Feladat

Magyarországon 1957 óta lehet ötös lottót játszani. A játék lényege a következő: a lottószelvényeken 90 szám közül 5 számot kell a fogadónak megjelölnie. Ha ezek közül 2 vagy annál több megegyezik a kisorsolt számokkal, akkor nyer. Az évek során egyre többen hódoltak ennek a szerencsejátéknak és a nyeremények is egyre nőttek.

Adottak a `lottosz.dat` szöveges állományban egy év 51 hetének ötös lottó számai. Az első sorában az első héten húzott számok vannak, szóközzel elválasztva, a második sorban a második hét lottószámai vannak stb.

Például:
37 42 44 61 62
18 42 54 83 89
...
9 20 21 59 68

Az állományból kimaradtak az 52. hét lottószámai. Ezek a következők voltak: 89 24 34 11 64.

Készítsen programot a következő feladatok megoldására!

1. Olvassa be a lottosz.dat szöveges állományt (51 sor), tárolja az adatokat <**Hét**> *hetek*: listában, a lottószámok minden sorban emelkedő számsorrendben szerepeljenek.

2. Kérje be a felhasználótól az 52. hét megadott lottószámait,
   - a program rendezze a bekért lottószámokat emelkedő sorrendbe!
   - a rendezett számokat írja ki a képernyőre és tárolja a *hetek* listába!

3. Kérjen be a felhasználótól egy egész számot 1-51 között! A bekért adatot nem kell ellenőrizni!
   - ➢ írja ki a képernyőre a bekért számnak megfelelő sorszámú hét lottószámait

(+)

1. Állapítsa meg, hogy hányszor volt páratlan szám a kihúzott lottószámok között! Az eredményt a képernyőre írja ki! (az objektum tudja, hogy hány páratlan száma van, saját adatmezőbe tárolja …)

2. Írja ki az összes lottószámot a `lotto52.txt` szöveges állományba! A fájlban egy sorba egy hét lottószámai kerüljenek, szóközzel elválasztva egymástól!

EXTRA ELLENŐRZÉSEKET NEM KELL VÉGEZNI!

Megoldás

Az `AppLottó` osztálydiagramja:

Hét
- számok[5]: egészszám
+ **Hét** (tmp[]: String) fillSzámok (tmp) //egésszé alakítás
- eljárás fillSzámok (tmp[]: String) - eljárás rendezés() - eljárás csere() + getSzámok (): String

AppLottó
hetek <Hét>: lista
adatokBe (path: String) hét52 ( ) melyikHét ( )

Az `AppLottó` a vezérlő osztály, ebben minden használati esetnek felveszünk egy-egy metódust. A vezérlő egy listán keresztül kapcsolódik a `Hét` osztályhoz.

A `Hét` osztálynak biztosítania kell az adatok tárolásán és az adatokon végzett metódusokat.

AppLottó:

- `adatokBe:` rekordok beolvasása fájlból,
  a rekordok (sor) átadása **`Hét`** osztály konstruktorának
- `hét52:` az 52. hét számainak beolvasása konzolról,
  a rekord (sor) átadása a **`Hét`** osztály konstruktorának
- `melyikhét:` tetszőleges hét számainak lekérése a hetek listából, a **`Hét`** osztály getSzámok metódusának hívása

Hét:

- `fillSzámok:` a konstruktor String típusú változóját felszeleteli (split) és a számokat rendezve betölti a saját számok[]-be
  hívja a `rendezés` alprogramot
- `rendezés:` a tmp[]-be egészszámként splitelt adtokat A-Z sorrendbe rendezi,
  hívja a `csere` alprogramot
- `csere:` felcseréli a tmp[] két elemét
- `getSzámok:` a számok adatmező adatait összefűzi egy Stringgé és átadja a főprogramnak formázott kiírásként

Főprogram

eljárás adatokBe(path: String) beolvassa az adatokat a path-ból
    ...
    ciklus i = 0-tól 51-ig egyesével
        hetek.add(new Hét(sor)) a **Hét** osztály konstruktorának hívása
    ciklus vége
    ...
eljárás vége

eljárás hét52() // standard inputtal
    new Scanner osztály: sc legyen inkább globális – static ...
    sor = sc.sorbeolvasása
    tmp[]: String split…
    hetek.add(new Hét(sor)) a **Hét** osztály konstruktorának hívása
    Ki: 52. hét rendezett számai
eljárás vége

eljárás melyikHét() standard inputtal
    new Scanner osztály: sc legyen inkább globális – static ...
    i = sc.egészszám beolvasása
    Ki: (i-1). hét rendezett számai a **Hét** osztály getSzámok metódusa
eljárás vége

Hét osztály

Adattagok
- számok [] = new egészszámok [5]
- páratlanDb: egész

+ **Hét** (sor: String) **konstruktor**
tmp [] = sor.split(" ") mindenki a saját adatait szeleteli
fillSzámok(tmp)
setPáratlanDb() megszámolja a páratlanokat a számok[]-ben és beállítja a **páratlanDb** értékét

- **eljárás** fillSzámok (tmp []: String)
ciklus i = 0-tól 5-ig egyesével
számok [i] = egészszám(tmp[i])
ciklus vége
rendezés()
eljárás vége

- **eljárás** rendezés ()
ciklus i = 0-tól 5-ig egyesével
minIndex = i legyen ő a minimum indexe
ciklus j = i+1-től 5-ig egyesével
ha számok[j] < számok[minIndex]
minIndex = j most meg ő a minimum indexe
ha vége
ciklus vége
csere(i, minindex) helyére tesszük a minimumot
ciklus vége
eljárás vége

- **eljárás** csere (i: egész, j: egész)
int tmp: egész
tmp = számok[i]; számok[i] = számok[j]; számok[j] = tmp;
eljárás vége

+ **függvény** getSzámok (): String
s: String
s=Integer.toString(számok[0]) számból String
ciklus i = 1-től 5-ig egyesével
s=s+" "+Integer.toString(számok[i]);
ciklus vége
return s
függvény vége