1. Melyik nem agilis szoftverfejlesztési módszertan szerinti modell az alábbiak közül?

a) Lean

b) Rational Unified Process (RUP)

c) Scrum

d) Kanban

2. Melyik nem része a rendszertervnek?

a) Költségbecslés

b) Felhasználói felület

c) Statikus terv

d) Dinamikus terv

3. Melyik nem funkciója a projektmenedzsment eszközöknek?

a) UML diagramok elkészítése és elhelyezése a tervben (case tooling).

b) Programverziók és változások áttekintése.

c) Hibák bejelentése, kapcsolódó információk (pl. eseménynapló) feltöltése.

d) Feladatok (issue) létrehozása, célszemélyhez (assignee) rendelése.

4. A szoftverfejlesztési csapatnak számos tagja lehet, akik különböző szerepeket töltenek be. Az alábbi szerepek közül melyik van helytelenül meghatározva?

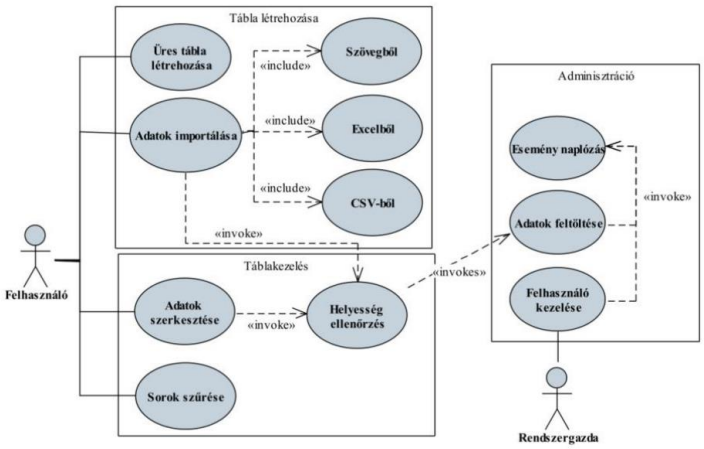
a) termékgazda (product management): üzleti folyamatok, prioritások és elfogadási feltételek kezelése

b) programgazda (program management): fejlesztés ütemezése, feladatok elosztása és követése

c) fejlesztő (developer): szoftver implementációja

d) minőségbiztosító (quality assurance): szoftver magas szintű tervének elkészítése, technikai döntések kezelése

5. Melyik állítás nem helytálló a következő diagramra?



a) Üres tábla létrehozása esetén nem történik adatfeltöltés.

b) Adatok importálása, valamint szerkesztése hatására is történik eseménynaplózás.

c) Az adatok szöveges, Excel és CSV formátumú fájlokból importálhatóak.

d) Amennyiben megszűrjük a sorokat, a tábla létrehozási funkciókat nem használhatjuk.

6. Mely álltás hamis? A követelmények feltárását nehezítheti, hogy...

a) a vevők nem egyértelműen fejtik ki az elvárásokat.

b) a vevők a szoftver közvetlen felhasználói.

c) a vevők bizonytalanok az elvárásokban.

d) a vevők nem rendelkeznek informatikai ismeretekkel.

7. A használati esetek diagramja különböző alkotó elemekből áll össze. Az alábbiak közül melyik nem tartozik közéjük?

a) függőség

b) reláció

c) funkció

d) aktor

8. A használati esetek diagramja különböző relációkat tartalmazhat. Az alábbiak közül melyik nem tartozik közéjük?

a) általánosítás (generalization)

b) kiterjesztés (extend)

c) tartalmazás (include)

d) aggregáció (aggregation)

9. Melyik a használati történet (user story) szerkezete?

a) WHEN tevékenység APPLYING funkció IN ORDER TO cél

b) GIVEN környezet WHEN tevékenység THEN hatás

c) AS A szerepkör USE funkció TO cél

d) USER felhasználó IN USE CASE használati eset WITH RELATION kapcsolat

10. Mely reláció típus nem része a használati eset diagramnak?

a) Származtatás

b) Kompozíció

c) Előfeltétel

d) Tartalmazás

11. Melyik nem "nem funkcionális" követelmény?

a) Menedzselési követelmények

b) Szolgáltatások, reakciók leírása

c) Termék követelmények

d) Külső követelmények

12. Melyik lépés nem része a szoftver specifikációnak?

a) követelmény validáció

b) megvalósíthatósági elemzés

c) követelmény feltárás és elemzés

d) rendszer szerkezetének meghatározása

13. Az alábbi SOLID elvek közül melyik van helyesen megfogalmazva?

a) Dependency inversion principle (DIP): absztrakciókat csak a függőségek között állítunk fel

b) Open/closed principle (OCP): a programegységek nyitottak a módosításra, de zártak a kiterjesztésre

c) Interface segregation principle (ISP): sok kisebb speciális interfész helyett kevesebb, de általános interfészt használjunk

d) Liskov substitution principle (LSP): az objektumok felcserélhetőek altípusaik példányára a program viselkedésének befolyásolása nélkül

14. Hány alapelvet célszerű követnünk a SOLID elv szerint?

a) **5** 4 6 3

15. Mi a szoftver architektúra?

a) A szoftver fejlesztése során meghozott elsődleges tervezési döntések halmaza.

b) A szoftver komponens diagramja.

c) Az az osztályszerkezet, amelyből a csomagdiagramot építjük fel.

d) A szoftvernek a hardver architektúrájára való kitelepülési módja.

16. Melyik nem része a rendszertervnek?

a) Dinamikus terv

b) Statikus terv

c) Felhasználói felület

d) Költségbecslés

17. A modell/nézet architektúrára vonatkozóan az alábbi állítások közül melyik van rosszul megfogalmazva?

a) a nézet tartalmazza a grafikus felhasználói felület megvalósítását, beleértve a vezérlőket és eseménykezelőket

b) a felhasználó a modellel kommunikál, a modell és a nézet egymással

c) a modell nem függ a nézettől, függetlenül, önmagában is felhasználható, ezért könnyen átvihető másik alkalmazásba, és más felülettel is üzemképes

d) a modell tartalmazza a háttérben futó logikát, azaz a tevékenységek végrehajtását, az állapotkezelést, valamint az adatkezelést, ezt nevezzük alkalmazáslogikának, vagy üzleti logikának

18. Mely architektúra esetében a legnehezebb elkülöníteni a program funkcióit?

a) Model-nézet-kontroller architektúra.

b) Model-nézet architektúra.

c) Monolitikus architektúra.

19. Melyik állítás helyes?

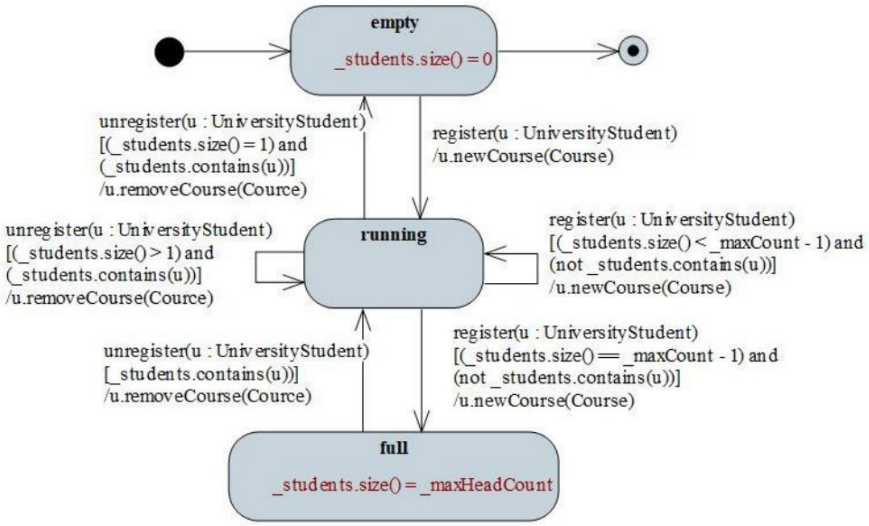
a) Az UML szekvencia diagram (sequence diagram) célja, hogy az objektumok közötti interakció lefolyását az interakciók és az interakciókban felhasznált adatok szempontjából közelítse meg

b) Az UML kommunikáció diagram (communications diagram) célja, hogy az objektumok közötti kommunikáció lefolyását a kommunikációk és a kommunikációkban felhasznált adatok szempontjából közelítse meg

c) Az UML kommunikáció diagram (communications diagram) célja az objektumok közötti kommunikációban felhasznált adatok sorrendjének megállapítása

d) Az UML tevékenység diagram (activity diagram) célja, hogy a végrehajtás lefolyását a tevékenységek és a tevékenységekben felhasznált adatok szempontjából közelítse meg

20. Az alábbi állapot diagramra melyik állítás igaz?



a) A hallgató mérete megegyezhet a fejméretével.

b) Az állapot átmenet feltétele lehet a newCourse metódus meghívása

c) A UniversityStudent objektum newCourse metódusa meghívásra kerülhet

d) A UniversityStudent objektum removeCourse metódusa nem kerülhet meghívásra

21. Melyik diagram nem jó az objektumok és osztályok közötti interakciós folyamatok modellezésére?

a) tevékenység diagram

b) állapot diagram

c) kommunikációs diagram

d) szekvencia diagram

22. A git mely parancsával szinkronizálhatjuk a távoli tárolóba a lokális tárolónkban létrehozott új verziót?

a) git push

b) git commit

c) git pull

d) git synchronize

23. Melyik nem funkciója a projektmenedzsment eszközöknek?

a) Feladatok (issue, ticket) létrehozása, célszemélyhez (assignee) rendelése.

b) Programverziók és változások áttekintése.

c) UML diagramok elkészítése és elhelyezése a tervben (case tooling).

d) Hibák bejelentése, kapcsolódó információk (pl. eseménynapló) feltöltése.

24. Mik a centralizált verziókövető rendszerek hátrányai?

a) Peer-to-peer kommunikáció.

b) A verziókezeléshez hálózati kapcsolat szükséges.

c) Konkurenciakezelés kizárólagos zárak által történik.

d) Fájl alapú műveletvégzés (1 verzió 1 fájl változásai).

25. Melyik állítás igaz az alábbiak közül a Git verziókövető rendszerre?

a) A Git szétválasztja a verziókezelési és a hálózati műveleteket.

b) A Git a beküldés előtti egyesítés konkurenciakezelési módszert alkalmazza (merge before commit).

c) A .gitignore fájlban azt adhatjuk meg, hogy mely állományokat nem szeretnénk a távoli tárolóról letölteni.

d) A Git elosztott verziókövető rendszer, ezért minden kliensnél csak a verziótörténet egy része található meg.

26. Mely állítás hamis a verziókövető rendszerekkel kapcsolatban?

a) A segítségével az összes eddigi program változatot eltárolhatjuk.

b) Lehetővé teszi a módosítások ellenőrzését.

c) Megengedi a változtatások visszavonását.

d) A használatának segítségével nincs szükségünk a konfliktusok kezelésére.

27. Mely állítás vonatkozik az elosztott verziókövető rendszerekre?

a) A kommunikáció peer to peer elven történik, de kitüntetett szerverek felállítására van lehetőség.

b) Fájl alapú műveletvégzést végez.

c) A konkurenciakezelés jellemzően beküldés előtti egyesítéssel történik.

d) Ismert megvalósításai pl.: CVS, SVN, SourceSafe

28. Git verziókezelő eszköz esetén mit értünk a staging area alatt?

a) Azokat a változtatásokat a helyi munkakönyvtárban, amelyeket még nem vontunk verziókövetés alá.

b) Azokat a változtatásokat, amelyeket már egy új verzióban rögzítettünk (commit), de nem küldtük be a távoli tárolóra (push).

c) Azokat a változtatásokat, amelyeket tesztelési célból egy külön fejlesztési ágra küldtünk be.

d) Azokat a változtatásokat, amelyeket már verziókezelés alá vontunk, de még nem mentettük el egy új verzióba (commit).

29. Igaz-e, hogy Git merge esetén nem lehet konfliktus?

a) Igaz, mivel minden merge egyben egy újabb commit is.

b) Hamis, mivel minden merge esetén van konfliktus a kollégák között.

c) Igaz, mivel csak rebase esetén alakulhat ki konfliktus.

d) Hamis, mivel előfordulhat, hogy a git nem tudja megoldani a változások automatikus integrálását.

30. Minek jelölésére nem való a feladatok (issuek) használata egy projektmenedzsment eszközben?

a) A futam lezárásához kapcsolódó megbeszélés

b) Dokumentációs feladat

c) Új funkcionalitás

d) Hiba

31. Mely feladatot nem látja el egy build rendszer?

a) Függőségek kezelése

b) A megváltozott projekt fájlok automatikus feltöltése a verziókezelőbe.

c) Automatizált tesztek végrehajtása

d) Program lefordítása

32. Melyik nem része egy szabálynak a Make build rendszerben?

a) cél

b) változók

c) függőségek

d) utasítások

33. Melyik állítás hamis a CMake build rendszerre vonatkozóan?

a) Generátor build rendszer.

b) Csak C/C++ projektekhez használható.

c) Képes külső függőségek (szoftverkönyvtárak) automatikus betöltésére.

d) Támogatja a telepítési (kihelyezési) folyamatot is.

34. Mi a build rendszerek elsődleges célja?

a) A forráskód felosztása fordítási egységekre.

b) A forráskód fordíthatóvá tétele continuous integration (CI) környezetben

c) A forráskód fordításának a definiált szabályok szerinti automatizálása.

d) A forráskód fordítása konzolos eszközökkel, ha integrált fejlesztőkörnyezet (IDE) nem áll rendelkezésre.

35. Mi a .NET Standard?

a) A .NET Framework új-generációs, cross-platform futtatókörnyezete.

b) A .NET Framework standard library-je.

c) A .NET Framework implementációja Linux operációs rendszerre, korábbi nevén Mono Framework.

d) Olyan API specifikáció, amelynek az összes .NET platform megfeleltethető.

36. Melyik állítás hamis a GNU Make build rendszerre vonatkozóan?

a) A hamis (phony) célok azokat a célokat jelölik, amelyek fájlként nem fognak előállni a szabály eredményeként.

b) Képes külső függőségek (szoftverkönyvtárak) automatikus betöltésére.

c) A Make egy önálló (standalone) build rendszer.

d) Csak a frissített függőségekhez tartozó célokat generálja újra.

37. Hogy nevezik a CMake build rendszer projekt fájlját?

a) Makefile

b) CMakeLists.txt

c) CMakeFiles

d) build.pom

38. Mely állítás igaz?

a) A kiadás tesztet a fejlesztő csapat végzi.

b) A fejlesztői teszt jellemzően fekete doboz tesztekből áll.

c) A füst tesztet a tápegységből felszálló füst mennyiségének mérésével végzik.

d) A felhasználói teszt jellemzően fekete doboz tesztekből áll.

39. A tesztelés...

a) ...futási idejű hibák, rendellenességek, kompatibilitási problémák keresésére használatos.

b) ...garantálja, hogy a program minden körülmény között helytáll.

c) ...célja fordítási időben felderíteni a hibákat.

d) ...garantálja, hogy a program hibamentes.

40. A fejlesztői teszt lépéseinek helyes sorrendje:

a) integrációs teszt, egységteszt, rendszerteszt

b) egységteszt, integrációs teszt, rendszerteszt

c) rendszerteszt, integrációs teszt, egységteszt

d) egységteszt, rendszerteszt, integrációs teszt

41. Az alábbiak közül melyik egy Lehman törvény?

a) egy szoftvert folyamatosan használni kell, vagy különben folyamatosan csökken a használhatósága és minősége

b) egy szoftvernek változnia kell, vagy különben folyamatosan csökken a használhatósága és minősége

c) egy szoftvert folyamatosan használni kell, hogy folyamatosan nőjön a használhatósága és minősége

d) egy szoftvernek változnia kell, hogy folyamatosan csökkenjen a használhatósága és minősége

42. Melyik nem fejlesztői teszt?

a) egységteszt (unit test)

b) rendszerteszt (system test)

c) integrációs teszt (integration test)

d) kiadásteszt (release test)

43. Mi a tesztelés helyes sorrendje?

a) egységteszt, integrációs teszt, felhasználói teszt, rendszerteszt, kiadásteszt

b) egységteszt, integrációs teszt, rendszerteszt, kiadásteszt, felhasználói teszt

c) kiadásteszt, egységteszt, felhasználói teszt, rendszerteszt, integrációs teszt

d) integrációs teszt, felhasználói teszt, kiadásteszt, egységteszt, rendszerteszt

44. Mely állítás hamis?

a) A tesztgyűjtemények által letesztelt programkód mértékét nevezzük kód lefedettségnek.

b) A kiadásteszt nem foglalja magába a kihelyezést (pl. telepítés).

c) Az integrációs és rendszertesztek első lépése a füst teszt.

d) A kiadásteszt és a felhasználói teszt során a szoftvernek már általában a célkörnyezetben, tényleges adatokkal kell dolgoznia.

45. Hogy hívjuk a folyamatos integráció és teljesítés egymásra épülő feladatait?

a) task

b) module

c) milestone

d) job

46. Az alábbiak közül melyik nincs kapcsolatban a folytonos integrációval?

a) GitLab

b) Jetlag

c) Travis

d) AppVeyor

47. Mi a folyamatos teljesítés (continuous delivery) célja?

a) Az önszerveződő, kis csapatok folytonos interakciójának biztosítása gyors visszajelzésekkel.

b) A folyamatos kiadások automatizálása.

c) A gyors alkalmazásfejlesztés megvalósítása, inkrementális alapon.

d) A programkódok egy központi tárhelyre küldésre, verziókezelő rendszer segítségével, naponta többször.

48. Melyik állítás nem igaz a container framework-ökre (pl. Docker)?

a) Minden container osztozik a gazda számítógép hardveres erőforrásain.

b) A containerekben futó alkalmazások belső hálózati kapcsolaton kommunikálhatnak egymással.

c) A containerek saját, elkülönített, virtualizált környezetben futnak.

d) A containerek futó alkalmazások annak saját virtuális operációs rendszerén (pl. Docker OS) futnak.

49. Mi a célja a folyamatos integrációs (continuous integration, CI) gyakorlati módszernek?

a) A lehetséges hibák, integrációs problémák azonnali, automatizált kiszűrése, visszajelzés a fejlesztőnek.

b) A manuális tesztelés teljes kiváltása.

c) Az elbukott integrációs tesztek automatikus újra futtatása, ameddig meg nem javulnak.

d) Objektum orientált programozási nyelvre való átállást segíti elő.

50. Milyen nyelven írható le a Git CI/CD konfigurációja?

a) XAML

b) XML

c) PHP

d) YAML

51. Melyik koncepció része a Clean Code-nak?

a) Ugyanazt a nevet ne használjuk különböző célra

b) A break és continue utasításokat elővigyázatosan kell alkalmaznunk.

c) Rövidítsük mindig a változó neveket

d) Használjunk prefixeket az elnevezéseknél

52. Mely tulajdonságok jellemzőek a Clean Code-ra?

a) Jól dokumentált, tesztelt, elegáns

b) Olvasható, karbantartható, tesztelhető, elegáns

c) Olvasható, tömör, öndokumentáló

d) Könnyen olvasható, nem tartalmaz kódismétlést, tesztelhető

53. Mi NEM jellemző oka a megírt kód folytonos változásának?

a) új funkciók bevezetése

b) hibajavítások

c) kód minőségének javítása

d) követelmény változások

54. Melyik állítás hamis a metódusokkal kapcsolatban a Clean Code-nál?

a) A blokkoknak egyértelmű be- és kilépési pontja kell legyen (break, continue nem megengedett).

b) A megvalósítás férjen rá egy képernyőre.

c) Egy metódus több absztrakciós szintet is megvalósíthat.

d) Ne ismételjük önmagunkat a kódban (DRY).

55. Melyik NEM LÉTEZŐ tervminta osztály?

a) Viselkedési

b) Szerkezeti

c) Létrehozási

d) Végrehajtási

56. Melyik tervezési mintát alkalmazhatjuk abban az esetben, ha egy adott osztály példányosítását szeretnénk a hozzátartozó alosztályokra átruházni?

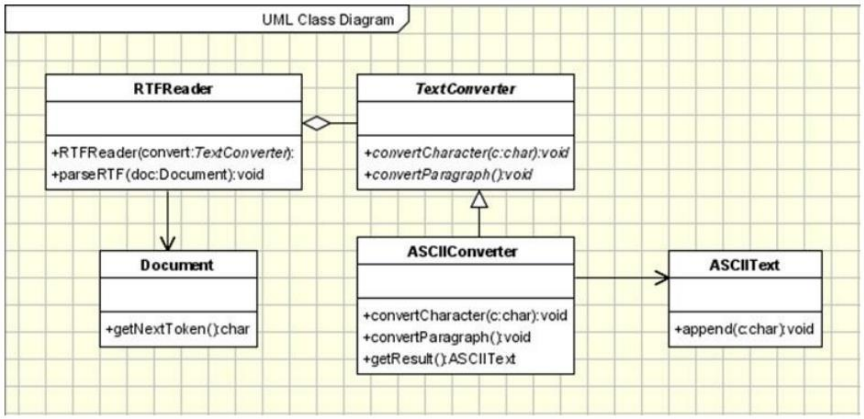
a) Factory method (Gyártó függvény)

b) Observer (Megfigyelő)

c) Command (Parancs)

d) Builder (Építő)

57. Melyik tervezési minta alkalmazása látszik a képen?



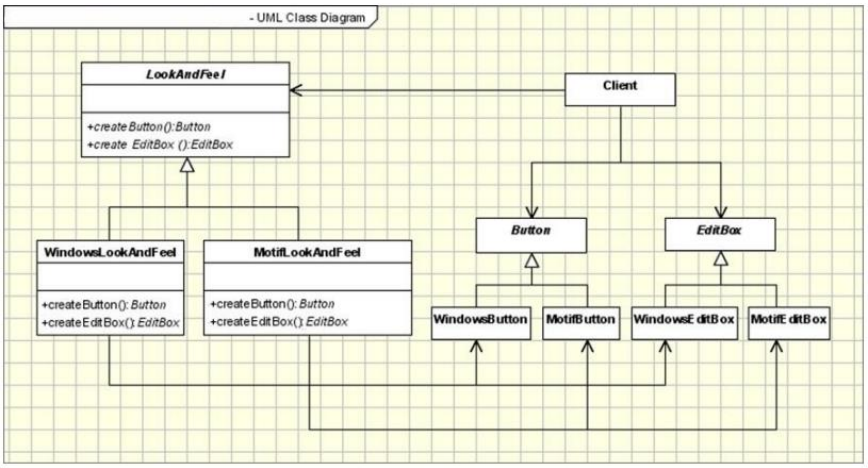
a) építő

b) híd

c) bejáró

d) homlokzat

58. Melyik tervezési minta alkalmazása látszik a képen?



a) gyártófüggvény

b) díszítő

c) megjelenítő

d) elvont gyár

59. Melyik tervmintát soroltuk rossz osztályba?

a) Pehelysúlyú - Viselkedési minta

b) Felelősséglánc - Viselkedési minta

c) Építő - Létrehozási minta

d) Egyke - Létrehozási minta

60. Melyik tervezési mintát alkalmazhatjuk abban az esetben, ha konkrét osztály megadása nélkül szeretnénk kapcsolódó vagy egymástól függő objektumok családjának létrehozására felületet biztosítani?

a) Builder (Építő)

b) Abstract Factory (Absztrakt gyár)

c) Factory method (Gyártó függvény)

d) Adapter (Illesztő)

61. Mely tervminta ad egyszerűsített felületet egy könyvtárhoz, keretrendszerhez stb.?

a) Homlokzat (Facade)

b) Illesztő (Adapter)

c) Helyettes (Proxy)

d) Híd (Bridge)

62. Mely tervminta fordítja le egy osztály interfészét egy kompatibilis másik interfészre?

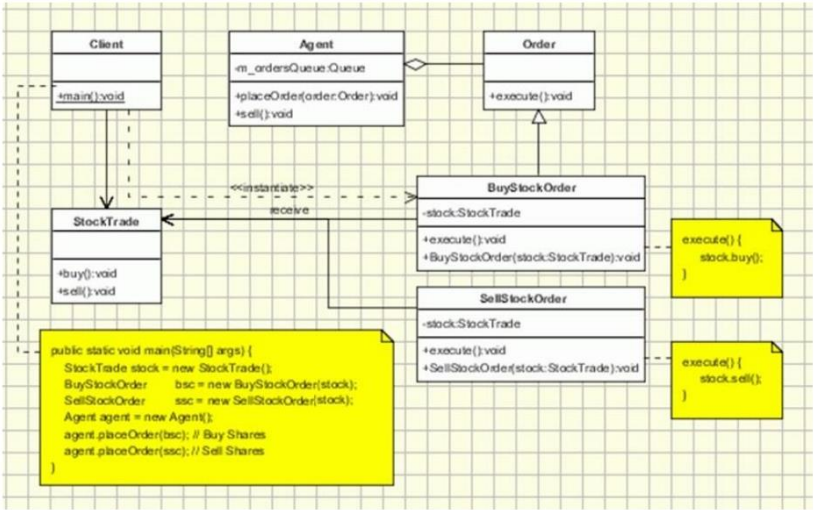
a) Illesztő (Adapter)

b) Homlokzat (Facade)

c) Helyettes (Proxy)

d) Híd (Bridge)

63. Melyik tervezési minta alkalmazása látszik a képen?



a) parancs

b) közvetítő

c) bróker

d) emlékeztető

64. Mely tervminta tudja csökkenteni az objektumok közötti függőségeket?

a) Közvetítő (Mediator)

b) Híd (Bridge)

c) Illesztő (Adapter)

d) Gyártó művelet (Factory method)

65. Melyik tervezési minta alkalmazható a hosszú paraméterlistájú konstruktorok elkerülésére?

a) Builder (Építő)

b) Observer (Megfigyelő)

c) Command (Parancs)

d) Factory (Gyártó)

66. Mely tervmintával csökkenthetjük a szükséges memóriát úgy, hogy megosztjuk az állapot közös részeit több objektum között egy új objektumban?

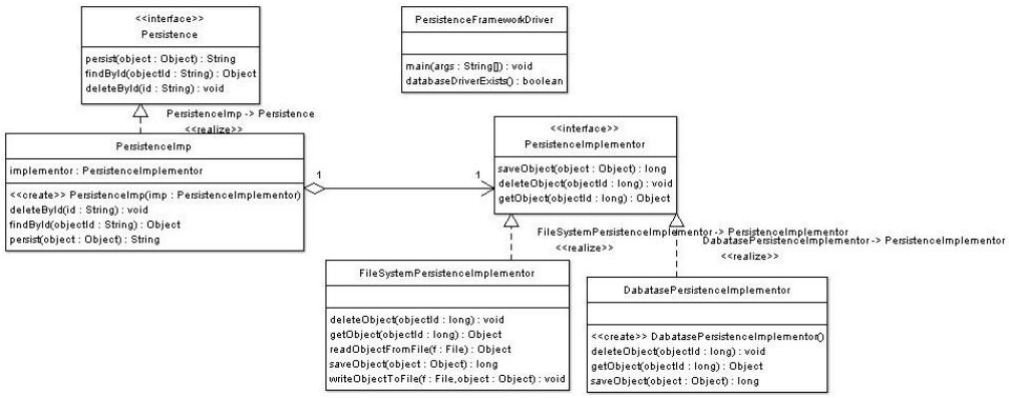
a) Pehelysúlyú (Flyweight)

b) Összetétel (Composite)

c) Helyettes (Proxy)

d) Illesztő (Adapter)

67. Melyik tervezési minta alkalmazása látszik a képen?



a) híd

b) homlokzat

c) perzisztencia

d) közvetítő

68. Melyik tervezési minta nyújt megoldást arra a problémára, ha több objektumot szeretnénk értesíteni, amikor egy másik objektumnak megváltozik az állapota?

a) Observer (Megfigyelő)

b) Singleton (Egyke)

c) Adapter (Illesztő)

d) Factory (Gyártó)

69. Mely tervminta ad lehetőséget egy gyűjtemény bejárására anélkül, hogy az elemek ábrázolását ismernénk?

a) Bejáró (Iterator)

b) Közvetítő (Mediator)

c) Emlékeztető (Memento)

d) Stratégia (Strategy)

70. Mely tervminta teszi lehetővé a kérések továbbítását a kezelők lánca mentén?

a) Felelősséglánc (Chain of Responsibility)

b) Építő (Builder)

c) Híd (Bridge)

d) Közvetítő (Mediator)

71. Melyik állítás igaz a RAII (Resource Allocation Is Initialization) paradigmára vonatkozóan?

a) Jelentése, a biztonságos memóriakezelés azáltal, hogy a hatókörén kívül került objektumokat a szemétgyűjtő felszabadítja (garbage collection).

b) Jelentése, hogy minden erőforrás foglalás megfeleltethető egy objektum példányosításnak, ezért a paradigma C++ nyelven nem alkalmazható, mert nem csak objektum-orientáltan lehet programozni a nyelvben.

c) Jelentése, hogy a kezelendő erőforrás az osztály invariánsa: annak lefoglalására / megnyitására az objektumok létrehozásakor, a felszabadítására / bezárására az objektum megsemmisítésekor kell sor kerüljön.

d) Jelentése, hogy minden objektum példányosítás megfeleltethető egy erőforrás foglalásnak.

72. Mi a kiéheztetés (starvation)?

a) Olyan prioritási hiba, amely esetén a kis prioritású vagy nagy erőforrás igényű folyamatok sosem képesek lefutni.

b) Olyan logikai hiba, amelynek során egy közös erőforráshoz több szál is egyszerre hozzáférhet, ezzel inkonzisztens állapotba juttatva a programot.

c) Olyan jogosultságkezelési hiba, amely során egy szál nem tudja a szükséges erőforrásokat (pl. fájlokat) megnyitni.

d) Olyan ütemezési hiba, amely során egy erőforrásra több szál is várakozik és egyikük sem jut soha hozzá, így "lefagy" a program.

73. Melyik állítás igaz a kölcsönös kizárásra (mutual exclusion)?

a) Nincsen olyan többszálú program, amely kölcsönös kizárás nélkül helyesen tud működni,

b) A kölcsönös kizárás célja, hogy a többszálú program egyszerre mindig csak egy szál futhasson.

c) A kölcsönös kizárás garantálja, hogy a közös erőforráshoz egyszerre csak egy szál férhessen hozzá, kizárva ezzel a versenyhelyzetet (race condition).

d) A kölcsönös kizárás célja a szálak szinkronizációja: a kritikus szakasz mindig ugyanazon a szálon fusson le.

74. Mi a különbség a folyamat (process) és a szál (thread) között?

a) Egy szál több folyamatot is tartalmazhat.

b) Nincs különbség, a kettő egymás szinonimája.

c) A folyamatoknak saját végrehajtási környezetük (pl. memóriaterület) van, a szálak osztozkodnak ezen.

d) A folyamatokat Linux operációs rendszeren szálaknak hívjuk.

75. Melyik állítás hamis az std::future<T> típusra?

a) Az std::future<T> típusú típus alkalmas kivételek átadására is szálak között.

b) Az std::future<T> típus objektum nem feltétlen aszinkron végrehajtást jelent.

c) Egy std::future<T> típus objektum egy ígéret arra, hogy egy T típusú eredmény előállításra kerül majd.

d) Egy std::future<T> típusú objektumba beleírható egy másik szálon majd kiolvasható érték.

76. Melyik lineáris szoftverfejlesztési modell az alábbiak közül?

a) Waterfall (Vízesés)

b) Spiral (Spirális)

c) Kanban

d) Scrum

77. A három Scrum-termék (artifacts) a következő:

a) termékvízió (product vision), termék kívánságlista (product backlog), felhasználói történet (user story)

b) termék kívánságlista (product backlog), futam teendőlista (sprint backlog), Scrum tábla (Scrum table)

c) termék kívánságlista (product backlog), futam teendőlista (sprint backlog), inkrementum (increment)

d) termék kívánságlista (product backlog), Scrum tábla (Scrum table), haladási diagram (progress diagram)

78. Mi az előadáson elmondott planning poker?

a) kártyajáték

b) projektmenedzsment becslési módszere

c) szoftver tervezésének becslési módszere

d) szerencsejáték

79. Melyik nem agilis szoftverfejlesztési módszertan szerinti modell az alábbiak közül?

a) Lean

b) Scrum

c) Rational Unified Process (RUP)

d) Kanban

80. Melyik nem tartozik az előadáson felsorolt SCRUM folyamat elemek közé?

a) bemutató

b) bemutató tervezés

c) visszatekintés

d) futam

81. Melyik nem iteratív szoftverfejlesztési módszertan szerinti modell az alábbiak közül?

a) Extreme Programming (XP)

b) Scrum

c) Kanban

d) Spiral (Spirális)

82. Melyik állítás igaz a Scrum master-re?

a) A Scrum mester nem felel azért, hogy külső hatásoktól védje a Scrum csapat munkáját.

b) A Scrum master a folyamatokért felel.

c) A Scrum mester vezeti a napi Scrumot.

d) A Scrum master a Scrum csapat menedzsere.

83. Melyik állítás igaz a napi Scrum-ra?

a) A napi Scrum-megbeszélés célja, hogy a Scrum csapat tagjai összehangolják tevékenységüket. A megbeszélés ideje maximum 15 percre korlátozott.

b) A napi Scrum-megbeszélés célja, hogy a maximum 1 óra hosszúra korlátozott megbeszélés során a Scrum Master megismerje, hogy ki mennyit haladt a projekttel az előző megbeszélés óta.

c) A napi Scrum-megbeszélésen bárki részt vehet és beszélhet a Scrum Master-rel jelentkezéses alapon.

d) A napi Scrum során az a cél, hogy felszámoljuk a csapatot érintő akadályokat.

84. Melyik SOLID elv ad lehetőséget a polimorfizmus megvalósítására?

a) ISP

b) OCP

c) LSP

d) SRP

85. Mely állítás hamis a Single Responsibility Principle-el kapcsolatban?

a) Ha az SRP elvet megszegjük, akkor az Open/Closed Principle-t is.

b) Feltöredezheti a programszerkezetet a túlzott használata.

c) Egy programegység csak egy felelősséggel rendelkezhet.

d) Elősegíti a programegységek laza összekapcsolását

86. Melyik állítás hamis? A Liskov helyettési elv...

a) ...tiltja a kivételek típusának bővítését.

b) ...elvárja a paraméterek kontravarianciáját.

c) ...elvárja a visszatérési értékek kovarianciáját.

d) ...megengedi az invariánsok gyengítését.

87. Mely állítás állja meg a helyét a Függőségek megfordítása elvnél?

a) Az objektum az osztályok példányosítását közvetlen saját maga végzi.

b) Az osztály mezői a konkrét osztályok példányait tartalmazzák.

c) A konkrét osztályok az absztrakció segítségével lépnek kapcsolatba egymással.

d) A konkrét osztályokat nem alakíthatjuk át.

88. Milyen módon NEM fecskendezhetünk be függőséget?

a) Interfész segítségével, ahol a kliens megvalósítja a beállító műveletet.

b) Beállító műveleten keresztül.

c) Egyke tervminta használatával.

d) Konstruktor paraméteren keresztül.

89. Melyik állítás hamis a tervezés fázisait illetően?

a) Minden fázisban tovább pontosíthatók a már létező osztályok.

b) Minden fázisban bevezethetünk új osztályokat.

c) A funkcionálisan elaprózódott osztályokat egy csomagba kell tennünk.

d) A bonyolulttá váló osztályokat felbonthatjuk.

90. Mi nem lehet program komponens?

a) Alkotóelem. (Artifact)

b) Programkönyvtár. (Class library)

c) Végrehajtható állomány. (Executable)

91. Melyik nem build rendszer?

a) Maven

b) Cthulhu

c) Gradle

d) Ant

92. Hogy nevezik a Maven build rendszer projekt fájlját?

a) pom.xml

b) pom.pom

c) build.pom

d) build.xml

93. Minek a rövidítése a GAV a Maven rendszerben?

a) Graphical Advanced Visualizer

b) Genuine Application Verifier

c) Group, Application Id, Version

d) Group, Artifact Id, Version

94. Mely állítás igaz a pom.xml tartalmával kapcsolatban?

a) groupId: a modult fejlesztő csapat cégen belüli azonosítója

b) artifactId: a projekt egyedi neve

c) version: a fordításhoz szükséges minimális Maven verzió

d) modelVersion: az alkalmazás modell rétegének verziószáma

95. Mely állítás igaz a verifikációval kapcsolatban?

a) A dinamikus elemzés felfedheti a programegységek együttműködéséből származó hibákat.

b) A dinamikus elemzés nem alkalmas a program teljesítményének mérésére.

c) A statikus elemzés ellenére nem garantált a hibák egymás általi elfedése.

d) A statikus elemzés a teljes programkód ismeretében végezhető el.

96. A folytonos integráció célja...

a) ... a több, kisebb részből álló kód egy nagyobb, összefüggő kódbázissá alakítása.

b) ... az elkészített szoftver automatikus eljuttatása a felhasználókhoz.

c) ... a programverziók tárolása, visszakereshetősége.

d) ... a lehetséges hibák, integrációs problémák azonnali, automatizált kiszűrése, visszajelzés a fejlesztőknek.

97. Minek jelölésére nem való a cédulák (tickets) használata egy projektmenedzsment eszközben?

a) Hiba

b) Dokumentációs feladat

c) Új funkcionalitás

d) A futam lezárásához kapcsolódó megbeszélés

98. Mit nem ad meg egy cédula (ticket)?

a) Feladat leírása

b) Feladat költsége

c) Feladat felelőse

d) Feladat határideje

99. Melyik NEM része a TDD három alapszabályának?

a) A sikeres tesztelést mindig refaktorálás követi.

b) Csak annyi kódot írjunk, amennyi éppen elegendő a sikeres teszthez.

c) NE kódoljunk semmit, kivéve ami ahhoz kell, hogy a programunk átmenjen a sikertelen teszten.

d) Tesztből csak éppen elegendő mértékűt írjunk a hiba demonstrálásához.

100. Mi igaz a TDD elvű fejlesztésre?

a) Végeredményként egy struktúrált kódot kapunk, ami kielégíti a teszteket, de még esetlegesen refaktorálásra szorul.

b) A végső kódnak elegendő jól működnie az utoljára bevezetett tesztesetekben.

c) Elegendő, ha csak annyit kódolunk, ami pillanatnyilag szükséges.

d) A tesztek a lehetséges használati eseteknek csak egy részhalmazát fedik le.

101. Melyik állítás igaz a Clean Code-al kapcsolatban?

a) Hibakódokat és kivételeket egyaránt használatunk.

b) A függvények egyszerre válaszolhatnak egy kérdésre és hajthatnak végre valamit.

c) A komment segít a nehezen érthető kódot megérteni.

d) A kód és a komment nem feltétlenül él együtt.

102. Mely szál futtatja az eseménykezelésért felelős tevékenységeket?

a) ELT

b) DBT

c) DDT

d) EDT

103. Melyik nem létező szál állapot?

a) Resumed

b) Stopped

c) Running

d) Suspended

104. Mi a szálak alapértelmezett prioritása?

a) 5

b) 10

c) 0

d) 1

105. A Thread objektum mely metódusával indítható el új szál?

a) run

b) execute

c) start

d) spawn

106. Mely metódussal tud egy szál lemondani a CPU időről?

a) interrupt

b) wait

c) suspend

d) sleep

107. Egy szál mely metódusának meghívásával lehet megvárni a szál befejeződését?

a) join

b) await

c) wait

d) stop

108. Mit jelent párhuzamos környezetben a kiéheztetés?

a) A szálak nem mondanak le a használt erőforrásaikról.

b) Egy szál nem tud hozzáférni a kívánt erőforráshoz huzamosabb ideig, mert más, hosszú futásidejű szálak korábban kapják azt meg.

c) A párhuzamos program szálai nem tudnak elindulni, mert az operációs rendszer korábban zombi állapotba került.

d) Mire a program hajlandó befejeződni, a felhasználó már régen éhen halt.

109. Egy lock-ként használt objektum mely metódusával ébreszthető fel az arra várakozó szál?

a) wait

b) wake

c) spotifiy

d) notify

110. Mi igaz egy immutable objektumra?

a) Attribútumai csak setter metódusok segítségével változtathatók meg

b) Az objektumban nem tárolhatunk referenciát más objektumokra.

c) Az objektum állapota nem változtatható meg a konstruktor lefutása után

d) Az objektumban attribútumai nem változtathatók meg, de a referenciák esetében a rajtuk keresztül elérhető objektumokon ez megengedett.

111. Mi volt az ún. szoftverkrízis kiváltó oka az 1960-as és 70-es években?

a) A szoftverfejlesztés alulfinanszírozottága miatti pénzügyi okok.

b) A szoftverprojektek méretben, komplexitásban, időben és a résztvevő fejlesztők számában is növekedni kezdtek.

c) A hidegháborús versengés miatti tudástranszfer akadályozása.

d) A magasabb szintű programozási nyelvek megjelenése.

112. Mi a Single Responsibility Principle (SRP) elv célja?

a) Minden osztály reprezentációját egyetlen adattagban kell definiálni, egy komplex adatstruktúra létrehozásával.

b) A felhasználói felület minden vezérlője csak egyetlen funkcióért felelhet, szoftverergonómiai megfontolásból.

c) Minden komponens, osztály, metódus csak egy felelősségi körrel rendelkezzen, ami megváltoztatásának oka lehet.

d) Minden metódus csak egyféle típusú kivétel kezeletkenül hagyását teheti lehetővé.

113. A modell/nézet architektúrára vonatkozóan az alábbi állítások közül melyik van rosszul megfogalmazva?

a) a nézet tartalmazza a grafikus felhasználói felület megvalósítását, beleértve a vezérlőket és eseménykezelőket

b) a modell függ a nézettől, egy modellt mindig egy adott felülethez készítünk el

c) a felhasználó a nézettel kommunikál, a modell és a nézet egymással

d) a modell tartalmazza a háttérben futó logikát, azaz a tevékenységek végrehajtását, az állapotkezelést, valamint az adatkezelést, ezt nevezzük alkalmazáslogikának, vagy üzleti logikának

114. Melyik helyes?

a) Az UML szekvencia diagram (sequence diagram) célja az objektumok közötti kommunikáció sorrendjének megállapítása.

b) Az UML tevékenység diagram (activity diagram) célja az objektumok közötti kommunikáció sorrendjének megállapítása.

c) Az UML kommunikáció diagram (communications diagram) célja az objektumok közötti kommunikáció sorrendjének megállapítása.

114. Melyik helyes?

a) Az UML szekvencia diagram (sequence diagram) célja az objektumok közötti interakció időrendi ábrázolása.

b) Az UML tevékenység diagram (activity diagram) célja az objektumok közötti interakció időrendi ábrázolása.

c) Az UML kommunikáció diagram (communications diagram) célja az objektumok közötti interakció időrendi ábrázolása.

115. Az alábbiak közül a git mely parancsával szinkronizálhatjuk a távoli tárolóból a lokális tárolónkba az oda mások által beküldött új verziót?

a) git pull

b) git commit

c) git push

d) git synchronize

116. Melyik nem projektvezető szolgáltatás?

a) Redmine

b) GitLab

c) Azure Devops

d) GitHub

117. Milyen fajta kommentet kerüljük?

a) Kikommentezett kód

b) TODO

c) Következményre figyelmeztető komment

d) Szándékot, pontosítást tartalmazó komment