

A személyszám 11 jegyű.

Az első jegy a személy nemét jelöli, az alábbi táblázat alapján.

1999.12.31-ig született		1999.12.31 után született	
férfi	nő	férfi	nő
1	2	3	4

A 2 – 7 számjegyek a születési év két utolsó jegyét, a születési hónapot és napot tartalmazza.

A 8 – 10 számjegyek az azonos napon születettek születési sorszáma.

4080107199 → **nő 2008.01.07 199.**

2490709322 → **nő 1949.07.09 322.**

1. Olvassa be a *szemszam.txt* fájl adatait - tabulátorokkal tagolt rekordok.

rekord szerkezet: **név, személyszám első 10 számjegye.**

pl.:	Karam Ella	4080107199
	Para Zita	2490709322
	Seft Elek	1340110416

2. Készítsen a fájlt *nőkAZ.txt* néven a nők névsoráról.
3. Írja ki a képernyőre a minta szerint, ki mikor született.
4. (+) Ki a legidősebb?

Olvassa be a *szemszam.txt*
fájl adatait

személyek lista <**Személy**>

Személy osztály

→ konstruktor

név

személyszám

Készítsen a fájl *nőkAZ.txt*
néven a nők névsoráról.

Személy osztály

→ név

neme // n/f

Ki mikor született.

Személy osztály

→ név

születésidátum // 2015.01.23

Személy

- név: String
- személyszám: String
- neme: char
- születésidátum: String

+ **Személy** (sor: String)

tmp[]:String // split...

this.név= tmp[0]

this.személyszám=tmp[1]

setNeme (személyszám)

setSzüldátum(személyszám)

AppSzemélyszám

személyek <**Személy**>: lista

nők <String>: lista

adatokBe (path: String)

nőiNévsor (path: String)

név_szüldátum ()

`adatokBe(path: String)` *beolvassa az adatokat a path-ból*

előltesztelő ciklus

feltölti a *személyek* listát

//a Személy osztály konstruktorának hívogatása

```
személyek.add(new Személy(sor))
```

ciklus vége

`nőiNévsor(path: String)` *előállítja a nők névsorát és kiírja*

ciklus végig a *személyek* listán

ha a *személyek* lista *i.* elemének `getNeme()` = 'n'

akkor ezt hozzáadjuk a *nők* listához

```
nők.add(személyek.get(i).getNév())
```

ciklus vége

`rendezés()`

`adatokKi(path: String)`

+ **getNév** (): String

└─ return név

- **setNeme** (személyszám): char

┌─ neme = 'n' //nő

char c = *személyszám* első karaktere

ha (c = '1' vagy c = '3')

 akkor neme = 'f' //férfi

ha vége

└─ return neme

+ **getNeme** (): String

└─ return neme

Személy

- név: String
- személyszám: String
- neme: char
- születésidátum: String

```
+ Személy (sor: String)  
  tmp[:String] // split...  
  this.név= tmp[0]  
  this.személyszám=tmp[1]  
  setNeme (személyszám)  
  setSzüldátum(személyszám)
```

- ```
+ getNév: String
- setNeme: char
+ getNeme: char
- setSzületésidátum: String
```



**rendezés()** *előállítja a nők névsorát*

*tetszőleges rendezése a női listának a csere eljárással*

```
nők.get(j).compareTo(nők.get(minIndex)) < 0 //névsor feltétele
```

**adatokKi**(path: String) *kiírja a nők névsorát*

```
f.writeBytes(nők.get(i) + "\n") //egy sor + soremelés kiírása
```

név\_szüldátum()

ciklus végig a *személyek* listán

ki: *személyek*.get(i).getNév(), getSzületésidátum()

ciklus vége

```
- setSzületésidátum(személyiszám: String): String
 születésidátum = "19" //1900, ha 3 vagy 4 akkor 2000
 char c = személyiszám első karaktere
 ha (c = '3' vagy c = '4')
 akkor születésidátum = "20"
 ha vége
 év = születésidátum + személyiszám 2.-3. karaktere + "."
 hó = személyiszám 4.-5. karaktere + "."
 nap = személyiszám 6.-7. karaktere
 születésidátum = év + hó + nap
 return születésidátum //pl: 1917.08.06

+ getSzületésidátum (): String
 return Születésidátum
```