|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NÉV: |  | NEPTUN KÓD: |  |  |  |  |  |  | **X** CSOPORT |
| **!** | A beadandó solution elnevezési mintája: NEPTUNKOD\_X, ahol a neptunkód a saját neptunkódot jelöli. | | | | | | | | |
| A beadott dolgozat fordítási hiba esetén nem értékelhető. Kommentezett részre pont nem szerezhető. | | | | | | | | |
| A feltöltendő file-t a megfelelő kurzushoz töltse fel, egyéb esetben a ZH nem értékelhető. | | | | | | | | |
| Minden részfeladat elvégzését, a megadott nevű külön metódussal, illetve osztállyal valósítsa meg. | | | | | | | | |

**Feladat 1. 3 pont**

Készítsen egy Food osztályt a következő tulajdonságokkal: *string Name, int Price, int Qty*

Készítsen egy DiscountException osztályt, amely string-et kapjon bemenetnek és adja tovább az ősnek.

Készítsen egy GroceryShop osztályt, amiben helyezzen el egy *Food listát* tulajdonsággal, valamint a következő metódusokat:

* AddToCart: egy Food elemet a listához ad
* RemoveFromCart: index alapján törlünk a listából
* SelectFoodByCriteria: a bemeneti predicate delegált alapján leválogatunk elemeket a listából és azokat visszaadjuk
* CalculateFinalSumPrice: bemenetnek egy int discountValue értéket kap, amely ha kisebb vagy egyenlő mint nulla, akkor dobjon saját készítéső kivételt; egyéb esetben számolja meg, hogy mennyi a kosár jelenlegi értéke (vegye figyelembe a terméket, annak árát és darabszámát) majd alkalmazza rá a paraméternek kapott kedvezményt és térjen vissza az eredménnyel 3 tizedesre kerekítve

**Feladat 2. 10 pont**

Készítsen egy új projektet amely tesztek írására legyen alkalmas. Hozzon létre egy Tests.cs állományt a projekten belül, és a következő metódusokat tesztelje a leírtak alapján:

* készítsen egy inicializáló metódust, amely minden teszt előtt **automatikusan** lefut; ebben hozzon létre egy GroceryShop példányt és töltse fel pár elemmel a bevásárlókosarat (javasolt a példány referenciáját osztály szintre kihelyezni)
* tesztelje le az AddToCart metódus működését, ellenőrizze, hogy a kosárba rakott elem tényleg belekerül-e
* tesztelje le a FinalSumPrice metódus működését, hogy megfelelő paraméter esetén valóban dob-e kivételt
* tesztelje le a FinalSumPrice metódust, hogy valóban jó értéket számol-e ki
* teszteljele a SelectFoodByCriteria metódust, hozzon létre egy predicate delegáltat amely az 1 darabszámú termékeket nézi

**Feladat 3. 10 pont**

Egészítse ki a GroceryShop osztályt egy PlaceOrder metódussal, amely Task-ok segítségével fájlba írja a kosárban lévő termékek nevét / árát / darabszámát. A három tulajdonságra három külön „szálat” hozzon létre, tehát egy szál dolgozik a neveken, egy az árakon és egy a darabszámokon. A működés idejét szimulálja a Thread.Sleep segítségével (2 msp, 5 msp és 10 msp). Írja ki a konzolra, hogy a task elkezdődött vagy befejeződött. A taskokat szinkronizálja és continuation segítségével írja ki a konzolra, ha minden task befejezte futását. A fájlok neveinek ezeket használja: names.txt, prices.txt, quantities.txt

**Feladat 4. 3 pont**

A Main részből tesztelje le a párhuzamosságért felelős részt, valamint adjon választ kommentezve az alábbi kérdésekre:

* Mutassa be a dependency injection működését egy egyszerű példán (DI nélküli kód ==> kód + DI)?
* Mik az ORM használatának előnyei?
* Mi a különbség a static linking és a dynamic linking között?

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*

*A feladat megoldása közben alkalmazza a tanult elveket és szintaktikákat.*

*A feladat megoldására 90 perc áll rendelkezésre.*

*A kész feladatot* ***teljes egészében*** *(egész Solution mappa), .zip (nem .rar / .7zip / stb.) formába tömörítse be, majd az ismertetett ZH leadó felületen, a megfelelő kurzust kiválasztva töltse fel!*