Programozási nyelvek (BSc, 18) Java 5. gyakorlat

1. feladat

Módosítsa az 1. gyakorlat 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.

2. feladat

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.

3. feladat

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.

4. feladat

A bemeneti fájlunk sorai vesszővel elválasztott egész számokat tartalmaznak. Soronként adjuk össze őket, és írjuk ki egy másik fájlba!

```
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.

5. feladat

Egy parancssori argumentumként megadott 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
```

1. 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.

2. 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
```

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!