

## 75. președintii de scără

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

În cadrul soluției un administrator trebuie să poată anunța tutu locatari din bloc atunci când arțează 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 asociatiile de proprietari dintr-un cartier de blocuri. În cadrul cartierului sunt mai multe meseriași (instalațori, 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 meseriași cartierului este liber, va prelua această problemă și o va rezolva. Să se implementeze modulul care permite gestionarea cozii 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 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 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 lăsarea unor măsuri adecvate și care să protejeze viața locatarilor. Soluția procesează diferite mesaje/evenimente primite de la senzori/sisteme din bloc și va lăsa 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 asociatiile 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 caldurii, 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/controllat 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 destinață 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 predefinită 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 călăorească 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 bilete 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 clienți 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 următe 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), vîrstă (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. Clientii 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 Photolmage. 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, comfort 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 înregistrare 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 gestionăți 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 așteptare moștră, încărcare eprubetă, procesare date, afișare rezultate etc). La pornire, echipamentul este în așteptare, 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 intermediu browser-ului. Această soluție este bazată pe module ce vor verifica acțiunile utilizatorilor, module ce au interfață IAntivirus (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 virusi, dacă da atunci download-ul este întrerupt (se afișează mesaj), altfel se trage 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 trage 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 utilizator. 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, 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;

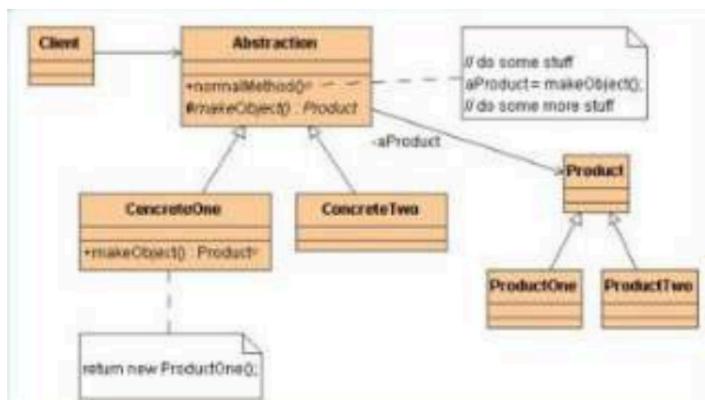
    if(valoare > 100)
        rezultat += 5;
    else
        rezultat += 10;

    return rezultat;
}

```

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

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



3. Fiind data funcția pentru inițializarea varstei și următoarele restricții:

- zona ia valori în 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 primita 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 afirmații 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 zi din lanț
- 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 sa 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 comenziilor**

**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. Defineste strategia adoptate 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

6

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

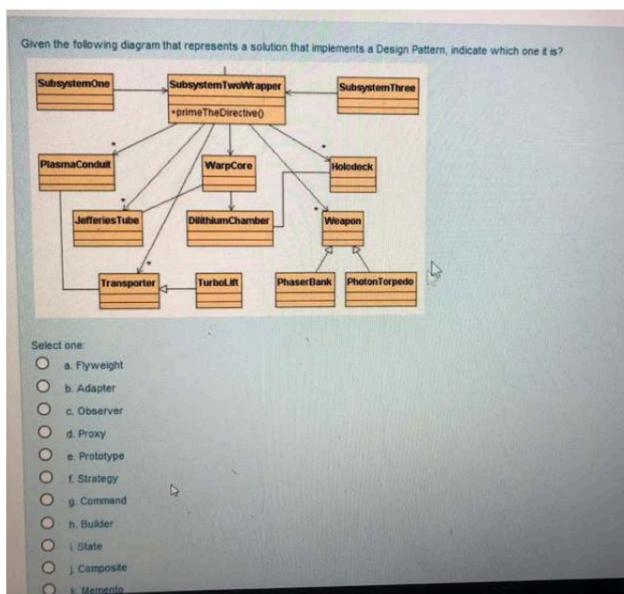
35. Ce dp oferă capacitatea de a restabili un obiect la starea anterioară? R: Memento

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.

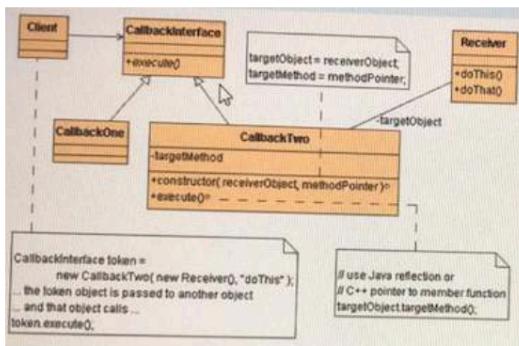
Indicați care din metode poate fi folosit pt Oder test? **Niciun parametru**

```
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;  
}
```

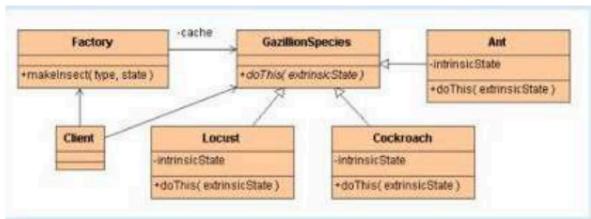
41. Care din următoarele enunțuri descrie modelul Adapter corect?

- 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?

- a. int calculSuma (int valoare1, int valoare2)
- b. int calcul(int valoare)
- c. **int calcul (ArrayList <Integer>valori)**
- d. **int calcul (int [] valori)**
- e. 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



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

50. „Identifice aspectele aplicației care variază și le separă 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. Datează 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 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