**4. Tiszta kód alapelvek - Tesztkérdések**

**1. Miért fontos a kód olvashatósága?**

* Csak a fejlesztőnek kell értenie
* **Könnyebb fenntartani és továbbfejleszteni** ✅
* Csökkenti a kódfájlok méretét

**2. Mit jelent a SOLID elvek első betűje, az "S"?**

* **Single Responsibility Principle** ✅
* Simple Code
* Structured Design

**3. Mit jelent a DRY (Don't Repeat Yourself) elv?**

* **Kerüld az ismétlődő kódrészleteket** ✅
* A kód legyen rövid
* Írj minél több megjegyzést

**4. Mi a KISS (Keep It Simple, Stupid) elv lényege?**

* **A kód legyen minél egyszerűbb és érthetőbb** ✅
* Minél kevesebb változót használjunk
* Ne dokumentáljuk a kódot

**5. Mit jelent a YAGNI elv?**

* **Ne fejlesszünk felesleges funkciókat előre** ✅
* Mindig használjunk design patternöket
* Írjunk minél több unit tesztet

**6. Mi a célja a refaktorálásnak?**

* **A kód minőségének javítása anélkül, hogy megváltozna a működése** ✅
* Új funkciók hozzáadása
* A kód hosszának csökkentése

**7. Melyik nem tartozik a SOLID elvek közé?**

* Single Responsibility Principle
* Liskov Substitution Principle
* **Dependency Reduction Principle** ✅

**8. Mit jelent az egységfelelősség elve (Single Responsibility Principle)?**

* Egy osztály több dolgot is csinálhat
* **Egy osztálynak csak egyetlen felelőssége lehet** ✅
* Egy metódusnak csak egy sorból kell állnia

**9. Miért fontos a függőségek minimalizálása?**

* **Csökkenti a kódtörések esélyét és könnyebb karbantartani** ✅
* Növeli a kód végrehajtási sebességét
* Megakadályozza a memóriahasználat növekedését

**10. Mit jelent az LSP (Liskov Substitution Principle)?**

* **Egy alosztály helyettesítheti a szülőosztályt anélkül, hogy a program viselkedése megváltozna** ✅
* Minden osztály csak egy metódust tartalmazhat
* Minden osztályt absztraktként kell definiálni

**11. Mi az Interface Segregation Principle (ISP) fő elve?**

* **Egy osztály csak azokat a metódusokat kell, hogy implementálja, amelyeket valóban használ** ✅
* Egy osztály minden metódust meg kell, hogy örököljön
* Minden interfésznek egyetlen metódusa lehet

**12. Mi a Dependency Inversion Principle (DIP) lényege?**

* Az alacsony szintű modulok függenek a magas szintű moduloktól
* **A magas szintű modulok ne függjenek közvetlenül az alacsony szintűektől** ✅
* Minden osztályt absztraktként kell megvalósítani

**13. Melyik elv segít elkerülni az óriási (God Object) osztályokat?**

* **Single Responsibility Principle** ✅
* Liskov Substitution Principle
* Open/Closed Principle

**14. Mi a kód szagmentessé tételének célja?**

* **A kódban lévő rossz tervezési döntések kiküszöbölése** ✅
* A kommentek számának növelése
* Az összes változó globálissá tétele

**15. Mi a Magic Number probléma?**

* **A kódban keménykódolt számok használata, amelyek jelentése nem egyértelmű** ✅
* A kód túl sok változót használ
* Az osztályok túl hosszúak

**16. Melyik az alábbiak közül egy rossz kódolási gyakorlat?**

* **Hosszú, több funkciót ellátó metódusok írása** ✅
* Kód dokumentálása
* Moduláris programozás

**17. Mit jelent az Open/Closed Principle (OCP)?**

* **Az osztályok nyitottak kell legyenek a bővítésre, de zártak a módosításra** ✅
* Egy osztály csak egy metódust tartalmazhat
* Egy osztály csak egy interfészt valósíthat meg

**18. Hogyan lehet egy kódot könnyebben érthetővé tenni?**

* **Beszédes változónevekkel és megfelelő formázással** ✅
* Rövid változónevekkel (a, b, c)
* Egyetlen hosszú metódussal

**19. Milyen esetben kell refaktorálni a kódot?**

* Ha a program gyorsan fut
* **Ha a kód nehezen érthető és fenntartható** ✅
* Ha a projekt már befejeződött

**20. Mi a különbség a refaktorálás és az újraírás között?**

* A refaktorálás törli a régi kódot
* **A refaktorálás megtartja a funkcionalitást, az újraírás teljesen új kódot jelent** ✅
* Nincs különbség közöttük