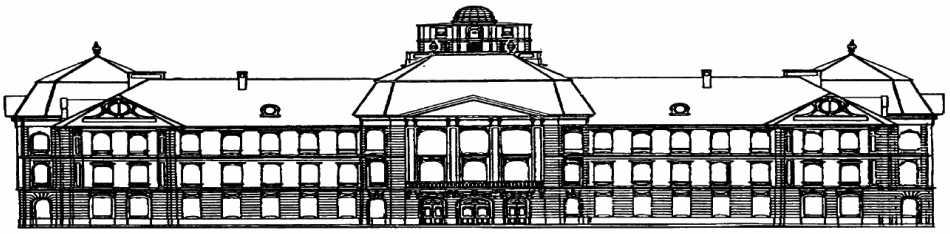

Eszterházy Károly Katolikus Egyetem

Magasszintű programozási nyelvek II.

2022/2022 tavaszi félév

2. zárthelyi dolgozat

Általános információk

Hozzon létre egy *.NET Framework* projectet a **C:\temp** mappába, melynek neve az Ön **neptunkódja**! A kijelölt feladatokat ebben implementálja C# nyelven, majd munkája végeztével ugyanilyen néven tömörítse a programot és töltse föl a megadott címre!

A kicsomagolás után nem beazonosítható dolgozatok automatikusan elégtelenek, ezek újraírására nincs lehetőség!

A dolgozat írása közben csak és kizárólag a **C:\temp**, valamint a **Letöltések** mappák lehetnek megnyitva a projecten kívül!

A dolgozat megírása során semmilyen segédanyagot, órai feladatot nem használhat!

A dolgozat megírása során semmilyen kommunikációs csatorna (levelezőrendszer, chat, stb. . .) nem lehet nyitva és nem léphet kapcsolatba senkivel.

Bármilyen a fentiekre utaló magatartás esetén azonnal elégtelen a zh eredménye. Ennek újra írására nincsen lehetőség!

Törekedjen részmegoldásokra!

A második zárthelyi dolgozat fókuszában az osztálystruktúra és a program összetettsége áll. Éppen ezért ahol nem fejtjük ki, hogy egy adatot milyen módon kell validálni, ott elég, ha egy ellenőrzések nélküli property szignatúrát készít!

Feladatkiírás

A zárthelyi dolgozatban egy 3×2 *km*-es területen elhelyezett szenzorok (hőmérők és kamerák) tárolását és menedzselését végző alkalmazást kell implementálnia!

A feladat megoldásához mellékelve kap néhány fájlt forráskódokkal, melyeket beilleszthet a forráskódjába! Éppen ezért figyeljen arra, hogy kövesse a dolgozat által előírt elnevezési konvenciókat!

DLL + Console alkalmazás

A feladatot úgy készítse el, hogy az enumok, interfészek és osztályok mind egy DLL-be legyenek implementálva, majd ezt egy másik konzolos alkalmazásban használja fel a főprogramban!

Ne feszüljön rá, ez csak 2 pontot ér! Ha nem biztos magában, írjon mindent egyetlen konzolalkalmazásba!

Csatolt forrásfájlok

Annak érdekében, hogy ne kelljen mindent teljesen elölről megírnia, mellékelten talál néhány forrásfájlt. Kezdje azzal, hogy a **Pozicio.cs** fájl tartalmát elhelyezi a project-ben!

Interfészek

A feladatban egyetlen saját interfészt kell létrehoznia!

IHomero

Hozzon létre egy interfészt **IHomero** néven, és implementálja benne az alábbi metódusok és propertyk szignatúráit!

- **HomersekletetMer** nevű paraméter nélküli valós értékkel visszatérő függvény!
- **HatarokatBeallit** nevű kettő darab egész értékű paraméterrel (egész: alsoHatar, egész: felsoHatar) rendelkező eljárás.

Osztályhierarchia

A program lényegi részét a különböző szenzorok képezik. Ezek alapja a *szenzor* absztrakt osztály, amelynek egy gyermekosztálya van, a *hőmerő*.

Szenzor

Hozzon létre egy osztályt **Szenzor** néven, és implementálja benne az alábbiakat!

Propertyk

- **Pozicio** nevű Pozicio típusú property, melynek van **get**tere és kívülről nem látható **set**tere.
- **Aktiv** nevű csak **get**terrel rendelkező **absztrakt** property.

Metódusok

- **Adatkuldes** nevű paraméter nélküli **absztrakt** eljárás.
- A **Szenzor** osztály implementálja az **ICloneable** interfészt! Mivel az osztály absztrakt (nem lehet belőle példányosítani), így a kapott **Clone** metódust tegye **absztrakt**tá!
- Írja felül a **ToString** metódust! A mellékelt fájlban megtalálja az ide másolható forráskódot!

Konstruktor

Készítsen egy konstruktort, mely bekéri a pozíciót és eltárolja az abban található értéket!

Hőmérő

Hozzon létre egy osztályt **Homero** néven, jelölje meg ősként a **Szenzor** osztályt, és implementálja az **IHomero** interfészt!

Propertyk

- **AlsoHatar** nevű egész típusú property, melynek van **get**tere és kívülről nem látható **set**tere. Fejse ki a propertyt és ellenőrizze, hogy az alsó határ nem lehet $-60°C$-nál kisebb! Ellenkező esetben dobjon saját kivételt **AlacsonyAlsoHatarException** néven!
- **FelsoHatar** nevű egész típusú property, melynek van **get**tere és kívülről nem látható **set**tere.
- **Aktiv**: Írjon egy **SetAktiv** property metódust a szokott módon, mely egy privát mező értékét kezeli, majd ehhez igazítva fejtse ki a **get**tert.

Metódusok

- **HatarokatBeallit**: Elmenti a paraméterben kapott értékeket.
- **HomersekletetMer**: Ha a szenzor nem aktív, dobjon kivételt! Ehhez (ha még nincs) készítsen saját kivételt **SzenzorInaktivException** néven! Ellenkező esetben térjen vissza egy véletlenül generált 2 tizedesjegy pontosságú valós értékkel az $[AlsoHatar,\ Felsohatar)$ intervallumból!
- **Adatkuldes**: Meghívja a HomersekletetMer függvényt, és a generált értéket felhasználva a minta szerint megjeleníti az alábbi szöveget. Pl.:

  Hőmérséklet a(z) (50;730) pozición 2022.05.13 16:53 időpontban: 32,05°C
- **Clone**: készítsen egy klónt az adott példánnyal és térjen vissza vele!
- Írja felül a **ToString** metódust! A mellékelt fájlban megtalálja az ide másolható forráskódot!

Konstruktor

Készítsen konstruktort, mely bekér két egész számot, mint a pozíció x és y koordinátái, valamint két további egész számot (alsó határ és felső határ)! A koordinátákból készítsen Pozíció példányt és adja át az ős konstruktorának! Mentse el a határokat a **HatarokatBeallit** metódus segítségével, és állítsa aktívra a szenzort!

Konténerosztály

A konténerosztály feladata kezelni a szenzorok listáját, és bizonyos lekérdezések eredményeit szolgáltatni.

Szenzorhálózat

Hozzon létre egy osztályt **SzenzorHalozat** néven és implementálja az **IEnumerable** interfészt!

Mezők

- **szenzorok**: kívülről nem elérhető **Szenzor** típusú adatokat tartalmazó lista. Ne írjon hozzá property-t!

Metódusok

- **Telepit**: Szenzor típusú paraméterrel rendelkező eljárás. Feladata elmenteni a paraméterben kapott szenzort a listába. Nem kell ellenőriznie semmit!

Propertyk

- **AktivSzenzorok**: Szenzor típusú objektumok listájával visszatérő property, melyet lambda kifejezéssel kell implementálnia! A visszaadott listában csak az aktív szenzorok legyenek benne x koordinátájuk szerint növekvő, továbbá y koordinátájuk szerint csökkenő sorrendben! (Extra pont: Ha meg tudja oldani, akkor a példányok klónjai kerüljenek a listába!)
- **AktivAtlag**: A függvény kérjen egy delegate-et paraméterben és térjen vissza egy valós értékkel, mely a rendszerben levő aktív szenzoroknak ezen delegate által megadott értékeinek átlaga! Ehhez készítsen delegate-et, mely Homero típusú objektumot kap paraméterként és egy egész értéket ad vissza!
- **GetEnumerator**: Adja vissza az aktív szenzorok klónjait!

Főprogram

Töltse be a mellékelt Program.cs fájl tartalmát! Ez tartalmazza az adatok beolvasásának vázát!

– Készítse fel a beolvasást arra, hogy esetleg rossz fájlnevet adtak meg!

A következő részfeladatokat úgy oldja meg, hogy csak a hibás sor ne legyen beolvasva, az utána levők ettől függetlenül működjenek!

– Készítse fel a beolvasást az Ön által definiált kivételekre!

További feladatok:

– Jelenítse meg a beolvasott szenzorokat az enumerator segítségével egy foreach ciklusban!

– Jelenítse meg az aktív szenzorokat rendezve a megírt metódus segítségével!

– Írja ki a képernyőre az **AktivAtlag** metódus segítségével az aktív hőmérők x koordinátáinak átlagát!