6. gyakorlat anyaga

Feladatok

1. feladat

Írjuk meg a `text.to.numbers.SingleLineFile.addNumbers()` metódust, ami egy fájlnevet vesz át, és az első sorában található számok összegével tér vissza.

- Ha nem létezik a fájl, az `IOException` kivétel terjedjen ki a metóduson kívülre.
- Ha a fájl üres, váltsunk ki `IllegalArgumentException` kivételt `Empty file` üzenettel.
- A fájl első sorának minden olyan szavát, ami nem értelmezhető `int` értékként, írjuk ki a `System.err` hibafolyamra. Erre ugyanúgy lehet írni, mint a `System.out` folyamra.
- A fájl használatának a végén erőforráskezelő try eszközzel garantáljuk a fájl bezárását.

Készítsük el az alábbi teszteseteket.

- A két kivételes eset tesztelése.
  - A tesztelő metódus egyszerűen kapja el a kivételeket.
    - Ha mégse váltódna ki kivétel, hívjuk meg a `fail()` metódust. Ez kaphat egy hibaüzenetet is.
    - A második esetben a kivétel szövege így kérhető le: `e.getMessage()`.
  - Mj.: a JUnit `assertThrows()` metódusával még elegánsabb megoldás adható, de ez egy haladóbb szintű eszköz, nem célunk használni.
- Egy érvényes fájl tesztelése, melynek első sora `1 2 not text 3 -123` tartalmú.
  - A kimenetre kiírt szövegeket nem teszteljük JUnit segítségével, de ott kell lenniük.
- Mj.: a fentiek nem valódi egységtesztek, mert a fájlrendszert is használják.

2. feladat

Írjuk meg az `text.to.numbers.MultiLineFile.addNumbers()` metódust. Ez a fájl összes sorának összes számát adja meg. Ennek legyen még egy paramétere, az elválasztójel karaktere.

Ezt is teszteljük. Az előzőhöz képest különbség: üres fájl esetén nem váltódik ki kivétel. Az érvényes fájlt teszteljük szóköz és vessző elválasztójellel egyaránt.

3. feladat

Az előző két megoldás ne csak a hibafolyamra, hanem a `wrong.in.txt` fájlba is írja ki a hibás szövegeket (ha a bemeneti fájl neve `in.txt`).

Ehhez nem kell JUnit tesztelőt írni.

4. feladat

A `textfile.Statistics.numberOfLines()` metódus kapjon meg egy fájlnevet, és adja vissza, hány sorból áll. Ha a fájl nem létezik, a visszatérési érték legyen `0`.

Ugyanebben az osztályban `numberOfCharacters()` adja meg egy fájlról, hány karaktert tartalmaz (sorvége jelek nélkül).

Mindkettő tesztelendő.

5. feladat

A `textfile.lookup.FileContent.contentLineCount()` metódus egy fájlnevet és egy szöveget kap meg, és megadja, a fájl hány sorában található meg ez a szöveg.

- Tipp: a `String` osztály dokumentációjában található egy hasznos metódus ehhez.

A `textfile.lookup.FileContent.contentLineNumbers()` metódus szintén egy fájlnevet és egy szöveget kap meg, és egy tömbben visszaadja az összes olyan sor sorszámát (egytől számozva), amelyben a szöveg megtalálható.

Mindkettő tesztelendő.

Gyakorló feladatok

1. gyakorló feladat

Készítsük el a `plane.util.CircleUtil.writeCircle()` metódust, ami egy fájlnevet és egy `plane.Circle` objektumot vesz át. Ez nyissa meg a fájlt, és írja bele a kör adatait a következőképpen:

```
12.34
-54.12
54.3
```

Készítsük el a `plane.util.CircleUtil.readCircle()` metódust, ami egy fájlnevet vesz át. A metódus a fájlból a fenti formátum szerint olvassa fel egy kör adatait, és ebből elkészít és visszaad egy `Circle` objektumot. Feltehető, hogy a fájl létezik, és a megfelelő módon tartalmazza az adatokat.

Készítsünk tesztelőt, amely egy kör adatait beleírja egy fájlba, majd visszaolvassa azokat, és megvizsgálja mindegyik adattagot.

- Az `assertNotSame()` meghívásával az is ellenőrizhető, hogy az eredeti kör és a beolvasott különböző objektumok.

2. gyakorló feladat

A `file.line.SumChecker.main()` kódja nyissa meg olvasásra a `sum_input.txt` fájlt, írásra a `sum_output.txt` fájlt. Az előbbi tartalma ilyen:

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

Készítsünk kódot, ami soronként megkeresi, hogy az első szám előáll-e a felsoroltak közül két szám összegeként.

- Két, egymásba ágyazott ciklussal kell bejárni a számokat.
- Ugyanazt a számot nem szabad kétszer felhasználni: a harmadik sorban nem `12=6+6` az eredmény.

A kimenetbe ez kerüljön:

```
7=2+5
-2=5+-7
12 none
12=6+6
```

Tipp: az alábbi `continue` utasítás hasznos lehet:

```
outer:
ciklus (...) {
    ciklus (...) {
        ...
        continue outer;
    }
}
```

Extra feladat

Az extra feladatok nem képezik a tananyag részét. A megoldásaid teamsen vagy emailben is elküldheted, ha szeretnél visszajelzést, pluszpontot vagy pacsit kapni cserébe.

Az alábbi feladathoz érdemes a ByteBuffer osztályt használni, amit pl. egy FileChannel segítségével bírhatunk rá fájlok olvasására és írására.

Fejlett fájlkezelés

A zip fájlok szerkezete ismert (pl. ennek a leírásnak a 4.3.7. fejezete tartalmazza). Tömöríts be tetszőleges fájlokat (fájlkezelő segítségével), majd az előállt zip fájlt megvizsgálva írd ki a betömörített fájlok

- nevét,
- méretét,
- az alkalmazott tömörítési módszert (ha 0, egyszerűen tárolva van a fájl tartalma)

Tipp: a `ByteBuffer` alapértelmezés szerint MSB bájtsorrendet használ, mint a Java általában; a zipből való megfelelő olvasáshoz meg kell hívni rá a `.order(ByteOrder.LITTLE_ENDIAN)` műveletet.

A betömörített fájlokat mentsd is el, ha szimplán tárolva vannak! (A bájtsorrendet ekkor nem kell megforgatni.)

Készíts zip "tömörítőt", ami a tetszőleges számú parancssori argumentumban megadott fájlt egy zipben tárol!