Feladat – NagyZH – FONTOS INFÓK

- Alkalmazni kell a megoldás során a tanult objektum-orientáltsági elveket.
- A bemeneti fájlokat (**order1.json**, **order2.json**, **order3.json**, **stock.json**) teszteléskor a build könyvtárba kell másolni.
- Feltehetjük, hogy a bemenő/teszt adatok helyesek, érvényesek (kivéve, ha a feladat mást nem állít).
- A megoldás teljes forráskódját egyetlen ZIP fájlba csomagolva kell feltölteni.

A feladat leírása

A feladatban egy faáru üzlet készletét kell kezelni. Az üzlet különféle fából készült termékek készítését, nyilvántartását, eladását kezeli. Jelen programban háromfajta terméket szeretnénk kezelni: széket, asztalt, illetve ruhás szekrényt. A faáruk esetén nyilván tartunk egy egyedi termék-azonosítót (szöveg), a felhasznált fa típusát (szöveg), a súlyt (kg-ban, nem feltétlenül egész), a méreteit (szélesség, magasság, hosszúság, cm-ben), valamint az egységárát (Ft-ban). Székek esetén ismerjük még a szék stílusát (szöveg), asztal esetében a lábak számát, valamint, hogy a lábak állíthatóak-e, szekrény esetében pedig az ajtók számát, valamint, hogy van-e rajta tükör.

Az átfogó működést a **Store** osztályon keresztül kell megoldani. Legyen egy *loadStock* metódusa, amely egy JSON fájl nevét kapja, és onnan betölti az üzlet kínálatát. A JSON két részre van osztva. A **"Products"** kulcs alatt vannak a termékek leírásai, míg a **"Stock"** kulcs alatti tömb pedig minden termék-azonosítóra megadja, hogy az adott termékből mennyi darab van éppen raktáron. Feltehetjük, hogy minden termékhez pontosan egy bejegyzés van a raktárkészletben, de a sorrend eltérhet.

A **Store** osztálynak kell egy *printProducts* metódus, amely megjeleníti az összes terméket, valamint egy *printStock* metódus, amely ezen felül a termékek aktuális mennyiségeit is kiírja.

A projekthez tartozik egy másik fajta JSON fájl is, ami egy vásárló által leadott rendelés adatait írja le. Ebben egy tömb van, amelynek minden eleme egy termék-azonosítót és egy darabszámot tárol. A **Stock** osztályban legyen egy statikus *loadOrder* metódus, amely megkapja egy ilyen fájl nevét, beolvassa a tartalmát, és visszatér a fájlban leírt **Order** objektummal (nem tárolja el).

A **Store** osztálynak legyen egy *priceOfOrder* metódusa, amely megkap paraméterben egy rendelést, és visszaadja a teljes költségét. A költségbe az áruk összes árán kívül fel kell számolni egy egyszeri ügyintézési díjat is, ami legyen egységesen 1000 Ft, de (bár a *main* ezt nem teszteli) legyen lehetőség az átállítására.

Legyen egy *deliverOrder* metódus, amely a raktár készletből levonja a rendelt mennyiségeket, feltéve, ha a rendelés teljesíthető. Ha bármelyik termékből nincs elég a raktáron, a rendelés nem teljesíthető. A visszatérési érték jelezze, hogy a rendelés teljesült-e.

Legyen egy *produce* metódus, amely egy termék-azonosítót kap és egy darabszámot, és az adott termékből elérhető mennyiséget megfelelően növeli.

Legyen egy *exportQuantities* metódus is, ami egy fájlnevet kap, és abba a fájlba JSON formátumba kimenti a tárolt termékek mennyiségeit, valamint az elérhető mennyiségeket (a formátum ugyanaz, mint ami a **stock.json** fájlban a **"Stock"** kulcs alatt van).

Pontozás

1. Termékek adatszerkezete, beolvasása, és tárolása a **Store** osztályban **(8 pont)**
2. Aktuálisan elérhető mennyiségek beolvasása és tárolása **(4 pont)**
3. *printProducts* metódus **(3 pont)**
4. *printStock* metódus **(4 pont)**
5. Rendelés adatszerkezete (**Order** osztály) és betöltése (*loadOrder* metódus) **(6 pont)**
6. *priceOfOrder* metódus **(4 pont)**
7. Az egyszeri ügyintézési díj mértéke program-szinten átállítható **(2 pont)**
8. *deliverOrder* metódus **(4 pont)**
9. *produce* metódus **(2 pont)**
10. *exportQuantities* metódus **(3 pont)**

Összesen: **40 pont**.