantilor și a primăriilor din cadrul cărora acestea aparțin în scopul reprezentării ierarhice generale gestionate de municipiului. **COMPOSITE** -ierarhie de obiecte

Existând mai multe tipuri de comercianți: magazine alimentare, benzinarii, restaurante, etc, să se implementeze un model de creare de obiecte de tipul Comerçant. Modulul trebuie să ofere posibilitatea extinderii cu noi tipuri de comercianți înănd cont de principiile Solid de implementare a aplicației. **Factory Method**

37. Ce va afișa următorul test case în momentul rulării?

```
public class TestCase {  
    @BeforeClass  
    public static void setUpClass() {  
        System.out.print("A");  
    }  
    @Before  
    public void setup() {  
        System.out.print("B");  
    }  
    @After  
    public void teardown() {  
        System.out.print("");  
    }  
    @Test  
    public void test1() {  
    }  
    @Test  
    public void test2() {  
    }  
}  
Alegeți o opțiune:  
. a. ABCBC  
b. ABBCC  
C. AABBCC  
d. ABC  
e. AABC
```

ABCBC

38. Ai o clasa care acceptă și returnează valori în unități Britanice (feet, miles, etc.), dar trebuie să folosești unități metrice. Ce design pattern rezolvă aceasta problema?

ADAPTER

39. You develop a software solution that allows customers to order food through similar Glovo / FoodPanda services. Customers generate an order with the respective products. The order will be taken over by a courier available in the system. The order will be delivered in the future but the delivery time depends on several external factors (traffic, congestion in the store, etc.). **COMMAND**

Orders registered in the system can no longer be canceled and are not lost. At this moment each customer can include in the order any kind of products but the solution must allow the addition of filters/rules in the future (does not allow orders less than a certain amount, does not allow orders for certain products, adds the cost of transport to order or increases restaurant prices by an amount) that limits/controls customer options. These changes must be included in the back-end services when the customer adds a product. What are the 2 design patterns that optimally solve these 2 problems?

33. You develop a software solution for an equipment, from a medical laboratory, equipment that is used to completely automate the process of conducting blood tests. The equipment has a display that shows its current status and is able to manage the complete

analysis of a blood sample without the involvement of laboratory employees. The display permanently displays messages specific to the status of the medical equipment (waiting for the sample, loading the test tube, processing the data, displaying the results, etc.). At start-up, the equipment is on stand-by, but from the moment a test tube is loaded, the equipment goes through all the phases on its own until the result is displayed. The display also shows the equipment name. Messages are displayed as soon as the equipment changes to a new state.

Implement the module that allows the efficient management of medical equipment, given that it is autonomous and goes through several states / phases to display the final result.

It is also desired that the device software can efficiently manage different types of medical tests, which derive from the MedicalAnalysis family of classes. The solution must efficiently generate different types of analysis (Test-COVID, Cholesterol, Vitamins) being decoupled from the details of creating these types. Over time, the details of these types of medical tests may change, but this solution should not be affected. What are the 2 design patterns that optimally solve these 2 problems? **STATE, FACTORY**

34. În principiu open-closed, entitățile sunt deschise pentru extensie, dar închise pentru: **modificări**.

35. Ce design pattern permite clientului să creeze familii de obiecte fără a specifica tipul concret al claselor? f. Factory

36. Cine se asteapta să scrie Unit Tests?

- a. software testerul
- b. designerul
- c. software arhitectul
- d. clientul
- e. developerul

37. Ce design pattern ne permite să definim dependențe între obiecte, astfel încât când starea unui obiect se schimbă, toate dependințele să fie notificate și să se modifice automat? j. **Observer**

38. Cand ar trebui ca Unit Tests urile sa fie scrise in Developemnt Cycle (bazat pe TDD)?

- A. Ambele variante de mai sus
- B. Unit test sunt scrise după ce codul a fost scris ca să ajute coderii să testeze codul
- C. Niciuna dintre variante
- D. Înainte să fie scris codul de dezvoltatori pentru a îi ajuta să scrie cel mai bun cod

39. Care este cea mai mică Unitate pe care o testezi/targhezi în Unit testing?

- A. a. constantele și variabilele

- B. b. modulele si componentele
- C. c. pachetele
- D. d. clasele
- E. e. metodele si functiile

40. Se da functia pentru calculul pretului unui ticket, indica numarul minim de teste (cu diferite combinatii de valori) care trebuie implementat pentru a asigura 100% test coverage? **6**

```
public float getTicketPrice(int area, float basePrice, boolean groupReduction){  
    float finalPrice = 0;  
    switch(area){  
        case 1:  
            finalPrice = basePrice;  
        case 2:  
            finalPrice = basePrice * 1.2f;  
            break;  
        case 3:  
            finalPrice = basePrice * 1.5f;  
            break;  
    }  
  
    if(groupReduction)  
        finalPrice = (float) (0.85 * finalPrice);  
  
    return finalPrice;  
}
```

41.

75. președintii de scară

Dezvoltă o soluție software pentru asociațiile de proprietari dintr-un cartier. Fiecare bloc are locatari, are un administrator și fiecare scară de bloc are un președinte de scară. Președintii de scară se ocupă doar de problemele locatarilor de pe scară sa, iar administratorii gestionează toate scările unui bloc. Administratorii la rândul lor vor raporta unui reprezentant al primăriei aceste probleme. Să se implementeze modulul care permite realizarea unei structuri pentru reprezentarea ierarhică a locatarilor, președintilor de scară și a administratorilor de bloc.

În cadrul soluției un administrator trebuie să poată anunța toți locatarii din bloc atunci când afișează la avizier costurile cu întreținere. Să se realizeze un modul care să permită administratorului să gestioneze eficient o listă de locatari pe care să-i anunțe automat atunci când face acest lucru.

Care sunt cele 2 design patternuri care rezolvă optim aceste 2 probleme ?

R: composite & observer

74. meseriași

Dezvoltă o soluție software pentru asociațiile de proprietari dintr-un cartier de blocuri. În cadrul cartierului sunt mai multe meseriași (instalaitori, electricieni, etc) care repară și îndreaptă lucrurile din tot cartierul. Atunci când apare o problemă, locatarii solicită intervenția acestora și solicitarea lor este salvată într-o coadă de așteptare, iar atunci când unul dintre meseriași cartierului este liber, va prelua această problemă și o va rezolva. Să se implementeze modulul care permite gestionarea cozi de comenzi de probleme către meseriași cartierului.

De asemenea, se dorește implementarea unei soluții care să permită locatarilor unui bloc să aleagă modul în care sunt publicate anunțurile din bloc. Asociația poate decide în cadrul ședințelor anuale ca anunțurile să fie printate la avizier sau să fie trimise prin email sau prin sms, etc. Odată soluția aleasă este făcută, trebuie să se mențină pana la ședința următoare.

Care sunt cele 2 design patternuri care rezolvă optim aceste 2 probleme ?

R: Command & Strategy

73.112, sistem național de alerte

Dezvoltă o soluție software pentru asociația de proprietari dintr-un bloc care să permită gestiunea eficientă a evenimentelor ce pot apărea, evenimente care necesită luarea unor măsuri adecvate și care să protejeze viața locatarilor. Soluția procesează diferite mesaje/evenimente primite de la senzori/sisteme din bloc și va lua următoarele decizii:

- dacă s-a declanșat alarmă de incendiu atunci este notificat serviciul 112
- dacă este anunțată o inundație atunci este notificat administratorul
- dacă este doar o informare atunci sunt notificați locatarii
- (optional) pentru alarmele de incendiu și inundație sunt anunțați și locatarii

Independent de implementarea scenariului anterior găsiți o soluție care să permită integrarea sistemului existent de procesare a alertelor (cu interfața IProcesareEveniment) în sistemul național de alertare ce este construit în jurul interfeței IAlertarePublică.

Care sunt cele 2 design patternuri care rezolvă optim aceste 2 probleme ?

R: Chain & Adapter

72. Întreținere, cameră video

Dezvoltă o soluție software pentru asociațiile de proprietari dintr-un cartier de blocuri care să le permită să gestioneze detalii legate de costurile de întreținere aferente unei luni pentru fiecare proprietar de apartament. Aceste date (numărul apartamentului, costul apel calde, costul calduril, costul gazelor, etc.) sunt stocate în fiecare lună. Soluția trebuie să permită salvarea acestor date la finalul lunii astfel încât să poată oferi în orice moment informații cu privire la istoricul acestor costuri pe fiecare apartament.

Independent de cerința anterioară, blocul are la intrare un interfon prin care este solicitat/controlat accesul. Deoarece copiii din cartier abuzează de acest serviciu și deranjează locatarii, se dorește adăugarea în sistem a unei camere video care să verifice dacă apelantul este adult sau copil. Această cameră va fi conectată la sistemul existent (nu se modifică interfonul) și va controla dacă se poate fi folosit interfonul sau nu. Dacă camera detectează un copil atunci interfonul nu va suna la apartamentul apelat.

Care sunt cele 2 design patternuri care rezolvă optim aceste 2 probleme ?

R: Memento & Proxy

71. Barieră

You develop a software solution that allows the implementation of an autonomous barrier that treats differently the situations specific to a railway crossing. The barrier is not controlled from the outside. Depending on the state of some sensors the barrier implements specific actions. The barrier is normally raised and it starts the audible warning when the approach of the train is identified. After 10 seconds the barrier lowers automatically. Also, the messages displayed by the barrier differ depending on the state of the barrier.

To avoid situations where the barrier is not raised after the train has passed or is raised even though the train is still in the area, it is desired to add to the system an external sensor to make additional checks. The existing sensor will notify the new sensor instead of transmitting the information directly to the barrier but the proposed solution should not involve modifying the current system. Find a solution that allows you to add the new component but keep the current solution in use.

What are the 2 design patterns that optimally solve these 2 problems?

R: State & Adapter

70. Dezvoltați o soluție software pentru editarea imaginilor de tip Bitmap destinată copiilor. Soluția trebuie să le permită utilizatorilor să vadă istoricul modificărilor și să le permită revenirea la un moment în timp ales de ei.

Fiind o aplicație pentru copii, soluția le permite să înceapă să deseneze de la o colecție predefinată de template-uri. Aplicație trebuie să permită eficiența managementului și creării acestor template-uri diferite. Fiecare template e alb&negru generat cu resurse minime.

R: Memento & Flyweight

69. Dezvoltați o soluție care să permită călătorilor să aleagă eficient o combinație de mijloace de transport care să le permită să ajungă la destinație. Soluția va sugera o soluție bazată pe preferințele lor (cursă directă, costurile etc) și pe disponibilitatea transportului din apropiere. Orașul este împărțit în zone de acces, astfel dacă călătorul vrea să călătorească în aceeași zonă (maxim 5km) sugerează autobuzul, dacă vrea în zona 2 (maxim 10 km) sugerează tramvaiul, zona 3 metroul, în afara orașului sugerează trenul. Implementați soluția care permite călătorilor să aleagă transportul pe care îl doresc bazat pe criteriile lor. Soluția software sugerează o soluție de fiecare dată când este introdusă proximitatea.

De acum, soluția le permite pasagerilor să plătească bilet online. Cum la anumite căi de transport (de ex metrou) este permisă plata doar fizică, soluția trebuie să permită în viitor plată NFC sau funcția bluetooth a telefonului să fie folosită și operatorii să ofere interfață specifică pentru a taxa.

R: Chain & Adapter

62. Dezvoltați o soluție care permite cumpărătorilor să comande mâncare similar cu Glovo/Panda. Cumpărătorii generează o comandă cu respectivele produse. Comanda va fi luată de un curier disponibil în sistem. Comanda va fi livrată în viitor, dar timpul de livrare depinde de factori externi(trafic etc). Comenzile înregistrate în sistem nu pot fi anulate.

La acest moment, fiecare cumpărător poate include în comandă orice fel de produse, dar soluția trebuie să permită filtre adiționale în viitor (nu permite comenzi sub o anumită sumă, cost de transport etc). Aceste schimbări trebuie să fie incluse în back-end atunci când cumpărătorul adaugă un produs.

R: Command & Decorator

61. Dezvoltați o soluție care permite eficiența managementului diferitelor adrese de email ale clientilor. Email clients trebuie să fie capabili să trimită și să primească emailuri pentru o adresă de mail specifică, dar de asemenea pot fi folosite pentru manage group emails cu

multe adrese de email. Utilizatorii ar trebui să fie capabili să își gestioneze structura de la soluție.

Utilizatorii au abilitatea de a modifica/customiza email client în timpul folosirii. Soluția propusă trebuie să îl lase pe clienti să modifice/customizeze soluția adăugând filtre sau criterii. De asemenea, transmisia funcției mesajelor poate fi modificată adăugând în plus o confirmare sau să întârzie transmiterea sau să specifice orele specifice.

R:Composite & Decorator

60. Dezvoltați o soluție care permite eficiența managementului unei situații școlare a mii de elevi cu același profil. Principala problemă este reducerea spațiului ocupat de date în RAM. Soluția trebuie să permită a lua în considerare informațiile (discipline, obiective, topicuri) în ceea ce privește cursurile următoare de studenți sunt comune.

Soluția trebuie să permită a da examenele. Pot fi orale, quiz, întrebări cu răspuns liber etc. Modul în care se desfășoară examinarea este stabilit de profesorul coordonator în concordanță cu opțiunile profesorilor care coordonează același curs.

R: Flyweight & Strategy

51. Dezvoltați o soluție care să permită profesorilor și studenților să comunice prin intermediul mesajelor în cadrul unor grupuri de discuții. În momentul înregistrării, utilizatorii trebuie să își definească un profil definit de o serie de atribute: nume (obligatoriu), prenume (obligatoriu), tip (obligatoriu, prof sau student), vârstă (opt), gen (opt), grupa (opt), fac (opt), cursul coordonat (opt). Găsiți o soluție care să le permită programatorilor să gestioneze eficient crearea acestor obiecte complexe.

Soluția permite participanților să comunice prin mesaje text ce sunt transmise pe diferite grupuri. Mesajele sunt gestionate la nivel de grup (Whatsapp) și transmiterea unui nou mesaj va genera afișarea acestuia pe interfața tuturor clientilor abonați la acel grup. Clienții au posibilitatea să se aboneze la diferite grupuri și de asemenea să iasă când doresc.

R: Builder & Observer

20.

Dezvoltați o soluție software pentru editarea imaginilor de tip Bitmap destinată copiilor. Soluția trebuie să le permită utilizatorilor să vadă istoricul modificărilor și să le permită revenirea la un moment în timp ales de ei. Imaginele ce pot fi procesate în cadrul aplicației sunt de tip ImagineBitmap. Soluția software trebuie să permită stocarea salvărilor și trebuie să permită utilizatorului să le vadă și să le recupereze în funcție de data salvării.

În timp soluția aceasta este achiziționată de o altă companie care are un produs similar, dar care permite procesarea de imagini de tip PhotoImage. Ca parte a vânzării trebuie găsită o soluție care să permită utilizarea imaginilor de tip Bitmap în noul context.

R: Memento&Adapter

21.

Dezvoltați o soluție software care să permită implementarea unui modul care să poată gestiona o casă inteligentă. Modulul primește informații de la diferiți senzori instalati în casă și poate lua decizii într-un mod independent privind încălzirea sau răcirea casei, udarea plantelor, închiderea geamurilor etc. Toate aceste acțiuni sunt determinate de starea incintei. Utilizatorii nu controlează direct modulul, acesta luând decizii în mod autonom.

În timp se dorește ca să se permită utilizatorilor să seteze în mod ambiental generic (cald, răcoros, confort etc). Aceștia vor putea seta modul generic de lucru al modulului după care acesta va monitoriza mediul și va lua decizii autonom, însă urmând direcția dată de utilizator.

R: State&Strategy

22.

Dezvoltați o soluție software care să permită realizarea unei librării pentru prelucrări grafice. Librăria conține foarte multe clase, pachet etc care acoperă o arie mare de prelucrări. Pentru a facilita integrarea librăriei în alte proiecte software trebuie găsită o soluție care să simplifice utilizarea ei pentru scenarii des întâlnite.

De asemenea, anumite prelucrări trebuie făcute prin parcurgerea unui set predefinit de pași ce trebuie realizată într-o ordine fixă. Programatorii pot controla detaliile fiecărui pas însă nu pot modifica ordinea lor în execuție.

R: Facade&Template

23.

Dezvoltați o soluție software folosită de un laborator medical pentru a gestiona cererile privind analiza unor mostre de material biologic (analize de sânge). Laboratorul primește de la diferite cabinete medicale cererile privind realizarea de diferite analize, pe care le salvează într-o coadă. Analizele sunt realizate de diferite departamente medicale ce au echipamente speciale pentru fiecare tip de analiză. Laboratoarele solicită aceste analize indicând tipul lor și numele pacientului. În momentul în care o solicitare de analiză este înregistrată în sistem, aceasta este asociată departamentului specializat în acel tip de analiză. În momentul în care este obținut rezultatul analizei, departamentul care a procesat analiza va trimite rezultatul pe email-ul pacientului (email-ul și numele pacientului sunt înregistrate în solicitare).

De asemenea se dorește ca în soluție, managerul care se ocupă de coada de comenzi sau de lista de clienți să fie unic, astfel încât să nu genereze probleme legate de modul în care sunt gestionati clienții sau analizele medicale.

R: Command & Singleton

26.

Dezvoltați o soluție software de tip trading care să permită clienților să cumpere acțiuni pe diferite burse prin intermediul unor brokeri. Clienții pot comanda cumpărarea sau vânzarea unui număr de acțiuni pentru anumite companii listate la bursă. Clienții își aleg brokerul în momentul în care lansează comanda în sistem. Comenzile sunt înregistrare în sistem și vor fi procesate în ordinea în care au fost generate de platforme de trading, în funcție de gradul de încărcare al platformei la momentul în care comanda este procesată și executată de broker.

În timp platforma se va extinde și vor fi diferite tipuri de servicii financiare: schimb valutar, vânzarea sau cumpărarea de criptomonede, investiții etc. Aceasta dezvoltare trebuie susținută de o soluție care să permită extinderea tipurilor de servicii.

R: Command & Decorator

37.

Dezvoltați o soluție software pentru un echipament, dintr-un laborator medical, echipament ce este utilizat pentru a automatiza complet realizarea de analize de sânge. Echipamentul are un display pe care e afișată starea curentă și este capabil să gestioneze analiza completă a unei mostre de sânge, fără implicarea angajaților laboratorului. Pe display sunt afișate în permanență mesaje specifice stării echipamentului medical (în aşteptare mostră, încărcare eprubetă, procesare date, afișare rezultate etc). La pornire, echipamentul este în aşteptare, însă din momentul în care este încărcată o eprubetă, echipamentul trece singur prin toate fazele până când e afișat rezultatul. Pe display este afișat și numele echipamentului (un atribut la alegere). Mesajele sunt afișate imediat ce echipamentul trece într-o stare nouă.

Să se implementeze modului care permite gestiunea eficientă a echipamentului medical, având în vedere că este unul autonom și că trece prin mai multe stări/faze pentru a afișa rezultatul final.

De asemenea, se dorește ca software-ul aparatului să poată gestiona eficient diferite tipuri de analize medicale, ce provin dintr-o familie de tipul AnalizaMedicala. Soluția trebuie să genereze eficient diferite tipuri de analize (Test Covid, Colesterol, Vitamine) fiind decuplată de detaliile creării acestor tipuri. În timp detaliile aferente acestor tipuri de analize medicale se pot modifica, dar soluția aceasta nu trebuie să fie afectată.

R: State & Factory

48.

Dezvoltă o soluție software/plugin care să permită verificarea fișierelor descărcate de pe Internet și a link-urilor Web accesate de utilizatori prin intermediul browser-ului. Această soluție este bazată pe module ce vor verifica acțiunile utilizatorilor, module ce au interfață lAntivirus (enumerarea se poate muta într-o clasa separată). Modulele vor face o verificare completă a acțiunii utilizatorului conform următoarelor condiții:

- dacă fișierul descărcat este de tip .exe (dacă numele fișierului conține .exe) se verifică dacă conține virus; dacă da atunci download-ul este întrerupt (se afișează mesaj), altfel se trece la următoarea verificare;
- dacă fișierul descărcat este de tip .pdf se verifică dacă conține malware; dacă da atunci download-ul este întrerupt (se afișează mesaj), altfel se trece la următoarea verificare;
- dacă se accesează un link se verifică dacă este unul de tip https; dacă nu este atunci este alertat utilizatorul printr-un mesaj; în ambele situații link-ul este accesat;
- ultimul modul va accesa link-ul sau va descărca fișierul și va consemna această acțiune în istoricul acțiunilor (istoricul se poate simula printr-o colecție standard ce este gestionată de acest modul)

În timp trebuie să se găsească o soluție prin care un modul al plugin-ului să poată fi modificat/customizat în timp real prin adăugarea de setări noi, dorite de utilizatori. Această soluție trebuie să permită utilizatorului să modifice modulul în timpul execuției plugin-ului prin adăugarea de setări noi. Acestea vor afecta și modul în care funcționează modulul respectiv verificarea acțiunii utilizatorilor. Modulele pot fi modificate doar dacă utilizatorii doresc, altfel modulul va funcționa normal.

Care sunt cele 2 design pattern-uri care rezolvă optim aceste 2 probleme ?

R: Chain & Decorator

1. Fiind data funcția următoare, indicați care este numărul minim de teste unitare (cu diferite combinații de valori) ce trebuie implementate pentru a asigura code coverage de 100%?

```

public int calcul(int valoare){
    int rezultat = 0;

    if(valoare > 0)
        rezultat = 100;
    else
        rezultat = 200;

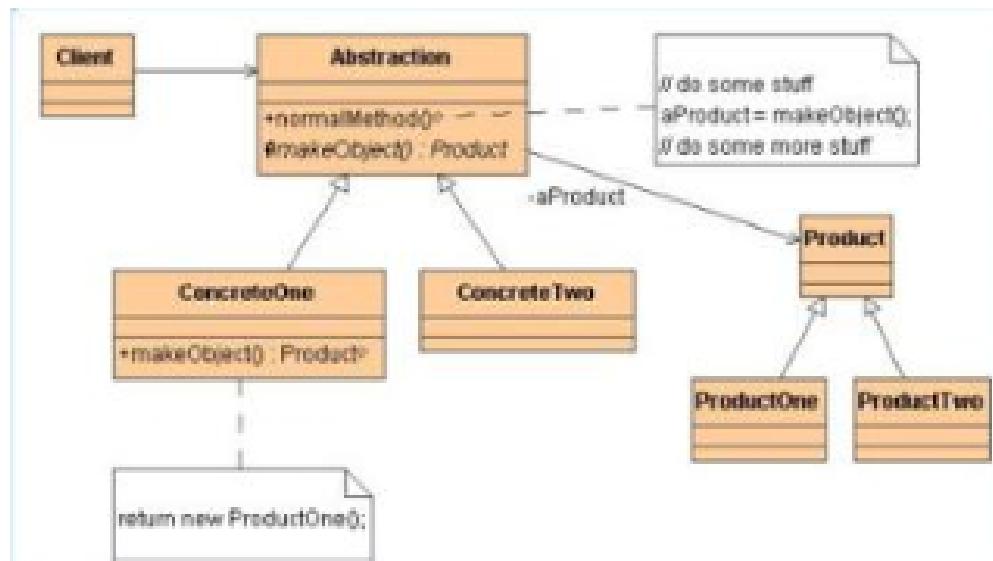
    if(valoare > 100)
        rezultat += 5;
    else
        rezultat += 10;

    return rezultat;
}

```

- a. 3
- b. 6
- c. 2
- d. 5

2. Fiind data diagrama urmatoare ce reprezinta o solutie ce implementeaza un Design Pattern, indicati care este acesta? **Factory**



3. Fiind data functia pentru initializarea varstei si urmatoarele restrictii:

- zona ia valori in intervalul (9,100] (interval deschis la stanga)

Indicați care dintre următoarele teste unitare sunt considerate de tip BOUNDARY pentru valoarea primită (selectați unul sau mai multe răspunsuri)?

```
public void setVarsta(int valoare){  
    int varsta = valoare;  
    System.out.println("Varsta primită este " + varsta);  
}
```

Alegeți una sau mai multe opțiuni:

- a. setVarsta(101)
- b. setVarsta(100)**
- c. setVarsta(8)
- d. setVarsta(10)**
- e. setVarsta(9)
- f. setVarsta(99)

4. Cum este implementat Adapter-ul de obiecte?

- a. Clasa Adapter mosteneste clasa existenta și implementează interfața la care trebuie să facă adaptarea
- b. Clasa Adapter conține o instanță a clasei existente și implementează interfața la care trebuie să facă adaptarea**
- c. Niciun răspuns corect
- d. Clasa Adapter conține instanțe ale celor două clase existente

5. Care dintre următoarele variante reprezinta contextul în care se utilizează Observer?

Alegeți o opțiune:

- a. Baze de date
- b. DNS Resolver
- c. Niciuna dintre variante
- d. Model View Controller**

6. Participanții din cadrul Proxy:

Alegeți o opțiune:

- a. Niciuna dintre variante
- b. O interfață+Clasa concreta Proxy
- c. O interfață+Clasa concreta Entitate+Clasa concreta Proxy**
- d. O interfață+Clasa abstractă Entitate+Clasa abstractă Proxy

7. Care este partea componentă care se ocupă de faptul că poate fi creat un singur obiect al clasei Singleton?

- a. Niciun răspuns corect

- b. Utilizatorul
- c. Apelatorul
- d. Clasa

8. Care dintre urmatoarele afirmatii este adevarata?

Alegeți o opțiune:

- a. Facade este folosit pentru framework-uri open-source disponibile
- b. Clasa Facade ascunde complexitatea prin apelurile sale
- c. Toate variantele sunt adevărate
- d. Clasa Facade cuprinde metode care utilizează metodele în clasele framework-ului

9. Ce reprezinta numele Chain of Responsibility?

Alegeți o opțiune:

- a. Aceste obiecte posibile se ordonează într-un lanț, apoi cel care are problema de rezolvat apelează pentru prima vezin lant
- b. Este folosit atunci cand cel care are nevoie de rezolvarea unei probleme are o lista de posibile obiecte ce pot rezolva problema
- c. Toate variantele
- d. Este folosit atunci cand cel care are nevoie de rezolvarea unei probleme nu știe exact cine poate să rezolve problema

10. Ce reprezinta numele Template Method?

Alegeți o opțiune:

- a. Este folosit atunci cand un algoritm este cunoscut și urmează anumiți pași precisi
- b. Toate variantele
- c. Nu există nicio metodă care să implementează algoritmul și se apelează toate celelalte metode
- d. Toți pașii sunt realizati de aceeași metodă

11. Participantii din cadrul State:

Alegeți o opțiune:

- a. Niciuna dintre variante
- b. O interfață+Clase concrete ce definesc stările obiectului+Clasa concreta ce definește obiectul
- c. O interfață+Clase concrete ce definesc obiecte în diferite stări
- d. O interfață+Clase abstracte

12. Facade este util când:

Alegeți o opțiune:

- a. Se dorește adăugarea de funcționalități
- b. Niciuna dintre variantele de mai sus

- c. Se dorește crearea de obiecte cu multe proprietăți opționale
- d. Se dorește simplificarea unui proces

13. Ce reprezinta participantul Invoker din design pattern-ul Command?

Alegeți o opțiune:

- a. Interfața care definește la nivel abstract comenzi sau acțiunile
- b. Obiectul responsabil cu execuția acțiunilor
- c. Clasele concrete pentru fiecare comandă
- d. Clasa care se ocupă cu gestiunea comenzielor

14. Care este design pattern-ul care nu este prezentat în cartea GoF?

Alegeți o opțiune:

- a. Simple Factory
- b. Builder
- c. Prototype
- d. Singleton

15. Ce tip de design pattern este Proxy?

Alegeți o opțiune:

- a. Structural
- b. Niciuna dintre variantele de mai sus
- c. Comportamental
- d. Creational

16. Ce reprezinta Strategy?

Alegeți o opțiune:

- a. Alegerea implementării se face la compilare
- b. Definește strategia adoptată la run-time
- c. Toate comportamentele sunt date de aceeași clasa
- d. Este folosit atunci cand avem un singur algoritm pentru rezolvarea unei probleme

17. Pentru ce tip de obiecte este recomandata folosirea design pattern-ului Prototype?

Alegeți o opțiune:

- a. obiecte a căror construire consumă foarte multe resurse
- b. Ambele variante sunt corecte
- c. obiecte a căror construire durează foarte mult

18. Care dintre următoarele afirmații despre Factory Method este adevărată?

Alegeți o opțiune:

- a. Toate variantele sunt corecte
- b. Poate fi găsit și sub denumirea de Virtual Constructor
- c. Pentru apeluri se folosesc abstractizări, nu obiecte concrete

d. Nu folosește structuri switch sau if-else

19. Ce dp permite construirea în pași? R: Builder

20. e sus

21. e sus

22. e sus

23. e sus

24. Ce dp permite unui client să creeze fam de obiecte fără a specifica clasele concrete din acestea? R: Factory

25. Pentru a testa funcția și pt a asigura code coverage de 100%, care este nr minim de teste unitare cu diferite valori pe care trebuie să le implementezi?

```
public static int calculeazaCeva(int valoare1, int valoare2) {
    int rezultat = 0;

    if(valoare1 > 0 || valoare2 < 0) {
        rezultat = 100;
    }

    switch(valoare1) {
        case 10:
            rezultat += 10;
            break;
        case 20:
            rezultat += 20;
            break;
        case 30:
            rezultat += 30;
            break;
        default:
            rezultat += 50;
    }

    return rezultat += valoare2;
```

R:4

26. e sus

27. Ce dp încapsulează o solicitare/task ca obiect, urmând ca acesta să fie executată mai târziu? R: Command

28. Pt următoarea metodă, ce afirmație este corectă pt a asigura un code coverage de 100% prin testarea ei?

```

int oMetoda(int valoare) throws Exception {

    int rez = 0;

    switch (valoare) {

        case 1:
            rez = 10;
            break;

        case 5:
            rez = 20;
            break;
        case 10:
            rez = 30;
            break;

        default:
            throw new Exception("Valoare gresita");
    }

    return rez;
}

```

- a. Nu se poate asigura code coverage 100% pt această metodă, deoarece generează excepție
- b. Se poate asigura code coverage 100% dacă se testează valorile 1, 5, 10 și 20**
- c. Se poate asigura code coverage 100% dacă se testează valorile 1, 5, 10
- d. Nu se poate asigura un code coverage de 100% pt această metodă deoarece switch-ul are multe ramuri
- e. Se poate asigura code coverage 100% dacă se testează valorile 1 și 5.

29. Ai de testat o funcție ce determină minimum dintr-un sir de valori primit ca parametru de tip int [] valori. Pentru a realiza teste unitare (diferite între ele) de tip Cardinality = 2 și Ordering, ce date de teste ai folosi? Alege una sau mai multe.

- a. 9, 10, 8
- b. 10
- c. 9,10**
- d. 10, 9, 6, 4
- e. 7, 8, 9, 10**
- f. 7, 9, 5
- g. 10, 9, 10, 9, 9

30. Pentru a testa funcția următoare și pentru a asigura code coverage 100%, care este numărul minim de teste unitare (cu diferite valori) pe care trebuie să le implementezi?

```
public static int calculeazaCeva(int valoare1, int valoare2) {
    int rezultat = 0;

    if(valoare1 > 0 || valoare2 < 0) {
        rezultat = 100;
    }

    switch(valoare1) {
        case 10:
            rezultat += 10;
            break;
        case 20:
            rezultat += 20;
            break;
        case 30:
            rezultat += 30;
            break;
        default:
            rezultat += 50;
    }

    return rezultat += valoare2;
```

- a. 4
- b. 5
- c. 6
- d. 1
- e. 3
- f. 2

31. Ce dp permite definirea de dependențe între obiecte, astfel încât, atunci când un obiect își schimbă starea, toți dependentii săi sunt notificați și actualizați automat?

R: Observer

32. Se consideră următorul TestCase JUnit4. Ce se afișează în urma execuției acestuia?

```

public class TestCaseExamen {

    @Before
    public void before () { System.out.print("Before ");}

    @Test
    public void test1() { System.out.print("Test1 ");}

    @Test
    public void test2() { System.out.print("Test2 ");}

    @AfterClass
    public static void after() { System.out.print("After ");}

}

```

- a. Before Test1 Before Test2 After
- b. Before Test1 After Before Test2 After
- c. Before Test1 After Test2 After
- d. Before Test1 After Test2 After
- e. Before Test1 Test2 After

33. Ai o clasă care acceptă și returnează valori în unități imperiale britanice (feet, miles), dar trebuie să o folosești pentru unități în sistem metric. Ce dp rezolvă problema?

- a. Prototype
- b. Observer
- c. Singleton
- d. Flyweight
- e. State
- f. Adapter
- g. Proxy
- h. Builder
- i. Memento
- j. Facade
- k. Chain
- l. Command
- m. Strategy
- n. Factory
- o. Composite
- p. Decorator

34. Ai de testat o metodă setVarsta() din clasa Student pentru care se știe că acceptă valori în intervalul (9, 100) (deschis la ambele capete). Dacă dorești să implementezi

teste de tip RANGE pentru care aștepți o excepție de tipul ValoareEronataException cu care dintre valorile următoare ai testa metoda? (una sau mai multe variante)

- a. 10
- b. 100
- c. 9
- d. 0
- e. 25
- f. -10
- g. 99

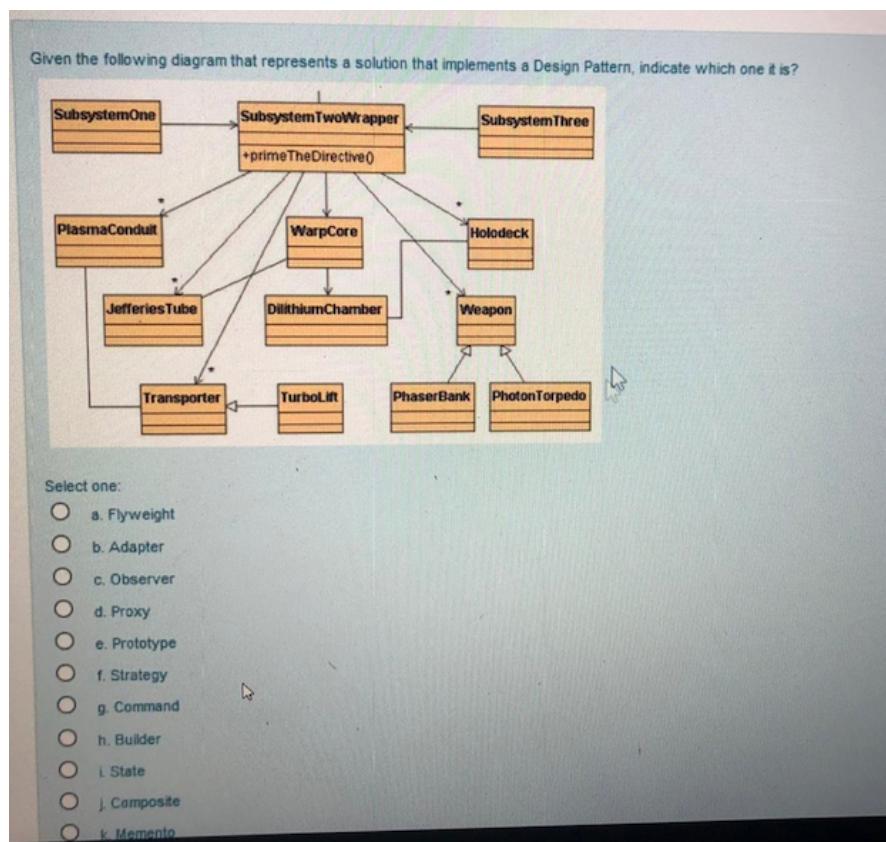
35. Ce dp oferă capacitatea de a restabili un obiect la starea anterioară? R: Memento

36. Ce dp incapsulează comportamentele dependente de stare și folosește delegarea pentru a comuta între acestea? R: State

37. e sus

38. Ce design pattern permite notificarea obiectelor atunci când are loc un eveniment? R: Observer

39. Dată fiind următoarea diagramă, ce dp e? R: FACADE



40. Dată fiind funcția pentru a calcula prețul unui bilet și următoarele restricții:

- Aria ia următoarele valori 1,2,3
- prețul de bază e pozitiv și mai mic decât 1000.

Indicați care din metode poate fi folosit pt Oder test? Niciun parametru

```
public float calculPretBilet(int zona, float pretBaza, boolean reducereGrup){
    float pretFinal = 0;
    switch(zona){
        case 1:
            pretFinal = pretBaza;
        case 2:
            pretFinal = pretBaza * 1.2f;
            break;
        case 3:
            pretFinal = pretBaza * 1.5f;
            break;
    }

    if(reducereGrup)
        pretFinal = (float) (0.85 * pretFinal);

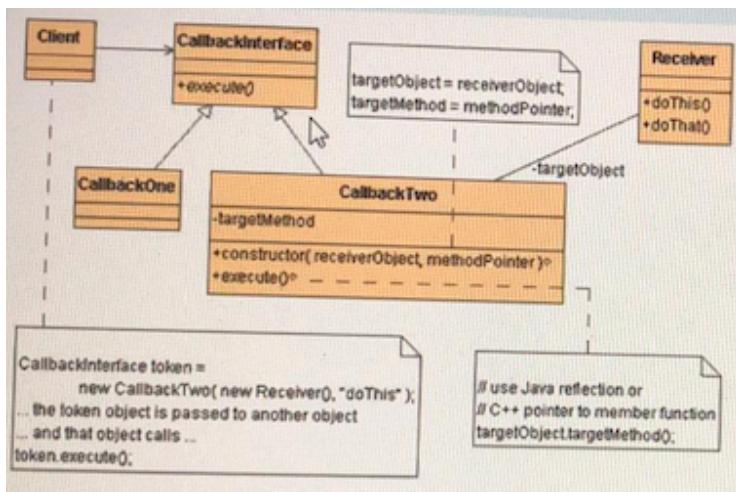
    return pretFinal;
}
```

41. Care din următoarele enunțuri descrie modelul Adapter corect?

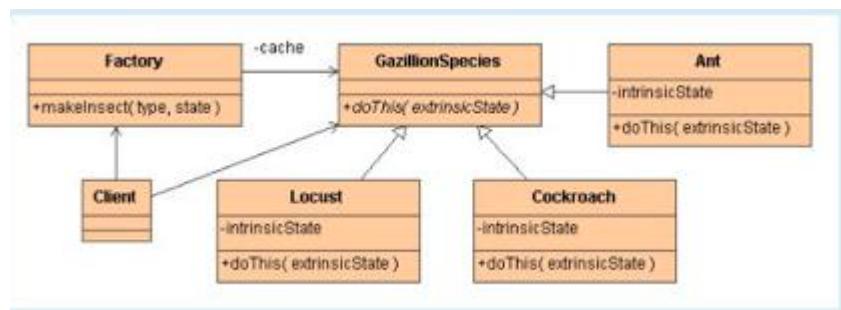
a. Niciuna dintre variante

- Acest model permite definirea unei familii de algoritmi, fiecare fiind încapsulat într-un obiect. Clientul poate folosi orice algoritm, fără să fie afectat de variația acestora.
- Acest model permite utilizatorului să adauge funcționalități noi la un obiect existent fără a-i modifica structura
- Acest model e utilizat acolo unde trebuie să tratăm un grup de obiecte în mod similar ca un singur obiect
- Acest model ascunde complexitatea sistemului/modulului și oferă clientului o interfață pe care o poate folosi pentru a accesa/utiliza sistemul
- Acest model este utilizat în principal pentru a reduce numărul de obiecte create și pentru a reduce amprenta de memorie.

42. Dată fiind următoarea diagramă, ce dp e? R:COMMAND



43. Ce diagramă e? **FLYWEIGHT**



44. Care dp implementează principiul Hollywood - Don't call us, we'll Call U? **TEMPLATE**

45. Fiind date semnăturile unor funcții ce fac diferite prelucrări, indicați pentru care dintre acestea se justifică teste de tip Ordering?

- int calculSuma (int valoare1, int valoare2)
- int calcul(int valoare)
- int calcul (ArrayList <Integer>valori)**
- int calcul (int [] valori)**
- Nu există acest tip de testare

46. Fiind dată clasa următoare, indicați cu ce valori se poate implementa un test unitar de tip Cardinalitate =0? **R: C (lista goală)**

```
public float prelucrareValori(ArrayList<Integer> valori){  
    float rezultat = 0;  
  
    for(int valoare : valori) {  
        if(valoare %2 == 0)  
            rezultat += valoare;  
    }  
  
    return rezultat;  
}
```

Alegeți o opțiune:

- a. Nu se poate implementa un astfel de test
- b. ArrayList<Integer> input = new ArrayList<>();
 input.add(1);
 input.add(3);
- c. ArrayList<Integer> input = new ArrayList<>();
- d. ArrayList<Integer> input = new ArrayList<>();
 input.add(10);
- e. ArrayList<Integer> input = new ArrayList<>();
 input.add(10);
 input.add(20);
- f. ArrayList<Integer> input = null;

47. Dacă se consideră funcția următoare, indicați cu ce valori se poate implementa un test unitar de tip Existence pentru care este așteptată și o excepție? R: F(cea cu null)

```
public float prelucrareValori(ArrayList<Integer> valori){  
    float rezultat = 0;  
  
    for(int valoare : valori) {  
        if(valoare %2 == 0)  
            rezultat += valoare;  
    }  
  
    return rezultat;  
}
```

Alegeți o opțiune:

- a.

```
ArrayList<Integer> input = new ArrayList<>();
    input.add(1);
    input.add(3);
```
- b. Nu se poate implementa un astfel de test
- c.

```
ArrayList<Integer> input = new ArrayList<>();
```
- d.

```
ArrayList<Integer> input = new ArrayList<>();
    input.add(10);
    input.add(20);
```
- e.

```
ArrayList<Integer> input = new ArrayList<>();
    input.add(10);
```
- f.

```
ArrayList<Integer> input = null;
```

48. sus

49.

```
public class GeneratorStudenti {
    private static HashMap<Integer, IStudent> studenti = new HashMap<Integer, IStudent>();

    public GeneratorStudenti() {
        studenti = new HashMap<Integer, IStudent>();
    }

    public static IStudent getStudent(int nrMatricol, String nume) {
        IStudent student=studenti.get(nume);
        if(student==null) {
            student=new Student(nrMatricol, nume);
            studenti.put(nrMatricol, student);
        }
        return student;
    }
}
```

Alegeți una sau mai multe opțiuni:

- a. Este un exemplu de lazy instantiation pentru crearea studentilor
- b. Este o implementare corecta de Factory Method
- c. Pentru a fi o implementare de Singleton, este obligatorie o metoda getInstance()
- d. Este un exemplu de eager instantiation pentru crearea Studentilor
- e. Este o implementare corecta de Prototype
- f. Este o implementare corecta de Singleton
- g. Este o implementare corecta de Simple Factory



A
G

R: Flyweight (dacă e printre rasp), dacă nu Factory cu Lazy

50. „Identifice aspectele aplicației care variază și le separi de partea comună”

R: Mai multă flexibilitate în sistem

51. Care e cel mai mic Unit pe care îl testezi/targetezi în testele Unitare?

- a. Constante și variabile
- b. Module și componente
- c. Pachete
- d. Metode și funcții**
- e. Clase

52. Funcția DoSomething(..) are 10 argumente. Ce poate fi făcut pentru a reduce numărul lor?

- a. Encapsularea logică a parametrilor
- b. Folosirea Builder Pattern pentru a manageui mai eficient
- c. Clasă și constructor
- d. Separarea funcției și ia rezultatul sub părții ca input**
- e. Păstrăm metoda și scriem comentarii pentru a explica semnificația parametrilor.

53. Dată funcția pentru a calcula prețul unui bilet, indicați numărul minim de teste unitare (cu diferite valori) care trebuie implementate pentru 100% code coverage.

```
public float getTicketPrice(int area, float basePrice, boolean groupReduction){  
    float finalPrice = 0;  
    switch(area){  
        case 1:  
            finalPrice = basePrice;  
        case 2:  
            finalPrice = basePrice * 1.2f;  
            break;  
        case 3:  
            finalPrice = basePrice * 1.5f;  
            break;  
    }  
  
    if(groupReduction)  
        finalPrice = (float) (0.85 * finalPrice);  
  
    return finalPrice;  
}
```

- a. 2**
- b. 5
- c. 6
- d. 1
- e. 4
- f. 3

54. Cine scrie testele unitare?R: developerul (programatorul)

55. Care din următoarele clase e un Composite of Testes?R: TestSuite

56. În Liskov substitution, principiul, „obiectele dintr-un program ar trebui să fie înlocuite cu instanțele”:

- a. Cod
- b. Părinte
- c. Subtipuri**
- d. Programe
- e. Command

57. Spaghetti Code:

- a. descrie o soluție cu relații dependențe complexe

- b. e cunoscut ca și „big ball of mud”
- c. poate fi evitat cu ajutorul dp
- d. poate fi evitat prin scrierea unor funcții mici (1-2 argumente)
- e. nu există acest concept.

58. Când ar trebui să fie scrise testele TDD? R: înainte de cod, în timpul development.

59. Dată fiind funcția pentru a calcula prețul unui bilet și următoarele restricții:

- Aria ia următoarele valori 1,2,3
 - prețul de bază e pozitiv și mai mic decât 1000.
- Indicați care metodă poate fi folosită pentru un test de tip Order Test.

```
public float getTicketPrice(int area, float basePrice, boolean groupReduction){  
    float finalPrice = 0;  
    switch(area){  
        case 1:  
            finalPrice = basePrice;  
        case 2:  
            finalPrice = basePrice * 1.2f;  
            break;  
        case 3:  
            finalPrice = basePrice * 1.5f;  
            break;  
    }  
  
    if(groupReduction)  
        finalPrice = (float) (0.85 * finalPrice);  
  
    return finalPrice;  
}
```

- a. oricare dintre parametrii
- b. reducere de grup
- c. aria pentru că dacă avem aria 1, atunci prețul pentru aria 2 e mai mică și mai mică pentru aria 3
- d. nici unul dintre parametrii
- e. basePrice pentru că pentru prețuri mai mari se obțin valori mai mari ale prețului final

63. Repository-ul tău poate lua o persoană după id sau query. Care din următoarele metode e potrivită?

- a. Person GetById(string id), Person GetByQuery(Query query)
- b. Person GetBy(string id), Person GetBy(Query query)
- c. Person GetById(string id, Query query)
- d. Person Get(string id), Person Get(Query query)

64. Ce dp implementează Lazy loading? R: Proxy

65. Fiind dată următoarea clasa, selectați (una sau mai multe) opțiuni care o descriu cel mai bine

```
public class DbConnection {  
    String ip;  
    String details;  
  
    public final static DbConnection conexiune;  
  
    static {  
        Settings settings = new Settings();  
        conexiune = new DbConnection(settings.getIp(), settings.getDetalii());  
    }  
  
    private DbConnection(String ip, String detalii) {  
        super();  
        this.ip = ip;  
        this.details = detalii;  
    }  
}
```

- a. e o implementare corectă de Simple Factory
- b. e o implementare corectă de Prototype
- c. **e o implementare corectă de Singleton**
- d. **e un exemplu de eager instantiation**
- e. Ca să fie Singleton, getInstance() e cerută
- f. Implementare greșită de Singleton
- g. Nu poate fi considerată o implementare de dp
- h. Are erori de compilare pentru că conexiune e final și neinitializat
- i. Este un exemplu de lazy instantiation

66. Interfețele client-specific sunt mai bune decât cele generale în:

- a. **Interface Segregation P**
- b. Single Responsibility P
- c. Niciun P solie
- d. Open/Close P
- e. Liskov

67. Care din următoarele e o reprezentare bună a lui Singleton? (PrintManager)

```

public class PrintManager1 {
    public PrintManager1 INSTANCE = new PrintManager1();
    public PrintManager1() {}
    public static PrintManager1 getInstance() {
        return INSTANCE;
    }
}

public class PrintManager2 {
    private PrintManager2 INSTANCE = new PrintManager2();
    private PrintManager2() {}
    public static PrintManager2 getInstance() {
        return INSTANCE;
    }
}

public class PrintManager3 {
    private static final PrintManager3 INSTANCE = new PrintManager3();
    private PrintManager3() {}
    public static PrintManager3 getInstance() {
        return INSTANCE;
    }
}

public class PrintManager4 {
    private final PrintManager3 INSTANCE = new PrintManager4();
    private PrintManager4() {}
    public PrintManager4 getInstance() {
        return INSTANCE;
    }
}

```

R: PrintManager3

68. Dată fiind funcția pentru a calcula prețul unui bilet și următoarele restricții:

- Aria ia următoarele valori 1,2,3
- prețul de bază e pozitiv și mai mic decât 1000.

Indicați ce este sunt considerate de tipul BOUNDARY pentru aria?

```

public float getTicketPrice(int area, float basePrice, boolean groupReduction){
    float finalPrice = 0;
    switch(area){
        case 1:
            finalPrice = basePrice;
        case 2:
            finalPrice = basePrice * 1.2f;
            break;
        case 3:
            finalPrice = basePrice * 1.5f;
            break;
    }

    if(groupReduction)
        finalPrice = (float) (0.85 * finalPrice);

    return finalPrice;
}

```

- a. getTicketPrice(2, 100, false)
- b. getTicketPrice(2, 500, true)
- c. getTicketPrice(4, 700, false)
- d. **getTicketPrice(3, 700, true)**
- e. **getTicketPrice(1, 500, false)**
- f. getTicketPrice(0, 700, true)

TXT

Ce model permite subclaszelor să decidă cum să implementeze pașii într-un algoritm?

-> Template

Composite: Acest model este utilizat acolo unde trebuie să trămă un grup de obiecte în mod similar ca un singur obiect

Ce pattern permite încapsularea unui obiect pentru a-i oferi un comportament specific unei alte interfețe?

-> Adapter

Ce pattern permite încapsularea unui obiect în vederea controlării accesului la acesta?

-> Proxy

Facade: Acest model ascunde complexitatea sistemului/modulului și oferă clientului o interfață pe care o poate folosi pentru a accesa/utiliza sistemul

Flyweight: Acest model este utilizat în principal pentru a reduce numărul de obiecte create și pentru a reduce amprenta de memorie

Adapter: Nici una din aceste afirmații