Programozás 2

– 2. zárthelyi dolgozat, pót-ZH –

2021. december 9., 18.00

1. feladat – szuperhősök (1 pont)

Tekintsük a `superhero.csv` állományt, melynek minden egyes sora egy-egy szuperhős nevét és képességeit tartalmazza. Példa:

```
Mr Cypher;Sonar;Entropy Projection;Vitakinesis
```

Az egyes oszlopok `';'`-vel vannak elválasztva. Az első oszlopban a szuperhős neve szerepel, majd a további oszlopokban a képességei. Egy szuperhősnek egy vagy több képessége is lehet. A fenti példában Mr Cypher három képességgel rendelkezik.

Készítsen egy `Superhero` nevű osztályt, mellyel egy szuperhőst tud reprezentálni. Ezt a `Superhero.java` állományban helyezze majd el. Olvassa be a `superhero.csv` tartalmát, s minden egyes sort alakítson át `Superhero` típusú objektummá. Egy `Superhero` objektumban tárolja el a hős nevét, ill. tárolja el a képességeit is egy sztringeket tartalmazó listában. A `Superhero` objektumokat gyűjtse össze egy listában.

Írjon egy programot, aminek interaktív módon meg kell adni egy szuperképességet. A szuperképesség megadásakor nem számít a kis- és nagybetű (nincs köztük különbség). A program írja ki a képernyőre az adott képességgel rendelkező hősök nevét ábécésorrendben. A hősök neve mellett zárójelben tüntesse fel azt is, hogy az adott hős hány darab képességgel rendelkezik! Ha az adott képességgel egyetlen hős sem rendelkezik, akkor ne legyen semmilyen kimenet sem.

Futási példák:

```
$ java Main
Képesség: lebegés
$

$ java Main
Képesség: sonAr
HeroA (2)
HeroB (3)
HeroC (5)
...

$ java Main
Képesség: SoNaR
HeroA (2)
HeroB (3)
HeroC (5)
...
```

A `lebegés` esetén a `$` azt jelenti, hogy visszakaptuk a promptot, nem volt semmilyen kimenet sem. A szonáros példánál nem a valódi kimenet lett feltüntetve. A kimeneten a “HeroA (2)” sor jelentése: a HeroA nevű hős 2 db szuperképességgel rendelkezik.

2. feladat – szuperképességek (1 pont)

Vegyük ismét az előző feladatban már látott `superhero.csv` állományt. Az állomány felépítésének a leírását az előző feladat elején találja meg.

Írjon egy programot, ami beolvassa az állomány tartalmát s kiírja, hogy az állományban hány *különböző* szuperképesség található.

Vegyük például a következő állományt (`example.csv`):

```
HeroA;Portal Creation;Armor;Vision;Sonar
HeroB;Sonar;Entropy Projection;Vitakinesis
HeroC;Healing Factor;Echokinesis;Healing Factor;Vision
```

Ebben a példában 8 *különböző* szuperképesség található.

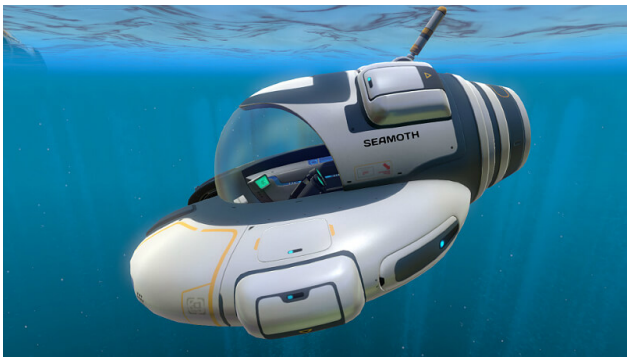
Futási példa:

```
$ java Main
234
```

234 helyett természetesen a helyes értéket kell majd feltüntetni.

Figyelem! A programnak a `superhero.csv` állományt kell feldolgoznia!

3. feladat – mélytengeri varázs (1 pont)



Az egyszemélyes tengeralattjárónkkal felfedezőútra indulunk. A tengeralattjáró háromféle utasítást ért meg:

- a `forward X` hatására vízszintes irányban előremegy `X` egységet
- a `down X` hatására lesüllyed `X` egységet (ekkor *nő* a mélység)
- az `up X` hatására felmegy `X` egységet (ekkor *csökken* a mélység)

A tengeralattjáró már rendelkezik egy betáplált útvonallal (ez lesz a programunk bemenete). Vegyük például a következő utasítássorozatot:

```
forward 5
down 5
forward 8
up 3
down 8
forward 2
```

Kezdetben mind a vízszintes helyzetünk, mind a mélységünk 0. Ez a két érték a fenti utasításokat követve a következőképpen változik:

- `forward 5`: 5-öt hozzáadunk a vízszintes pozícióhoz (aktuális értéke: 5)
- `down 5`: a mélységhez hozzáadunk 5-öt (aktuális értéke: 5)
- `forward 8`: 8-at hozzáadunk a vízszintes pozícióhoz (aktuális értéke: 13)
- `up 3`: csökken a mélység 3-mal (aktuális értéke: 2)
- `down 8`: a mélységhez hozzáadunk 8-at (aktuális értéke: 10)
- `forward 2`: 2-t hozzáadunk a vízszintes pozícióhoz (aktuális értéke: 15)

Az utasítássorozat végrehajtása után a vízszintes pozíciónk 15 lett, míg a mélységünk 10. A kettő szorzata: 150.

Feladat: kövessük az `input.txt`-ben található utasításokat s állapítsuk meg, hogy hova érkezünk. Mennyi lesz a vízszintes pozíciónk és a mélységünk *szorzata*?

Futási példa az `example.txt` esetén:

```
$ java Main
vízszintes pozíció: 15
mélység: 10
---
végeredmény: 150
```

Figyelem! A programnak az `input.txt` állományt kell feldolgoznia!

4. feladat – palindróm prímek (1 pont)

Legyen adott a következő PHP állomány (lásd `example.php`):

```
<?php $data = array (
  0 => 2,
  1 => 3,
  2 => 5,
  3 => 7,
  4 => 11,
  5 => 13,
  6 => 17,
); ?>
```

Az állományban a nyíl (`=>`) után a prímszámok vannak felsorolva. A nyíl előtt az indexek láthatók.[1]

Olvassuk be az input állományt, s írjuk ki a palindróm prímszámokat az `output.txt` nevű állományba.[2] A kiírás során soronként egy számot írjunk ki.

A fenti példa (`example.php`) esetén az `output.txt` fájl tartalma a következő lenne:

```
2
3
5
7
11
```

Ezek mind palindróm számok, míg a 13 és 17 nem palindróm.

A programot úgy írja meg, hogy az a `primes.php` fájl tartalmát dolgozza fel. A program a futtatás során adjon egy kis visszajelzést:

```
$ java Main
-> primes.php beolvasva
-> output.txt létrehozva
```

Figyelem! A programnak a `primes.php` állományt kell feldolgoznia!

[1] A primes.php az első százezer prímszámot tartalmazza.
[2] Egy szám akkor palindróm, ha visszafelé olvasva is ugyanazt a számot kapjuk. Pl. a 101 palindróm.