Programozás I. Nagy zh

SZTE Szoftverfejlesztés Tanszék

2024. tavasz

Általános követelmények, tudnivalók

- A feladat elkészítésére 50 perc áll a rendelkezésre. Ez szigorú határidő, a Bíró előre megadott időben zár.
- A feladatokat számítógép előtt kell megoldani, tetszőleges fejlesztői környezetben, tetszőleges operációs rendszer segítségével.
- Az elkészült programot **20** alkalommal lehet benyújtani, a megadott határidőig.
- Csak a leírásban szereplő osztályokat, metódusokat és adattagokat kell megvalósítani, egyéb dolgokért nem jár plusz pont.
  - Aki Windowst használ, annak a gép elindítása után érdemes egyből a fejlesztőkörnyezetet elindítani, és létrehozni egy új projektet, és csak utána a böngészőt, mivel az elején egy néhány percig indexel, addig pont el lehet olvasni a feladatot.
- Bármely segédanyag használata **tilos** (a fejlesztőkörnyezetek nyújtotta segítségen kívül), aki mégis ilyet tesz, vagy próbálkozik vele, annak a dolgozata nem értékelhető és a ZH nem teljesített. Ha valakinek a padtársa segít, akkor mérlegelés nélkül mindkettő hallgató dolgozata sikertelen, a ZH nem teljesített.
- A feladat megoldása során minden megadott előírást pontosan követni kell! Tehát, ha a feladat leírása szerint egy adattag neve a "elsoFoku", akkor az alábbi elnevezések nem megfelelőek: "elsőFokú", "elsofoku", "elso_foku", "elsőFoq". Ugyanez igaz a metódusok, osztályok elnevezésére is!
- A metódusok esetében a visszatérési típus, a név, módosítók és a paraméterek típusai (és azok sorrendje) kerülnek ellenőrzésre, azonban a paraméterek nevei tetszőlegesek lehetnek.
- A Java elnevezési konvenciókat követni kell (getter/setter elnevezés, toString, indentálás, stb). Abban az esetben is, ha ezt a feladat külön nem emeli ki, az ellenőrzés során erre is építünk.
- A nem forduló kódok nem kerülnek kiértékelésre, ezt utólagosan a gyakorlatvezető sem bírálhatja felül. (Hiszen mindenki rendelkezésére áll a saját környezete, ahol fordítani, futtatni tudja a forráskódot, így feltöltés előtt ezt mindenképpen érdemes megnézni!)
- Az adattagok és konstruktorok hiányában garantáltan 0 pontos lesz a kiértékelés, mert ezek minden teszt alapját képezik.
- Ha végtelen ciklus van a programban, akkor ezt a Bíró ki fogja dobni 3 másodperc után (ha többször is meghívásra kerül ilyen metódus, akkor ez többszöri 3 másodperc, összesen akár 2 perc is lehet). Ilyenkor NE kattints még egyszer a *Feltöltés* gombra, mert akkor

kifagyhat a Bíró, csak a böngésző újraindításával lehet megoldani a problémát (emellett elveszik 1 feltöltési lehetőség is).

- Kérdés/probléma esetén a gyakorlatvezetők tudnak segítséget nyújtani.
- **A feladat megoldása során a default csomagba dolgozz, majd a kész forrásfájlokat tömörítve, zip formátumban töltsd fel, azonban a zip fájlt tetszőlegesen elnevezheted!**
  - Zip készítése: Windowson és Linuxon is lehet a GUI-ban jobb klikkes módszerrel tömörített állományt létrehozni (Windowsban pl. a 7-Zip nevű ingyenes program használatával).
  - Linux terminálon belül például a "zip feladat.zip *.java" paranccsal is elkészíthető a megfelelő állomány.
- A feladatokban az alábbi dolgok az alapértelmezettek (**kivéve**, ha a feladat szövege mást mond)
  - az osztályok láthatósága publikus
  - az egész érték 32 bites
  - a lebegőpontos számok dupla pontosságúak
  - az olyan metódusok void visszatéréssel rendelkeznek, amelyeknél nincs specifikálva visszatérési típus.
  - a metódusok mindenki számára láthatóak
  - az adattagok csak az adott osztályban legyenek elérhetőek
- A *riport.txt* és a fordítási log fájlok megtekinthetőek az alábbi módon:
  1. Az *Eredmények megtekintése* felületen a vizsgálandó próba új lapon való megnyitása
  2. A kapott url formátuma: `https://biro.inf.u-szeged.hu/Hallg/IB204L-1/1/hXXXXXX/4/riport.txt`
  3. Az url-ből visszatörölve a 4-esig (`riport.txt` törlése) megkaphatók a 4-es próbálkozás adatai.
- Szövegek összehasonlításánál az egyezés a pontos egyezést jelenti, azaz ha kis-nagy betűben térnek el, akkor már nem tekinthetők egyenlőnek (pl. a "piros" != "Piros")
- A leírásokban bemutatott példákban a stringek köré rakott idézőjelek nem részei az elvárt kimenetnek, azok csak a string határait jelölik. Például ha az szerepel, hogy a példa bemenetre az elvárt kimenet az, hogy `"3 alma"`, akkor az elvárt kimenet idézőjelek nélkül a `3 alma`, de a szóköz szükséges!
- Az elvárt kimeneteknek karakterről karakterre olyan formátumúnak kell lennie, ami a feladatban le van írva (szóközöket és sortöréseket is beleértve).

A feladat megoldása során használd a `minta.zip` fájlban lévő kiindulási projektet! A projekt egy egyszerű boltot valósít meg, ahol a vásárlók majd termékeket tudnak vásárolni a rendelkezésre áll pénzük és a készlet erejéig.

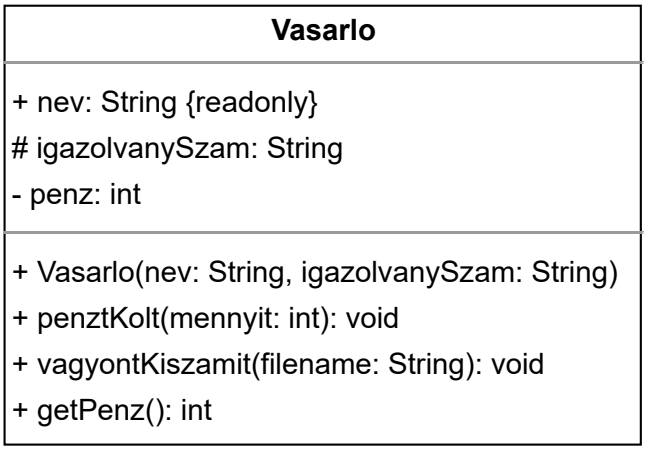
1. feladat: Vásárlók és vállalkozások (15 pont)

1.1. feladat: Vállalkozások (3 pont)

Módosítsd a `Vallalkozas` osztályt az alábbiak szerint!

- A *befektetesek* kollekció tartalmazza a vállakozásnak a különböző befektetéseiben lévő pénzmennyiségeit. A probléma az, hogy ha két befektetésben is ugyanannyi pénze van a vállalkozásnak, akkor a kettő közül csak az egyik kerül bele a kollekcióba. Módosítsd az adattagot (a többi kód módosítása nélkül), hogy azt is el tudja tárolni, ha több befektetésében is ugyanannyi pénzt tárol a vállalkozás. **(1 pont)**
- A *penztKolt()* metódus bizonyos esetekben egy `IllegalArgumentException` típusú kivételt dob. Cseréld ki a dobott kivételt egy *saját kivétel osztályra* úgy, hogy a dobott kivétel hibaüzenete ne változzon! A *penztKolt()* metódus fejlécén ne változtass! **(2 pont)**

1.2. feladat: Vásárlók (8 pont)



Készítsd el a `Vasarlo` osztályt a mellékelt UML diagram alapján! **(1 pont)**

Készítsd el az osztály konstruktorát! A konstruktor várja a vásárló nevét és az igazoványszámát, és azok alapján beállítja az adattagokat. Kezdetben a vásárlónak nincs pénze. **(1 pont)**

A *getPenz()* metódus visszaadja, hogy a vásárlónak mennyi pénze van. **(0 pont)**

A *penztKolt()* metódus paraméterben várja, hogy a vásárló mennyi pénzt szeretne költeni, tehát az ember pénzmennyisége ennyivel csökkenjen. Amennyiben a vásárlónak nincs elég pénze, akkor dobj egy ugyanolyan típusú kivételt, ugyanazzal a hibaüzenettel, mint a `Vallalkozas` osztály *penztKolt()* metódusában. **(2 pont)**

A *vagyontKiszamit()* metódus paraméterben egy fájlnak a nevét várja. A metódusnak ebből a fájlból kell olvasnia. A fájl minden sora egy bevételt vagy kiadást ír le. A fájlban nem tudjuk, hogy hány sor van. A metódus feladata, hogy a fájl tartalma alapján módosítsa a vásárló pénzét.

A fájl minden sora úgy néz ki, hogy először a bejegyzés dátuma látható, utána az, hogy bevételről vagy kiadásról van szó, végül pedig a pénzmozgás mennyisége olvasható. Az adatok pontosvesszővel vannak egymástól elválasztva.

Példa fájl:
2024.04.11;BEVETEL;200
2024.04.13;BEVETEL;1000
2024.04.17;KIADAS;500
2024.04.22;BEVETEL;700

Ha az embernek kezdetben 1000 pénze volt, akkor a végén 2400 pénze lesz (1000+200+1000-500+700). **(4 pont)**

1.3. feladat: Közös rész (4 pont)

Bár a vásárló és a vállalkozás egymástól elég távol áll, mégis van bennük valami közös. Minden olyan kódrészletet, amit tudsz, szervezz ki egy interface-be, amelyet ezután mind a két osztályod implementál.

2. feladat: Bolt (15 pont)

Készítsd el a `Bolt` osztályt!

Az osztály egy nagyon egyszerű bolt működését írja le, amely lehet nagykereskedés vagy kiskereskedés. A boltban különböző termékeket árulnak megadott áron (ez persze idővel változhat (akció/infláció)). A bolt mindegyik termékből véges árukészlettel rendelkezik, de természetesen időnként kapnak új szállítmányt az egyes termékekből. A vásárlók betérhetnek a boltba, és a raktáron lévő termékek közül válogathatnak, amelyekért - ahogy megszoktuk - pénzzel fizetnek.

A megoldást tetszőleges módon elkészítheted, ehhez tetszőleges új adattagokat, metódusokat, akár osztályokat is létrehozhatsz. Azonban fontos, hogy az osztály ne tartalmazzon publikus adattagot, illetve az elvárt 5+1 metóduson kívül ne legyen benne másik publikus metódus. A megoldás elkezdése előtt érdemes végigolvasni az összes feladatot, a feladat pontos megértése céljából!

Az osztály rendelkezzen egy 1 paraméteres konstruktorral, amely egy logikai értéket vár. Ez az érték igaz, ha nagykereskedésről van szó, hamis ha kiskereskedésről.

Először is szükségünk van egy adott termékek árának és mennyiségének lekérdezésére. Készíts egy metódust, amely paraméterben egy termék nevét várja, és egy 2 elemű tömbben adja vissza az adott termék adatait. A tömb első eleme a termék árát, míg a második eleme a termék mennyiségét tartalmazza. Amennyiben a keresett terméket a bolt nem árusítja, akkor a tömb mindkét eleme *null* legyen! Ha a bolt bármikor is árusított egy terméket, akkor a visszaadott értékek akkor se legyenek null-ok, ha éppen nincs a termék raktáron (lásd: későbbi metódusok)! **(0 pont)**

FONTOS! A megoldások tesztelése a fenti lekérdező metódus segítségével történik, ami azt jelenti, hogy ennek helyes megvalósítása nélkül az összes többi feladat 0 pontot ér. Feltöltés előtt ezt a metódust mindenképpen készítsd el, különben az osztályra a Bíró nem fog pontot adni!

Az osztálynak az alábbi 4 funkciót kell megvalósítania:

A boltnak tudnia kell fogadni új szállítmányokat. Az erre használható metódus paraméterben várja az áru nevét, illetve az áru mennyiségét, azaz, hogy hány darabot hoztak az adott termékből.

A kapott áruk az eddigi készlethez hozzáadásra kerülnek (pl. ha volt csokiból már eddig 10, de most hoztak még 5-öt, akkor most már 15 csoki van összesen a boltban).

Amennyiben egy új fajta termékről van szó, akkor a bolt felveszi a kínálatába az új terméket, 1000 Ft-os darabonkénti kezdőárral.

Ha egy bolt már korábban bármikor árusított egy terméket, akkor annak ára nem áll vissza az 1000-es értékre (még akkor sem, ha volt olyan időszak, amikor az adott termék nem volt raktáron). **(4 pont)**

A boltnak reagálnia kell az árváltozásokra. Az erre használható metódus várja a termék nevét, a minimum árát, illetve az ajánlott árát. Egy nagykereskedés az adott termék új árát a minimum árban, míg egy kiskereskedés az ajánlott árban határozza meg. Természetesen előfordulhatnak olyan esetek, amikor egy olyan termék ára módosul, amit az adott bolt nem árul. Ilyenkor a boltnak semmilyen teendője nincs az árváltozással kapcsolatban. **(2 pont)**

A bolt weboldalán szeretné listázni a termékeinek a nevét. Ehhez kétféle rendezést kell biztosítani: ABC szerinti, illetve ár szerinti (növekvő) sorrendet. Készíts egy metódust, amely paraméterben várja, hogy mi alapján szeretnénk rendezni (true: ár alapján; false: ABC szerint). A metódus egy megfelelő méretű tömbben adja vissza a termékek nevét, a kért módon rendezve! **(3 pont)**

A bolt legfontosabb funkciója azért mégis csak az, hogy a vásárlók tudnak vásárolni benne. Készíts egy metódust, amely a vásárlás lebonyolításáért felelős. A metódus megkapja a vásárló személyét, illetve egy bevásárlólistát. A bevásárlólista egy Map, amelyben a kulcsok a termékek nevei, az értékei pedig a vásárolni kívánt mennyiség. A metódus bonyolítsa le a vásárlást az alábbiak szerint:

- A pénztár előtt a vásárló kiszámolja, hogy van-e elég pénze a kosara tartalma alapján. Ha nincs elég pénze, akkor inkább mindent visszapakol a polcokra és üres kézzel távozik (ha egy termékből nincs elegendő mennyiség a boltban, akkor nyilván csak azzal számolunk, ami van).

- Ha van elég pénze, akkor a bevásárlólistáján szereplő termékekből a megadott mennyiségű terméket megvásárol. Ilyenkor (ahogy várnánk), a bolt raktárkészlete csökken a megadott mennyiséggel, illetve a vásárló kifizeti a termékeket a bolt részére. Előfordulhat, hogy egyes termékekből egyáltalán nincs a boltban, vagy nincs annyi, amennyit a vásárló szeretne. Előbbi esetben az adott tételt a vásárló kihagyja, utóbbi esetben annyit vásárol, amennyi készleten van a boltban.

- A metódus adja vissza, hogy a vásárló meg tudta-e vásárolni az összes terméket, amelyet a bevásárlólistájára felírt. **(6 pont)**

Jó munkát!