Név: .....

Neptun-kód: .....

Programozás 2
– 1. zárthelyi dolgozat –

2021. okt. 13., 8.00 órás csoport

Feladatok

1. (1 pont) Írjon egy Monitor nevű osztályt, mellyel egy monitort (kijelzőt) tudunk reprezentálni. A Monitor osztályt egy `Monitor.java` nevű állományban helyezze el!

   Az osztálynak egyetlen konstruktora legyen. A konstruktornak a képernyő felbontását (szélesség x magasság) kell megadni sztringként. Példák:

   ```
   Monitor m0 = new Monitor("1024x 768");
   Monitor m1 = new Monitor("1366 x 768");
   Monitor m2 = new Monitor("1920x1080");    // ez a full HD felbontás
   Monitor m3 = new Monitor("3840 x2160");
   ```

   Mint látható, a szeparátorként használt x előtt és után opcionálisan állhat egy-egy szóköz.

   Egy Monitor 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(m0);                        // Monitor(1024x768)
   System.out.println(m1.getWidth());             // 1366
   System.out.println(m1.getHeight());            // 768
   System.out.println(m2.getNumberOfPixels());    // 2073600
   System.out.println(m0.isFullHD());             // false
   System.out.println(m2.isFullHD());             // true
   System.out.println(m3.isFullHD());             // false
   System.out.println(m0.getRatio());             // 1.3333333333333333
   System.out.println(m0.compare(m2));            // -1
   System.out.println(m1.compare(m1));            // 0
   System.out.println(m3.compare(m1));            // 1
   ```

   Egy kis magyarázat

   A `getRatio()` a szélesség és a magasság hányadosa. Pl. egy 1024x768-as felbontású monitor képernyője 4:3-as arányú.

   A `compare()` két monitor felbontását (pixeleinek a számát) hasonlítja össze. Ha a bal oldali monitor kisebb felbontású, akkor `-1`-et adunk vissza. Ha a két monitor azonos felbontású, akkor `0` a visszatérési érték. Ha a bal oldali monitor a nagyobb felbontású, akkor `+1`-et kell visszaadni.

2. (1 pont) Írjon egy programot, ami parancssori argumentumként kap egy szót, illetve egy vagy több betűt. Példa:

```
# egy szó és egyetlen betű:
$ java Main Aladar a

# egy szó és három betű:
$ java Main Aladar a d x
```

A szónak és legalább egy betűnek kötelezően szerepelnie kell. Feltételezhetjük, hogy a szó legalább egy hosszúságú. A betűk garantáltan egy karakterből állnak (ezt nem kell külön leellenőrizni).

A program kimenete egyetlen szó legyen, melyet a következőképpen kell előállítani: vegyük az input szót, s minden olyan karakterét duplázzuk meg, ami szerepel az input szó utáni betűk között. Az input szó után szereplő betűk között lehet ismétlődés, de egy betűt ekkor is csak egyszer kell duplázni.

Példák:

```
$ java Main Aladar a
Alaadaar

$ java Main Aladar a a a a a a a
Alaadaar

$ java Main Aladar a d x
Alaaddaar

$ java Main Aladar x y z
Aladar

$ java Main Aladar r d l
Alladdarr
```

A program elején ellenőrizzük le, hogy megfelelő-e a paraméterezés. Ha nem, akkor írjunk ki egy hibaüzenetet és a program lépjen ki 1-es hibakóddal.

```
$ java Main
Hiba! Adj meg egy szót és legalább egy darab betűt!
$ echo $?
1

$ java Main na
Hiba! Adj meg egy szót és legalább egy darab betűt!

$ java Main na a
naa
```

3. (1 pont) Írjon egy programot, ami interaktív módon bekér a felhasználótól egy csupa számjegyeket tartalmazó sztringet.

   Feltételezhetjük, hogy a sztring csak számjegyeket tartalmaz (vagyis ezt nem kell külön levizsgálni). Azt is tudjuk, hogy a sztring hossza legalább 2.

   Példa:

```
$ java Main
Az egyes felhőkarcolók magassága: 2483971
Válasz: 25

$ java Main
Az egyes felhőkarcolók magassága: 020
Válasz: 4

$ java Main
Az egyes felhőkarcolók magassága: 65734434594654447653432334584572354347
Válasz: 61
```

   Vagyis: a felhasználó megadja a felhőkarcolók magasságát. Minden egyes számjegy egy-egy felhőkarcoló magasságát jelöli. A mi feladatunk az, hogy számítsuk ki a szomszédos felhőkarcolók magasságkülönbségének az összegét.

   Az első példa esetén a válasz 25, hiszen $2 + 4 + 5 + 6 + 2 + 6 = 25$.

   A tényleges munkát szervezze ki egy statikus metódusba, amit a következőképpen akarunk meghívni (példa):

```
int result = SkyScrapers.process("2483971");    // 25
```

   Tegyen róla, hogy a SkyScrapers osztályt még véletlenül se lehessen példányosítani!

4. (1 pont) Az `input.txt` fájl minden egyes sorában egész számok szerepelnek, egymástól egy vagy több szóközzel elválasztva. A feladatunk az, hogy számítsuk ki az állomány ellenőrző összegét (*checksum*). Ehhez minden sorban vegyük a legelső (jelölje a) és a legutolsó (jelölje b) számot, majd vegyük az (a modulo b) értékek összegét.

   Például legyen adott a következő állomány (`pelda.txt`):

   ```
   5 1 9 5
   7 5  3
   3
   2   4 6 8
   ```

   Ekkor $(5\ mod\ 5) + (7\ mod\ 3) + (3\ mod\ 3) + (2\ mod\ 8) = 0 + 1 + 0 + 2 = 3$.

   Írassuk ki a képernyőre a bemeneti állomány ellenőrző összegét.

   Futási példa:

   ```
   $ java Main pelda.txt
   3
   ```

   Figyelem! A programot az `input.txt` állományon is próbálja ki!