2. kvíz

1. kérdés 1 / 1 pont

Melyik a használati történet (user story) szerkezete?

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

3. kérdés 1 / 1 pont

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

- ( ) reláció
- ( ) funkció
- ( ) aktor
- (●) függőség

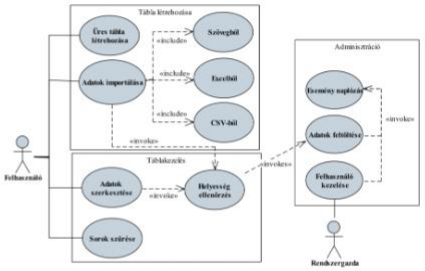
1. kérdés 1 / 1 pont

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

- (●) aggregáció (aggregation)
- ( ) általánosítás (generalization)
- ( ) kiterjesztés (extend)
- ( ) tartalmazás (include)

2. kérdés 1 / 1 pont

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



- ( ) Az adatok szöveges, Excel és CSV formátumú fájlokból importálhatóak.
- ( ) Üres tábla létrehozása esetén nem történik adatfeltöltés.
- (●) Amennyiben megszűrjük a sorokat, a tábla létrehozási funkciókat nem használhatjuk.
- ( ) Adatok importálása, valamint szerkesztése hatására is történik eseménynaplózás.

3. kérdés 1 / 1 pont

Melyik nem "nem funkcionális" követelmény?

- ( ) Külső követelmények
- ( ) Menedzselési követelmények
- (●) Szolgáltatások, reakciók leírása
- ( ) Termék követelmények

1. kérdés 1 / 1 pont

Mely reláció típus nem része a használati eset diagramnak?

- (●) Kompozíció
- ( ) Előfeltétel
- ( ) Tartalmazás
- ( ) Származtatás

1. kérdés 1 / 1 pont

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

- ( ) a vevők nem rendelkeznek informatikai ismeretekkel.
- ( ) a vevők nem egyértelműen fejtik ki az elvárásokat.
- ( ) a vevők bizonytalanok az elvárásokban.
- (●) a vevők a szoftver közvetlen felhasználói.

4. kérdés 1 / 1 pont

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

- ( ) követelmény validáció
- ( ) megvalósíthatósági elemzés
- (●) rendszer szerkezetének meghatározása
- ( ) követelmény feltárás és elemzés

3. kvíz

4. kérdés 1 / 1 pont

Hány alapelvet célszerű követnünk a SOLID elv szerint?

- (●) 5
- ( ) 6
- ( ) 3
- ( ) 4

1. kérdés 1 / 1 pont

Mi a szoftver architektúra?

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

5. kérdés 1 / 1 pont

Melyik nem része a rendszertervnek?

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

1. kérdés 1 / 1 pont

Mely architektúra esetében a legnehezebb elkülöníteni a program funkcióit?

- ( ) Model-nézet architektúra.
- ( ) Model-nézet-nézetmodell architektúra.
- (●) Monolitikus architektúra.
- ( ) Model-nézet-vezérlő architektúra.

4. kérdés 1 / 1 pont

Mi a *Single responsibility principle (SRP)* elv célja?

- (●) Minden komponens, osztály, metódus csak egy felelősségi körrel rendelkezzen, ami megváltoztatásának oka lehet.
- ( ) Minden metódus csak egyféle típusú kivétel kezeletlenül hagyását teheti lehetővé.
- ( ) Minden osztály reprezentációját egyetlen adattagban kell definiálni, egy komplex adatstruktúra létrehozásával.
- ( ) A felhasználói felület minden vezérlője csak egyetlen funkcióért felelhet, szoftverergonómiai megfontolásból.

2. kérdés 1 / 1 pont

Az alábbi SOLID elvek közül melyik van helyesen megfogalmazva?

- ( ) Open/closed principle (OCP): a programegységek nyitottak a módosításra, de zártak a kiterjesztésre
- ( ) Liskov substitution principle (LSP): az objektumok felcserélhetőek őstípusaik tetszőleges példányára a program viselkedésének befolyásolása nélkül
- ( ) Interface segregation principle (ISP): sok kisebb speciális interfész helyett kevesebb, de általános interfészt használjunk
- (●) Dependency inversion principle (DIP): függőségeket csak az absztrakciók között állítunk fel, és nem a konkrét megvalósítások között

2. kérdés 1 / 1 pont

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

- (●) a modell függ a nézettől, egy modellt mindig egy adott felülethez készítünk el
- ( ) a nézet tartalmazza a grafikus felhasználói felület megvalósítását, beleértve a vezérlőket és eseménykezelőket
- ( ) a modell tartalmazza a háttérben futó logikát, azaz a tevékenységek végrehajtását, az állapotkezelést, valamint az adatkezelést, ezt nevezzük alkalmazáslogikának, vagy üzleti logikának
- ( ) a felhasználó a nézettel kommunikál, a modell és a nézet egymással

1. kérdés 1 / 1 pont

Melyik **nem** UML csomagdiagrammokban alkalmazható reláció?

- ( ) import
- ( ) use
- (●) provide
- ( ) merge

4. kvíz

1. 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 kommunikáció diagram (communications diagram) célja az objektumok közötti kommunikációban felhasznált adatok sorrendjének megállapítása
- 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 tevékenység diagram (activity diagram) célja, hogy a végrehajtás lefolyását a tevékenységek és a tevékenységekben felhasznált adatok szempontjából közelítse meg**

2. kérdés

1 / 1 pont

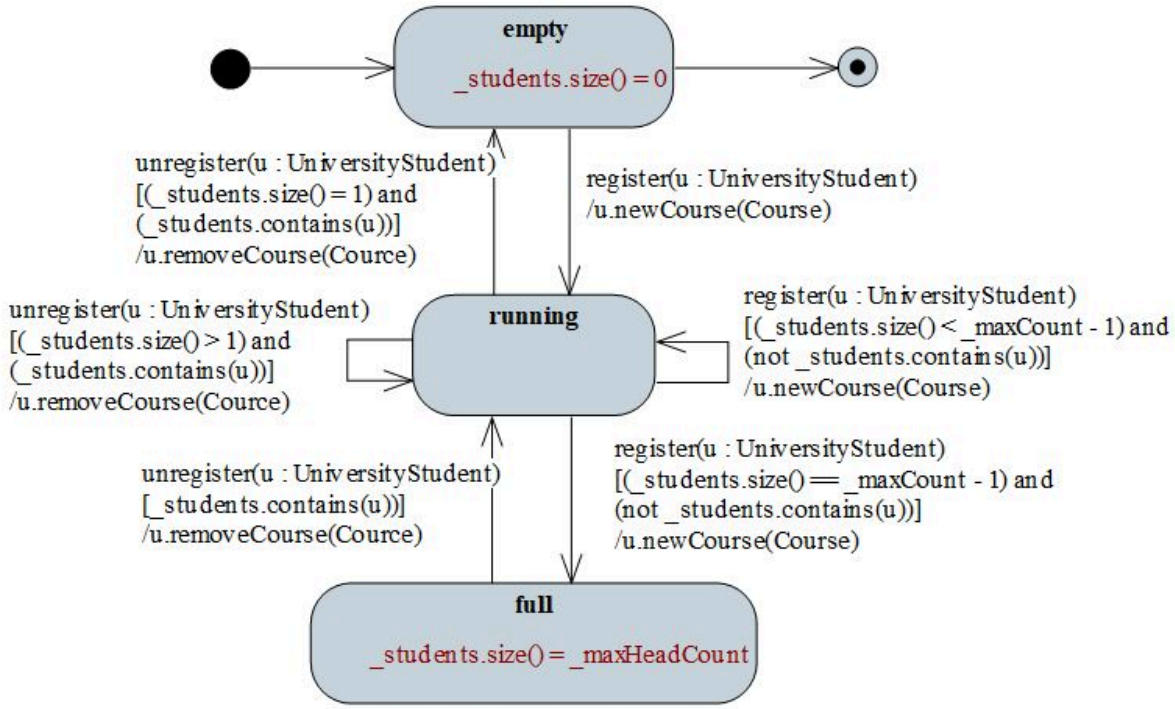
Melyik helyes?

- Az UML kommunikáció diagram (communications diagram) célja az objektumok közötti kommunikáció sorrendjének megállapítása
- Az UML szekvencia diagram (sequence diagram) célja az objektumok közötti kommunikáció sorrendjének megállapítása
- Az UML tevékenység diagram (activity diagram) célja az objektumok közötti kommunikáció sorrendjének megállapítása

3. kérdés

1 / 1 pont

Az alábbi állapot diagramra melyik állítás igaz?



- A hallgató mérete megegyezhet a fejméretével.
- Az állapot átmenet feltétele lehet a newCourse metódus meghívása
- A UniversityStudent objektum removeCourse metódusa nem kerülhet meghívásra
- A UniversityStudent objektum newCourse metódusa meghívásra kerülhet

4. kérdés

1 / 1 pont

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

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

5. kérdés

1 / 1 pont

Melyik helyes?

- Az UML tevékenység diagram (activity diagram) célja az objektumok közötti interakció időrendi ábrázolása
- Az UML szekvencia diagram (sequence diagram) célja az objektumok közötti interakció időrendi ábrázolása

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

Kvízeredmény: **5** az összesen elérhető 5 pontból

5. kvíz

3. kérdés — 1 / 1 pont

Melyik **nem** projektvezető szolgáltatás?

- GitLab
- Redmine
- Azure Devops
- GitHub

4. kérdés — 1 / 1 pont

Mik a centralizált verziókövető rendszerek hátrányai?

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

5. kérdés — 1 / 1 pont

Melyik **nem** funkciója a projektmenedzsment eszközöknek?

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

4. kérdés — 1 / 1 pont

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

- Azokat a változtatásokat, amelyeket már egy új verzióban rögzítettünk (*commit*), de nem küldtük be a távoli tárolóra (*push*).
- Azokat a változtatásokat, amelyeket tesztelési célból egy külön fejlesztési ágra küldtünk be.
- Azokat a változtatásokat, amelyeket már verziókezelés alá vontunk, de még nem mentettük el egy új verzióba (*commit*).
- Azokat a változtatásokat a helyi munkakönyvtárban, amelyeket még nem vontunk verziókövetés alá.

5. kérdés — 1 / 1 pont

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

- git commit
- git synchronize
- git push
- git pull

1. kérdés

1 / 1 pont

Melyik **nem** projektvezető szolgáltatás?

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

1. kérdés

1 / 1 pont

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

- (•) A futam lezárásához kapcsolódó megbeszélés
- ( ) Dokumentációs feladat
- ( ) Hiba
- ( ) Új funkcionalitás

2. kérdés

1 / 1 pont

Mely állítás vonatkozik az elosztott verziókövető rendszerekre?

- (•) A kommunikáció peer to peer elven történik, de kitüntetett szerverek felállítására van lehetőség.
- ( ) Ismert megvalósításai pl.: CVS, SVN, SourceSafe
- ( ) Fájl alapú műveletvégzést végez.
- ( ) A konkurenciakezelés jellemzően beküldés előtti egyesítéssel történik.

4. kérdés

1 / 1 pont

Mely állítás hamis a verziókövető rendszerekkel kapcsolatban?

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

3. kérdés

1 / 1 pont

Melyik állítás **igaz** az alábbiak közül a Git verziókövető rendszerre?

- ( ) A `.gitignore` fájlban azt adhatjuk meg, hogy mely állományokat nem szeretnénk a távoli tárolóról letölteni.
- (•) A Git szétválasztja a verziókezelési és a hálózati műveleteket.
- ( ) A Git elosztott verziókövető rendszer, ezért minden kliensnél csak a verziótörténet egy része található meg.
- ( ) A Git a beküldés előtti egyesítés konkurenciakezelési módszert alkalmazza (*merge before commit*).

6. kvíz

1. kérdés

1 / 1 pont

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

- Program lefordítása
- A megváltozott projekt fájlok automatikus feltöltése a verziókezelőbe.
- Függőségek kezelése
- Automatizált tesztek végrehajtása

2. kérdés

1 / 1 pont

Mi a build rendszerek elsődleges célja?

- A forráskód fordíthatóvá tétele continuous integration (CI) környezetben
- A forráskód fordításának a definiált szabályok szerinti automatizálása.
- A forráskód fordítása konzolos eszközökkel, ha integrált fejlesztőkörnyezet (IDE) nem áll rendelkezésre.

A forráskód felosztása fordítási egységekre.

3. kérdés

1 / 1 pont

Mi a .NET Standard?

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

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

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

A .NET Framework standard library-je.

4. kérdés

1 / 1 pont

Melyik nem része egy szabálynak a Make build rendszerben?

változók

cél

- függőségek
- utasítások

5. kérdés

1 / 1 pont

Melyik állítás **hamis** a GNU Make build rendszerre vonatkozóan?

- A hamis (*phony*) célok azokat a célokat jelölik, amelyek fájlként nem fognak előállni a szabály eredményeként.
- Képes külső függőségek (szoftverkönyvtárak) automatikus betöltésére.
- A Make egy önálló (*standalone*) build rendszer.
- Csak a frissített függőségekhez tartozó célokat generálja újra.

Kvízeredmény: **5** az összesen elérhető 5 pontból

7. kvíz

Az alábbiak közül melyik egy Lehman törvény?

- egy szoftvernek változnia kell, hogy folyamatosan csökkenjen a használhatósága és minősége
- egy szoftvert folyamatosan használni kell, vagy különben folyamatosan csökken a használhatósága és minősége
- **egy szoftvernek változnia kell, vagy különben folyamatosan csökken a használhatósága és minősége**
- egy szoftvert folyamatosan használni kell, hogy folyamatosan nőjön a használhatósága és minősége

2. kérdés

1 / 1 pont

Mely állítás igaz?

- A füst tesztet a tápegységből felszálló füst mennyiségének mérésével végzik.
- **A felhasználói teszt jellemzően fekete doboz tesztekből áll.**

- A kiadás tesztet a fejlesztő csapat végzi.
- A fejlesztői teszt jellemzően fekete doboz tesztekből áll.

3. kérdés

1 / 1 pont

Mi a tesztelés helyes sorrendje?

- kiadásteszt, egységteszt, felhasználói teszt, rendszerteszt, integrációs teszt
- integrációs teszt, felhasználói teszt, kiadásteszt, egységteszt, rendszerteszt
- egységteszt, integrációs teszt, felhasználói teszt, rendszerteszt, kiadásteszt
- egységteszt, integrációs teszt, rendszerteszt, kiadásteszt, felhasználói teszt

4. kérdés

1 / 1 pont

A tesztelés...

- ...garantálja, hogy a program minden körülmény között helytáll.

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

5. kérdés

1 / 1 pont

Melyik **nem** fejlesztői teszt?

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

Kvízeredmény: **5** az összesen elérhető 5 pontból

8. kvíz

1. kérdés

1 / 1 pont

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

- A containerek saját, elkülönített, virtualizált környezetben futnak.
- A containerekben futó alkalmazások belső hálózati kapcsolaton kommunikálhatnak egymással.
- **A containerek futó alkalmazások annak saját virtuális operációs rendszerén (pl. Docker OS) futnak.**
- Minden container osztozik a gazda számítógép hardveres erőforrásain.

2. kérdés

1 / 1 pont

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

- Az elbukott integrációs tesztek automatikus újra futtatása, ameddig meg nem javulnak.
- Objektum orientált programozási nyelvre való átállást segíti elő.
- A manuális tesztelés teljes kiváltása.
- **A lehetséges hibák, integrációs problémák azonnali, automatizált kiszűrése, visszajelzés a fejlesztőnek.**

3. kérdés

1 / 1 pont

Mi a folyamatos teljesítés (continuous delivery) célja?

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

4. kérdés

1 / 1 pont

Hogy hívjuk a folyamatos integráció és teljesítés egymásra épülő feladatait?

- **job**
- milestone
- module
- task

5. kérdés

1 / 1 pont

Milyen nyelven írható le a GitLab CI/CD konfigurációja?

- **YAML**
- XAML
- XML
- JSON

9. kvíz

1. kérdés

1 / 1 pont

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

- [ ] új funkciók bevezetése
- [ ] hibajavítások
- [ ] követelmény változások
- [x] kód minőségének javítása

2. kérdés

1 / 1 pont

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

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

3. kérdés

1 / 1 pont

Mely tulajdonságok jellemzőek a Clean Code-ra?

- Jól dokumentált, tesztelt, elegáns
- Olvasható, karbantartható, tesztelhető, elegáns
- Könnyen olvasható, nem tartalmaz kódismétlést, tesztelhető
- Olvasható, tömör, öndokumentáló

4. kérdés

1 / 1 pont

Melyik fajta kommentet kerüljük?

- Következményre figyelmeztető komment
- Kikommentezett kód
- Szándékot, pontosítást tartalmazó komment

- TODO

5. kérdés

1 / 1 pont

Melyik koncepció része a Clean Code-nak?

- Ugyanazt a nevet ne használjuk különböző célra
- Rövidítsük mindig a változó neveket
- Használjunk prefixeket az elnevezéseknél
- A break és continue utasításokat elővigyázatosan kell alkalmaznunk.

Kvízeredmény: **5** az összesen elérhető 5 pontból

10. előadás kvíz

1. kérdés

1 / 1 pont

Melyik tervmintát soroltuk **rossz** osztályba?

- Egyke - Létrehozási minta
- Felelősséglánc - Viselkedési minta
- Építő - Létrehozási minta
- Pehelysúlyú - Viselkedési minta

1. kérdés

1 / 1 pont

Melyik **NEM LÉTEZŐ** tervminta osztály?

- Végrehajtási
- Szerkezeti
- Létrehozási
- Viselkedési

2. kérdés

1 / 1 pont

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

- Observer (Megfigyelő)

- Builder (Építő)
- Factory method (Gyártó függvény)
- Command (Parancs)

3. kérdés

1 / 1 pont

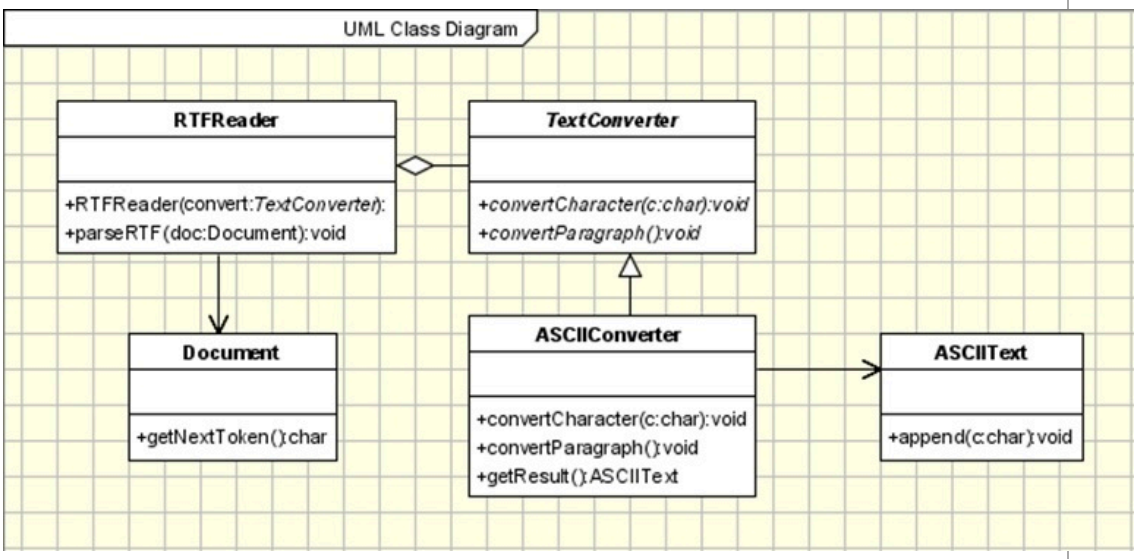
Melyik tervezési mintát alkalmazhatjuk abban az esetben, ha konkrét osztály megadása nélkül szeretnénk kapcsolódó vagy egymástól függő objektumok családjának létrehozására felületet biztosítani?

- Factory method (Gyártó függvény)
- Abstract Factory (Absztrakt gyár)
- Builder (Építő)
- Adapter (Illesztő)

4. kérdés

1 / 1 pont

Melyik tervezési minta alkalmazása látszik a képen?



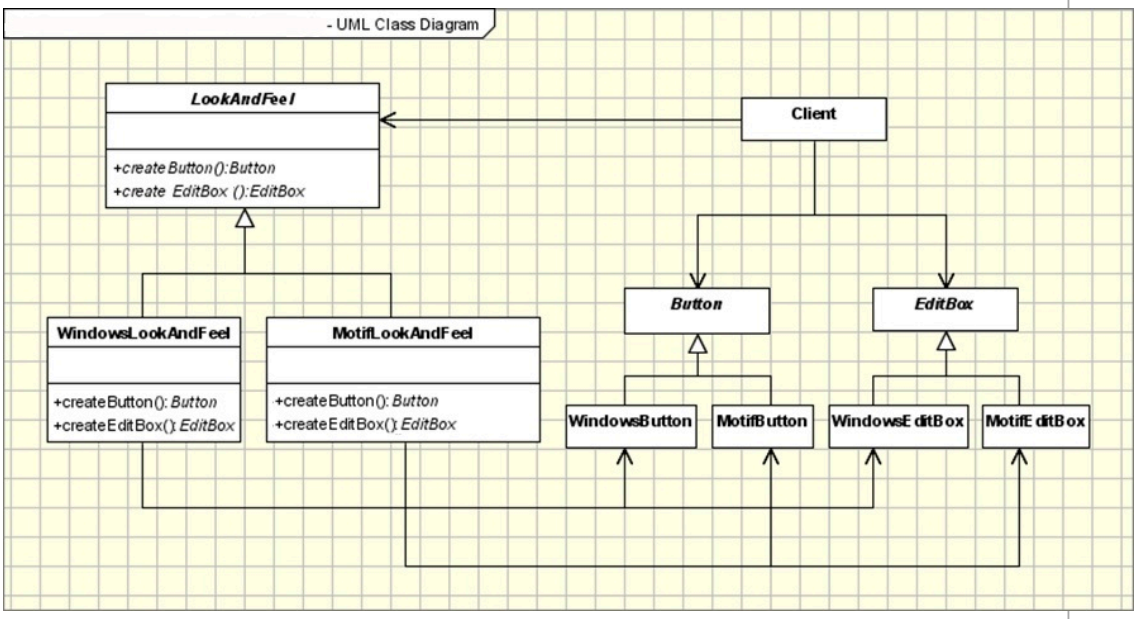
- híd
- építő

- bejáró
- homlokzat

5. kérdés

1 / 1 pont

Melyik tervezési minta alkalmazása látszik a képen?



- (●) elvont gyár
- megjelenítő
- gyártófüggvény
- díszítő

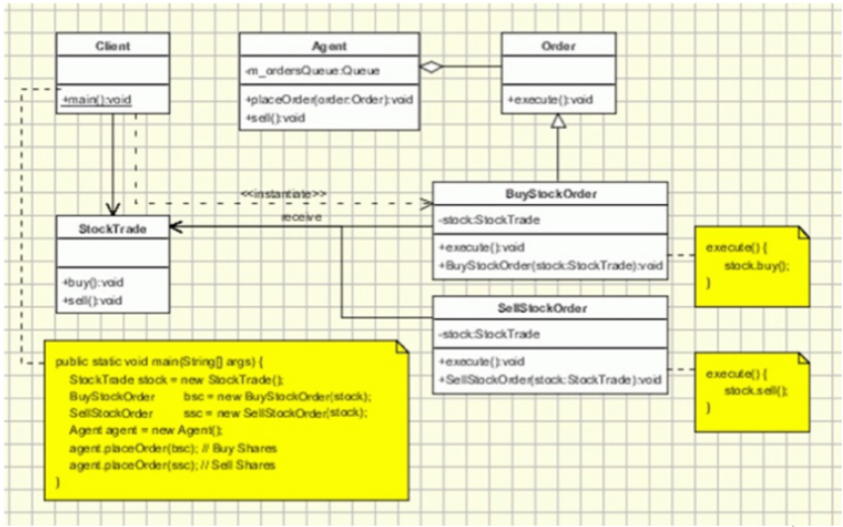
Kvízeredmény: **5** az összesen elérhető 5 pontból

11. előadás kvíz

2. kérdés

1 / 1 pont

Melyik tervezési minta alkalmazása látszik a képen?



- emlékeztető
- parancs
- közvetítő
- bróker

1. kérdés

1 / 1 pont

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

- Híd (Bridge)
- Felelősséglánc (Chain of Responsibility)
- Közvetítő (Mediator)
- Építő (Builder)

2. kérdés

1 / 1 pont

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

- Singleton (Egyke)
- Adapter (Illesztő)
- Observer (Megfigyelő)
- Factory (Gyártó)

3. kérdés

1 / 1 pont

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

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

4. kérdés

1 / 1 pont

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

- Összetétel (Composite)
- Helyettes (Proxy)
- Illesztő (Adapter)
- Pehelysúlyú (Flyweight)

2. kérdés

1 / 1 pont

Melyik tervezési minta alkalmazható a hosszú paraméterlistájú konstruktorok elkerülésére?

- Factory (Gyártó)
- Observer (Megfigyelő)
- Command (Parancs)
- Builder (Építő)

5. kérdés

1 / 1 pont

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

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

4. kérdés

1 / 1 pont

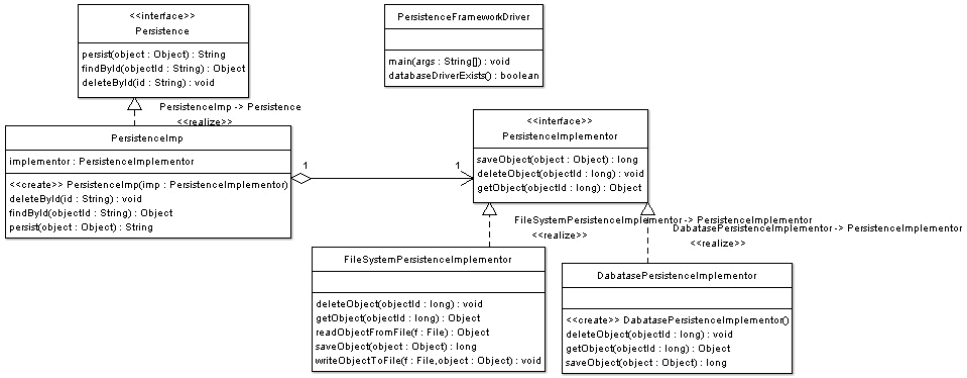
Mely tervminta fordítja le egy osztály interfészét egy kompatibilis másik interfészre?

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

5. kérdés

1 / 1 pont

Melyik tervezési minta alkalmazása látszik a képen?



- híd
- homlokzat
- perzisztencia
- közvetítő

Kvízeredmény: **5** az összesen elérhető 5 pontból

12. előadás kvíz

1. kérdés

1 / 1 pont

Melyik állítás **hamis** az `std::future<T>` típusra?

- Az `std::future<T>` típusú típus alkalmas kivételek átadására is szálak között.
- Egy `std::future<T>` típusú objektumba beleírható egy másik szálon majd kiolvasható érték.
- Az `std::future<T>` típus objektum nem feltétlen aszinkron végrehajtást jelent.
- Egy `std::future<T>` típus objektum egy ígéret arra, hogy egy T típusú eredmény előállításra kerül majd.

2. kérdés

1 / 1 pont

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

- Jelentése, a biztonságos memóriakezelés azáltal, hogy a hatókörén kívül került objektumokat a szemétgyűjtő felszabadítja (*garbage collection*).
- 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.
- Jelentése, hogy a kezelendő erőforrás az osztály invariánsa: annak lefoglalására / megnyitására az objektumok létrehozásakor, a felszabadítására / bezárására az objektum megsemmisítésekor kell sor kerüljön.

Jelentése, hogy minden objektum példányosítás megfeleltethető egy erőforrás foglalásnak.

3. kérdés

1 / 1 pont

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

Egy szál több folyamatot is tartalmazhat.

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

Nincs különbség, a kettő egymás szinonimája.

A folyamatokat Linux operációs rendszeren szálaknak hívjuk.

4. kérdés

1 / 1 pont

Mi a kiéheztetés (*starvation*)?

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

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

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

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

5. kérdés

1 / 1 pont

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

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,

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*).

Kvízeredmény: **5** az összesen elérhető 5 pontból

13. előadás kvíz

1. kérdés

1 / 1 pont

Melyik állítás igaz a napi Scrum-ra?

- A napi Scrum során az a cél, hogy felszámoljuk a csapatot érintő akadályokat.
- 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-megbeszélésen bárki részt vehet és beszélhet a Scrum Master-rel jelentkezéses alapon.
- A napi Scrum-megbeszélés célja, hogy a Scrum csapat tagjai összehangolják tevékenységüket. A megbeszélés ideje maximum 15 percre korlátozott.

2. kérdés

1 / 1 pont

Melyik nem agilis szoftverfejlesztési módszertan szerinti modell az alábbiak közül?

- Lean
- Kanban

- Rational Unified Process (RUP)
- Scrum

3. kérdés

1 / 1 pont

Melyik állítás igaz a Scrum master-re?

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

4. kérdés

1 / 1 pont

A három Scrum-termék (artifacts) a következő:

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

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

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

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

4. kérdés

1 / 1 pont

Melyik lineáris szoftverfejlesztési modell az alábbiak közül?

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

5. kérdés

1 / 1 pont

Melyik **nem** tartozik az előadáson felsorolt SCRUM folyamat elemek közé?

- futam
- bemutató
- bemutató tervezés
- visszatekintés

4. kérdés

1 / 1 pont

Melyik nem iteratív szoftverfejlesztési módszertan szerinti modell az alábbiak közül?

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