Eseményvezérelt és Vizuális Programozás PótZH 2020. ősz

A ZH során egy 3 projektből álló solutiont kell létrehozni, ezekbe készítsd el az alábbi feladatok megoldását, majd a laborokhoz hasonlóan github pull request formájában add be a befejezési időpontig.

A solutionbe az alábbi típusú és nevű 3 projektet hozz létre:

- Console Application (.NET Core), C#: "ConsoleApplication" néven
- Class Library (.NET Standard) C#: "Common" néven
- xUnit Test Project (.NET Core) C#: "Tests" néven

Ne feledd, hogy egyik projektből csak akkor tudod elérni a másikban lévő dolgokat, ha a projekt References részében felveszel rá egy hivatkozást a Solution Explorerben.

Commonban ősosztály létrehozása **3p**

Töröld az alapból létrejövő Class.cs-t.

Hozz létre egy Person nevű (publikus) osztályt, benne egy Name és Age nevű property-vel.

Ebben a Person osztályban definiálj egy konstruktort 2 paraméterrel, az első egy string típusú name, a második egy int típusú age. Add értékül a konstruktor paramétereket a megfelelő property-knek.

Ebben a Person osztályban definiáld felül a ToString() metódust úgy, hogy egy olyan szöveget adjon vissza, ami tartalmazza az adott ember nevét és korát (Pl. "Hello, I'm ... and ... years old", ... részek értelemszerűen kitöltendőek).

A stringek összerakásához használj string interpolationt.

(Itt érdemes commitolni egyet.)

Testsben triviális unit teszt létrehozása **3p**

A Tests projektben töröld az alapból létrejövő UnitTest1.cs fájlt.

Hozz létre egy PersonTests unit teszt osztályt, benne egy PersonToStringTest nevű teszttel, ami példányosít egy Person objektumot "Aladár" névvel és 22 korral.

Ellenőrizd, hogy a létrehozott Person objektumnak megfelelő értékű property-jei vannak illetve a ToString() metódus is az elvárt értéket adja.

(Itt érdemes commitolni egyet.)

Konzol alkalmazás elkészítése **3p**

A ConsoleApplication projektben hozz létre a Program osztályban egy ShowPersonDetails metódust, ami paraméterül kap egy Person objektumot és kiírja konzolra a kapott Person nevét, korát és a ToString() hívás eredményét.

A Main() metódusban hozz létre egy Person objektumot a saját neveddel és koroddal, majd hívd meg a ShowPersonDetails metódust ezzel a létrehozott Person objektum paraméterrel.

Ellenőrzésként futtasd le a konzol alkalmazást!

(Itt érdemes commitolni egyet.)

Leszármazott osztályok **6p**

A Common projektben hozz létre egy (publikus) Person leszármazottat Student néven, melynek vagy egy NeptunCode property-je és egy 3 paraméteres konstruktora (név, kor, neptun). A megfelelő konstruktor paramétereket add tovább az ősosztálynak!

Készíts a konstruktorba egy neptun kód validálást reguláris kifejezéssel: egy jó neptun kód 6 karakter hosszú és csak nagybetűkel illetve számokat tartalmazhat. Ha nem valid neptun kóddal akarunk létrehozni Student objektumot, akkor dobjunk kivételt. (Figyelj rá, hogy megfelelő típusú kivételt dobj, amely arra utal, hogy itt egy paraméter nem volt jó.)

(Itt érdemes commitolni egyet.)

Kódos unit teszt **6p**

A Tests projektben hozz létre egy (publikus) StudentTests osztályt, mely egy StudentTestSuccess nevű tesztben ellenőrzi, hogy ha létrehozunk egy Student objektumot "Béla" névvel, 20 korral és "B1TM4N" neptun kóddal, akkor az létre jön a megfelelő property-kkel.

StudentTests osztályban hozz létre egy StudentTestException nevű tesztet, amellyel ellenőrzöd, hogy egy "Cecil" nevű, 42 korú és "Cecil.42" neptun kódú Student létrehozása kivételt dob. (Figyelj az elvárt kivételt típusára!)

(Ha valamiért nem működik a programod, a unit teszteket is tudod debuggolni úgy, hogy a Test Explorer ablakban a teszten jobb klikk - Debug-ot választassz.)

(Itt érdemes commitolni egyet.)

A konzol alkalmazás kiegészítése **3p**

A ConsoleApplication projektben a Program osztályt egészítsd ki egy ShowStudentDetails() függvénnyel, ami egy Student objektumot kap paraméternek, kiír mindent, amit a ShowPersonDetails() illetve még a Student neptun kódját is (sorrend tetszőleges, törekedj arra, hogy kevés parancsból írd meg, használd fel, amit már egyszer megírtál).

A Main() metódust egészítsd ki úgy, hogy hozz létre egy Student objektumot a saját adataiddal, majd hívd meg rá a ShowStudentDetails() függvényt.

(Itt érdemes commitolni egyet.)

Megoldások leadása **1p**

A megoldást a laborfeladatokhoz hasonlóan, github pull requestként add be és a laborvezetődet rendeld hozzá reviewerként, ha ez kimarad érvénytelen a dolgozatod és nem javítjuk ki! Figyelj rá, hogy a forráskód esztétikus is legyen (felesleges üres sorok mellőzése, kikommentezett forráskód ne legyen), a commitokban csak verziókövetendő fájlok legyenek (forráskód, projekt és solution file igen, bin és obj könyvtár és .user file nem). A pull request szövegébe pedig rakj bele egy screenshotot, amin látszanak a Test Explorerben a zöld unit tesztek, valamint a konzolos alkalmazás futási eredménye. (A konzol alkalmazás ablaka ha futtatás után egyből bezárul, akkor Ctrl-F5-tel indítsd el, akkor biztosan nyitva marad.)

A ZH befejezési ideje az utolsó commit és pull request időpontjára vonatkozik. Későbbi leadásra percenként -1 pont jár. Ha a pull requestből hiányzik a két screenshot, az -4 workflow hibapont.