1. Intre doua obiecte dintr-o diagrama de comunicare poate exista o conexiune de coumincare oricand este posibil ca unul dintre obiecte sa transmita un mesaj catre celalalt obiect. Clasele celor doua obiecte care interactioneaza pot fi conectate printr-o relatie de:

- Asociere
- Dependenta

2. Coeziunea de comunicare este realizata cand toate modulele (elementele de procesare) care acceseaza sau manipuleaza sunt pastrate …

3. Cuplarea de marca se poate reduce prin:

- Utilizarea unei interfete ca tip al argumentului metodei
- Transmiterea de date simple

4. Efecte laterale pot sa apara numai pentru module sau subsisteme care manifesta:

- Coeziune de nivel

5. In proiectarea prin contract comportamentul unei metode este descris printr-un set de asertiuni care stabilesc:

- Preconditii (solocitate de metoda inaintea executiei)
- Postconditii (generate de metoda la sfarsitul executiei)
- Invariatii (pe care metoda se angejeaza sa nu le modifice in timpul executiei)

6. Participati la dezvoltarea unui program care va stoca un set de obiecte, ce contin atat informatie comuna (ce poate fi partajata de mai multe obiecte), precum si informatie specifica fiecarui obiect. Doriti sa evitati duplicarea pentru a evita inconsistenta datelor. In acest scop se poate utiliza Abstraction Occurrence Pattern.

7. Pentru a reduce dependentele intre pachete (subsisteme) se poate utiliza facade, care ofera o interfata simplificata … .

8. Se considera urmatoarele faze si activitati software din Procesul Unificat. Selectati fazele care pot cuprinde mai mult decat o iteratie:

- Initiere
- Constructie
- Tranzitie
- Elaborare

9. Care din urmatoarele paradigme de dezvoltare software includ o faza de modificare a cerintelor?

- Componente reutilizabile

10. Se considera urmatoarele activitati software si faze din Procesul Unificat. In Procesul Unificat arhitectura sistem se dezvolta in principal in:

- Elaborare

11. Procesul Unificat este orientat pe componente, utilizeaza limbajul UML, pentru construirea modelelor orientate obiect, si are urmatoarele 3 caracteristici distinctive:

- Ghidat pe cazurile de utilizare
- Centrat pe arhitectura
- Iterativ si incremental

12. Un caz de utilizare este descriere a unui set de secvente de actiuni pe care le efectueaza un sistem pentru a produce un rezultat observabil si semnificativ pentru un anumit actor particular.

13. In timp ce toate formele de testare prezentate mai jos sprijina activitatea de verificare software, care dintre acestea sprijina in cea mai mare masura activitatea de validare software?

- Testarea de acceptanta

14. Scrieti formula matematica care defineste complexitatea ciclomatica:

- noEdges – noNodes + 2 = noBinaryDecisionPredicates + 1

15. Se considera urmatoarele notiuni din vocabularul UML ce pot fi utilizate in diagrame de interactiune sau in diagrame de stare. Selectati notiunile utilizate pentru a reprezenta un comportament instantaneu:

- Actiune
- Tranzitie

16. Complexitatea ciclomatica atasata unui CFG reprezinta numarul maxim de cai independente care trebuie sa fie testate pentru a acoperi toate muchiile posibile.

17. Trebuie sa proiectati cazuri de testare pentru un sistem software … 4 tipuri de intrari cu 5, 10, 4, respective 15 clase de echivalenta … Numarul de cazuri de testare pe care trebuie sa le proiectati pentru acest sistem se poate reduce la:

- 15 (cel mai mare nr din cerinta) – MAI ESTE O GRILA CU 14

18. O diagrama de activitate este un caz special de diagrama de tranzitie. … Intr-o diagrama de activitate starile sunt stari de activitate si fluxul controlului trece automat la starea urmatoare Atunci cand activitatea starii curente se termina .

19. Cuplarea externa apare atunci cand un modul are o dependenta semantica fata de sistemul de operare, pachete software de la alta firma, componente hardware.

20. Efectele laterale pot sa apara numai pentru module sau subsisteme care manifesta

- Coeziune de nivel

21. Ordonati urmatoarea lista de tipuri de coeziune in ordinea descrescatoare a gradului de coeziune:

- Functionala
- De nivel
- De comunicare
- De secventiere
- Procedurala
- Temporala

22. In cazul anumitor clase, costul operatiilor de creare si initializare a instantelor este ridicat … Pentru a reduce necesitatea de a crea instante ale unei astfel de clase se poate utiliza proxy pattern.

23. Completati campurile lipsa conform dictomiei tip/instanta din UML:

- Clasa / obiect
- Caz de utilizare / scenariu
- Asociere / conexiune

24. Relatia “universitatea are mai multe departamente” ar trebui modelata ca:

- Compozitie

25. Se considera urmatoarea lista de categorii de cerinte. Selecatati categoriile de cerinte nonfunctionale:

- Fiabilitate
- Timpul de raspuns al sistemului
- Mentenabilitate

26. Selectati practicile si instrumentele utilizate in programarea Extrema (XP) pentru stabilirea si exprimarea cerintelor software:

- Scenarii XP
- Ritm sustinut, fara excese
- Refactorizare
- Dezvoltare ghidata de testarea

27. Intr-un domeniu model exista o colectie de obiecte care partajeaza informatie … Pentru a reprezenta aceasta colectie de obiecte fara a se duplica informatia comuna se poate utiliza singleton.

28. Componenta asigura o interfata unificata la un set de interfete dintr-un subsistem. Facade definseste interfata simplificata (de nivel inalt) care face subsistemul mai usor de utilizat.

29. Intr-un model software modelul cascada (liniar) este adecvat atunci cand:
- Cerintele software nu se schimba (sau este putin probabil sa se schimbe)

30. In testarea incrementala:
- Se proiecteaza module test driver pentru testarea de integrare bottom up
- Se proiecteaza module simulator pentru testarea de integrare top down

31. Design Pattern Observer defineste o dependenta de tip one-to-many intre obiecte astfel incat atunci cand un obiect isi modifica starea toate obiectele care depind de obiectul respective sunt notificate si actualizate automat.

32. Coeziunea de comunicare este realizata cand toate modulele (elementele de procesare) care acceseaza sau manipuleaza anumite date sunt pastrate impreuna si orice alta functionalitate este plasata in alta parte a sistemului.

33. Paradigma Object Oriented favorizeaza coeziunea de comunicare.

34. Proxy Pattern furnizeaza un inlocuitor (sau un subsitut) pentru un alt obiect pentru a controla accesul la obiectul respectiv.

35. Design Pattern Adapter poate fi utilizat pentru a se converti interfata unei clase intr-o alta interfata ceruta de client, permitand astfel ca unele clase cu interfata incompatibila sa functioneze impreuna.

36. Pentru reducerea gradului de cuplare trebuie sa se reduca:
- Numarul conexiunilor intre module
- Taria conexiunilor (sugerata in figura prin grosimea sagetilor)

37. Cuplarea de marca apare atunci cand una din clasele aplicatiei este utilizata ca tip pentru argumentul unei metode.

38. Sablonul de design facade poate reduce cuplarea externa prin simplificarea interfetei spre functiile externe.

39. Ordonati urmatoarea lista de tipuri de cuplare (modul sau subsistem) in ordinea descrescatoare a gradului de cuplare:

- Continut
- Prin date globale
- Control
- Marca
- Date (simple)
- Prin apel de rutina
- Utilizare tip
- Import

40. Modulul afecteaza o sarcina de calcul specifica (unica) si returneaza un rezultat, fara a produce efecte laterale.

41. Un modul este lipsit de efecte laterale faca nu modifica starea sistemului atunci cand efectueaza o sarcina de calcul.

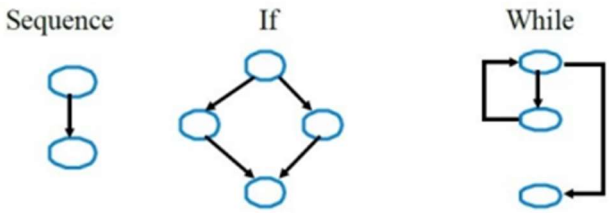
42. O diagrama use case trebuie sa:

- Sa descrie interactiunea unui user cu sistemul
- Sa includa doar actiunile prin care actorul interactioneaza cu computerul
- Sa descrie designul interfetei utilizator din perspectiva unui actor particular

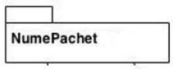
43. Se considera o metoda care implementeaza …

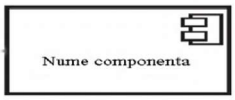
- Un graf orientat cu un singur punct de intrare si mai multe puncta de iesire
- Toate muchiile CFG pentru aceasta metoda

44. Desenati graful de fluenta al controlului pentru constructiile de baza din programarea functionala:



45. Desenati simbolurile UML pentru urmatoarele concepte de utilizare in descrierea arhitecrurii software:

Pachet: 

Componenta: 

Desfasurare: 

Nod: 

46. Desenati diagrama UML de clase care descrie designul pentru Junit 3.x