Általános tudnivalók

Ebben az ismertetésben az osztályok, valamint a minimálisan szükséges metódusok leírásai fognak szerepelni. A feladatmegoldás során fontos betartani az elnevezésekre és típusokra vonatkozó megszorításokat, illetve a szövegek formázási szabályait.

Segédfüggvények létrehozhatóak, a feladatban nem megkötött adattagok és elnevezéseik is a feladat megoldójára vannak bízva. Törekedjünk arra, hogy az osztályok belsõ reprezentációját a lehetõ legjobban védjük, tehát csak akkor engedjünk, és csak olyan hozzáférést, amelyre a feladat felszólít, vagy amit azt osztályt használó kódrészlet megkíván!

A beadott megoldásodnak mûködnie kell a mellékelt tesztprogramokkal, de ez nem elégséges feltétele az elfogadásnak. A megírt forráskód legyen kellõen általános és újrafelhasználható!

Figyelem! Az a metódus, mely fordítási hibát tartalmaz, automatikusan nulla pontot ér!

A feladat összefoglaló leírása

A feladatban egy piac nagyon leegyszerûsített mûködését fogjuk szimulálni.

A feladat részletes ismertetése

1. rész (6 + 1 pont)

`market.Fruit` osztály:

Az osztály egy gyümölcsöt reprezentál.

- Az osztálynak két rejtett adattagja van: egy szöveges típusú `name`, amely a gyümölcs nevét tárolja és egy egész szám típusú `price`, ami a gyümölcs árát tárolja.
- Az osztálynak legyen egy rejtett konstruktora, amely paraméterben megkapja a nevet és az árat, és beállítja a megfelelõ adattagokat.
- Legyen egy statikus `make` metódus, amely szintén egy nevet és egy árat kap. A metódusnak ellenõriznie kell a paramétereket, és amennyiben azok megfelelõek, akkor hozza létre, és adja vissza a paramétereknek megfelelõ `Fruit` objektumot. Ha a paraméterek nem jók, akkor a metódus `null`-t adjon vissza. A nevet tartalmazó paraméter akkor megfelelõ, ha csak betût tartalmaz és legalább 2 karakter hosszú. Az árat tartalmazó paraméter pedig akkor helyes, ha pozitív, de legfeljebb 5000 és 0-ra vagy 5-re végzõdik. Segítség: használhatod a `Character` osztály `isLetter` metódusát.
- Legyen egy paraméter nélküli `getPrice`, ami visszaadja a gyümölcs árát.
- Az osztály tartalmazzon egy `cheaperThan` metódust, ami eldönti, hogy az aktuális gyümölcs olcsóbb-e, mint a paraméterben kapott gyümölcs.
- Az osztályban legyen egy paraméter nélküli `show` metódus, amely visszaadja az objektum szöveges reprezentációját. A formátum legye a következõ: `név (ár-ezres-tagolással Ft)` (ha az ár 1000-nél nagyobb, akkor ezres tagolásként egy szóközt kell használni, pl. `3 065`).
- Az osztály tartalmazzon egy `cheapestFruit` nevû osztályszintû adattagot, ami a legolcsóbb gyümölcsöt tartalmazza, amit valaha létrehoztak (ha több ilyen van, akkor ezek közül az elsõt). Ha még nem hoztak létre `Fruit` objektumot, akkor az adattag értéke legyen `null`. Figyelj rá, hogy a konstruktor mindig aktualizálja ezt az objektumot, amikor az összes eddiginél olcsóbb gyümölcsöt hoz létre.
- Legyen egy osztályszintû `getCheapestFruit` metódus, ami visszaadja az elõbbi `Fruit` objektumot.

Tesztelõ: `tests.Part1`

2. rész (7 + 2 pont)

`market.Market` osztály:

Az osztály egy egyszerûsített piacot reprezentál.

- Az osztály egy rejtett láncolt lista típusú adattagban tartsa nyilván, hogy milyen gyümölcsöket lehet kapni a piacon (`Fruit` típusú

objektumok).

- Az osztálynak egy publikus konstruktora legyen, ami egy fájlnevet kap paraméterként, amely gyümölcsöket tartalmaz. A metódus dolgozza fel a fájlt, szûrje ki belõle a hibás adatokat, majd töltse fel a gyümölcsöket a láncolt listába. Ha a fájl nem létezik, vagy nem olvasható, akkor a gyümölcsök listája legyen üres, a konstruktor pedig ne engedje ki a keletkezõ kivételt. A fájl olvasásához használható pl. a BufferedReader vagy a Scanner osztály. Az inputfájl minden sora egy gyümölcsöt tartalmaz `név,ár` formában. Ha a sor nem ilyen szerkezetû, vagy az `ár` nem konvertálható számmá, vagy a megadott adatokból a `Fruit` objektum nem hozható létre, akkor a sort figyelmen kívül kell hagyni és a feldolgozást a következõ sorral kell folytatni.

  Azt, hogy egy szöveg egy egész számot tartalmaz kétféleképpen is ellenõrizhetjük.

  a. Elkapjuk az `Integer.parseInt()` által sikertelen számmá konvertáláskor dobott `java.lang.NumberFormatException` kivételt.

  b. Megvizsgáljuk a szöveget, hogy minden karaktere számjegy-e vagy sem a `Character.isDigit()` metódussal.

- Legyen egy `numberOfFruits` metódus, amely visszaadja a piacon még kapható gyümölcsök számát. Kezdetben az összes gyümölcs kapható.

- Legyen egy paraméter nélküli `show` metódus, ami szöveges típusban visszaadja a még kapható gyümölcsöket. A szöveg összeállításakor a gyümölcsök olyan sorrendben szerepeljenek, amilyen sorrendben a konstruktor beolvasta õket, és olyan formában, ahogy a `Fruit show` metódusa elõállítja. Figyeljünk rá, hogy az utolsó gyümölcs után már ne legyen sortörés!

Tesztelõ: `tests.Part2`

3. rész (6 + 1 pont)

A `market.Market` osztályba vegyük fel az alábbi publikus metódusokat:

- `cheaperThan`: a metódus egy `Fruit` típusú objektumot vár, és ilyen objektumok láncolt listáját adja vissza. A metódus gyûjtse össze azokat a gyümölcsöket, amelyek olcsóbbak, mint a paraméterben kapott gyümölcs. A gyümölcsök a visszaadott listában olyan sorrendben szerepeljenek, ahogy a konstruktor beolvasta õket.

- `average`: a metódus egy valós számot (`double`) adjon vissza, azt, hogy mennyi a gyümölcsök átlagos ára. Ha nincsenek gyümölcsök, akkor a metódus adjon vissza `-1`-et.

  Figyelem: ha két egész számot osztunk egymással, akkor az eredmény mindig egész lesz (lefelé kerekít).

Tesztelõ: `tests.Part3`

4. rész (6 + 1 pont)

A `market.Market` osztályba vegyük fel az alábbi publikus metódusokat:

- `buyCheapestFruit`: a metódusnak nincsen paramétere, és egy vásárlást szimulál. Egy vásárló mindig a legolcsóbb terméket veszi meg. Ha a legolcsóbb gyümölccsel azonos árú is van, akkor azt, amelyiket a konstruktor korábban vett fel a listába. A megvásárolt gyümölcsöt el kell távolítani a listából, és ez a gyümölcs lesz a metódus visszatérési értéke is. Ha a vásárlás elején egyetlen gyümölcs sem volt már a piacon, akkor a metódus `null`-t adjon vissza.

- `sale`: a metódusnak nincs paramétere és egy kiárusítást szimulál. A kiárusítás addig tart, amíg el nem fogy az összes gyümölcs. A metódus egy gyümölcsöket tartalmazó láncolt listába gyûjtse össze, hogy a gyümölcsöket milyen sorrendben vásárolták meg. Ez a lista lesz a metódus visszatérési értéke.

Tesztelõ: `tests.Part4`

Pontozás (elmélet + gyakorlat)

- 0 -- 20: elégtelen (1)
- 21 -- 25: elégséges (2)
- 26 -- 30: közepes (3)
- 31 -- 35: jó (4)
- 36 -- 40: jeles (5)