Delegáltak

Delegate

- Olyan típus, aminek példányaiban metódusokat lehet elhelyezni.
- Amilyen típusú a delegált, pont ugyanolyan típusú metódusok tehetők bele.
- Meghívása mint egy metódusé.

Multicast delegate

- C#-ban a delegate-ek multicast tulajdonságúak.

Delegate null ellenőrzés

- myDelegate != null
- myDelegate?.Invoke()

Több metódus meghívása

- Ha nincs visszatérési érték, akkor lefut minden feliratkoztatott metódus.
- Ha van visszatérési érték, akkor az utolsó lefuttatott metódus eredményét menti el.
    - Hívási sorrend nem garantált.

Probléma a delegáltakkal

- Kívülről írhatóvá kell tenni, hogy lehessen beletenni metódust.
    - Így is kivehető belőle másik.
- Megoldás:
    - Saját feliratkoztató és leiratkoztató metódus.
        - Nehéz/Körülményes

Eseménykezelés

Event

- event kulcsszóval ellátott delegált példány
- Előnyei:
    - Csak a deklaráló osztályból lehet meghívni/elsütni.
        - Delegáltat bárhonnan.
    - Csak += és -= operátor megengedett kívülről publikus esetben is.
        - Delegáltál felülírható a teljes példány.
    - Korlátozott képességű tulajdonság készíthető add és remove kulcsszavakkal.
        - Delegáltnál get és set kulcsszavak vannak
    - Szerepelhet interface-ben
        - Delegált nem írható elő.

Névtelen függvények

- Nincs neve, amivel máshol hivatkozhatunk rá.
- Nem lehet máshonnan elérni.
- Egy helyben kifejtés.
- Nem local/inline függvények.

Lambda kifejezések

- Névtelen függvényt jobban lehet egyszerűsíteni.
- Paraméterek => kimenetet meghatározó kifejezés (a nyíl a lambda operátor)
- Ha több paraméter van, akkor zárójelbe kell rakni őket.
- Nem kell a paraméter típusát kiírni.
- Return kulcsszó kell, ha több ág van a kifejezésben.

Altípusok

- Kifejezéslambda (Expression lambda)
    - Egyetlen kifejezés, ami meghatározza a kimenetet.
    - Delegált helyett kifejezésfára is fordulhat.
- Kifejezéslambda (Statement lambda)
    - Többsoros kód (kapcsos zárójelek között)
    - Más függvény hívás
    - Return kulcsszó

Outer Variable Trap

- A névtelen metódusok és lambdák belsejében felhasználhatók a külső változók, de a külső változókat a függvény referencia formájában kapja meg – az érték típusúakat is!

XML/JSON

XML

- Hierarchikus adatleíró formátum
- Elemekből (element/node) és attribútumokból (attribute) áll.
- Az Al-elemek és az attribútumok ugyanazok.
- Kötelező gyökérelemnek lennie.
- Egymásba ágyazás lezárásainak megfelelő sorrendben kell történnie és mindent le kell zárni.
- Kisbetű-nagybetű érzékeny.
    - Well-formed XML
        - W3C elvárásoknak megfelelő a formátuma.
    - Valid XML
        - Megadott sémának megfelel a formátuma.
- Kezelése
    - XDocument, XElement, XAttribute osztályok segítségével.
    - XElementnek van Name és Value tulajdonsága.
    - Node-ok egymásbaágyazhatóak az Add() metódussal.
    - XElement paraméterezése: XName name (string), object content (bármilyen obj)

XML elmentése

- XDocument
  - XML fájl reprezentáló objektum létrehozása.
- XElement root
  - Gyökérelem
- xdoc.Add()
  - Gyökérelem dokumentumához adása.
- XElement element
  - Egy al-elem elkészítése
- element.Add(new XElement())
  - Al-elem feltöltése al-elemekkel.
- root.add(element)
  - Al-elem gyökérelem alá-szúrása
- xdoc.Save(„file.xml”)
  - Dokumentum elmentése

XML bejárása

- new XDocument()
  - Üres objektum, feltöltjük és kimentjük
- XDocument.Load(uri)
  - Fájl elérési út vagy URL cím alapján betölt egy XML fájlt egy XDocument példányba és a példányt visszaadja.
- XDocument.Parse(text)
  - Ha az xml tartalom egy string változóba be van töltve, akkor abból elkészíti a feltöltött XDocument példány

XDocument és XElement metódusai

- Element(XName)
  - Visszaadja az ilyen nevű al-elementnek közül az elsőt (XElement).
- Elements()
  - Visszaadja az összes al-elementet (IEnumerable<XElement>)
- Elements(XName)
  - Visszaadja az ilyen nevű összes al-elementet (IEnumerable<XElement>)
- Descendants(XName)
  - Visszaadja az ilyen nevű összes al-elementet rekurzívan (IEnumerable<XElement>)
- Attribute(XName)
  - Visszaadja az ilyen nevű attribútumok közül az elsőt (XAttribute)
- Attributes(XName)
  - Visszaadja az ilyen nevű attribútumok közül az összeset (IEnumerable<XAttribute>)

JSON

- Javascript Object Notation
- Gyökérelem nem kötelező

Különbségek

- Nem kötelező gyökérelem
- Objektum-orientált megközelítéshez jobban passzol szerkezetre
- Nincs zárótag
- Tömböket is lehet benne használni.

LINQ

Language Integrated Queries

- Gyűjtemények forrás és struktúra független kezelése egyszerűen
    - Bármilyen adatforráson ugyanúgy
- Lambda kifejezések
- Névtelen osztály, var
- Kiegészítő metódusok
- LINQ operátorok

Var

- Nem kell a bal oldalra kiírni a típust, mert a fordító meg tudja határozni, hogy az objektum milyen típusú.
- Kötelező neki értéket adni!

Névtelen osztályok

- Egyszer használatos
- „Anonymus”
- Gyorsan tudunk kezdőértéket adni a tulajdonságoknak.

Kiegészítő metódusok

- Egy meglévő típushoz futásidőben új metódusok „ragasztása”.

Select

- Egy tulajdonság alapján új gyűjtemény.

SelectMany

- Egy összetett objektum simítható ki egy szimpla gyűjteménnyé.

LINQ láncolás

- Interfésszel tér vissza, általában IEnumerable<T>
- Deklaratív megközelítés = Query syntax

Aggregáló metódusok

- Sum
- Average

Csoportosítás

- Egy gyűjtemény széttördelése több gyűjteményre.
- Kategorizálás
- IEnumerable<IGrouping<TKey, TElement>>

DLL

- Bizonyos osztályok és metódusok kiszervezése egy olyan fájlba, ami nem forráskód, hanem már lefordított bináris kód.
- Értelme
    - Felhasználhatóság későbbre
    - .NET összes beépített típusát is így érjük el.

Fordítás és linkelés

- Statikus linkelés
    - Függőségek is belekerülnek a végső futtatható állományba.
    - Régen így készültek a programok, de a fájlméretük hatalmas volt.
    - Ugyanazt a függvényt több program is külön-külön tartalmazza, ez így pazarlás.
- Dinamikus linkelés
    - Futásidőben belehívunk a DLL-ben egy metódusba.
    - A betöltést az operációs rendszer végzi.
    - Ugyanazt a DLL-t több program is hívhatja (Shared Object)
        - .NET-ben AppDomain miatt nem mindig.
    - Legtöbb mai modern program így működik.
    - Frissítésnél DLL csere

Futtatható állomány típusok

- Futtatható kód és minden a futtatáshoz szükséges adat.
- Linux: ELF
    - Executable and Linkable Format
    - Kiterjesztés nélküli bináris futtatható állomány vagy SO fájl.
    - Extra funkció: fatELF
        - Több platformfüggő állomány egy nagy platformfüggetlen fájlban.
- Windows: PE
    - Portable Executable
    - Minden exe és DLL fájl ide tartozik.
    - DLL
        - Nincs belépési pontja.
    - EXE
        - Van belépési pontja.
    - Extra funkció
        - Ikonok

Unmanaged/managed állományok

- Unmanaged/natív állomány
    - OS/CPU számára értelmezhető bytekód
    - Közvetlen hardver elérés
    - Bármilyen nyelven írható
    - Platformfüggő
- Managed/felügyelt állomány
    - Egy vagy több osztály van benne
    - .NET értelmező tud vele dolgozni csak
    - Nyelvfüggetlen
    - Platformfüggetlen, de .NET értelmező kell hozzá.
        - .NET runtime

Natív DLL-ek

- Aktuális könyvtárból vagy a %PATH%-ból töltődnek be.
- Lassú betöltődés
- DLL HELL: Verziózás nem megoldott
    - Különféle alkalmazások, különféle verziót igényelnek.
    - DLL stomping a megoldás vagy minden DLL az exe mellett.
    - Linux megoldás
        - Fájl szintű csomagkezelő.
    - .NET megoldás
        - Global Assembly Cache = GAC
- Windows API
    - Natív DLL gyűjtemény
    - OS minden publikus funkcionalitása elérhető.
    - Háttérben WINAPI hívás
- Problémák
    - Platform és OS függő kód.
    - Paraméterek és visszatérési értékek típusai.
    - Ki, mit szabadít fel a memóriában.
    - Nem tudjuk mi érhető el a DLL-ben = dumpbin
    - Nem tudjuk mik a DLL függőségei = dependency walker
    - Csak futásidőben derül ki, hogy létezik-e a DLL és a metódusok.

Felügyelt DLL-ek

- Dependencies szekcióban tároljuk el.
    - CSPROJ fájl
- Fordító ellenőrzi a meglétét.
- Gyors betöltődés.
- .NET Assembly

Futtatási folyamat

1. EXE fájl futtatása.
2. PE formátum validálása, 32/64 bites Process készítése a header alapján.
3. Annak eldöntése, hogy az APP felügyelt/natív.
4. Ha felügyelt, akkor a megfelelő verziójú CLR betöltése (FW 4.6 vagy 5.0, stb).
5. App Domain létrehozása (sandboxolt közeg)
    a. EXE és DLL-ek egységbezárva.
6. Függőségek (assembly-k betöltése a Fusion komponenssel).
7. CLR meghívja a Main() metódust.

Reflexió

- A program önmaga struktúráját és viselkedését futásidőben analizálni és alakítani tudja.
    - Meta-adatok kinyerése.
- Jellemzői
    - Magasszintű nyelv kell hozzá.
    - C# használati módok:
        - Futásidejű típusanalízis
            - Milyen tulajdonságai vannak ennek az objektumnak?
            - Milyen értékei vannak?
        - Futásidejű típusgyártás
            - System.Reflection.Emit
- Intellisense
    - Metódusok, tulajdonságok listázása.
- Szerializáció
    - Tulajdonságok és hozzájuk tartozó értékek kinyerése/beállítása.
- Tesztek
    - Teszt metódus kigyűjtése egy vezérlőpultra.

Típusinformációk kinyerése

- Type típusba kapjuk vissza a jellemzőket.
    - typeof(DateTime)
- Egy generikus paraméterből is megkaphatjuk a futásidőben behelyettesített típus infóit.
- Egy objektumból is megkaphatjuk a típusinfókat
    - Gyakoribb használati eset.
- Stringként is beírható a típus neve, de akkor teljes névtér és a típusnév kell.
- A jelenleg futtatott Assembly-ben lévő is elérhetjük.

PropertyInfo/FieldInfo

- Ha rendelkezésre áll egy PropertyInfo, rajta keresztül lekérhető egy adott objektum ezen PropertyInfo-jú tulajdonságához tartozó konkrét érték.
- GetProperty()
    - Tulajdonság lekérése
- GetValue()
    - Értéke lekérése
- SetValue()
    - Érték beállítása

MethodInfo

- Ha rendelkezésre áll egy MethodInfo, meghívhatjuk ezen metódusát valamilyen objektumnak.
- Statikus osztályoknál
    - Invoke első paramétere null.

CreateInstance, PropertyType

- CreateInstance
    - Adott típus példányosítása
- PropertyType
    - Kinyert tulajdonság/adattag típusának lekérése.

Assembly elérés

- DLL-ből típusok kigyűjtése futásidőben.
    - Jelenlegi Assembly-ből is lehet.
- Hasznossága:
    - DLC (játékokhoz)

Attribútumok

- Intelligens, programozható cetlik, amiket osztályokra, metódusokra, tagokra, stb tehetünk.
- Más nyelvekben:
    - Annotáció
- Célja
    - Információk, kikötések, intelligens kommentek
    - Adott elem működését nem befolyásolja
    - Kinyerhetjük minden tulajdonságra, stb, hogy van-e attribútuma és van-e benne adat.

Saját attribútumok írása

- Attribute őstől származik
- AttributeUssage
    - Hol lehessen használni?
- AllowMultiple
    - Többször is szerepelhessen.

Reflexió hátrányai

- Lassú futásidő.
    - Objektumok feldolgozása miatt.
    - Ott használjuk, ahol nem lehet másképp megoldani.
- Nem arra való, hogy a láthatóságokat megkerüljük vele.
- Gyorsabb megoldás
    - dynamic típus

Adatbázis

Probléma a fájlkezeléssel

- 1 felhasználó/1 szálú környezetben jó csak.
- Rossz teljesítmény.
- Nem biztosít
    - Tranzakciókezelést
    - Redundáns tárolást
    - Elosztott tárolási lehetőséget
- Mikor kell adatbázis?
    - Több szálú alkalmazásoknál
    - Egyik szálon fájl írás, másik szál nem fér hozzá
        - Exception
            - Megoldása, hogy random ideig várakozás, de így lassú a válaszidő.

Mi az Adatbázis?

- Szoftver, ami
    - Fel van telepítve a gépre/kiszolgálóra.
    - Adott porton válaszol.
    - Tárolási igények elküldése.
    - Adatok lekérése
- Az alkalmazások adattárolás/perzisztencia részét kiszervezzük egy programnak, ami
    - Párhuzamos kérések kezelésére képes.
    - Redundánsan tárolja az adatokat.
    - Tranzakcióként kezel minden kérést.
    - Rollback az előző jó állapotra, ha hiba van.

MSSQL

- Kis-és középvállalatok által használt relációs adatbázis-kezelő szoftver.
    - **SQL Express**: Kisebb változat, 10GB/adatbázis, 1 CPU, 1 GB RAM.
    - **LocalDB**: Szerverszolgáltatás helyett egy azonos funkcionalitást nyújtó könyvtár, ami .mdf fájlba dolgozik.
        - ConnectionString
        - Táblák visszaállnak az eredeti állapotra.
    - **InMemoryDatabase**
        - Memóriában vannak tárolva az adatok.
        - Teszteléshez jó.
        - Code-First megközelítés

Kapcsolódás

- ADO.NET: DbConnection technika
    - SQL utasítások küldése stringként.
    - Object tömb az eredmény
        - Kasztolni kell
    - Connected mód
- DataSet technika
    - GUI csak
    - Nem használt már
- Entity Framework Technika
    - SQL réteg fölé ORM réteget helyez
        - Object Relational Mapping = Rejtett fizikai adatelérés
    - Táblákat objektumgyűjteményként érjük el.
    - Connected/Disconnected mód
    - Database-First
        - Van már adatbázis
        - Adatbázis feltöltése és ConnectionString
    - Code-First
        - Nincs még adatbázis
        - ConnectionString

Optimalizálás

- Mindig minden szűrést/rendezést/al-lekérdezést bele kell írni a query-be.

Adatbázis szerverek

- Mező szerint szétosztott működés (Sharding)
- Több kiszolgáló a read műveletekre (Replica Set)
- Kulcs alapján hasítás
- Cachelés
- SSD, gyors memória
- Memóriában tartott táblák
- Minden feladatot az adatbázis-szerver végezzel el, mi csak az eredményeket jelenítsük meg.

Közvetlen SQL kommunikáció hátránya

- SQL injection
    - Jelszó nélkül is be tud lépni a támadó
    - 1=1, mindig igaz
- Üzleti logikai kódba kell helyezni az SQL kódot
- Valamilyen réteg kell az SQL fölé, ami elrejti az SQL kódot.
    - LINQ

ORM előnyök és hátrányok

- Előnyök
    - Nincs dialektusfüggő SQL utasítás a kódban.
    - Nincs string formátumú SQL utasítás a kódban.
    - SQL injection ellen védett.
    - Lekérdezés eredménye típusos gyűjtemény.
- Hátrányok
    - Nehezebb konfiguráció
    - Nehéz optimalizálni
    - Nagyobb memóriai és CPU igény

Attribútumok

- Key: Elsődleges kulcs
- DatabaseGenerated(DatabaseGeneratedOption.Identify): Automatikus növelés
- Required: Kötelező, nem lehet null.
- StringLength(240): Max string hossz
- Range(0, 10): Tartomány
- Column(„id”): Adatbázis mező neve
- ForeignKey(nameof(Director)): Idegen kulcs
- NotMapped: Ne kerüljön az adatbázisba, csak futásidőben van rá szükség.
- JsonIgnore: JSON szerializációnál ne kerüljön bele a JSON-be.

DbSeed

- Minta adatok
- Mindig felülírja az előzőleg bemásolt MDF fájlt.
- Mindig belekerülnek újra a seed adatok.
- Explicit meg kell adni az Id-t.
- Miért jó?
    - Nem éles adatbázis
    - Kiinduló pont
    - Véletlenül kitörlünk vmit, következő buildeléskor ott lesznek a seed adatok.

LazyLoading

- Minimalizálja az alkalmazás indítási idejét.
- Az alkalmazás kevesebb memóriát fogyaszt az igény szerinti betöltés miatt.
- Szükségtelen kérések elkerülhetők

Eager Loading

- Töltsön be minden szükséges x dolgot a „renderelés” előtt.

Adatbázis fizikai megvalósítások

- Publikus IP címmel vagy lokálisan elérhető SQL szerver.
    - ConnectionString
- LocalDb-vel MDF fájlok
    - MDF és LDF fájlok létrehozása (Bele lehet nézni a táblákba)
        - Csak Windows
- InMemoryDatabase
    - Memóriában dolgozik
    - Nem kell semmilyen fájl.
        - Nincs fizikai tárolás

Rétegek

Célja

- Az implementációs részletek elrejtése, hogy a felső réteg ne függjön a nem szomszédos alsó rétegektől.
- Minden réteg csak az alatta lévő réteget ismeri.

Tier

- Fizikai felbontása az alkalmazás komponenseknek.
    - Szoftver vagy hardverkomponensek, amik a rétegeket futtatják.
    - Nem feltétlenül azonos felbontású, mint a rétegek.

Layer

- Logikai felbontása az osztályoknak
    - Az egy rétegbe tartozó osztályok egy közös, magasabb rendű feladatot látnak el.
    - Egyértelmű, hogy mi érhető el kívülről.
    - Rétegek között interfészekkel teremtünk kapcsolatot.
    - Akkor jó egy alkalmazás, ha egy réteg cserélhető egy máshogy implementált, de azonos interfészekkel dolgozó másik komponensre.

Rétegzett/fenntartható megoldás

- Fenntartható
    - Könnyen továbbfejleszthető, átültethető más környezetbe.

SOLID elvek

- Single Responsibility
  - Egy osztály, egy felelősségi kör elve
  - Ne legyen olyan osztály, ami adatot kezel, megjelenít, átalakít, adatbázist/fájlt ír.
- Open/Closed Principle
  - Egy osztály zárt a módosításra és nyitott a bővítésre.
  - Leszármazottakban bővítsünk.
- Liskov Substitution
  - Ős osztály helyett bárhol a kódban legyen lehetőség leszármazottat használni hiba nélkül.
- Interface Segregation
  - Sok kisebb interfész egy nagy interfész helyett.
- Dependency Inversion
  - Egy osztály ne függjön egy másik osztálytól.
    - Helyette interfésztől
  - A függőség létrehozása nem az adott osztály feladata.
  - Megvalósítása
    - Dependency Injection: Interfész típusú konstruktor paraméterrel.
    - Factory tervezési minta
    - Inversion of Control (IoC)

Felosztási gondok

- Spaghetti code/Big Ball of Mud
  - Átláthatatlan kód
- Ravioli code
  - Kicsi és könnyen érthető osztályok
  - Megértése nehéz
- Lasagne code
  - Ránézésre rétegzett kód
  - Káosz, a kereszthivatkozások miatt kezelhetetlen
- Cél
  - Legyen fenntartható a kód

Verziókövetés

Verziókövető rendszer

- Lényege, hogy tárolja a fájlok módosításait.
- Funkciói
  - Korábbi állapotra visszaállás
  - Összehasonlíthatóság
  - Ki/Mikor/Mit/Miért módosított
- Típusai
  - Local Version Control System
  - Centralized Version Control System
  - Distributed Version Control System

LVCS (helyi verziókövető)

- Legegyszerűbb
  - Külön könyvtárak a verzióknak
  - Timestamp
- GNU Revision Control System
  - Egyszerű fájl tracking
  - Fájl újraépítés
  - Korábbi verzió visszaállás

CVCS (központosított verziókövető)

- Teameknél
- Mindent egy központi szerver tárol.
- Előnyei
  - Egyszerű kezelni
  - Átlátható
- Hátrányok
  - Single point of failure (szerver)
  - Nincs biztonsági mentés
  - Disk hibánál a fájlok és a history is elveszik.
- Szoftverek
  - CVS, Subversion, Perforce

DVCS (elosztott verziókövető)

- Workflow-k támogatása
- Kliensek teljes tükrei a szervernek.
- Előnyök
  - Offline is lehet használni
  - Bárki kódtárából a szerver visszaállítható.
  - Ritkán kell a szerverhez szólni.
- Hátrányok
  - Bonyolultabb a CVCS-nél
- Szoftverek
  - Git, Mercurial, Bazaar, Darcs

GIT beállítások

- Három beállítási szint
  - SYSTEM: eszközre érvényes
  - GLOBAL: fiókra érvényes
  - LOCAL: repositoryra érvényes
- Felhasználó teljes neve
  - git config –global user.name „XYZ ZXY”
- Felhasználó email címe
  - git config –global user.email xzy@email.com
- Beállítások listázása
  - git config –list

Lokális verziókövetés alapjai

- Ha még nincs központi szerver.
- GIT általános használata
    - Letükrözzük a távoli szervert vagy létrehozunk egy helyi repositoryt.
    - Helyben dolgozunk, committolunk.
    - Időnként felküldjük a központi szerverre a commitjainkat (push).
- GIT működése
    - Minden fájl egy 40 karakteres SHA1 hash-t kap a tartalma alapján.
    - Minden apró változás következménye egy teljesen új hash (lavina hatás).
    - Minden apró változásról azonnal tud a GIT.
    - Minden apró változás egy-egy újabb rekord a GIT adatbázisában.
    - Könyvtárakat nem ismer a git (csak elérési utakat).

Fájlok lehetséges állapotai

- UNTRACKED
    - Még nincs verziókövetve.
- TRACKED
    - Szerepelt az előző snapshotban
        - MODIFIED
            - Módosítva lett.
        - STAGED
            - Kijelölve committolásra.
        - COMMITTED
            - Elmentve a repositoryba.

Első lépések

- Új projekt
    - git init
        - Létrejön egy .git mappa.
        - Még nem trackelt.
    - git add *
        - Trackelve
            - Staged
- Létező projekt
    - Tükrözés
        - git clone https://github.com/repo
    - Teljes copy
        - Minden snapshot és history

Fájlok életciklusa

- ADD parancs értelmezése
    - Trackelni egy untracked fájlt.
    - Egy modified fájlt stagingelni.
    - Egyszerűbb felfogás
        - Add ezt a fájlt a következő commithoz.

Committolás

- Staging area tartalmát hozzáadja a repositoryhoz.
- Commit készítése
    - git commit
        - Üzenettel
            - git commit -m „üzenet”
- Staging area kihagyása
    - Stagel minden modified fájlt és committol.
        - git commit -a -m „üzenet”
- Jó commit üzenet jellemzői:
    - Miért kérdésre is válaszoljon.
    - Ne legyen benne „és” kötőszó, az rossz jell.

GIT status

- Megmutatja a working directory és a staging area állapotát.
- Rövid fájl lista mutatása.
    - git status -s
- Jelentés
    - A: staged
    - M: modified

Gitignore

- Adott kiterjesztésű fájlok kihagyása.

DIFF

- Status eszköz
- Mi nincs még stagelve, de már modified? (WD vs SA)
    - git diff
- Mi is kerül per pillanat a következő commitba?
    - git diff –staged

Fájlok törlése

- Ha nem akarjuk többé verziókövetni (-f ha stagelve van)
    - git rm fajl.txt
        - Commit
- Nem akarjuk verziókövetni, de WD-ben maradjon.
    - git rm –cached fajl.txt
        - Pl.: Elfelejtettük .gitignore-ba tenni.
- Átnevezés
    - git mv fajl.txt ljaf.txt
        - git rm és git add

Commit history

- git log -p
  - Változások
- git log –stat
  - Hány sor változott meg?
- git log –pretty=oneline
  - Egy sorban legyen egy commit.
- git log –graph
  - Gráfos megjelenítés

Visszavonások

- git commit –amend
  - Előző commitból kihagytunk vmit.
- git reset HEAD^
  - Töröljük az előző commitot.
    - Untracked
- git reset HEAD fajl.txt
  - Stagelés visszavonása.
- git checkout – fajl.txt
  - Változtatások elvetése egy fájlon. (előző állapot)

Remote repository

- Remote
  - Egy szervergépre, központi repository céljából telepített git program.
- Hostok
  - Github
  - Gitlab
  - stb
- Remote haszna
  - Publisholjuk valaki számára a kódot.
  - Több remote is lehet egy localhoz.

Remote repo készítése

- git remote add origin https://github.com/repo
  - Remote hozzáadása localhoz
- git fetch origin
  - Remote repo-ból letöltése.
- git pull origin master
  - Remote repo-ból letöltés és összefésülés.
- git push origin master
  - Remote repo-ba feltöltés.

Ellentmondások

- Remote hozzáadásakor van egy **master** és egy **origin/master** branch.
  - Eltérnek
- Amíg eltérnek nem lehet pusholni!

Műveletek a remoteon

- git remote show origin
    - Infók kérése a remoteról.
- git remote rename origin bitbucket
    - Remote átnevezése
- git remote remove origin
    - Remote törlése.

Taggelés

- A history egy bizonyos pontját megjelöljük egy stringgel.
- Általában release pontokat jelölünk.
- Ahol állunk azt taggeljük (utólag is).
    - Lightweight tag (pointer)
        - git tag v1.2-lw
    - Annotated tag (objektum)
        - git tag -a v1.3 -m ’1.4 version release’
- Tag-ek listázása
    - git tag
- Listázáskor szűrés egy mintára.
    - git tag -l „v1.8.5*”

Tesztelés

Mi a tesztelés?

- Mindenki a saját kódjának teszteléséért felelős.
- Közelebb került az implementációhoz, mert
    - Korai feedbacket ad.
    - Védi a lefedett kódot a későbbi véletlen hibák ellen.
    - Követelmények tisztázását segíti.
    - Tiszta, jól strukturált kód írását kényszeríti ki.

Egységteszt/Unit teszt

- A különböző metódusaink lefuttatása automatizáltan előre definiált bemeneti adatokkal és az eredmények összevetése egy elvárt eredmény listával.
    - Teszteléshez külön osztály készítése.
    - Rétegzett alkalmazástól független tesztelő réteg.

Egyéb tesztek

- Integrációs teszt
    - Egy szoftver ellenőrzi, hogy egy több modulból álló nagyobb alkalmazás az összeépítés után működőképes lett-e.
- UI teszt
    - Webalkalmazásoknál egy megfelelő szoftver végig kattintgatja a felületen a gombokat. Ír az űrlapmezőkbe, ellenőrzi a visszakapott értékeket.
- Terheléses teszt
    - Egy megfelelő szoftver a webalkalmazásunkat/API-nkat hívja egy szálon párhuzamosan és figyeli a válaszidőket.
        - N felhasználónál mekkora a rendszer válaszideje.

NUnit

- Unit teszt keretrendszer a .NET nyelvekhez.
- Jellemzői
    - Jól olvasható szintaxis.
    - Saját grafikus felület.
    - Command Line Test Runner
    - Visual Studio Test Explorer felületről is vezérelhető.
    - Elterjedt
- Használati elve
    - Új Class Library projekt a teszteléshez
        - NUnit telepítése
        - Referenciaként hozzáadása
        - Tesztelés

Vizsgálatok/Assert

- Assert.That(result, Is.EqualTo(42));
    - Egyenlőség
- Assert.That(result, Is.Null);
    - Null-e az érték.
- Assert.That(result, Is.LessThan(50));
    - Kisebb-e, mint…
- Assert.That(result, Is.SameAs(another));
    - Ugyanoda mutat-e a két referencia.
- Assert.That(() => mf.Area(-1, -1), Throws.TypeOf<ArgumentException>());
    - Dob-e kivételt.

Gyűjtemények vizsgálata

- Eredménylista
    - Egyezik-e az elvárt listával.
- Elvárja az Equals() és a GethashCode() felüldefiniálását.
    - Assert.AreEqual(object a, object b)

Több vizsgálat egyszerre

- TestCaste: Független unit tesztként futnak le.

Sorozatos vizsgálatok

- Sequential
  - Megadott értékek sorban beillesztése.
- Combinatorial
  - Összes lehetséges kombináció.
- Pairwise
  - Combinatorial, de kevesebb teszesettel.

Dinamikus vizsgálatok

- Gyakorlatilag bárhonnan beszerezhető a tesztadat
  - Külső fájl
  - Adatbázis
  - Hardver
  - stb

Hasznos attribútumok metódusokra

- SetUp
  - Minden teszteset külön-külön lefut.
- TearDown
  - Minden teszteset után külön-külön lefut.
- TestFixtureSetUp vagy OneTimeSetUp
  - Mielőtt lefutnának a tesztesetek, lefut egyszer.
- TestFixtureTearDown vagy OneTimeTearDown
  - Miután lefutott minden teszteset, lefut egyszer.
- SetUpFixture
  - Osztályra tehető, namespace-szintű setup/teardown.

Jól tesztelni nehéz

- Legyenek a tesztek a gyorsak.
- Legyenek egymástól függetlenek.
- Olvasható nevekkel.
- Nem kell és nem is szabad minden inputlehetőséget lefedni.
- Egyszerre egyetlen művelet és egy osztály tesztelése.

Tesztelési módszertanok

- Code-First
  - Nehéz és nem hatékony.
- Test-First
  - Fejlesztő először a teszteket írja meg.
  - Eközben tisztázódik sok követelmény.
- Test-Drive Development (TDD)
  - Először tesztek, mindegyik piros. Kódolás, tesztek futtatása, egyre több zöld lesz.
  - A kód 100%-ban tesztelhetőre íródik meg.
  - Általában pair-programming-ben csinálják.
  - Fejlesztési idő duplázódik.
  - Algoritmikusabb feladatokhoz illeszkedik igazán.

Tesztlefedettség

- Code Coverage
  - Hány %-át járták be a Unit tesztek.
  - Minden utasítás lefutott? (Statement coverage)
  - Bementek-e minden elágazás minden ágába? (Branch coverage)
  - Végigment-e minden ciklus? (Loop coverage)
  - Minden bool volt igaz és hamis is? (Condition coverage)
- Alsó határ szokott lenni.
- 100% elérése kb. lehetetlen, nem érdemes rá hajtani.

Rétegzett alkalmazás tesztelése

- Egyszerre egyetlen művelet és egy osztály tesztelése.
  - Mindig függetlenül az éles adattól, ami változhat.
    - Nem olvasunk éles adatbázist.
  - Osztályok függőségeit is helyettesítjük.
    - Dependency Injection + tesztduplikátum (test doubles)

Fake előnyök és hátrányok

- Előnyei
  - Könnyen, gyorsan implementálható
  - Egyszerű megérteni a kódot.
- Hátrányai
  - Sok üres metódus elszennyezi a kódot.

Testability Code Smells

- Jelentése
  - Gyanús kódok, amik a tesztelhetőség problémáját jelzik.
- Más osztályra közvetlen referencia van.
  - Interfészen át kérjük be konstruktoron keresztül.
- new() szerepel a kódban
  - Függőség, kiszervezendő dependency injection-re.
- Túl sok konstruktor paraméter
  - Sérül a Single Responsibility elv.
- Statikus osztályok és változók a kódban.
  - Átírás nem statikus osztályokra.
  - Statikus osztályok nem mockolhatóak!
- Felhasználói interakciótól függő kód (Console.ReadLine())
  - Kiszervezés külön osztályba (Humble Object Pattern)
- Adatbázisba, fájlba írás
  - Kiszervezés külön osztályba (Repository, Humble Object Pattern)
- Túl sokat kell mockolni
  - Túl sok dologért felelős az osztály.

Web

Mi az internet? Mi a web?

- Internet: telekommunikációs technológia
    - Világméretű összekapcsolt hálózat.
- WEB/WWW: hivatkozásokkal összekötött dokumentumrendszer.
    - Kiszolgálók dokumentumokat/szolgáltatásokat publikálnak.
    - Kiszolgálókra DNS címmel hivatkozunk (google.com).
    - Kiszolgálók által hosztolt weboldalak más weboldalra hivatkoznak (link).
- WEB szabványai
    - URL: Uniform Resource Locator
        - Dokumentumok/szolgáltatások elérése távolról.
        - Helyi:
            - C:\Users\xyz\index.html
        - Távoli
            - https://xyz.com/index.html
    - HTML: HyperText Markup Language
        - Weboldalak kinézetének leírónyelve.
        - HTML5 jelenleg
    - Http: HyperText Transfer Protocol
        - Üzenetek átvitele az interneten
        - Kérés-válasz alapú, állapotmentes protokoll
            - Kliens kezdeményez, kiszolgáló válaszol.
            - Két kérés között nincs állapotmegőrzés.
                - Küldünk egy parancsot és paramétereket (request).
                - Kapunk egy választ (reply).
        - HTTP request
            - Dokumentum neve
            - Elvárt formátum
            - Előzőleg látogatott oldal
            - Elvárt nyelv
        - HTTP reply
            - Dátum
            - Utolsó módosítás
            - Hossz
            - Formátum
            - Maga a dokumentum

HTTP metódusok

- GET
  - Dokumentum elkérése a kiszolgálótól.
- HEAD
  - Dokumentum törzsét még nem kérjük el, csak a jellemzőit.
- POST
  - Adatok küldése a szerverre (űrlapba írt adatok).
- PUT
  - Dokumentum feltöltése a szerverre.
- DELETE
  - Dokumentum törlése a szerverről.

Webszerver szoftver

- Egy olyan szoftver, ami a számítógépre telepítve figyeli a bejövő HTTP kéréseket és reagál rájuk két lehetséges módon:
  - Kikeresi az említett fájlt és tartalmát HTTP válaszban visszaküldi.
    - HTML tipikus példa
    - Videók, képek, zenék, tömörített állományok, stb.
    - Statikus weboldal készíthető így HTML-ben.
    - Csak akkor kap a felhasználó más tartalmat, ha a HTML dokumentum frissült a szerveren.
  - A kérést átadja egy tőle független programnak, ami feldolgozza és a webszerver ennek a független programnak az eredményét küldi vissza.
    - PHP
    - Ha .php kiterjesztésű fájl van a request-ben, akkor a PHP értelmező program kapja meg a kérést.
    - Előállít valamilyen dinamikus kimenetet.
    - Ezt kapjuk vissza válaszként.
    - Dinamikus weboldal készíthető így.

ASP.NET Core

- ASP: Active Server Pages
- Szerveroldali C# nyelvű keretrendszer webes fejlesztéshez.
- Ő maga a webszerver szoftver is egyben (Kestrel szerver).
- Amilyen portszámot konfiguráltunk be, azon a porton válaszol a kérésekre.
- ASP.NET MVC
  - MVC tervezési mintát valósít meg.
  - HTTP kérésekre HTML választ küldd vissza.
- ASP.NET WEBAPI
  - MVC tervezési minta View réteg nélkül.
  - HTTP kérésekre JSON válasz küldd vissza.

MVC architektúra végrehajtási ciklusa

1. A felhasználó egy kérést küldd a szervernek.
2. A vezérlő/controller fogadja a kérést, majd a modellben végrehajtja a megfelelő action methodot.
3. A modellben végrehajtott akció állapotváltozást okoz.
4. A vezérlő/controller begyűjti az action method eredményét/action resultot, majd létrehozza az új nézetet.
5. A felhasználó megkapja a választ (nézetet).

HTML világ gondjai

- Problémafelvetés
    - A hagyományos webes alkalmazások nem univerzálisak.
    - HTML választ küldenek vissza.
    - Csak böngészővel jeleníthető meg.
- Egyéb platformok térhódítása
    - Okostelefonok, tabletek
        - Natív alkalmazások igénye.
    - Viselhető eszközök
    - Beágyazott rendszerek
- Mi a gond?
    - Ezek a platformok nem HTML elemeket jelenítenek meg.
    - Saját UI megjelenítő
    - Nyers objektumgyűjtemények kellenének, nem a formázott HTML kód.

API világ

- Nyers adatok egységes továbbítása
    - Szöveges formátum kellene
        - Parse
        - Hierarchikus szerkezetre lenne szükség.
        - JSON vagy XML
- API alkalmazások lényege
    - HTTP kéréseket a törzsükben JSON adattal.
    - JSON válaszok.
    - Teljesen univerzális, nyelvfüggetlen megoldás.

REST API

- Representative State Transfer
- HTTP metódusokat használja fel.
- Jól szétválasztja az alkalmazásokat.
    - Backend
        - Repository, Logic, API endpoint
    - Frontend
        - API consumer, View renderer
- Inkább architekturális megközelítés, semmint protokoll

IoC konténer

- Milyen interfész függőségekre milyen implementációt adjon.

OpenAPI

- /swagger
- Reflexióval kikeresi az API endpointokat és ad hozzájuk egy grafikus tesztelő felületet.

Navigation Property probléma

- Amikor JSON formátumra alakítanak a Controllerek, akkor a Navigation Property-k miatt ciklikusság alakulna ki a JSON eredményben.

Szoftverfejlesztés

Régen

- Otthon dolgozik a fejlesztő a kódon.
- Elviszi a megrendelőhöz.
- Kipróbálja
- Nem megy, felírja a hibákat és hazamegy.
- Otthon dolgozik a fejlesztő a kódon.

Korai deployment

- Szerver
    - Túl kellett méretezni, elromolhatott.
    - Több felelősségi körrel is rendelkezett (levelezés, web, storage, stb…).
    - Nincs staging szerver.
- FTP protokollal történt a kódfrissítés.
    - Ki? Mikor? Mit?
    - Rollback nincs
    - Manuális deployment.
    - Nincsenek konfiguráció menedzsment eszközök.

Függőségi rémálom

- Egy számítógépen egy oprendszer fut.
- Egy oprendszeren egy szoftververzió.
    - Gátat szabott a fejlődésnek.

Virtuális gépek

- 1960-as évek IBM. Egy fizikai gépen több oprendszer fut párhuzamosan.
- Előnyök
    - Erőforrások jobb szétosztása.
    - Függőségek megoldódnak.
    - Másolhatók, áthelyezhetők, stb…
- Hátrányok
    - Kevesebb erőforrás marad a szoftverekre.
    - Versengés a gazda OS erőforrásaiért.
- Hypervisorok
    - VMWare, VirtualBox, Xen, QEMU, Hyper-V

Felhő rendszerek

- Fejlesztő szemszögéből
    - Egy adott szerveren vannak valahol az adatok.
    - Valaki elvégzi a frissítéseket, szerver karbantartást.
    - Valamennyit fizetünk majd használati alapon.
- A felhő egy magas absztrakciós szinten elfedi a napi szinten fejfását okozó technikai részleteket.

Szolgáltatási szintek

- IAAS (Infrastructure-as-a-service)
    - Virtuális gépek
    - Software defined networking (SDN).
    - Tűzfalak
- PAAS (Platform-as-a-service)
    - Adatbázisok
    - Fájl szerverek
    - Alkalmazás szerverek
- SAAS (Software-as-a-service)
    - Állománytároló
    - Levelezés

Cloud providerek

- Microsoft Azure
- Amazon Web Services
- Google Cloud Platform

Eldobható infrastruktúrák

- Régi világ: Adott a szerver és megy.
- Virtualizáció: Adott a virtuális szerver és megy.
- Tesztelés = egyész napos géptelepítgetés?
    - Infrastructure as code.
    - Szöveges fájlba, hogy mit szeretnénk és elkészül.
    - Virtuális gépekre: Vagrant nevű szoftver
    - Felhőbe: YAML fájlok, GUI kattintgatás.

C# alkalmazástípusok

- Console app/WPF app/Windows Forms App
    - Kliensoldali alkalmazás
        - .exe build
    - A felhasználó gépének erőforrásait használja.
- ASP.NET Core MVC/ASP.NET WEBAPI
    - Szerveroldali alkalmazás
    - Kliens szoftverek HTTP kéréseire válaszol az interneten keresztül.
        - JSON üzenetekkel
    - Kliensszoftver
        - Böngésző/Desktop app (WPF)/Mobil App (Xamarin)
    - Build
        - DLL
    - DLL hoszt
        - Webszerver szoftver
    - A cég szervereinek erőforrásait használja.

Új verzió kiadása

- Desktop/Console/Mobile app
    - Delivery
        - Frissebb weboldal verzió.
        - Google Play/App Store
        - Frissítő program.
    - Deployment
        - Új funkció a webalkalmazásban.
        - A kiszolgálón van minden.
        - Bármikor megváltoztatható.

Continuous Integration / Folyamatos integrálás

- Verziókat definiálunk.
- Megvan előre a cél, hogy az adott verzióban milyen feature-öket kell kiadni.
- GIT-en kezeljük
    - Fejlesztünk egy Development branchen, róla leágazva feature brancheken.
    - Masterre való mergelés általában egy új verziót jelent.
    - Merge commit ellátható egy labellel: v3.0
- Kliens gépen futó alkalmazás (desktop/mobil) esetén ezek a verziók letölthetők a githubról.
- Szerver alkalmazás esetén szükséges egy pipeline.

Continuous Deployment / Folyamatos élesítés

- GIT rendelkezik webhookokkal (esemény szerűség)
- Feliratkozunk a master branch merge eseményére.
- A webhook behív ilyenkor egy build szerverre (tokenizált HTTP kérés/SSH).
- A build szerver git pull kérést küld.
- A build szerver a letöltött forrásra kiadj a dotnet build parancsot.
- A buildelt binárist átküldi az alkalmazásszerverre.
- Alkalmazásszerver leállítja az előző verziót, eldobja, és hostolja az újat.
- A felhasználó egy oldalfrissítés után már látja az új feature-t.

Konténer technológia

- Mintha egy virtuális gép lenne.
- Oprendszer kernele közös.
- Konténer
    - Csak az alkalmazás + függőségei.
- Miért jó ez?
    - Izolált alkalmazás környezetek.
    - Nincsenek elpazarolt erőforrások.
    - Kicsi a konténerek mérete.
- Legismertebb konténer technológia
    - Docker
- Mit lehet csinálni egy konténerrel?
    - Kitörölni
    - Elindítani több példányban
        - + több fizikai gépen átívelve
    - Skálázás
        - Ha monitorozzuk a konténer válaszidejét, CPU és RAM kihasználtságát, akkor a reakció egy új konténer indítása és a forgalom felének ide terelése.
            - Növekedés: Új gép, megadjuk az IP-jét az orchestratornak.

Párhuzamosítás igénye

- Gyorsabb program futás.
- Több számítás ugyanazon időegység alatt.
- Mi teszi ezt lehetővé?
    - Többmagos architektúra
    - Nem lehet minden feladatot párhuzamosítani.

Egy magos rendszerek

- Neumann-elvű számítógépek kizárólag szekvenciális formában tudnak utasításokat végrehajtani.
- Időosztás
- Hyperthreading

Hyperthreading

- Processzor bizonyos részeinek fizikai duplázása.
- A konkrét végrehajtó egységből egy marad.
- Oprendszerek számára két különálló CPU-ként látszik.

Dekompozíció

- Adat vagy a probléma részekre osztása.
    - Adat dekompozíció
        - Egy adathalmazt részekre bontunk és több párhuzamos folyamat keresgél az egyes részekben.
    - Probléma dekompozíció
        - Egy szoftvernél a megjelenítés és a logolás történhetne egyszerre párhuzamosan.
            - Felesleges ezeket egymás után végrehajtani, mert nem függenek egymástól.
- Ha a konkurrens folyamatos száma több, mint a végrehajtó egységek száma, akkor torlódás lesz.
    - 4 mag esetén 4 részre osszuk a problémát.

Process

- Az oprendszer egyidejűleg több programot is futtathat.
    - Az éppen végrehajtás alatt álló programot folyamatnak nevezzük.
- .NET segítségével:
    - Futó folyamatok lekérése
    - Folyamatok indítása programozottan
    - Paraméterek küldése
    - Adatok fogadása

Párhuzamosítás fajáti

- Multiprocessing
    - Több folyamat dolgozik a probléma megoldásán (Process megoldás).
    - Különálló CPU (különálló gépek).
    - Külön memóriaterület minden Processnek.
    - Kommunikáció nehézkes.
- Multithreading
    - Egy folyamaton belül több szál dolgozik a probléma megoldásán (Thread megoldás).
    - Akár különálló CPU magok az egyes szálaknak.
    - Közös memóriaterületen dolgoznak.
    - Kommunikáció egyszerűbb.

Multiprocessing

- Hogyha egy Process-t elindítunk, akkor a környezet nem blokkol, amíg nem végzett.
    - Párhuzamosításra.
- Megvalósítás
    - Egy ciklussal elindítjuk a Processeket (start)
    - Egy ciklussan összevárjuk a Processeket (WaitForExit)
- Más módszer
    - Exited eseményre feliratkozunk
        - Több párhuzamos Processnél bonyolult.

Multithreading

- Folyamatokon belül szálak létrehozása.
- Különbség
    - Nem egy programot futtatunk több példányban, hanem egy metódust.
    - Összevárjuk az összes példányt.
- Thread
    - Előnye
        - Jobb erőforrás kihasználás.
        - Adatmegosztás sokkal könnyebb.
    - Hátránya
        - Versenyhelyzet lehetősége.
        - Holtpont lehetősége.
        - Nehéz debuggolás.
        - Nehezen kezelhetők az egymásra épülések.
        - Nagyon költséges létrehozni szálakat és váltani a szálak között.
        - Nincs beépített lehetőség korrekt leállításra.
        - Nincs visszatérési értékre lehetőség.
        - Paraméterezés nem elegáns (object).

ThreadPool, Taskok, Versenyhelyzet, Holtpont

ThreadPool

- A szálak egyenkénti létrehozása költséges.
- Ha lesz pool-ban szabad Thread, akkor végrehajtódik.

Task

- Aszinkron módon elvégzett feladat.
- A háttérben a ThreadPool egy eleme van.
    - Threadre épített technika a Task.
- Előnyei a Thread-hez képest
    - Visszatérési értékkel rendelkezhet.
    - Kivételkezelés megoldott.
    - Leállítható könnyen.
    - Paraméterezés kényelmes.
    - Szinkronizációban több lehetőség van.
- Mikor kell Thread?
    - Ha normáltól eltérő prioritással kell futnia egy szálnak.
        - Taskok csak normal-ok.
    - Előtérszálra van szükség.
        - Taskok csak háttérszálak.
    - A szál extrém hosszú műveletet végez.
        - Alkalmazás teljes élettartama alatt fut.

Continuation

- Feliratkoztathatunk a Task-ra egy olyan metódust, ami akkor fut le, hogyha a Task végzett.
- Mi az értelme?
    - A ContinueWith() nem blokkol, mint a Wait(), futhat tovább a program és majd lefut a Continuation.
        - Átírható, mert a WaitAll() lehet később a kódban.
    - Különböző megkötések
        - Csak akkor fusson le a Continuation, ha hiba van/ha minden rendben, stb.

Feltételes Continuation

- Lehetőségek
    - OnlyOnRanToCompletion
        - Ha minden rendben.
    - OnlyOnCanceled
        - Ha leállították.
    - OnlyOnFaulted
        - Ha hibára futott.
- Több Continuation is lehet egy Task-hoz.
    - Egyik akkor fut, ha minden rendben, egyik ha hiba van, stb…

Kivételkezelés Task-ban

- Ha egy Task-ban hiba keletkezik, akkor a kivétel elnyelődik.
- A Wait() hívásakor vagy a Result lekérésekor dobódik el AggregateException formájában.
    - Ez több Exception-t is tartalmazhat (InnerExceptions), mert a Taskoknak is lehetnek gyermek Task-jai.

Task leállítása

- Nincs Kill(), Abort() metódus a leállításra.
- A Task-ban elvégzett műveletben kell kezelni a leállítási kérést.
- Módja
    - Főprogramból jelzést küldünk (cancel request), hogy le akarunk állni.
    - Taskokban futó műveletek megkapják a jelzést és nem futnak tovább.
- Ellenőrzési módok
    - cancellatiokToken.IsCancellationRequested
        - Bool-t ad vissza.
        - Pl.: hosszan futó ciklusan ellenőrizzük.
    - cancellationToken.ThrowIfCancellationRequested()
        - Eldob egy kivételt, amint a kérelem beérkezik.
        - Ennek hatására OnlyOnFaulted Continuation indulhat.
            - Nem derül ki, hogy hiba történt, csak, hogy végzett a Task.

Versenyhelyzet

- Összevissza sorrend alakul ki minden futtatásra. Oka:
    - A konzol egy szálas környezet.
    - A szálak egyszerre indulnak.
    - Aki hamarabb odaér, az következik be először.

Bekövetkeztetés

- Amikor nem kell ezzel foglalkozni.
    - Szálak nem használnak közös változókat és erőforrást.
- Amikor kell ezzel foglalkozni.
    - Ha nem mindegy a végrehajtási sorrend.
    - Átlapolás (interleaving) adatokon.

Versenyhelyzet megoldási ötlet

- Saját ID-jának megfelelő sorba tegye azt.
- Hatás:
    - Üres sorok, összevisszaság, néha 1-1 szám elveszik

Kölcsönös kizárás megvalósítása

- Biztosítja, hogy a kritikus szakaszba egyszerre csak egy szál léphessen be.
- Gyakorlati megvalósítása
    - Zár
    - Szemafor
    - Monitor
    - Atomi végrehajtás
- Elvük
    - Egy szál magánál tartja a szinkronizációs objektumot, addig a többi szál kénytelen várakozni.
    - Amikor elkészült, akkor elengedi az objektumot.

Versenyhelyzet megoldása lock-kal

- Szükséges hozzá egy tetszőleges referencia típusú példány.
- lock blokkal jelöljük meg a kritikus szakaszt.

Versenyhelyzet megoldása monitorral

- Valójában a lock is ezt használja.

Versenyhelyzet megoldása szemaforral.

- Vasúti terminológia alapján (Edgar Dijkstra).
- Megadható paraméterként, hogy hány szál léphet be egyszerre a kritikus szakaszba.

Egyéb párhuzamos problémák

- Holtpont
  - A szálak olyan eseményre vagy feltételre várnak, ami sosem következik be.
    - Ha egy szál kritikus szakaszhoz ér, de nem tud belépni.
- Többféle holtpont is definiálható
  - Ön-holtpont
    - A szál olyan lock-ra vár, ami már az övé.
  - Rekurzív-holtpont
    - A kritikus szakaszban lévő szál olyan funkciót próbál elérni, ami visszahivatkozik a lockkal védett részre.
  - Zár-sorrend holtpont
    - Két szál, két lock
      - Egyiknél az egyik, másiknál a másik és egymásra várnak.
- Élőpont
  - Egyes részfeladatok ugyanazokat a rövid lépés sorozatokat követik és ezen nem tudnak túljutni.
  - Kivédése nehéz.
- Éhezés
  - Megosztott erőforrásokhoz bizonyos szálak nem tudnak hozzáférni.
    - Más szálak egy ideig elérhetetlenné teszik.
  - Megoldása policy-kkel lehetséges
    - Kell egy szabályozott rendszer, hogy ki, mikor, mihez férhet hozzá.

Párhuzamos alkalmazások tervezése, Task Parallel Library, Aszinron

Párhuzamos tervezési ökölszabályok

- Nehéz meglátni, hogy hol lehet jól párhuzamosítani és hol érdemes.
- Hosszan futó szekvenciális folyamatokat érdemes egyszerre végezni, amik nem függenek egymástól.
  - N db fájl letöltése webről.
  - Távoli erőforrásokkal adatcsere.
  - Adatpárhuzamos feladatok.
  - Egymástól nem függő általános folyamatok
    - Megjelenítéshez táblázat rajzolása és közben logolás.
- Szemcsézettség (granularity)
  - Minden szálnak jusson elég munka.
  - Túl kevés munka és túl sok szál is rontja a teljesítményt.
- Terheléskiegyenlítés
  - A felosztás akkor a leghatékonyabb, ha a szálak mindegyike egyenlő mennyiségű munkát végez.
  - A már elkészül szálak tétlenül várakoznak.

Task Parallel Library

- Publikus API, ami kiegészíti a System.Threading és System.Threading.Task névtereket.

Aszinkron metódusok

- Nem feltétlenül párhuzamos metódusok.
- Létrehozhatók olyan metódusok, amik elindulnak, közben a kód továbblép és a metódus konkurrens módon végzi a dolgát.

Async-await

- Async-el megjelölt metódust be kell várni az await-el jelölt helyen.
- Ha nem használunk await-et, akkor elindul az x metódus és fut tovább a főprogram.
    - Visszatérési érték gondot okoz és Continuation-t érdemes használni.
- Érdemes pl. mentési folyamatot async metódusba tenni, nem várunk választ tőle.

Kérdésbank

- **Mit jelent agilis keresztfunkcionális csapatban fejleszteni?**
    - A fejlesztés összes aspektusáért a team felel.
- **Melyik szolgáltatás nem része egy PAAS rendszernek?**
    - tűzfalak
- **Mit jelent a szerializáció?**
    - Objektumok tárolható formába alakítása.
- **Melyik állítás HAMIS a GIT CLONE paranccsal kapcsolatban?**
    - Szükséges hozzá egy meglévő local repository.
- **Mit jelent a build folyamatban a linkelés?**
    - Több lefordított bináris egy db binárissá egyesítése.
- **Melyik C# alkalmazástípusnál beszélhetünk Continous Deliveryről?**
    - WPF app
- **Melyik állítás HAMIS a barátságos URL-el kapcsolatban?**
    - Gyorsabb lesz tőle a válasz, hiszen a szerveroldalon egyszerűbb feldolgozni.
- **Melyik http metódusnak szokás megfeleltetni ezt a CRUD metódust: adott elem módosítása**
    - PUT
- **Melyik állítás igaz az alábbiak közül?**
    - Lockoláshoz valamilyen referencia típusú objektumot szükséges használni.
- **Hogyan találja meg az NUnit a teszt célból létrehozott metódusokat?**
    - Reflexióval keresi attribútumok alapján.
- **Melyik állítás HAMIS klasszikus weboldalak/API alapú weboldalakkal kapcsolatban?**
    - Klasszikus weboldalaknál a szerver egy JSON alapú objektumlistát küld és a böngésző rendereli ki a HTML kódot.
- **Melyik CRUD funkciónak szokás megfeleltetni ezt a http metódust: DELETE**
    - Adott id-jű elem törlése.
- **Mi nem igaz a delta alapú tárolásra verziókövetés esetén?**
    - A teljes mappák különbségeit tárolja.
- **Mik tartoznak a Business rétegbe?**
    - Domain modelek és szolgáltatások
- **Mely állítás igaz az alábbiak közül?**
    - A szemafor alkalmazható kölcsönös kizárásra is, így biztosítva a kritikus szakaszhoz való megfelelő hozzáférést.

- **Melyik igaz a THREAD-re az alábbiak közül?**
  - Visszatérési értéke nincs.
- **Melyik CRUD funkciónak szokás megfeleltetni ezt a http metódust: POST**
  - Új elem létrehozása
- **Melyik állítás nem igaz a jó tesztre?**
  - A jó teszt minden inputlehetőséget lefed.
- **Mi nem igaz a THREAD-re az alábbiak közül?**
  - Magas szintű eszköz szálak létrehozására.
- **Melyik párhuzamos programozási technikára igaz: Felhasználói felületen futó szálak és hosszú műveletek elkülönítése.**
  - BackgroundWorker
- **Melyik állítás nem igaz a natív kódra?**
  - Platformfüggetlen
- **Melyik rossz URL az alábbiak közül?**
  - http://asd.com/alma.php? a = 6?b=7
- **Mi nem a célja attribútumok létrehozásának?**
  - Kód fordítás optimalizálása
- **Mely állítás igaz az alábbiak közül a Task-ok esetében?**
  - Task létrehozásakor egy szál kerül hozzárendelésre az elvégzendő feladathoz, amely a Threadpool-ból automatizáltan kerül meghatározásra.
- **Mi a hátránya egy CVCS rendszernek egy DVCS rendszerhez képest?**
  - A szerver az egyetlen tároló és egyben single pont of failure.
- **Melyik állítás nem igaz az XML fájlokra?**
  - Létezik God-Formatted XML és Valid XML
- **Melyik beépített delegate-re illeszkedik a double(int, int) függvény?**
  - Func<int, int, double>
- **Melyik cég nem cloud provider az alábbiak közül?**
  - Asus
- **Melyik párhuzamos programozási technikára igaz: Eldönthetjük, hogy adott processt kivárjuk-e vagy továbblépünk és addig futtatunk egyéb kódokat.**
  - Async-await
- **A tesztelés melyik fázisára igaz a kijelentés: objektumok előkészítése**
  - Arrange
- **Melyik állítás igaz a [Setup] attribútummal jelölt metódusokra NUnit használata esetén?**
  - Minden teszteset előtt lefut.
- **Melyik C# alkalmazástípusnál beszélhetünk Continuos Deploymentről?**
  - webalkalmazás
- **A LINQ OrderBy operátort mivel lehet paraméterezni?**
  - Kulcskijelölő lambdával
- **Melyik állítás HAMIS a GIT COMMIT paranccsal kapcsolatban?**
  - Working directory tartalmát hozzáadja a repositoryhoz.
- **Melyik CRUD funkciónak szokás megfeleltetni ezt a http metódust: GET**
  - Adott id-jű illetve összes elem visszaadása.
- **Melyik párhuzamos programozási technikára igaz: Külső folyamatot tudunk vele futtatni párhuzamos (pl.: másik exe-t)**
  - Process
- **A tesztelés melyik fázisára igaz a kijelentés: tesztelendő egyetlen lépés végrehajtása.**
  - Act

- **Melyik állítás HAMIS a gittel kapcsolatban?**
  - Minden fájl és mappa egy 40 karakteres SHA1 hash-t kap a tartalma alapján.
- **Mi célt szolgál a DbContext OnModelCreating() metódusa?**
  - Táblák konfigurálását végzi.
- **A szoftverfejlesztésben mit jelent a rejtett függőség?**
  - Egy osztály kompozícióval használ egy másik osztályt és a szükséges objektumot önmaga példányosítja le.
- **Melyik állítás igaz a szálbiztonság kapcsán az alábbiak közül?**
  - Egy szálbiztos program megvédi adatait a memória konzisztencia hibáktól.
- **Melyik SOLID elvre igaz: ha egy ős helyébe tetszőleges utódot helyettesítünk, akkor ez nem ront a működésén.**
  - Liskov substitution.
- **Melyik XML formátumra igaz a kijelentés: hibás, hogyha a mellékelt szabályoknak nem felel meg.**
  - Valid XML
- **Melyik igaz a THREAD-re az alábbiak közül?**
  - Végét tipikusan eseményekkel érzékeljük.
- **Melyik LINQ operátor alkalmas arra, hogy két halmaz különbségét megadja?**
  - Except
- **Mi nem igaz egy virtuális gépre?**
  - A Docker a legismertebb hypervisor.
- **A REST API milyen klienseket támogat?**
  - Teljesen mindegy, bármit aminek a programozási nyelve támogatja a REST kéréseket.
- **Melyik állítás igaz a szoftver LAYER-re?**
  - Az egy rétegbe tartozó osztályok egy közös magasabb rendű feladatot látnak el.
- **Melyik http metódusnak szokás megfeleltetni ezt a CRUD metódust: minden elem visszaadása.**
  - GET
- **Mit ad vissza a Where LINQ operátor, hogyha egy objektumlistán (T típusú List) használjuk?**
  - T típusú IEnumerable
- **Mit csinál a Select LINQ operátor?**
  - T típusú gyűjteményt transzformál egy másik típusú gyűjteményre.
- **Mit jelent a SPA a szoftverfejlesztésben?**
  - Single Page Appication: JS alkalmazás, amely a funkciók hatására a böngészőbe betöltött HTML kódot változtatja.
- **Mivel nem valósítható meg a Dependency Inversion?**
  - statikus osztállyal
- **Milyen verziókövető rendszer típus nem létezik?**
  - Remote Version Control System (RVCS)
- **Mi az előnye egy CVCS rendszernek egy LVCS rendszerhez képest?**
  - Már alkalmas team munkára központi szerver használata miatt.
- **Melyik állítás HAMIS az alábbiak közül?**
  - A POST kérés tipikusan egy űrlap szerverről való letöltésére szolgál.
- **Melyik CRUD funkciónak szokás megfeleltetni ezt a http metódust: PUT**
  - Adott id-jű/tartalmú elem módosítása.
- **Mi volt a GIT újdonsága már meglévő DVCS-ekhez képest?**
  - Lehetett benne brancheket kezelni.

- **Melyik állítás nem igaz a FAKE-re?**
    - Nem kell hozzá új osztályt létrehozni.
- **Melyik párhuzamos programozási technikára igaz: Szálakat egy adott halmazból újra és újra felhasználunk.**
    - ThreadPool
- **Melyik HTTP metódus célja klasszikus webalkalmazások esetén az űrlapadatok szerverre való továbbítása?**
    - POST.
- **Mi nem igaz a snapshot alapú tárolásra verziókövetés esetén?**
    - Lassú
- **Melyik állítás nem igaz egy ASP.NET Core alkalmazásra?**
    - Kliensoldalon fut
- **Melyik SOLID elv mondja ki, hogy egy osztály a függőségét ne hozza létre, hanem kívülről várja?**
    - Dependency Inversion
- **Milyen attribútumot kell használnunk, hogyha natív DLL-t szeretnénk C# nyelven hívni?**
    - DllImport
- **Melyik szolgáltatás nem része egy SAAS rendszernek?**
    - tűzfal
- **Melyik állítás igaz egy natív (C++) kódra?**
    - Közvetlenül az OS/CPU értelmezi.
- **Melyik állítás HAMIS az alábbiak közül?**
    - A POST kérés tipikusan egy űrlap szerverről való letöltésre szolgál.
- **Melyik szolgáltatás nem része egy SAAS rendszernek?**
    - tűzfal
- **Melyik állítás nem igaz a MOCK-ra?**
    - Ugyanaz, mint a fake. Egy szinonima rá…
- **Melyik állítás nem igaz a konténertechnológiákra?**
    - tartalmazza a konténer a gazda operációs rendszer kernelét is.
- **Amdahl törvényét felhasználva, az elméletileg elérhető gyorsulás maximális mértéke X, az egyszálas feldolgozáshoz képest.**
    - X = 20
- **Melyik LINQ operátor alkalmas arra, hogy két halmazt egymás után fűzzön?**
    - Concat
- **Mire szolgál a GIT PULL utasítás?**
    - Remote repository tartalmát letölti egy meglévő local repositoryba és összefésüli a módosításokat.
- **Melyik LINQ operátor alkalmas arra, hogy egy elem létezését megállapítsa egy gyűjteményben?**
    - Contains
- **Melyik állítás igaz a TDD módszertanra?**
    - Sokszor párban programozva csinálják.
- **Melyik http metódus célja klasszikus webalkalmazások esetén az űrlapadatok szerverre való továbbítása?**
    - POST
- **LINQ to XML esetén mire szolgál a Descendants() függvény?**
    - Visszaadja az adott nevű közvetett és közvetlen gyermekelemeket is.

- **Mit jelent az, ha a .gitignore fájlban van egy ilyen sor: *.mp4**
    - Minden mp4 kiterjesztésű fájl maradjon ki a traceklésből.
- **Mi nem igaz egy felhő rendszerre?**
    - Nem tudjuk, hogy melyik cégnél vannak az adataink.
- **Melyik SOLID elvre igaz ez: sok kisebb osztály jobb, mint egy nagy egész, ami sok mindenért felel**
    - Single Responsibility
- **A tesztelés melyik fázisára igaz a kijelentés: állapotváltozás megismerése és ellenőrzése, hogy az elvárásnak megfelelő-e**
    - Assert
- **Melyik SOLID elvre igaz ez: a függőség konkrét létrehozása valaki másnak a feladata**
    - Dependency Inversion
- **Melyik SOLID elvre igaz ez: sok kisebb interfész jobb, mint egy nagy egész**
    - Interface Segregation
- **Melyik szolgáltatás nem része egy IAAS rendszernek?**
    - fájl szerverek
- **Melyik toolra igaz az alábbi állítás: DLL-eket tud regisztrálni a Global Assembly Cache-be**
    - gacutil.exe
- **Mire szolgál a GIT PUSH utasítás?**
    - Local repository eddig nem pusholt commitjait feltölti egy remote repositoryba.
- **Tesztelésnél mire nem tudunk Assert() segítségével elvárást beállítani?**
    - A felhasználó megnyomott X gombot a billentyűzeten
- **Milyen teszt nem létezik?**
    - konszolidációs teszt
- **Melyik SOLID elvre igaz ez: egy osztály legyen használható és bővíthető**
    - Open/Closed principle
- **Melyik állítás nem igaz a NuGet toolra?**
    - Központi .NET csomagkezelő, tipikusan natív DLL állományokhoz.
- **Melyik állítás igaz a statikus linkelésre?**
    - Az összes függvény a programba belefordítódik.
- **Melyik LINQ operátor alkalmas arra, hogy egy gyűjteményt halmazzá alakítson?**
    - Distinct
- **Melyik állítás HAMIS a GIT ADD paranccsal kapcsolatban?**
    - Hatására megjelenik a fájl githubon.
- **Melyik igaz a THREAD-re az alábbiak közül?**
    - A paraméter semmi vagy object típusú.
- **Mit nem támogat egy LVCS rendszer?**
    - Távoli hozzáférést
- **Melyik állítás igaz a jó tesztre?**
    - A jó teszt legyen független.
- **Melyik adatbáziskezelő szerver az, amelyet tipikusan kis és középvállalatok használnak éles környezetben?**
    - MSSQL
- **Melyik nem tipikus code smell?**
    - Interfész típusú konstruktor paraméterek
- **Mit jelent a de-szerializáció**
    - Valamilyen eltárolt adat (szöveges vagy bináris) objektummá visszaalakítása.

- **Ki fejlesztette a GIT-et?**
    - Linus Torvalds
- **Mire szolgál a GIT FETCH utasítás?**
    - Remote repository tartalmát letölti egy meglévő local repositoryba, de nem foglalkozik összefésüléssel.
- **Melyik párhuzamos programozási technikára igaz: Alacsony szintű szálkezelés, visszatérési értékek nincsenek**
    - Thread
- **Melyik kódsor képes megmondani, hogy egy elemek nevű T generikus lista elemeinek mi a típusa?**
    - typeof(T)
- **Az XDocument osztály melyik metódusa képes URL címről betölteni az XML fájl tartalmát?**
    - Load
- **Mi értelme van ebben a kódban a „?" operátornak? node.Element(„name")?.Value**
    - Hogyha nincs ilyen element, akkor nem próbálja lekérni a Value értékét.
- **Melyik állítás nem igaz a C# Eventre?**
    - bárhonnan meg lehet hívni.
- **Melyik érv nem támasztja alá, hogy interfész típusú paramétereket használjunk függőségek átvételére?**
    - A debuggolás és a kivételkezelés így könnyebbé válik.
- **Mire való az Activator.CreateInstance() függvény?**
    - Adott type-ból képes egy objektumpéldányt létrehozni.
- **Mit jelent a refaktorálás szó?**
    - A kód struktúrájának módosítása a viselkedés módosítása nélkül.
- **Melyik http metódust nem használjuk REST API kérésre?**
    - HEAD
- **Melyik állítás igaz a dinamikus linkelésre?**
    - Egyes függvények a program saját címterületén kívül vannak.
- **Mi az XML?**
    - Adatok tárolására szolgáló leíró nyelv.
- **Melyik lehetőség nem segíti elő a párhuzamos programozást?**
    - több memória
- **Melyik állítás nem igaz a REST-re?**
    - Könnyebb implementálni, mint a SOAP-ot.
- **Mi nem célja a párhuzamos programozás témakörének?**
    - egy program futtatása több gépen egyszerre.
- **Melyik adatbáziskezelő szerver az, ami vállalati célú alkalmazásnál szerényebb teljesítményű, de valós adatbázis motor?**
    - SQL Express
- **Melyik nem igaz egy DVCS rendszerre?**
    - Egyszerűbb kezelni, mint egy CVCS rendszert.
- **Melyik állítás nem igaz a managed kódra?**
    - OS/CPU számára értelmezhető bytekód.
- **Melyik nem része egy tipikus szoftver rétegzésnek?**
    - Encryption
- **Mi nem funkciója egy verziókövető rendszernek?**
    - Nem használt fájlok felderítésére.

- **Melyik párhuzamos programozási technikára igaz: Magasszintű szálmenedzsment, API biztosítása**
    - TPL/Task
- **Melyik állítás HAMIS a GIT CLONE paranccsal kapcsolatban?**
    - Távoli repositoryból készít adott nevű másolatot.
- **Melyik állítás igaz a C# delegate-ekre?**
    - Több függvény is lehet benne.
- **Melyik adatbáziskezelő szervert használtuk gyakorlaton?**
    - LocalDb
- **Melyik állítás nem igaz az XML fájlokra?**
    - Lehet megnyitni egy elemet, majd egy másikat, majd lezárni az egyiket és a másikat.
- **Mit jelent a Tier a szoftverfejlesztésben?**
    - Fizikai felbontása az alkalmazás komponenseinek.
- **Melyik LINQ operátor alkalmas arra, hogy két halmazt összes elemét visszaadja ismétlőd nélkül?**
    - Union
- **Hogyha egy meglévő, void visszatérésű metódust szeretnénk párhuzamosan több példányban futtatni, akkor melyik módszer NEM alkalmas erre?**
    - async-await
- **Mely állítás igaz az alábbiak közül?**
    - Thread létrehozásakor előtérszál kerül létrehozásra.
- **Melyik LINQ operátor alkalmas arra, hogy két halmaz különbségét megadja?**
    - Except
- **Melyik SOLID elvre igaz ez: sok kisebb osztály jobb, mint egy nagy egész, ami sok mindenért felel**
    - Single Responsibility
- **Mit csinál a Select LINQ operátor?**
    - T típusú gyűjteményt transzformál egy másik típusú gyűjteményre
- **Melyik XML formátumra igaz a kijelentés: hibás, hogyha a mellékelt szabályoknak nem felel meg?**
    - Valid XML
- **Melyik állítás igaz a [Setup] attribútummal jelölt metódusokra NUnit használata esetén?**
    - Minden teszteset előtt lefut.
- **A tesztelés melyik fázisára igaz a kijelentés: állapotváltozás megismerése és ellenőrzése, hogy az elvárásnak megfelelő-e**
    - Assert
- **Mi nem igaz a THREAD-re az alábbiak közül?**
    - Magas szintű eszköz szálak létrehozására.
- **Melyik szolgáltatás nem része egy IAAS rendszernek?**
    - fájl szerverek
- **Mire szolgál a GIT PUSH utasítás?**
    - Local repository eddig nem pusholt commitjait feltölti egy remote repositoryba.
- **Melyik igaz a THREAD-re az alábbiak közül?**
    - Visszatérési értéke nincs
- **A tesztelés melyik fázisára igaz a kijelentés: tesztelendő egyetlen lépés végrehajtása**
    - Act

- **Válassza ki, hogy melyik HTTP metódus alkalmazható teljesen új elem létrehozására (korábban még nem létezik)!**
    - Post
- **Válassza ki a folyamatokra (Processre) jellemző állításokat!**
    - Nagy szemcsézettségű párhuzamos feladat feldolgozást szolgáltat.
    - Saját memória területe van.
    - A folyamatok szerializált formában képesek a kommunikációra egymással.
- **Milyen összefüggést ismer a folyamatok standard outputjának átirányíthatóságáról és a rendszerhéjon való futtatásról?**
    - Akkor irányítható át az alapértelmezett kimenete a folyamatnak, hogyha nem a rendszer rendszerhéjon futtatjuk..
- **Válassza ki a jó teszt jellemzőit!**
    - A teszt lehetőleg legyen gyors
    - A teszt intervallum szemléletben teszteljen (ne az összes adat fusson le, hanem jellemző pontokat vegyen ki véletlen a határokon).
- **Párosítsa össze az előadáson tanultak alapján a Fake és Mock jellemzőket!**
    - Tipikusan kézzel kell megoldani ezt a fajta helyettesítést. → Fake
    - Körülményesebb a kialakítása a másikhoz képest. → Fake
    - Csak imitálja a működést, tényleges kód nincs mögötte. → Mock
    - A félév során ezt a módszert tanultuk, gyakoroltuk. → Mock
    - Tipikusan keretrendszer segítségével használjuk ezt a fajta helyettesítést. → Mock
    - A helyettesítés érdekében több sor kód írására is szükség van, például egy listával kell kiváltani az adatbázis működését. → Fake
- **Hogy nevezzük az alábbi kódot? valamiMetodus(x => x.Fontos)**
    - Lambda expression
- **Mi az a módszer, amellyel C#-ban ki lehet nyerni egy típus adattagjait, metódusait, stb?**
    - reflexió
- **Igaz, hogy egységtesztelés során tényleges adatbázist/éles settings fájlokat alkalmazhatunk?**
    - Hamis
- **Párosítsa össze a Thread-ekre és a Task-okra vonatkozó állításokat!**
    - Alapértelmezetten hosszú folyamatokra tervezett eszköz. → Thread
    - Threadpool-on lévő szálként jelenik meg. → Task
    - Visszatérési értékkel rendelkezhet. → Task
    - Bemeneti paramétere tetszőleges típus lehet és abból is több. → Task
    - Alacsony szintű szálkezelést nyújt. → Thread
- **Az alábbi kód mit képes elvégezni? List<int> eredmeny = lista.FindAll(delegate(int i) { return i % 2 == 0; });**
    - A lista elemei közül kiválogatja a páros számokat..
- **Párosítsa össze a CI és CD-re vonatkozó feladatokat.**
    - Automatizált deploy → CD
    - Kód fordítása → CI
    - Artifact-ek összeállítása → CD
    - Statikus kódelemzés → CI
    - Tesztek futtatása → CI
- **Mit nevezünk kritikus szakasznak többszálú környezetben?**
    - Kritikus szakasznak nevezzük a kód azon részét, amely olyan közös erőforrásokat olvas/módosít, melyet több szál is használ/elér..

- **Igaz, hogy a dinamikusan linkelt exe fájl mérete jóval kisebb, mint a statikusan linkelt alkalmazásé?**
    - Igaz
- **Az alábbi csomópont felépítés érvényes (valid)? <alma><csutka></alma></csutka>**
    - Hamis
- **Válassza ki, hogy a CD esetén, mely szoftverfejlesztési "lépésnél" jelenhet meg, nyújthat támogatást?**
    - Kész termék telepítőjének összeállítása, Ügyfélnél kitelepítés
- **Párosítsa a multiprocessing és a multithreadingre vonatkozó állításokat!**
    - Egy folyamaton belül több szál dolgozik a probléma megoldásán → multithreading
    - A szálak közös memóriaterületen dolgoznak, adatok hozzáférésekor számíthat a hozzáférési sorrend → multithreading
    - Több folyamat dolgozik ugyan azon probléma megoldásán → multiprocessing
    - Az egyes folyamatok különálló CPU-t használhatnak a számításaikhoz → multiprocessing
- **Egy delegált akkor tud egy metódust képviselni, hogyha...**
    - a metódus szignatúrája és visszatérési értéke is megegyezik a delegáltéval..
- **Igaz az az állítás, hogy a kódomat majd a tesztelő leteszteli, nekem nincs szükség rá? (tegyük fel, hogy mi nem tesztelők vagyunk)**
    - Hamis.
- **Mi a TDD tesztelési módszertan elve?**
    - Előbb a teszteket írjuk meg, majd ezt követően történik a kód fejlesztése, majd refaktorálás és indul elölről a folyamat..
- **Válassza ki a helyes állításokat a group by Linq kifejezésre!**
    - Csoportosítást valósít meg.
    - Az értékeket egy külön csoportosított adathalmazba képzi le, melyen belül a csoportosítás kulcsa nyerhető ki, valamint a csoportosított elemeken aggregáló függvények hajthatóak akár végre.
- **Az alábbi kódrészlettel mit érhetünk el a kódban? Task.WhenAll(tasks).ContinueWith(t => Console.WriteLine("\nAll task has finished!"));**
    - Amennyiben az összes feladat befejeződött, akkor aszinkron módon jelenítsük meg a konzolon, hogy befejeződtek..
- **Válassza ki az igaz állításokat a Mock-olásra vonatkozóan!**
    - Az osztályok függőségeinek imitálását végzi el, mintha ténylegesen ott lenne az objektum, viselkedését mi határozzuk meg a beállításakor.
    - Dependency Injection segítségével cserélhetők le a függőségek.
    - Tipikusan interfészt mockolunk.
- **Válassza ki a szálszinkronizálásra alkalmas eszközöket!**
    - lock, Interlocked.Increment, Semaphore