Név: .....

Neptun-kód: .....

Programozás 2
– 1. zárthelyi dolgozat, pót-ZH –

2020. nov. 11.

Feladatok

1. (1 pont) Implementálja a `Path` nevű osztályt. A `Path` osztály segítségével egy linuxos állományt vagy könyvtárat tudunk reprezentálni. Az osztályt a következőképpen szeretnénk használni:

```
Path f1 = new Path("/home/pisti/text/cucc.txt");
Path d1 = new Path("/usr/bin/");

// hívás                              elvárt kimenet
// -----                              --------------
System.out.println(f1);             // /home/pisti/text/cucc.txt
System.out.println(d1);             // /usr/bin/
System.out.println(f1.isFile());    // true
System.out.println(f1.isDir());     // false
System.out.println(d1.isFile());    // false
System.out.println(d1.isDir());     // true
System.out.println(f1.fileName());  // cucc.txt
System.out.println(d1.fileName());  // (üres sztring)
System.out.println(f1.fileExt());   // .txt
System.out.println(d1.fileExt());   // (üres sztring)
```

Tudjuk a következőket:

- A `Path` osztállyal csakis állományokat és könyvtárakat akarunk reprezentálni. Egyéb lehetőség (pl. szimbolikus link) nincs.
- A standard könyvtárban lévő `File` osztályt nem lehet használni.
- Ha egy könyvtárat adunk meg, akkor a könyvtár végén mindig van egy `'/'` jel, pl. `"/usr/bin/"`.
- Állomány esetén mindig van kiterjesztés.
- A `Path` osztály szimbolikus feldolgozást tesz lehetővé. Vagyis a segítségével nem létező állományokat, könyvtárakat is lehet reprezentálni. A `Path` osztály metódusai csupán sztringműveleteket végeznek.
- A `fileName()` és `fileExt()` metódusok csak állományokra vannak értelmezve. Egy könyvtár esetén ezek adjanak vissza egy üres sztringet!
- Vegyük észre, hogy a `fileExt()` metódus a pont karaktert is visszaadja a kiterjesztés előtt!

2. (1 pont) Írjon egy programot, ami pontosan 2 db argumentumot vár. Hibás argumentumszám esetén írjon ki egy hibaüzenetet s a program lépjen ki 1-es hibakóddal!

   A két argumentum egy-egy sztring. Az első sztringből csak azokat a karaktereket tartsuk meg, amik szerepelnek a második sztringben. Az eredménysztringet írjuk ki a képernyőre. Az eredménysztringben a betűk sorrendje egyezzen meg az első sztringben lévő betűk sorrendjével!

   Az eredménysztringet egy ilyen szignatúrával rendelkező metódussal állítsa elő:

   ```
   public static String filterChars(String s, String t);
   ```

   Ahol `s` az első paraméterként megadott sztring, `t` pedig a második paraméterként megadott sztring.

   Ez a metódus a `Main` osztályban szerepeljen!

   Példák:

   ```
   $ java Main
   Hibás paraméterezés!
   $ echo $?
   1

   $ java Main aa
   Hibás paraméterezés!
   $ echo $?
   1

   $ java Main aa bb cc
   Hibás paraméterezés!
   $ echo $?
   1

   $ java Main programming prg
   prgrg
   $ echo $?
   0

   $ java Main abcaabc ab
   abaab
   ```

3. (1 pont) Interaktív módon kérjen be a felhasználótól egy sztringet.

   Feltételezhetjük, hogy a sztringben csakis az angol ábécé kis- és nagybetűi szerepelnek, ill. a szóköz.

   Írjunk ki a képernyőre egy kis statisztikát.

   Példa:

```
// Az <E> jelentése: itt ütöttünk Enter-t.

$ java Main
Szöveg: Hello World<E>
Kisbetűk száma:  8
Nagybetűk száma: 2
Szóközök száma:  1
```

   A statisztikát a `StringUtils.statistics()` nevű metódussal állapítsuk meg.

   A metódus megkapja a felhasználó által megadott sztringet, s egy háromelemű tömbben visszaadja a három egész értéket.

   A tömb első eleme tartalmazza a kisbetűk számát. A tömb második eleme tartalmazza a nagybetűk számát. A tömb harmadik eleme tartalmazza a szóközök számát.

4. (1 pont) Gyakornokként a Nemzeti Kibervédelmi Felügyeletnél szeretnénk nyári munkát vállalni. Beugró feladatként fel kell törnünk egy jelszóval védett ZIP állományt.

   A jelszót a következőképpen tudjuk megállapítani:

   Adott egy szöveges állomány (`input.txt`), ami sorokat tartalmaz. Egy sorban pontosan két egész szám szerepel vesszővel elválasztva. Vagyis az állomány két oszlopot tartalmaz. A jelszó az első oszlopban szereplő érvényes számoknak az összege.

   Az első oszlopban akkor tekintünk egy számot érvényesnek, ha ez a szám maradék nélkül osztható a mellette lévő számmal.

   Vajon sikerül átjutnunk a beugrón, vagy már rögtön az első feladatnál elvérzünk?

   Tekintsük a `pelda.txt` állomány tartalmát:

```
196, 6
892, 6
938, 10
463, 5
343, 6
252, 2
211, 2
599, 2
153, 9
855, 5
```

   $196/6 = 32.6$, vagyis a 196 nem érvényes.

   Az első érvényes szám a 252, mivel ez maradék nélkül osztható 2-vel.

   Eredmény: $252 + 153 + 855 = 1260$.

   Vagyis a jelszó: 1260.

   Írjunk egy programot, ami az `input.txt` állományt feldolgozva megállapítja a jelszót, s ezt kiírja a képernyőre.

   Példa:

```
// a `pelda.txt` -n futtatva

$ java Main
1260
```

   Figyelem! Az `input.txt` -t kell feldolgozni!