1. Mi az objektum?

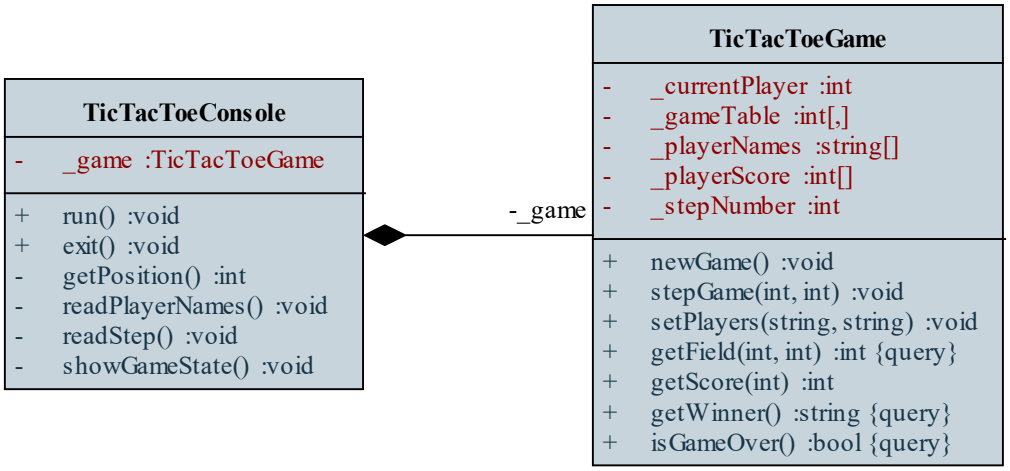
☐ **A** Egy olyan változó, vagy konstans, amely a program tetszőleges pontján felhasználható.

☐ **B** Egy olyan típus, amelyet a programozó valósít meg korábbi objektumokra alapozva.

☐ **C** Egy olyan változó, amely összetett szerkezettel rendelkezik, így többféle adatot is eltárolhat.

☐ **D** Egy olyan programegység, amely rendelkezik saját műveleteivel és adataival.

2. Melyik állítás helytálló a következő diagramra?



☐ **A** A `TicTacToeGame` kompozíciós szerepe a `_game`.

☐ **B** Az `exit()` művelet csak egy harmadik osztályból (vagy a főprogramból) hívható meg.

☐ **C** A `_playerNames` tömb 2 elemű, és a játékosok neveit tartalmazza.

☐ **D** A `TicTacToeGame` osztály meghívhatja a `readStep()` metódust.

3. Mit jelent a teszt vezérelt fejlesztés (TDD)?

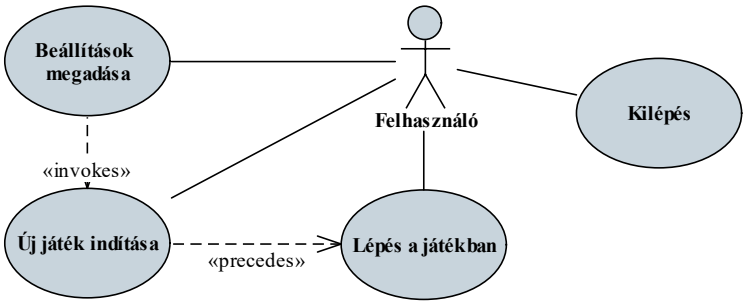
☐ **A** Szoftverfejlesztési módszer, amelyben a teszteket a tényleges programkód elkészítése előtt írják meg.

☐ **B** Tesztelési módszer, amelynek célja, hogy az tesztesetek minden programegységre kiterjedjenek, és megfelelő sorrendben hajtódjanak végre.

☐ **C** Egy általános elv, amely kimondja, hogy a programkód minden utasítását ellenőrizni kell egységtesztek segítségével (100%-os kódlefedettség).

☐ **D** Tesztelési módszer, amelyben először egységteszteket készítenek az osztályokra (és metódusaikra), majd integrációs tesztekkel ellenőrzik az osztályok együttes viselkedését, végül rendszertesztekkel a teljes szoftvert viselkedését ellenőrzik.

4. Melyik állítás helytálló a következő diagramra?



☐ **A** A felhasználónak lehetősége van új játékot kezdeni, de csak miután a beállításokat megadta.

☐ **B** A felhasználó a beállítások megadásával automatikusan új játékot indít.

☐ **C** A felhasználónak külön nem szükséges beállításokat megadni, vagy új játékot indítania, azonnal léphet a játékban.

☐ **D** A felhasználó csak akkor léphet ki a programból, ha elkezdett egy játékot.

5. Mi a helyes sorrendje a követelményelemzésnek?

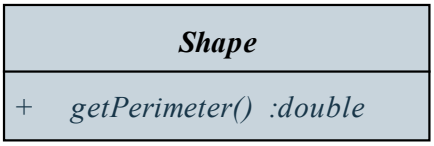
☐ **A** megvalósíthatósági elemzés, követelmény feltárás, követelmény specifikáció, követelmény validáció

☐ **B** követelmény feltárás, követelmény specifikáció, követelmény validáció, megvalósíthatósági elemzés

☐ **C** követelmény feltárás, követelmény validáció, követelmény specifikáció, megvalósíthatósági elemzés

☐ **D** követelmény feltárás, követelmény specifikáció, megvalósíthatósági elemzés, követelmény validáció

6. Melyik C++ kódrészlet felel meg a következő diagramnak?



☐ **A** `virtual double getPerimeneter() { … }`

☐ **B** `virtual double getPerimeneter() = 0;`

☐ **C** `double getPerimeneter() const { … }`

☐ **D** `double getPerimeneter();`

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

☐ **A** USER felhasználó IN USE CASE használati eset WITH RELATION kapcsolat

☐ **B** AS A szerepkör USE funkció TO cél

☐ **C** WHEN tevékenység APPLYING funkció IN ORDER TO cél

☐ **D** GIVEN környezet WHEN tevékenység THEN hatás

8. Melyek az UML használati eset (use case) diagram relációi?

☐ **A** függőség (dependency), kompozíció (composition), használat (usage), beágyazás (nesting)

☐ **B** előfeltétel (precedes), tartalmazás (include), használat (usage), általánosítás (generalization)

☐ **C** használat (usage), beágyazás (nesting), importálás (import), függőség (dependency)

☐ **D** felület (interface), megvalósítás (implementation), tartalmazás (include)

9. Melyik nem szoftverfejlesztési modell?

☐ A vízesés (waterfall)

☐ B továbbfejlesztés (evolution)

☐ C scrum

☐ D prototipizálás (prototyping)

10. Mely funkciót nem tudják biztosítani a teszt keretrendszerek (unit testing frameworks)?

☐ A Implementálhatunk teszteseteket, amelyeket egyenként, vagy egyben is futtathatunk.

☐ B Ha a tesztesetek mind sikerrel zárulnak, a program hibátlannak tekinthető.

☐ C A tesztekben a kapott eredményt összehasonlítjuk egy elvárt eredménnyel assert utasítások segítségével.

☐ D A tesztelés végeztével előáll egy tesztjelentés, amelyben látható, hogy mely tesztek lettek sikeresek/sikertelenek.

11. Mi a helyes tartalmazási sorrendje az alábbi, osztályok közötti relációknak (ahol előbbi tartalmazza az utóbbit)?

☐ **A** függőség, asszociáció, kompozíció

☐ **B** asszociáció, függőség, kompozíció

☐ **C** kompozíció, függőség, asszociáció

☐ **D** asszociáció, kompozíció, függőség

12. Mire szolgál az UML telepítési (deployment) diagram?

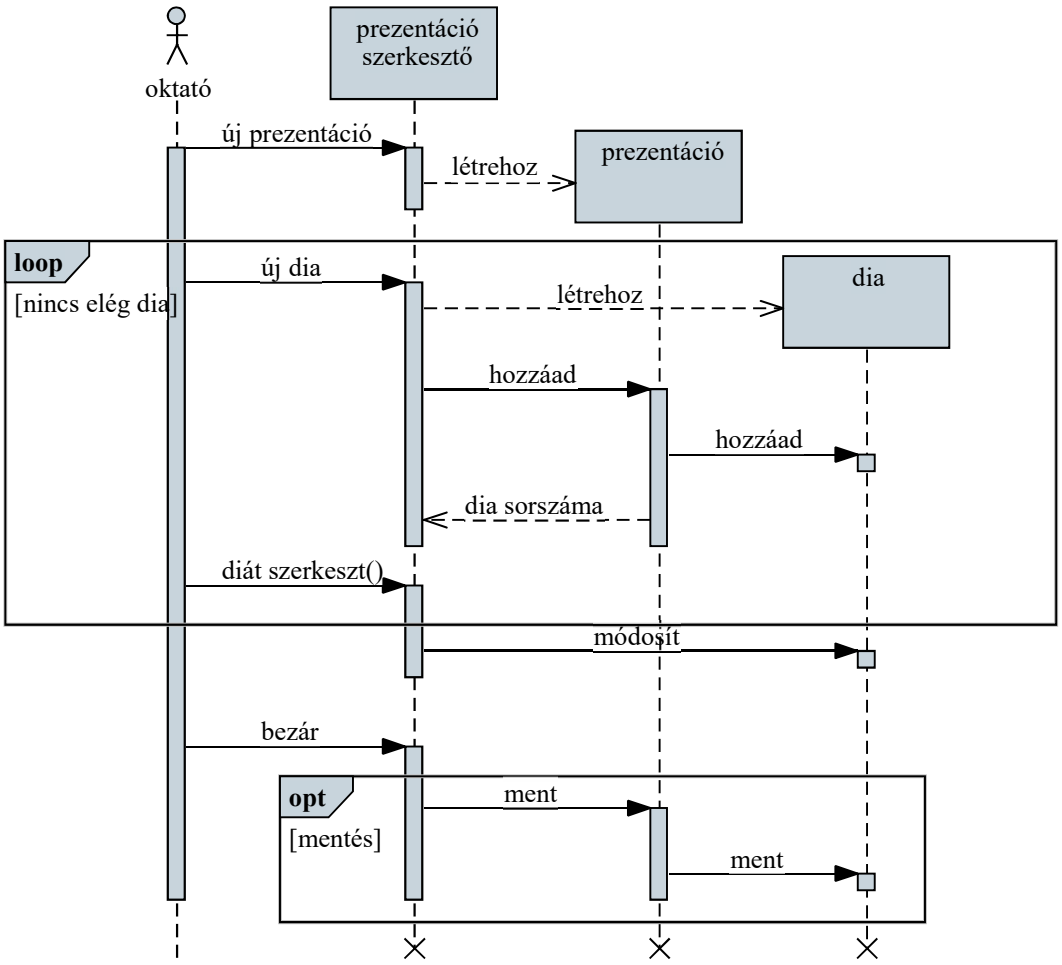
☐ **A** Ábrázolja azt a műveletsorozatot, amely a szoftver adott gépen történő üzembehelyezéséhez szükséges.

☐ **B** Ábrázolja az összes hibalehetőséget, amellyel a telepítés során találkozhatunk.

☐ **C** Ábrázolja a szoftver összetevőket, annak megfelelően, miként kell őket telepítőcsomagba helyezni.

☐ **D** Ábrázolja a szoftver összetevőinek fizikai (különböző gépeken történő) elhelyezését, a szükséges szoftverkörnyezettel.

13. Melyik állítás nem helytálló a következő diagramra?



☐ **A** Ha új diát hozunk létre a szerkesztőben, akkor azzal létrejön egy dia objektum.

☐ **B** Bezáráskor mindenképpen el lesznek mentve a diák.

☐ **C** A dia szerkesztéséhez nem vesszük igénybe a prezentáció objektumot.

☐ **D** A dia sorszámát a prezentáció objektum határozza meg hozzáadáskor.

14. A három rétegű (tri-tier) architektúrában mi a célja a modell (üzleti logika) rétegnek?

☐ **A** Leképezi a modellezett tartalmakat ténylegesen megjeleníthető információkra, pl. képekre.

☐ **B** Az üzleti funkciókat biztosítja az alkalmazásnak, pl. hitelkártya tranzakciók kezelése.

☐ **C** A program lényegi funkcionalitását látja el, a tevékenységek végrehajtását, valamint az állapotfenntartást biztosítja.

☐ **D** Felügyeli az adatok mentésének és betöltését valamilyen tárolóba, pl. szöveges fájl, adatbázis.

15. Mi a hard coding?

☐ **A** Olyan szoftverfejlesztési modell, amelyben az implementációs folyamat rövid, és rögzített időtartamokra van szabva, amelyek során a fejlesztők csak kis feladatokat valósítanak meg.

☐ **B** Egy fejlesztőkörnyezetek által nyújtott szolgáltatás, amely lehetővé teszi, hogy speciális jelölőket (pl. TODO, HACK) hagyjunk a kódunkban, és külön listázzuk őket.

☐ **C** Az UML modellek közvetlen generálása programkóddá, így a fejlesztőnek az osztályok definícióját már nem kell külön implementálnia.

☐ **D** Olyan kódolási forma, ahol konstansokat (pl. szám, szöveg) helyezünk el különböző helyeken a kódunkban, megnehezítve a karbantartást.