⚠ Ez a kvíz újra lett értékelve, ez az eredményét is befolyásolja.

Évfolyamzárthelyi 0524

Határidő Nincs megadva határidő **Pont** 45 **Kérdések** 45
Elérhető máj 24, 09:00-ig **Időkorlát** 45 perc

Instrukciók

Az előadás ismeretanyagából az aláírás megszerzésének feltételeként a félév végén zárthelyit kell írni. A zárthelyi értékelése kétfokozatú (megfelelt / nem felelt meg), a sikeres teljesítéshez a kérdések legalább 2/3-át kell helyesen megválaszolni.

Az elméleti zárthelyin 45 feleletválasztós kérdésre kell választ adni 45 perc alatt. A zárthelyi sikeres, amennyiben legalább 30 kérdés helyesen megválaszolásra kerül.

A zárthelyit a géptermi gépeken kell teljesíteni, és semmilyen segédeszköz nem használható.

Ezt a kvízt ekkor zárolták: máj 24, 09:00.

Próbálkozások naplója

	Próbálkozás	Idő	Eredmény	Újraértékelt
LEGUTOLSÓ	1. próbálkozás	14 perc	38 az összesen elérhető 45 pontból	39 az összesen elérhető 45 pontból

⚠ A helyes válaszok el vannak rejtve.

Ezen kvíz eredménye: **39** az összesen elérhető 45 pontból
Beadva ekkor: máj 24, 08:19
Ez a próbálkozás ennyi időt vett igénybe: 14 perc

1. kérdés 1 / 1 pont

Melyik állítás **igaz** a szoftver architektúrára?

- A szoftver architektúrát a program osztálydiagramja alapján határozzuk meg.

A szoftver architektúra célja a feladat megoldásához leginkább alkalmas programozási nyelv meghatározása.

A szoftver architektúra jellemzően folyamatosan változik, fejlődik a projekt fejlesztése folyamán.

A szoftver architektúra a rendszer magas szintű komponenseinek és kapcsolataiknak meghatározása.

2. kérdés

1 / 1 pont

Az alábbi, alkalmazások architektúrájára vonatkozó állítások közül melyik **hamis**?

A függőségeket úgy kell megvalósítani, hogy a konkrét megvalósítástól függjenek.

Az egyes rétegek között függőségek alakulnak ki, mivel felhasználják egymás funkcionalitását.

A befecskendezésnek különböző módjai lehetnek (például: konstruktor, metódus).

A függőség befecskendezés (*dependency injection*) jelentése, hogy a rétegek a függőségeknek csak az absztrakcióját látják, a konkrét megvalósítását külön adjuk át nekik.

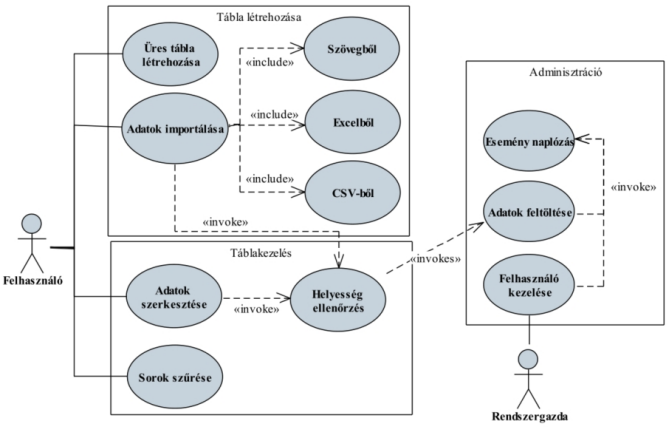
Helytelen

Question 3

Eredeti eredmény: 0 / 1 pont **Újraértékelt eredmény: 1 / 1 pont**

ⓘ Ez a kérdés újra lett értékelve.

Melyik állítás **nem** helytálló a következő diagramra?



- ○ Az adatok szöveges, Excel és CSV formátumú fájlokból importálhatóak.
- ○ Adatok importálása hatására nem történik eseménynaplózás.
- ● Üres tábla létrehozása esetén nem történik adatfeltöltés.
- ○ Felhasználó is tölthet fel adatokat a rendszerbe.

4. kérdés

1 / 1 pont

Mely reláció típus nem része a használati eset diagramnak?

- ● Kompozíció
- ○ Előfeltétel
- ○ Származtatás
- ○ Tartalmazás

Helytelen

5. kérdés

0 / 1 pont

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

- minőségbiztosító (quality assurance): tesztelés tervezése, magvalósítása, minőségi kritériumok ellenőrzése
- fejlesztő (developer): szoftver implementációja
- termékgazda (product management): szoftver magas szintű tervének elkészítése
- programgazda (program management): fejlesztés ütemezése, feladatok elosztása és követése

6. kérdés

1 / 1 pont

Melyik a használati történet (user story) szerkezete?

- GIVEN környezet WHEN tevékenység THEN hatás
- WHEN tevékenység APPLYING funkció IN ORDER TO cél
- AS A szerepkör USE funkció TO cél
- USER felhasználó IN USE CASE használati eset WITH RELATION kapcsolat

7. kérdés

1 / 1 pont

Mi a szoftver architektúra?

- A szoftver komponens diagramja.
- Az az osztályszerkezet, amelyből a csomagdiagramot építjük fel.
- A szoftvernek a hardver architektúrájára való kitelepülési módja.
- A szoftver fejlesztése során meghozott elsődleges tervezési döntések halmaza.

8. kérdés

1 / 1 pont

Melyik nem része a rendszertervnek?

- Üzemeltetési és karbantartási terv
- Felületi tervek
- Perzisztencia (adattárolási módszerek és formátumok)
- UML komponens diagramok

9. kérdés

1 / 1 pont

Melyik állítás helyes?

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

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

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

10. kérdés

1 / 1 pont

Melyik **nem** funkciója a projektmenedzsment eszközöknek?

- UML diagramok elkészítése és elhelyezése a tervben (case tooling).
- Hibák bejelentése, kapcsolódó információk (pl. eseménynapló) feltöltése.
- Feladatok (issue, ticket) létrehozása, célszemélyhez (assignee) rendelése.
- Programverziók és változások áttekintése.

11. kérdés

1 / 1 pont

Mik a centralizált verziókövető rendszerek hátrányai?

- Konkurenciakezelés kizárólagos zárak által történik.
- A verziókezeléshez hálózati kapcsolat szükséges.
- Fájl alapú műveletvégzés (1 verzió 1 fájl változásai).
- Peer-to-peer kommunikáció.

12. kérdés

1 / 1 pont

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

- Új funkcionalitás
- A futam lezárásához kapcsolódó megbeszélés
- Dokumentációs feladat
- Hiba

13. kérdés

1 / 1 pont

Mely állítás hamis a verziókövető rendszerekkel kapcsolatban?

- Megengedi a változtatások visszavonását.
- Lehetővé teszi a módosítások ellenőrzését.
- A segítségével az összes eddigi program változatot eltárolhatjuk.
- A használatának segítségével nincs szükségünk a konfliktusok kezelésére.

14. kérdés

1 / 1 pont

Mely feladatot **nem** látja el egy build rendszer?

- Automatizált tesztek végrehajtása
- Program lefordítása

- Függőségek kezelése
- A megváltozott projekt fájlok automatikus feltöltése a verziókezelőbe.

15. kérdés

1 / 1 pont

Melyik **nem** fejlesztői teszt?

- kiadásteszt (release test)
- rendszerteszt (system test)
- egységteszt (unit test)
- integrációs teszt (integration test)

16. kérdés

1 / 1 pont

A tesztelés...

- ...célja fordítási időben felderíteni a hibákat.
- ...garantálja, hogy a program minden körülmény között helytáll.
- ...garantálja, hogy a program hibamentes.
- ...futási idejű hibák, rendellenességek, kompatibilitási problémák keresésére használatos.

17. kérdés

1 / 1 pont

Mely állítás hamis?

- (•) A kiadásteszt nem foglalja magába a kihelyezést (pl. telepítés).
- ( ) Az integrációs és rendszertesztek első lépése a füst teszt.
- ( ) A kiadásteszt és a felhasználói teszt során a szoftvernek már általában a célkörnyezetben, tényleges adatokkal kell dolgoznia.
- ( ) A tesztgyűjtemények által letesztelt programkód mértékét nevezzük kód lefedettségnek.

18. kérdés

1 / 1 pont

Melyik **nem** projektvezető szolgáltatás?

- ( ) GitHub
- ( ) GitLab
- (•) Redmine
- ( ) Azure Devops

19. kérdés

1 / 1 pont

Az alábbiak közül a git mely parancsával szinkronizálhatjuk a távoli tárolóba a lokális tárolónkban létrehozott új verziót?

- ( ) git pull

- [x] git push
- [ ] git commit
- [ ] git synchronize

20. kérdés

1 / 1 pont

Mi a Git LFS (Large File Storage) célja és működési elve?

- [ ] A nagy méretű bináris állományokat Git helyett SVN verziókezelőrendszerbe helyezi, amely alkalmas a bináris állományok hatékony verziókezelésére.
- [ ] A nagy méretű bináris állományokat a GitHub szerverén tárolja, így az nem növeli a tároló (repository) méretét.
- [ ] A nagy méretű bináris állományoknak csak az utolsó állapotát őrzi meg, mert ezek verziókezelése jellemzően szükségtelen.
- [x] A nagy méretű bináris állományokat egy hivatkozással helyettesíti, és magukat a fájlokat külön tárolja.

21. kérdés

1 / 1 pont

Git verziókezelő eszköz esetén mit értünk a *staging area* alatt?

- [ ] Azokat a változtatásokat a helyi munkakönyvtárban, amelyeket még nem vontunk verziókövetés alá.

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

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

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

22. kérdés

1 / 1 pont

Milyen nyelven írható le a GitLab CI/CD konfigurációja?

- YAML
- XAML
- XML
- JSON

Helytelen

23. kérdés

0 / 1 pont

Milyen célt szolgál a GitLab CI cache konfigurációjának kulcsa (key)?

Használatával korlátozható, hogy mely jobok férhetnek hozzá a cache-hez.

Használatával a cache tartalma artifact-ként letölthetővé tehető.

Használatával megadható a Cache Server elérhetősége, amennyiben nem az alapértelmezettet kívánjuk használni.

Használatával különálló cache használható akár jobonként vagy fejlesztési áganként.

24. kérdés

1 / 1 pont

Mi a folyamatos teljesítés (continuous delivery) célja?

- (•) A gyors alkalmazásfejlesztés megvalósítása, inkrementális alapon.
- ( ) A folyamatos kiadások automatizálása.
- ( ) A programkódok egy központi tárhelyre küldésre, verziókezelő rendszer segítségével, naponta többször.
- ( ) Az önszerveződő, kis csapatok folytonos interakciójának biztosítása gyors visszajelzésekkel.

25. kérdés

1 / 1 pont

Melyik fajta kommentet kerüljük?

- ( ) TODO
- ( ) Szándékot, pontosítást tartalmazó komment
- (•) Kikommentezett kód
- ( ) Következményre figyelmeztető komment

26. kérdés

1 / 1 pont

Mi **NEM jellemző** oka a megírt kód folytonos változásának?

- ○ új funkciók bevezetése
- ○ hibajavítások
- ○ követelmény változások
- ● kód minőségének javítása

27. kérdés

1 / 1 pont

Melyik állítás **hamis** a metódusokkal kapcsolatban a Clean Code-nál?

- ○ Ne ismételjük önmagunkat a kódban (DRY).
- ○ A megvalósítás férjen rá egy képernyőre.
- ○ A blokkoknak egyértelmű be- és kilépési pontja kell legyen (break, continue nem megengedett).
- ● Egy metódus több asztrakciós szintet is megvalósíthat.

Helytelen

28. kérdés

0 / 1 pont

Melyik tervmintát soroltuk **rossz** osztályba?

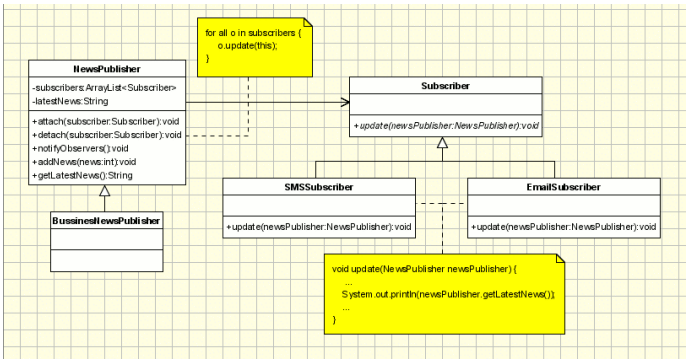
- ○ Felelősséglánc - Viselkedési minta

- Építő - Szerkezeti minta
- Egyke - Létrehozási minta
- Pehelysúlyú - Szerkezeti minta

29. kérdés

1 / 1 pont

Melyik tervezési minta alkalmazása látszik a képen?



- elvont gyár
- objektumkészlet
- homlokzat
- megfigyelő

30. kérdés

1 / 1 pont

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

- Pehelysúlyú (Flyweight)
- Helyettes (Proxy)

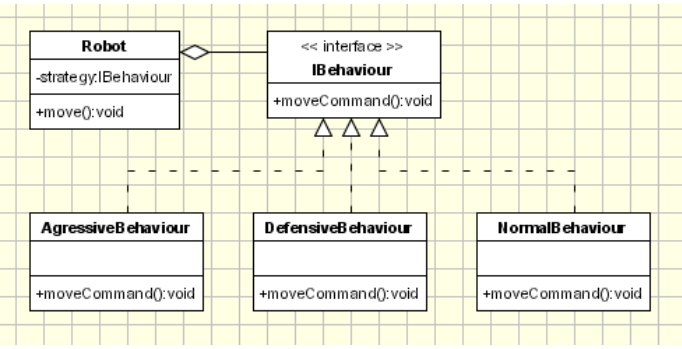
- Összetétel (Composite)
- Illesztő (Adapter)

Helytelen

31. kérdés

0 / 1 pont

Melyik tervezési minta alkalmazása látszik a képen?



- Command (Parancs)
- Decorator (Díszítő)
- Template method (Sablonfüggvény)
- Strategy (Stratégia)

32. kérdés

1 / 1 pont

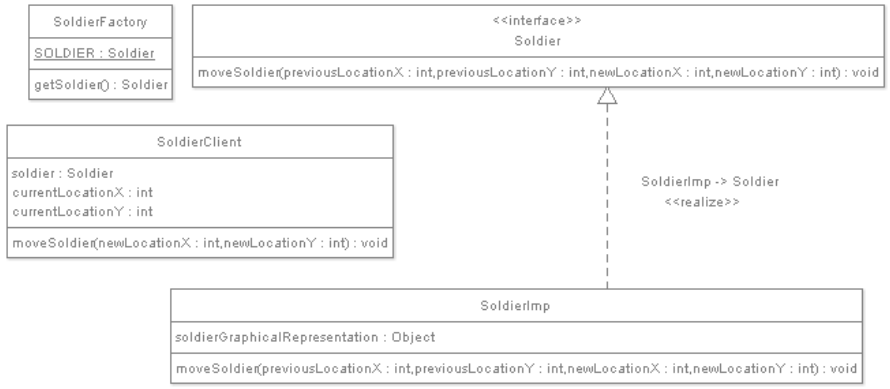
Melyik lépés **nem** része a szoftver specifikációnak?

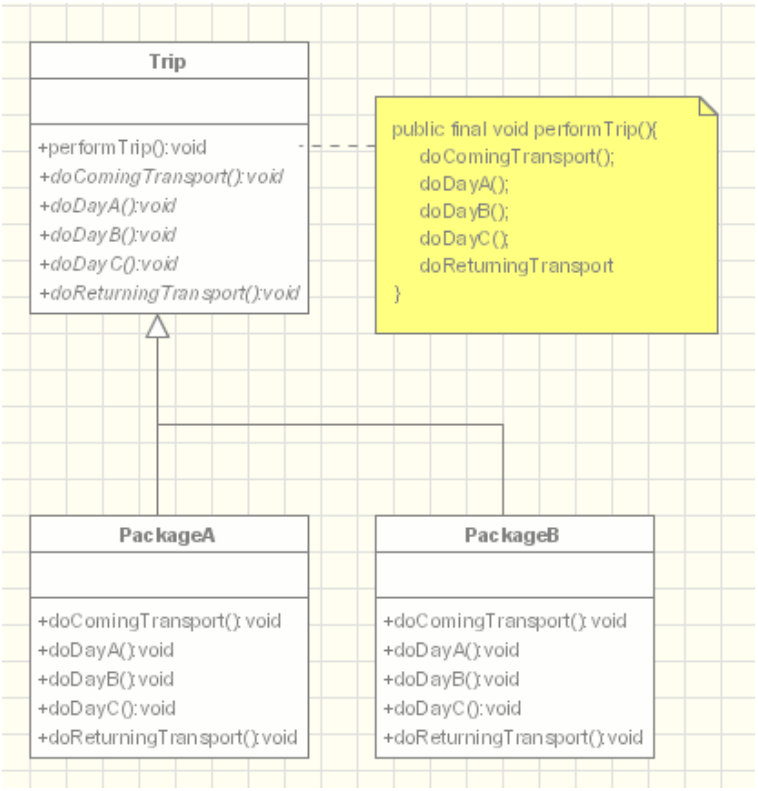
- megvalósíthatósági elemzés
- követelmény validáció
- követelmény feltárás és elemzés
- rendszer szerkezetének meghatározása

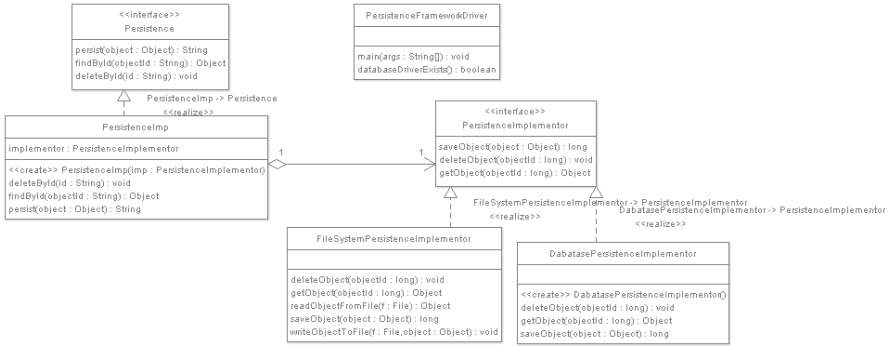
33. kérdés

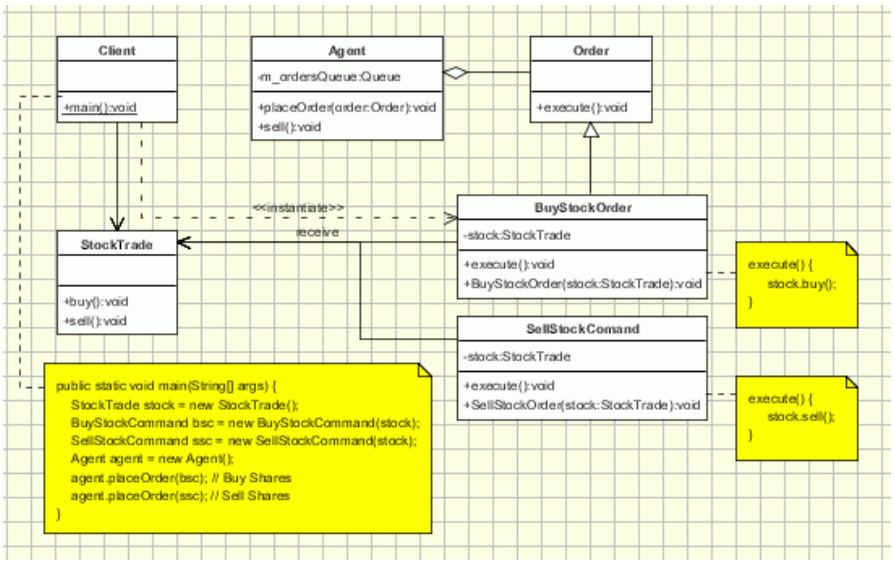
1 / 1 pont

Melyik diagram mutat példát a Parancs (Command) tervezési mintára?









34. kérdés

1 / 1 pont

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

- szekvencia diagram
- kommunikációs diagram
- tevékenység diagram
- állapot diagram

35. kérdés

1 / 1 pont

Mit mond ki a DRY elv?

- Ne implementáljunk előre olyan kódot, ami „majd a jövőben kelleni fog”, mert szinte biztos, hogy sose lesz rá szükségünk.
- A tökéletességet nem akkor lehet a legjobban megközelíteni, ha egy rendszerhez nem tudunk már semmit hozzáadni, hanem akkor, ha nem tudunk mit elvenni belőle.
- **A tudás minden darabkájának egyetlen, egyértelmű és megbízható reprezentációval kell rendelkeznie egy rendszeren belül.**
- Az biztosan elmondható, hogy javulni fog a kódbázisunk minősége, ha mindig úgy hagyjuk ott az aktuális kódunkat, hogy az egy kicsit „jobb”, egy kicsit tisztább annál, mint ahogy megtaláltuk.

36. kérdés

1 / 1 pont

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

- Homlokzat (Facade)
- Helyettes (Proxy)
- **Illesztő (Adapter)**
- Híd (Bridge)

Helytelen

37. kérdés

0 / 1 pont

Az alábbi SOLID elvek közül melyik van helyesen megfogalmazva?

- Open/closed principle (OCP): a programegységek nyitottak a módosításra, de zártak a kiterjesztésre
- Dependency inversion principle (DIP): absztrakciókat csak a függőségek között állítunk fel
- Interface segregation principle (ISP): sok kisebb speciális interfész helyett kevesebb, de általános interfészt használjunk
- Liskov substitution principle (LSP): az objektumok felcserélhetőek altípusaik példányára a program viselkedésének befolyásolása nélkül

38. kérdés

1 / 1 pont

Mely tervminta ad lehetőséget egy gyűjtemény felsorolására anélkül, hogy az elemek ábrázolását ismernénk?

- Bejáró (Iterator)
- Emlékeztető (Memento)
- Stratégia (Strategy)
- Közvetítő (Mediator)

39. kérdés

1 / 1 pont

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

- Nincs különbség, a kettő egymás szinonimája.
- A folyamatoknak saját végrehajtási környezetük (pl. memóriaterület) van, a szálak osztozkodnak ezen.
- Egy szál több folyamatot is tartalmazhat.
- A folyamatokat Linux operációs rendszeren szálaknak hívjuk.

40. kérdés

1 / 1 pont

Melyik állítás igaz a kölcsönös kizárásra (*mutual exclusion*)?

- A kölcsönös kizárás célja a szálak szinkronizációja: a kritikus szakasz mindig ugyanazon a szálon fusson le.
- A kölcsönös kizárás garantálja, hogy a közös erőforráshoz egyszerre csak egy szál férhessen hozzá, kizárva ezzel a versenyhelyzetet (*race condition*).
- A kölcsönös kizárás célja, hogy a többszálú program egyszerre mindig csak egy szál futhasson.
- Nincsen olyan többszálú program, amely kölcsönös kizárás nélkül helyesen tud működni,

41. kérdés

1 / 1 pont

Melyik lineáris szoftverfejlesztési modell az alábbiak közül?

- Scrum
- Kanban
- Spiral (Spirális)
- Waterfall (Vízesés)

42. kérdés

1 / 1 pont

Melyik nem iteratív szoftverfejlesztési módszertan szerinti modell az alábbiak közül?

- Kanban
- Spiral (Spirális)
- Scrum
- Extreme Programming (XP)

43. kérdés

1 / 1 pont

Melyik állítás igaz a Scrum master-re?

- A Scrum mester nem felel azért, hogy külső hatásoktól védje a Scrum csapat munkáját.
- A Scrum mester vezeti a napi Scrumot.

A Scrum master a folyamatokért felel.

A Scrum master a Scrum csapat menedzsere.

Helytelen

44. kérdés

0 / 1 pont

Melyik agilis szoftverfejlesztési módszertan szerinti modell az alábbiak közül?

Rapid application development (RAD)

V-modell

Rational Unified Process (RUP)

Lean

45. kérdés

1 / 1 pont

Melyik állítás igaz a napi Scrum-ra?

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.

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

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

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

Kvízeredmény: **39** az összesen elérhető 45 pontból