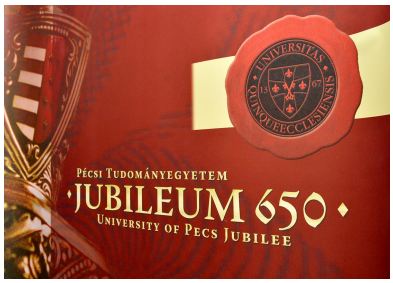
TUDNIVALÓK: Lévén, hogy nagyon kevés időnk van a konzultációkon, a feladatok zöme HÁZI FELADAT !
Ezek megoldása fontos a tárgy sikeres teljesítéséhez! (Nyilván minden feladatsorban vannak hasonló jellegű feladatok, ki-ki maga tudja, hogy mennyire boldogul velük, ha könnyen, akkor természetesen nem kell megoldani az összes „egyforma” feladatot.)

1. feladat (órai)



Ha már a PTE jubileumával kezdődik az év, vegyük ki mi is a részünket belőle. Bár szerencsére a rendezvényeket ingyenesen lehetett látogatni, a feladat kedvéért fizetőssé tesszük őket.

Az ünnepséghez tehát rendezvények tartoznak. Minden egyes rendezvény a címével, időpontjával (jelenleg sima String) és a belépőjegy árával jellemezhető.
Természetesen vannak résztvevők is, sőt, a PTE azonosítóval rendelkezők még kedvezményt is kapnak. Minden résztvevőnek van neve (a név egyébként sohasem lehet egyedi azonosító, de most az egyszerűség kedvéért feltesszük, hogy minden név más). A PTE-s résztvevőt ezen kívül még egy azonosító is jellemzi.
Egy résztvevő akkor vesz részt egy adott rendezvényen, ha kifizeti a rendezvény részvételi díját. A PTE-s résztvevők egységesen 10% kedvezményt kapnak.
Minden egyes résztvevő esetén tudjuk felsorolni azokat a rendezvényeket, amelyeken részt vett az illető, illetve minden rendezvény esetén állapítsuk meg a résztvevők számát és a rendezvény bevételét.

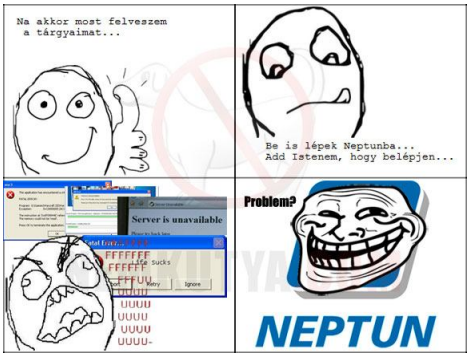
Írjunk programot a rendezvények szimulálására: Olvassuk be a tervezett rendezvényeket, ill. a potenciális résztvevőket, a résztvevők kapjanak véletlenszerűen valamennyi zsebpénzt, majd rendezvényenként véletlenszerűen döntsék el, hogy megpróbálnak-e részt venni rajtuk, vagy sem. (A részvételi kedv kb. 80%-os.)

A beolvasáshoz talál segítséget a feladatsor végén.

Oldjon még meg néhány „alapfeladatot”:
Határozza meg, mely rendezvény hozta a legtöbb bevételt; fejenként összesen hány Ft kedvezményt kaptak a PTE-s résztvevők; mekkora volt a teljes rendezvénysorozat bevétele, stb.

A további feladatok megoldásakor készítsen osztálydiagramokat is! (A feladatok sorrendje esetleges, nem nehézség szerint követik egymást.)

2. feladat



Írjon programot a diákok adminisztrálására.

a/ Egy diákot jellemez a neve, kódja és életkora (egyelőre csak évszám).

b/ Olvassa be a *diakok.txt* fájlban lévő adatokat, ezek alapján hozza létre a megfelelő diák-példányokat, majd számítsa ki az átlagos életkorukat, ill. határozza meg, hogy kik a legidősebbek. A beolvasáshoz talál segítséget a feladatsor végén.

c/ Bővítsük a feladatot. A diák vegyen fel tantárgyakat, és az aktuálisan felvett tantárgy kreditszámával növekedjen a krediteinek száma, a tantárgy pedig kerüljön be a diák felvett tantárgyainak listájába. Természetesen egy tárgyat nem vehet fel többször (legföljebb egy másik félévben, de az már egy másik feladat).

Szimulálja a tantárgyfelvételt (valahányszor futtassa le, hogy egy véletlen diák felvesz egy véletlen tárgyat), majd állapítsa meg az átlagos kreditszámot, ill. azt, hogy kik vették fel a legtöbbet/legkevesebbet.

d) Vannak költségtérítéses diákok is. Nekik fizetniük kell a felvett kreditek után, mégpedig a kreditszám valahányszorosát. Regisztrálásukkor meg kell adni a fizető felet is (cég vagy saját maga). Írassuk ki a névsorukat, és állapítsuk meg, hogy összesen mennyit kellett fizetniük.

e/ A diák azért diák, hogy időnként vizsgázhasson ☺. Vagyis egy olyan tantárgyból tudjon vizsgázni, amelyik szerepel az általa felvett tantárgyak listáján. Amikor vizsgázik, növekszik a vizsgáinak száma. Állapítsa meg az átlagos vizsgaszámot, ill. azt, hogy kik vizsgáztak a legkevesebbszer.

f/ Most már a vizsga is legyen „rendes”. Vagyis a vizsgán kapjon jegyet is. Bár alaposztály-ban szinte soha nincs random generálás, most kivételesen legyen. Ha a generált jegy érvényes, akkor értelemszerűen növekedjen a diák teljesített krediteinek száma, illetve számolja a diák átlagát is. (Odabízom, hogy az elégtelent beleszámolja-e az átlagba vagy sem.)

g/ Bővítse a vezérlést is: olvassa be a tantárgyak listáját egy adatfájlból. (Az adatfájl soronként a tárgy nevét, kódját és kreditszámát tartalmazza. Figyeljen rá, hogy amikor megírja a fájlt. UTF-8-as kódolásban mentse.

h/ Szimulálja a tanévet: először a tantárgyfelvételt, majd a vizsgaidőszakot.
Mindkettő azt jelenti, hogy véletlen sokszor futtassa a következőt: egy véletlenül választott diák felvesz egy véletlen tárgyat, ha tud (vagy vizsgázik belőle).
Utána azt is határozza meg, hogy melyik diák vette fel a legtöbb/legkevesebb kreditet, ki vizsgázott legtöbbször, és még, ami eszébe jut.

i/ Bővítse a feladatot a saját fantáziája szerint.

j/ Pihenjen. ☺

3. feladat

A JAVA-TRAVEL utazási iroda a Java szigetek körüli hajóutakat szervez.

Minden hajóút esetén adott annak neve és egy egyedi azonosítója, ezen kívül pedig az utaztatható személyek maximális száma. A hajóra akkor tud felszállni egy utas (vagyis akkor tud részt venni a hajóúton), ha az utaslétszám még ezen a korláton belül van, és az utas beszállhat. Ha beszállhat, akkor az utas bekerül a hajó utas-listájába (persze, csak ha még nincs benne). Egy-egy út költsége egy-egy adott hajón minden utas számára azonos.

Az utasokat a nevük és egy egyedi kód azonosítja. Mindenkinek van valamennyi pénze, és bármikor tud költeni valamennyit és kapni is valamennyit. A társaság csak akkor enged beszállni valakit, ha ki tudja fizetni az adott út költségét, plusz még ezen felül letétbe tud helyezni egy, az összes utas számára egyforma értékű kauciót.
Írjon programot az iroda működtetésére, vagyis olvassa be néhány hajóút és néhány utas adatait, szimulálja az utazást, majd írassa ki az adatokat.
A szimuláció ezt jelenti: minden hajóút esetén adjuk meg az illető hajóútra érvényes útiköltséget (bizonyos határok között véletlenül generált érték), majd valahányszor egymás után válasszunk ki egy véletlen hajót, amelyre egy véletlen utas megpróbál felszállni.

Bővítse az eddigieket:

- A cég kedvezményt ad a Java programozók számára ☺, de mivel nem mindenki egyformán jó programozó, ezért a kedvezmény százalékát egyénenként dönti el. (A Java programozók azonban reménykednek abban, hogy esetleg a családtagjuk, barátjuk is kedvezményt kaphat, ezért úgy írják meg a programot, hogy csak az legyen érdekes, hogy valaki kedvezményezett. ☺) Kiíratáskor az is kerüljön a kedvezményezett neve mellé, hogy hány százalék kedvezményt kap.

- További feladatként számoljuk ki a cég teljes bevételét. Gondolja végig, hogy ehhez mit, melyik osztályban és hogyan kell megadni.

További feladatok a „szokásosak”: melyik hajónak legtöbb a bevétele, melyiken utaznak a legtöbben/legkevesebben, valamilyen szempont szerinti rendezés, stb.

4. feladat

Egy **újság**ot egyértelműen jellemez a *neve* és *megjelenési dátum*a (sima String). Az újság *cikket közöl()*, vagyis a paraméterében megadott cikket hozzáadja az újságban megjelent *cikkek listájá*hoz.

Egy **cikk** egyértelműen megadható a *szerző nevé*vel, a *cikk címé*vel, a *cikk méreté*vel (karakterszám) és egy, a cikkben lévő hazugság nagyságára utaló *százalékláb* értékkel. A cikk címe szükség esetén megváltozhat, egyéb jellemzői nem.

Az újságban a cikk közlésekor azt is számolják, hogy a paraméterben lévő cikk mennyivel változtatja meg az *átlagos hazugságszázalék*ot. Ha még egyetlen cikket sem közöltek, akkor az átlagszázalék lekérdezésekor -1-et adjon vissza.

Olvassuk be valahány újság és cikk adatait, majd az *ujsagIras()* során véletlen sokszor egy-egy véletlenül választott újságban jelentessünk meg egy-egy véletlenül választott cikket. Végül írassuk ki az újságok adatait, és számoljuk ki, hogy mekkora volt az újságok átlaghazugsága (az egyes újságok átlaghazugságának átlaga). Számolja ki, hogy összesen hány cikk jelent meg, illetve azt is, hogy melyik a leghazugabb újság.

Szorgalmi: A megjelenés dátumát ne String-ként, hanem rendes dátumként kezelje. Segítség a feladatsor végén, vagy ha előre akar szaladni, akkor itt:
http://javarevisited.blogspot.hu/2015/03/20-examples-of-date-and-time-api-from-Java8.html .

Folytatás:

Továbblépve a következőt tudjuk még:

A médiabirodalom nyomtatott és internetes újságot is megjelentet.
A **nyomtatott újság**ot a néven és megjelenési dátumán kívül jellemzi még az újság *példányszám*a és a *méret*e (összesen hány karaktert tud megjelentetni). Mivel itt korlátozott a méret, ezért egy cikk közlése csak akkor lehetséges, ha annak mérete még belefér az újság méretébe. Egy cikk közlésekor természetesen csökkenteni kell a rendelkezésre álló *maradék méret*et.
A nyomtatott újság *olvasóinak száma* akkor növekszik, ha *elad()*ják az újság egy példányát. Nyilván csak a példányszám erejéig.

Az **internetes újság** a néven és megjelenési dátumon kívül tartalmazza még az újság *link*jét (most csak egy String). Olvasóinak száma *kattintás()*kor növekszik.
Egyelőre ennyi, de esetleg végiggondolhat még néhány, az internetes bulvárra jellemző dolgot, pl.:

1. Előfordulhat, hogy egy-egy cikk esetén *címmódosítás()*ra van szükség, mégpedig akkor, ha a vizsgálat pillanatában az *olvasók száma* nem haladott meg egy bizonyos létszámot.

2. Egy-egy cikkhez *fórumbejegyzés()* is születhet. Jó lenne figyelni a *fórumozók számá*t, de akár magukat a megjegyzéseket is.

a/ Alakítsa át a korábban megírt adatbevitelt úgy, hogy most már vegye figyelembe a fájl sorainak minden adatát – értelemszerűen látszik, hogy melyik internetes, melyik nyomtatott, és mit jelentenek a sorokban szereplő adatok.

b/ Válogassa szét az internetes és a nyomtatott újságokat, és külön-külön írassa ki őket.

c/ Bővítse ki a feladatot, és „terjessze” az újságokat. Ez azt jelenti, hogy néhányszor (véletlen sokszor) válasszunk ki véletlenszerűen egy-egy újságot, és növeljük olvasóinak számát. Utána írassuk ki az újságok nevét a hozzájuk tartozó olvasószámmal együtt. Az internetes vagy a nyomtatott újságoknak van nagyobb olvasóközönsége?

d/ Bővítse a feladatot a korábban említett internetes bulvárra jellemző tulajdonságokkal.

5. feladat – mielőtt hiányolná az italokat ☺

Az előző feladatok sikeres megoldása után „ihat” egyet, és folytathatja a múltkori feladatsor ide vonatkozó feladatát.
Az italokat jellemzi az ital fajtája (pl. tej, bor, stb.), vonalkódja, mennyisége (literben), literenkénti egységára (Ft/liter) és a tényleges eladási ár, amely már az ÁFA értékét is tartalmazza.

Vannak emberek, akik szeretnek inni. ☺ Ezeket az embereket jellemzi a nevük, és van valamennyi pénzük. Elvileg fizetnek az italért. Mivel többször is rendel()hetnek, ezért ilyenkor a fizetendő összeg az aktuálisan fogyasztott ital árával növekszik.

Az ember egy szöveges számlát „kap”, erre vagy a fizetendő összeg kerül, ha azt ki tudja fizetni, vagy az, hogy a különbözet (azaz a fizetendő és a pénze közötti különbség) értékben köteles mosogatni. Ekkor „büntetésképpen” és az adminisztrálhatóság kedvéért a neve is rákerül a számlára

Az ember fizet(), ha tud, azaz levonódik az illető pénzéből a fizetendő érték, de a maradék nem mehet át negatívba (hiszen „lemosogatja” a tartozást).

Játsszon kicsit a feladattal, vagyis hozzon létre italokból és/vagy emberekből álló listá(ka)t, és pl. az összes ember rendelhet valamilyen italt saját számlára, de akár úgy is, hogy egyikőjük fizeti a teljes összeget, vagy próbálkozhat avval, hogy egyetlen ember „végigkóstolja” a teljes italválasztékot, stb. Működtesse a fantáziáját.

Folytatás:

Hozzunk létre egy AlkoholosItal osztályt. Az alkoholos ital is Ital, de az ital jellemzőin kívül jellemzi még az ital márkája és alkoholfoka. Ezek nem is változhatnak meg az ital élettartama alatt. Megállapítandó, hogy az adott italmennyiség ártalmas-e már: akkor ártalmas, ha a benne lévő alkohol mennyisége több, mint egy általánosan rögzített alkoholhatár. (megart())

Egészítsük ki, illetve módosítsuk az Ember osztályt
Csak akkor ihat alkoholt, ha elég idős (azaz egységesen adott korhatárnál idősebb), és a rendelt italmennyiség nem árt meg.

Szorgalmi: Az aktuális év megadására használja a rendszeridőt (ehhez az aktuális calendar példány évszámát kell lekérni.)

Finomítsuk a „megártás” fogalmát, és egészítsük ki a feladatot avval, hogy valaki csak akkor ihat alkoholt, ha eleget tesz a korábbi feltételeknek (elég idős és nem árt meg az ital), és még a véralkoholszintje nem ér el egy, a törvényben rögzített határt. (A törvény nem tesz különbséget a nemek között, de a véralkoholszint kiszámításához szükséges faktor eltérő férfiak és nők esetén – de mindkét nemre egy-egy általánosan elfogadott érték jellemző.)
A véralkoholszint kiszámítási módja:
véralkoholszint = fogyasztott_alkoholmennyiség / testsúly_kg/Widmark_faktor

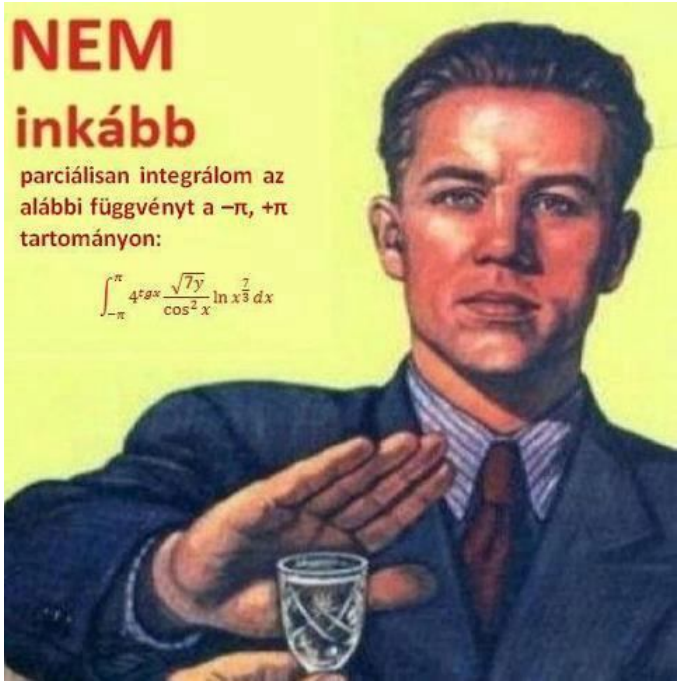
(A kapott érték ezreléket jelent, vagyis az olvashatóság kedvéért még szorozni kell 1000-rel.)

Ha már a véralkoholszint miatt meg kell különböztetnünk egymástól a férfiakat és a nőket, legyen még egy különbség: férfiak esetén figyeljük azt is, hogy nős-e az illető. Nős férfiak esetén egy metódus mondja meg, hogy sodrófával várják-e otthon, vagy sem – akkor jár sodrófa, ha az illető véralkoholszintje meghalad egy bizonyos – az otthoni toleranciától függő – határt.

Hol lehet beállítani a fogyasztott alkoholmennyiség értékét? Nyilván akkor, amikor iszik. Feltehetjük, hogy akkor iszik, amikor megrendeli az italt. Vagyis módosítsuk a rendel() metódust a következők szerint:
Ha ihat, akkor növelje a fizetendő értéket az ital árával. Ha az ital alkohol, akkor a fogyasztott alkoholmennyiség értékét is növelje a fogyasztásnak megfelelően. (a rendelt ital alkohol-mennyiségével)
Módosítsa úgy az előző metódust, hogy a vezérlő osztály azt is ki tudja írni, hogy az illető nem ihat, ezért nem is rendelhet. (Segítség: gondolja végig a metódus típusát.)

Adjon rá lehetőséget, hogy az illető többször is bulizhasson, azaz írjon egy ujBuli() metódust, amelyben kinullázza a szükséges kezdőértékeket.

Ha ezt mind-mind jól megoldotta, akkor jutalomból ihat egy igazi pohárral is a kedvencéből, de vigyázzon, meg ne ártson! ☺



6. feladat

Egy **könyv**et jellemez a szerzője, címe, oldalszáma, ára. Az ár az oldalszám valahányszorosa. A könyv ki tudja írni a saját jellemzőit.

Egy **tankönyv**et szintén jellemez a szerzője, címe, oldalszáma, ára, de ezen felül még az is, hogy melyik tárgy ajánlott irodalma. A tankönyvek árából lejön a jegyzettámogatás, amely az ár x százaléka. Ő is ki tudja írni a saját jellemzőit.

Az **idegen nyelvű könyv**et szintén jellemzi a szerzője, címe, oldalszáma, ára, de ezen felül még az is, hogy milyen nyelven íródott, és egy nehézségi szint: A, K, F (alap, középfokú, felsőfokú), amely azt jelzi, hogy milyen nehéz az idegen nyelvű szöveg. Az ő ára is az oldalszám valahányszorosa, de nem biztos, hogy ugyanaz a szorzó, mint egy sima könyvnél. És ő is ki tudja írni a saját jellemzőit.

- Írassa ki legalább egy regény és egy tankönyv adatait!
- Olvassa be n db idegen nyelvű könyv adatait, majd kérjen be egy idegen nyelvet, és írassa ki az azon a nyelven olvasható könyvek listáját.

Készítsen osztálydiagramot!

Szorgalmi: Gondolja végig, és oldja meg úgy a feladatot, hogy legyen még egy negyedik könyvtípus is: **idegen nyelvű tankönyv**, amely egyszerre tankönyv is és idegen nyelvű könyv is. A vezérlő osztályban hozzon létre egy-egy példányt mind a négy fajtából!

BEOLVASÁS:

Majd tanulunk másik fajta beolvasást is, de a legegyszerűbb az, ha a **Scanner** osztály megfelelő metódusait használja. (ld. Help).

Néhány tudnivaló:

A program elejére, a `package…` kezdetű sor után be kell illeszteni egy importot (az alap Java csomag nem tartalmazza a `Scanner` osztályt):

```
import java.util.Scanner;
```

Ezt nem muszáj „gyalog” beírni, lehet generáltatni is: ha a kódban elkezdjük gépelni a `Scanner` szót, de ctrl-space kombinációval fejezzük be, akkor automatikusan bekerül az import. Ha nem használjuk a ctrl-space billentyűket, hanem végigírjuk a szót, akkor a figyelmeztető kis sárga lámpácskára kattintva lehet generáltatni.

A beolvasás során definiálni kell egy scannert (ezt jelenleg a standard inputra, vagyis a billentyűzetre definiáljuk):

```
Scanner scanner = new Scanner(System.in);
```

Ha egész számot akarunk olvasni, akkor:

```
int szam;
System.out.print(”Kérem a számot: ”);
szam = scanner.nextInt();
```

Ha a szam `double`, akkor: `szam = scanner.nextDouble();`
Ha a beolvasandó adat String, akkor a `next()` vagy a `nextLine()` metódust használjuk.
A többit nézze meg a help-ben.

FÁJLBÓL VALÓ OLVASÁS

FONTOS: Ez egy „gyöngített” változat, később majd pontosítjuk, de mivel a billentyűzetről való olvasás roppant unalmas, ezért célszerű már evvel kezdeni.

A prog2-ben tanulthoz nagyon hasonló megoldás:

Nem a billentyűzetre, hanem egy fájlra irányítjuk a scanner-t:

```
Scanner scanner = new Scanner(new File("adatok.txt"));
```

Az *adatok.txt* fájl helye: a projekt gyökere. (Ezt majd javítjuk később.) Az adatfájl megírásakor figyeljen rá, hogy UTF-8-as kódolással mentsen (vagy egyelőre ne használjon ékezetes karaktereket).
Javaslat: ne másolja, hanem gépelje ezt a sort, mégpedig kódkiegészítéssel (Ctrl + space). Ekkor automatikusan beszúrja a szükséges importokat.

Fájlból való olvasás esetén listában célszerű tárolni az objektumokat. Ennek deklarálása és inicializálása:

```
private List<Tipus> listaNev = new ArrayList<>();
```

Ha menet közben kódkiegészítéssel dolgozik, akkor automatikusan beilleszti a szükséges importokat, ha nem így írja, akkor a kis lámpácska hatására is be lehet szúrni, ha ezt sem akarja, akkor be kell gépelnie. A `Scanner` osztály importja mellé még ez is kell:

```
import java.util.ArrayList;
import java.util.List;
```

A fájlból addig tudjuk olvasni a sorokat, amíg a scanner talál új sort (`hasNextLine()`).

Az adatfájlban egy objektum adatait célszerű egyetlen sorba írni, és valamilyen határolóval elválasztani egymástól (pl. vessző vagy pontosvessző). A fájlt soronként tudjuk olvasni, a sor szétvágásához használhatja a `String` osztály `split()` metódusát.

Listához az `add()` metódussal adhatunk új elemet.

FONTOS: Már prog2-ből is tanulta, hogy fájlból való olvasáskor mindig kötelező a kivételkezelés. A Java ezt valóban komolyan veszi, és anélkül nem is hajlandó lefordítani a programot. Ugyanakkor nem muszáj direktben leírnia (annál is inkább, mert még nem volt szó a kivételkezelés Java változatáról, ami egyébként nagyon hasonlít a prog2-ben tanulthoz). Ha nem írja le, akkor kap egy „unreported exception…" hibaüzenetet. Ha a sor melletti kis lámpácskára kattint, akkor két vagy három javítási ötletet kap (attól függ, hogy mikor kattint a lámpácskára, akkor, amikor már megírta a teljes metódust, vagy az első sor után azonnal). Ha érti a kiírt javaslatokat, akkor döntsön belátása szerint. Ha még nem igazán érti, akkor válassza a „Surround … with try-catch" változatot. Ha már a blokkot is felkínálja (ezt akkor teszi, ha több kódsort is megírt már), akkor válassza ezt, ha csak az adott sort kínálja fel, akkor azt, de ez utóbbi esetben is írjon mindent a `try` blokkba.

DÁTUMKEZELÉS

Természetesen ennél sokkal gazdagabb a dátumkezelés, de egy kis mintapélda talán segít elindulni a használatában.

```java
import java.util.Calendar;
import java.util.Date;

public class DatumPelda{

    private static Calendar naptar = Calendar.getInstance();
    private static Calendar most = Calendar.getInstance();

    public static Date setDatum(int ev, int ho, int nap) {
        naptar.set(Calendar.YEAR, ev);
        naptar.set(Calendar.MONTH, ho);
        naptar.set(Calendar.DAY_OF_MONTH, nap);
        return naptar.getTime();
    }

    public static void main(String[] args){
        System.out.println("Aktuális dátum: " +
                           most.get(Calendar.YEAR) + ". " +
                           (most.get(Calendar.MONTH)+1) + ". " +
                           most.get(Calendar.DAY_OF_MONTH) + "." );

        // Date szulEv = setDatum(2000,Calendar.APRIL +1 ,23);
        Date szulEv = setDatum(2000,4,23);
        naptar.setTime(szulEv);
        System.out.println("Születési dátum: " +
                           naptar.get(Calendar.YEAR) + ". " +
                           naptar.get(Calendar.MONTH) + ". " +
                           naptar.get(Calendar.DAY_OF_MONTH) + ".");
        int kor = most.get(Calendar.YEAR)- naptar.get(Calendar.YEAR);
        System.out.println("Életkora: " + kor + " év");
    } }
```