6. feladatsor

Kezdés: márc 22, 14:31

Kvízinstrukciók

1. kérdés 0 pont

Programozási nyelvek (BSc, 18) Java 6. feladatsor

1. feladat

a

Módosítsa az **1. feladatsor 7.** feladatának megoldását úgy, hogy az operandusok `double` típusúak legyenek, valamint az elvégezendő műveletet is parancssori argumentumként fogadja a program. A program csak a kért típusú alapműveletet végezze el.

Amennyiben nem megfelelő számú argumentummal hívták meg a programot, akkor a `main()` függvény dobjon `IllegalArgumentException` kivételt. Nullával való osztás esetén dobjon `ArithmeticException` kivételt; nem támogatott alapművelet esetén pedig `IllegalArgumentException` kivételt.

b

Módosítsa az előző megoldást úgy, hogy a `main()` függvény kapja el a dobott kivételeket, és ezek előfordulása esetén általános hibaüzeneteket írjon ki a képernyőre (például nem megfelelő számú argumentum esetén "Invalid program arguments provided."). A `parseDouble()` konverziós metódus érvénytelen sztring esetén `NumberFormatException` kivételt dob; kapja el ezt a kivételt is.

c

Módosítsa az előző megoldást úgy, hogy a kivételek konstruálásakor informatív üzenetet ad át a kivétel konstruktorának; a kivétel kezelésekor írja ki a kivétel objektumban tárolt üzenetet.

2. feladat

a

A bemeneti fájlunk sorai vesszővel elválasztott egész számokat tartalmaznak. Soronként adjuk össze őket, és írjuk ki az összegüket a képernyőre!

```
1,2,5,-2
10,20,0,7
3,2
2
0
```

```
1,2
3
```

Oldjuk meg `BufferedReader`-rel!

Keressünk a `String` osztályban olyan metódust, mely alkalmas rá, hogy egy speciális karakter (most a vessző) mentén feldarabolja a sorunkat. Feltehetjük, hogy a bemenet formátuma helyes.

b

Módosítsuk az előző megoldást úgy, hogy képernyő helyett egy másik fájlba írja ki az eredményeket.

3. feladat

Egy parancssori argumentumként megadott fájlnevű fájlban keressünk meg egy kapott szövegrészletet!

A szövegrészletet kérjük be a felhasználótól a billentyűzetről.

Írjuk ki, hogy hány sorban fordult elő a keresett szövegrészlet. Ne csak akkor számítsuk találatnak, ha az egész sorral megegyezik, akkor is vegyük figyelembe, ha a sor csak tartalmazza a keresett szövegrészletet! (Keressünk megfelelő metódust a `String` osztályban!)

```
hello
__hello2
hello
    hello

                hello
```

4. feladat

Ebben a feladatban a **3. feladatsor 4/b** feladatát érdemes továbbfejleszteni.

a

Írjon a `Circle` osztályhoz statikus `readFromFile()` metódust, amely betölti a paraméterként kapott fájlnevű fájlból egy kör adatait (`x` és `y` koordináta és sugár újsorral elválasztva), megkonstruál ezen adatokkal egy `Circle` objektumot, majd visszatér az objektum referenciájával. A függvény a fellépő kivételeket engedje tovább a hívóhoz.

Írjon a `Circle` osztályhoz `saveToFile()` metódust, amely az aktuális `Circle` objektum adatait a paraméterként megadott fájlnevű fájlba menti. Amennyiben a fájlba írás kivételes eseménybe ütközik, a függvény engedje tovább a kivételes eseményt a hívóhoz. Gondoskodjon arról, hogy ha a kiírás menet közben ütközik kivételes eseménybe, akkor a már kiírt adatok ne vesszenek el (azaz a `Writer` objektumot mindenképpen be kell zárni).

Próbálja ki mi a különbség ha a hívott függvények kivételeit elkapja illetve nem kapja el a főprogramban.

b

Készítse el az **a** megoldás egy olyan változatát, amelyben a `readFromFile()` metódus megpróbálja kezelni a fellépő kivételes eseményeket: ha valamilyen kivételes esemény miatt nem sikerül a beolvasás, akkor jöjjön létre kör objektum valamilyen értelmes kezdőértékekkel.

1. gyakorló feladat

Egy szöveges fájl minden sorában található egy egész szám, majd szóközzel elválasztva egész számok vesszővel elválasztott listája.

Olvassa be a fájl sorait, majd döntse el, hogy az egész számok listájában van-e két olyan egész szám, amelyek összege az első oszlopban lévő szám.

Az eredményeket írja ki egy szöveges fájlba: soronként a szám, amely összeg-felbontását keressük, majd szóközzel elválasztva a két listabeli szám, amelyek összege a vizsgált szám; ha nincs a listában két megfelelő egész, akkor a "none" sztringet írja a szám mellé.

Például:

in.txt:

```
7 2,5,-7,6,9
-2 2,5,-7,6,9
12 2,5,-7,6,9
```

out.txt:

```
7 2 5
-2 5 -7
12 none
```

2. gyakorló feladat

Készítsünk programot, amely karaktereket ír ki a standard kimenetre egy `in.txt` szöveges fájlból.

A standard bemenetről olvassa be, hogy hány karaktert szeretne a felhasználó kiíratni és parancssori argumentumként kapjon egy egész típusú értéket, amely azt határozza meg, hogy hány karakter maradjon ki minden beolvasás után (használja a `BufferedReader` `skip()` metódusát).

A karakterenkénti olvasáshoz használja a `BufferedReader` `read()` metódusát.

Kezelje le a `NoSuchElementException` (vagy `InputMismatchException`) kivételeket a parancssori argumentum beolvasás és parse-oláskor, illetve a felhasználói beolvasáskor.

3. gyakorló feladat

A NIO API-t használva készítsen programot, amely egy `BufferedReader` segítségével egy `nums.txt` fájlbeli számokról megállapítja, hogy párosak-e és ezt kiírja egy `BufferedWriter`-rel az `out.txt` fájlba a következő módon:

```
"x is an even number", ha x páros
"x is an odd number", ha x páratlan
```

A `nums.txt` fájlban soronként egy szám található. Használja a try-with-resources megközelítést!

4. gyakorló feladat

Készítsünk egy egyszerű `Color` felsorolt típust, mely a következő értékeket tárolhatja: `RED`, `GREEN`, `BLUE`.

Írjunk egy `Auto` osztályt, mely a következő információkat tárolja:

- rendszám (`String`)
- szín (`Color`)
- maximális sebesség (`int`)

Írjunk az osztályhoz egy konstruktort, mely ezt a három értéket várja paraméterként. Az osztályban legyen számláló, mely tárolja, hogy hány objektumot hoztunk eddig létre.

Írjunk egy paraméter nélküli konstruktort is, mely `AAA-000`, `BLUE` és `120` értékekkel hoz létre objektumot.

Írjunk egy osztályszintű összehasonlító metódust, mely két autó objektumot vár, és igazzal tért vissza, ha az első gyorsabb mint a második.

Helyezzük a `Color` osztályt az `auto.utils` csomagba, az `Auto`-t pedig az `auto` csomagba!

Hozzunk létre egy `input.txt` fájlt, melyben autók adatai vannak soronként megadva, vesszővel elválasztva. Pl: ABC-123,RED,100

Írjunk egy `Main` osztályt (a csomogokon kívül), amely a tesztprogramunkat fogja tartalmazni! A `Main` osztály `main()` metódusában olvassuk be az input fájl tartalmát, a létrehozott objektumok referenciáit pedig tároljuk el egy tömbben.

5. gyakorló feladat

Készítse el az előadáson már előforduló `Time` osztályt teszőlegesen választható időreprezentációval. Kezelje le az előfoduló kivételeket és legyen egy metódusa, ami `String`-ként visszaadja az időt.

Készítsen egy `Pizza` osztály, amelynek legyen az összes mezője `private` és `final`; egy mezője az átmérő (`double`), egy a feltétek listája (`String[]`), egy az elkészítési idő (`Time`) és egy a szállítási idő (`Time`). Írjon hozzá konstruktort, amely a feltétek szerint kiszámolja, hogy mennyi az elkészítési idő, tehát az ne legyen paraméterül átadva a konstruktornak. Az elkészítési idő a következő képlet szerint történjen: az összes feltétre

a következők összege: a feltét nevének hossza szorozva a pizza átmérőjével (centiméterenként és feltétbetűnként egy másodperc).

Pl.: 32 cm-es pizza "cheese" feltéttel: `32*6 = 192`, amit kerekítsünk 4 percre. A konstruktor dobjon kivételt, ha valamely paraméter nem értelmezhető. Dobjon `TypeNotPresentException`-t, ha a feltét nincs benne a következő listában: `beef`, `cheese`, `corn`, `fish`, `ham`, `mushroom`, `salami`, `tomato`. Jelenítse meg a `throws` kulcsszóval, hogy a `Pizza` osztály példányosításakor le kell kezelni a `TypeNotPresentException`-t.

Írjon `Main` programot, amelynek egy statikus metódusa beolvas a paraméterként átadott fájnévű fájlból pizzarendeléseket, és létrehozza egy listában az összes lehetséges pizzát, majd kiszámítja, hogy mennyi ideig kell aznap sütni. Kezelje le a kivételeket, amelyek felléphetnek a pizzák létrehozása során.

Végül mutassa be ennek a `Main` osztálynak statikus metódusának használatát a `Main` osztályon belül.

Feltöltés Fájl kiválasztása

Nincs elmentve Kvíz beadása