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 is létrehozhatók, a feladatban nem megkötött adattagok és elnevezéseik is a feladat megoldójára vannak bízva. Törekedni kell arra, hogy az osztályok belső reprezentációja a lehető legjobban legyen védve, tehát csak akkor és csak olyan hozzáférés megengedett, amelyre a feladat felszólít, vagy amit az osztályt használó kódrészlet megkíván!

A beadott megoldásnak a megadott outputfájlt kell produkálnia, de ez nem elégséges feltétele az elfogadásnak. A megírt forráskód kellően általános és újrafelhasználható legyen!

Az egyes feladatrészeknél két pontszám látható (pl. 5 + 1), ebből az első az alappont, amely a helyességért, a feladatleírásban szereplő követelmények teljesítéséért adható. A második egy pluszpont, amely az oktató megítélése alapján akkor jár, ha a hallgató az adott feladatrészt a félévben tanult alapvető programozási irányelvek betartásával készítette el, így levonandó például optimalizálatlan, kódismétlésekkel teli vagy konvenciókat erősen sértő kód esetén. konvenciók

Használható segédanyagok: Java dokumentáció, legfeljebb egy üres lap és toll. Ha bármilyen kérdés, észrevétel felmerül, azt a felügyelőknek kell jelezni, *NEM* a diáktársaknak!

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

A feladathoz mellékelve van egy minta input- és outputfájl, valamint egy hibásan megírt osztály, melyet ki kell javítani.letöltés

A feladat összefoglaló leírása

A feladatban hagyományos, játékokkal foglalkozó üzlet működését fogjuk szimulálni.

A feladat részletes ismertetése

1. rész (11 + 2 pont)

`shop.Platform` felsorolási típus (1 pont):

- hozzunk létre egy felsoroló típust mely a lehetséges platformokat tartalmazza, (azonos írásmóddal):

  ```
  XBOX, PS, PC, COMMODORE64
  ```

`shop.Game` osztály (10 + 2 pont):

Az osztály egy játékot reprezentál.

- Az osztálynak hat rejtett adattagja van: egy szöveg típusú fejlesztő cég, egy szöveg típusú cím, egy egész típusú darabszám és eladott darabszám, egy szintén egész típusó azonosító, illetve egy Platform típusú, a platformot tároló példányváltozó.
- Az osztálynak legyen egy rejtett konstruktora, amely paraméterként megkapja az cég nevét, a játék címét, a kezdő darabszámot, valamint a platformot (Platform típus), és beállítja a megfelelő adattagokat (eladott darabszám 0). Az azonosító legyen mindig a legutolsóként használt azonosítónál egyel nagyobb egész, 0-tól indulva (1 pont)

- Definiáljunk egy osztályszintű `make` nevű metódust is. A `make` metódus szintén az cég nevét, a játék címét és a darabszámot kapja meg paraméterként, valamint szöveges paraméterként a Platform nevét. A metódus először ellenőrzi, hogy a paraméterek megfelelőek-e. Amennyiben igen, akkor létrehozza és visszaadja a paramétereknek megfelelő `Game` típusú objektumot. Ha a paraméterek nem megfelelőek, akkor a metódus `null`-t adjon vissza.
    - A cég neve és a játék címe akkor megfelelő, ha nem egy `null` referencia, és nem üres String.
    - A darabszám akkor megfelelő, ha pozitív szám. (3 pont)
    - A platform akkor megfelelő, ha konvertálható egy megfelelő enum értékké.

  Segítség: a metódusban használható az `Enum` osztály `valueOf()` metódusa.

- Definiáljuk az osztályban az alábbi, paraméter nélküli lekérdező metódusokat: `getDeveloper()`, `getTitle()` és `getPiecesOnStock()` és `getPlatform()` amelyek rendre visszaadják a fejlesztő cég nevét, a címet, a darabszámot és a platformot. (1 pont)

- Az osztálynak legyen egy `buy` nevű metódusa, mely visszatérési érték nélküli, és egy pozitív egész paramétert vár, és amelynek segítségével vásárolni lehet az adott játékból. A vásárlás a következőképpen történik: ha a paraméter kisebb, mint az aktuális darabszám, akkor a darabszámot az adott értékkel csökkentjük, míg az eladott darabok számát ugyanannyival növeljük. Különben nem történik semmi. (3 pont)

- Definiáljunk egy paraméter nélküli `toString` nevű metódust is, amely visszaadja az objektum szöveges reprezentációját. A formátum legyen a következő: `<cég>: <játék címe> (<db> pcs)`. Pl. `Rockstar Games: Grand Theft Auto V (3 pcs)`. (1 pont)

- Definiáljunk egy `betterSellThan()` metódust, mely egy játékot vár paraméterül, és logikai igazat ad vissza, ha az aktuális játékból, melyen a metódust meghívták, többet adtak el, mint a paraméterül kapottból, továbbá a paraméter nem `null`. (1 pont)

2. rész (12 pont)

`shop.Shop` osztály:

Az osztály egy üzletet reprezentál, és hibásan lett megírva. Az alábbi leírás alapján javítsuk ki a hibás részeket! A hibák a következőképpen oszlanak el:

Fordítási hiba: 5 (5 pont)
Futási hiba: 2 (4 pont)
Szemantikai hiba: 1 (3 pont)

Az utóbbi két kategória hibáinak megtalálásában a 3. rész megoldása nyújthat plusz segítséget!

- Az osztály egy rejtett játéklista adattagban tartsa nyilván (ArrayList), hogy milyen játékok vannak raktáron.

  Segítség: `ArrayList`

    - létrehozás: `ArrayList<Game> list= new ArrayList<>();`
    - új elem hozzáadása: `list.add(game);`
    - indexelés: `Game g = list.get(i);`
    - méret: `list.size();`

- Az osztálynak legyen egy publikus konstruktora, amely játékok listáját kapja paraméterként. A konstruktor inicializálja a sorozat adattagot a listát használva, ügyelve arra, hogy a belső állapot ne szivárogjon ki. Feltesszük, hogy egyik elem sem `null`.

- Az osztály tartalmazzon egy a listát visszaadó `getGames()` metódust. Itt figyeljünk arra, hogy a visszaadott listához ne lehessen új elemeket hozzáfűzni illetve belőle törölni. Az egyes játékokhoz viszont megengedett kívülről hozzáférni.

- Definiáljunk egy `numberOfGames` nevű metódust, amely visszaadja az üzletben található játékok számát.

- Definiáljunk egy paraméter nélküli `toString` nevű metódust is, amely visszaadja az üzlet szöveges reprezentációját. Az egyes termékeket sortörés vagy szóköz karakter is elválaszthatja. A szöveg összeállításakor a termékek olyan formában szerepeljenek, ahogyan a `Game` `toString` nevű metódusa előállítja őket. Az utolsó játék után opcionálisan lehet sortörés vagy szóköz.

- `browseGames(Platform p)`: a metódus lehetővé teszi a játékok közötti böngészést. Egy platformot kap paraméterként és egy listában visszaadja azon játékokat, melyek az adott kategóriába tartoznak. Ha az üzlet nem rendelkezik egyetlen olyan termékkel sem, mely megfelel a követelménynek, akkor a metódus egy üres listát ad vissza. A végeredmény típusa `List` legyen, megadva az elemek típusát is.

- `bestSellingInList(List<Game> list)`: osztályszintű metódus, a paraméterként kapott lista legjobban fogyó játékát adja vissza (egy `Game` típusú objektumot). Ha a lista üres, akkor `null`-t adjunk vissza. Ha több egyformán jól fogyó van, akkor az elsővel térjünk vissza.

3. rész (15 + 3 pont)

A `main.Main` osztályban szimuláljuk az üzlet működését. Az osztály az alábbi publikus metódusokat implementálja:

- `readStock(String fileName)`: a paraméterként megadott fájlból beolvasva létrehozza és visszaadja játékok (`Game`) egy listáját. A fájlban többféle adat is szerepelhet, ebben a metódusban azokat fogjuk feldolgozni, melyek a következő formátumúak:

  `Game:<cég>;<cím>;<platform>;<db>.`

  Ha ilyen sort olvasunk be a fájlból, a megfelelő mezőkkel a létrehozzuk a játék objektumokat, majd egy listában visszaadjuk azokat.(Figyeljünk arra, hogy az utolsó paraméternek számnak kell lennie, a többi ellenőrzését elvégzi a játék `make` metódusa.) (6 pont)

- `simulate(Shop s, String fileName)`: a metódus az üzlet egy napját szimulálja. Az inputfájl

  `Sell:<cím>;<platform>;<db>`

  formátumban vételi ajánlatokat tartalmaz, ezeket végigolvasva a Shop objektumban tárolt játékokból kikeresi az egyező címűt és platformút, majd eladja az adott darabszámot, amennyiben ez lehetséges (`buy` metódus). (itt is figyeljünk arra, hogy más típusú sorok is lehetnek az input fájlban illetve hogy a darabszám egy egész szám legyen) Ha nincs meg a megadott darabszám a játékból, hagyjuk a sort figyelmen kívül. (4 pont)

- `print(Shop a, Sring fileName)`: a paraméterként kapott nevű fájlba írjuk ki minden platform legjobban fogyó játékát soronként a következő módon, ha létezik ilyen játék (azaz a `Shop.bestSellingInList` nem `null`-lal tér vissza): (3 pont):

  `<platform> : <cég>: <cím> (<eladott db> pcs)`

  itt használjuk ki, hogy már megírtuk a játék `toString` metódusát, mely majdnem ezt az eredményt adja.

- `public static void main(String[] args)`: a program belépési pontja, 2 szöveges paramétert vár, az első egy input-, a második egy outputfájl neve. Ha nincs meg a két paraméter, a metódus lépjen ki. Az inputfájlt beolvasva kapott játéklistából hozzunk létre egy `Shop` objektumot, majd ugyanazon fájl felhasználásával szimuláljunk egy eladási sorozatot rajta. Végül a `print` metódussal írassuk ki a végső állást az outputfájlba! (2 pont)