Programozás 2

– 1. zárthelyi dolgozat (Java) –

2022. október 12., 8.00

1. feladat – games (1 pont)

Írjon egy `Game` nevű osztályt, mellyel egy videójátékot tudunk reprezentálni. Az osztályt egy `Game.java` nevű állományban helyezze el!

Az osztálynak egyetlen konstruktora legyen. A konstruktornak a játék címét, az értékelését, ill. a teljes végigjátszáshoz szükséges időt kell megadni. Példák:

```
Game g1 = new Game("Subnautica", "84%", "43 hours");
Game g2 = new Game("Danganronpa 1", "82%", "28.5 hours");
Game g3 = new Game("Immortals Fenyx Rising", "77%", "41.5 hours");
```

A címet sztringként, az értékelést egészként, míg a végigjátszáshoz szükséges időt valós számként tárolja el az objektumokon belül. Egy `Game` típusú objektum legyen *immutable*, vagyis miután létrehoztunk egy ilyen objektumot, azt utólag ne lehessen módosítani!

Példányosítás után az objektumokat a következőképpen akarjuk használni. A megjegyzésekben az elvárt kimenet látható:

```
System.out.println(g1);                         // Subnautica (84%), 43 hours to beat
System.out.println(g1.getTitle());              // Subnautica
System.out.println(g1.getScore());              // 84
System.out.println(g1.getTime());               // 43.0
System.out.println(g1.isBetterThan(g2));        // true
System.out.println(g3.isBetterThan(g2));        // false
System.out.println(g3.isLongerThan(g2));        // true
```

Egy játékot akkor tekintünk jobbnak, ha magasabb az értékelése. Egy játék akkor hosszabb, ha több ideig tart végigjátszani.

Folytatás

Írjon továbbá egy `Playlist` nevű osztályt, mellyel egy játéklistát tudunk reprezentálni. Az osztályt egy `Playlist.java` nevű állományban helyezze el! Az osztályban a játékok tárolását dinamikus tömbbel oldjuk meg. Használata:

```
Playlist winter = new Playlist();
winter.add(g1);
winter.add(g2);
winter.add(g3);

System.out.println(winter.getBestGame());       // Subnautica (84%), 43 hours to beat
System.out.println(winter.getLongestGame());    // Subnautica (84%), 43 hours to beat
System.out.println(winter.getShortestGame());   // Danganronpa 1 (82%), 28.5 hours to beat
```

Mint látható, a `getBestGame()` a legmagasabb értékelésű játékot, a `getLongestGame()` a leghosszabb játékot, míg a `getShortestGame()` a legrövidebb játékot adja vissza.

2. feladat – szűrés (1 pont)

Írjunk egy programot, aminek parancssori argumentumként pontosan három darab sztringet kell megadni:

```
$ java Main  KissAladar1983  ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789  A123
köztes eredmény: 'KA1983'
végeredmény: 'A13'
```

Vesszük az első sztringet (“KissAladar1983”), majd ebből csak azokat a karaktereket hagyjuk meg, melyek szerepelnek a második sztringben. Az eredményt (“KA1983”) kiírjuk a képernyőre. Majd vesszük ezt a sztringet (“KA1983”), s ebből megint csak azokat a karaktereket hagyjuk meg, melyek szerepelnek a harmadik sztringben. Az eredményt (“A13”) ismét kiírjuk a képernyőre. A képernyőre kiírt sztringeket tegyük aposztrófok közé, hogy jobban látszódjanak.

Mind a köztes eredmény, mind a végeredmény is lehet üres sztring.

A program tartalmazzon hibakezelést. Ha a felhasználó háromnál kevesebb vagy több parancssori argumentumot adott meg, akkor legyen egy informatív hibaüzenet, s a program lépjen ki 1-es hibakóddal.

3. feladat – egyedi karakterek (1 pont)

Írjon egy programot, ami interaktív módon bekér egy sztringet, majd kiírja ezen sztring megtisztított változatát. Az eredeti sztringből csak azokat a karaktereket hagyja meg, melyek egyedi karakterek, vagyis az eredeti sztringben csak egyszer fordultak elő. Példa:

```
$ java Main
Input: hello
Output: heo
```

Mivel az ‘`l`’ betű 2x is szerepelt az input sztringben, ezért ezt nem tartottuk meg. A ‘`h`’, ‘`e`’ és ‘`o`’ betűk csak 1x szerepeltek, így ezeket megtartottuk.

4. feladat – soronkénti összeg (1 pont)

Tekintsük a `numbers.txt` fájlt. Dolgozza fel az állományt soronként, s minden sorban állapítsa meg a sorban lévő számok összegét. A program írja ki a képernyőre, hogy

- mi a legkisebb összeg
- melyik sorban a legkisebb a számok összege
- mi a legnagyobb összeg
- melyik sorban a legnagyobb a számok összege

A legelső sor sorszáma 1 legyen!

Vegyük pl. a következő állományt (`example.txt`):

```
1 2 3
10 20 30
4 5
40 50 60 100
```

```
$ java Main
A legkisebb összeg (6) az 1. sorban található.
A legnagyobb összeg (250) a 4. sorban található.
```

Figyelem! A programnak a `numbers.txt` fájlt kell feldolgoznia! Az input állomány természetesen NEM módosítható!