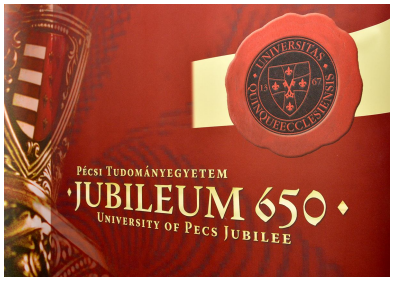
TUDNIVALÓK:
Most is és a következő gyakorlatokon is – akkor is, ha külön nem emeljük ki – az órán meg nem oldott feladatok HÁZI FELADAT-ként megoldandóak!!!
Ez fontos a tárgy sikeres teljesítéséhez!

1. feladat

Nézzük át közösen a legelso_Java_program.pdf fájlt.

2. feladat



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.

HÁZI FELADAT:

Az órán meg nem oldott feladatok mindegyike otthoni megoldásra vár!

Ha úgy érzi, hogy sokat felejtett, és szüksége van a vezérlőszerkezetek átismétlésére is, akkor oldjon meg néhányat az **alap_hf1.pdf** feladatai közül!

3. feladat

Ha már láttuk, hogy milyen szép a Java szigete, tegyünk ott egy hajókirándulást, és tegyük ezt lehetővé mások számára is.

A JAVA-TRAVEL utazási iroda több hajót működtet, ezeken utaztatja a vendégeit.

Minden hajóút esetén adott a hajó neve és egy, az utat definiáló egyedi azonosító, ezen kívül pedig az utaztatható személyek maximális száma. A hajóra akkor tud felszállni egy utas, ha az utaslétszám még ezen a korláton belül van, és az utas beszállhat. Ha beszállhat, akkor az utas bekerül a hajó utas-listájába. Egy-egy út költsége egy-egy adott hajóúton 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 marad nála egy minden utas számára egyformán kötelező alaptőke.

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.

Írjunk programot az iroda működtetésére, vagyis olvassuk be néhány hajóút és néhány utas adatait, szimuláljuk az utazást, majd írassuk 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 hajóúthoz tartozó hajót, amelyre egy véletlen utas megpróbál felszállni.
A beolvasáshoz talál segítséget a feladatsor végén.

További feladatok:
- A szimuláció során egy-egy véletlen utas időnként kapjon valamennyi pénzt.
- Ha már költségekbe veri magát, akkor utazhasson az utas, vagyis az utazik() metódusának hatására az aktuális utat adja hozzá a hajóútjai listájához.
- Számolja ki a cég teljes bevételét. Gondolja végig, hogy ehhez mit, melyik osztályban és hogyan kell megadni.
- A „szokásosak”: melyik hajónak legtöbb a bevétele, melyiken utaznak a legtöbben, legkevesebben, stb.

4. feladat

Most még elég kezdetlegesen, de szimuláljuk a tantárgyfelvételt!
Minden egyes **tantárgy**at jellemez a tantárgy neve, kódja és az elvégzéséért járó kreditszám.

a/ Adjunk meg két tantárgyat, és írassuk ki az adatait!

b/ A harmadik tantárgyat úgy hozzuk létre, hogy a billentyűzetről olvassuk be a szükséges adatokat. (Beolvasáshoz segítséget ld. a feladatsor végén.)

c/ Olvassuk be n db tantárgy nevét és kreditszámát! A tárgyak kódja a beolvasás sorrendjében legyen T1, T2, … Írassuk is ki a beolvasott tárgyak adatait. (Lehetőleg fájlból olvasson.)

d/ Számoljuk ki az össz-kreditszámot!

e/ Melyik tárgy(ak) a legnagyobb kredit-értékű(ek)?

5. feladat

Igyunk az eredményes tantárgyfelvétel örömére (vagy aki inkább úgy érzi, akkor a nagy ijedtségre)!
Az **ital**okat jellemzi az ital fajtája (pl. víz, sör, tej, stb.), vonalkódja, mennyisége, literenkénti egységára, és a tényleges eladási ár, amely már az ÁFA értékét is tartalmazza.

a/ Olvassa be n darab italrendelés adatait, és számolja ki az összes fizetendő értéket. (Természetesen állítsa be az ÁFA-kulcs értékét is.)

b/ Próbálja meg megoldani, hogy ne kelljen minden egyes rendeléskor megadnunk a rendelt ital egységárát is. (Erre később még visszatérünk.)

6. feladat

Egy **könyv**et jellemez a szerzője, címe, ISBN kódja, oldalszáma, ára. Az ár az oldalszám valahányszorosa. Ez a szorzó minden egyes könyv esetén azonos érték. A könyv String-ként vissza tudja adni a saját jellemzőit.
„Vásároljon” n db könyvet, számolja ki a teljes fizetendő összeg értékét, állapítsa meg, hogy melyik(ek) voltak a legolcsóbb, ill. legdrágább könyv(ek).

7. 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.

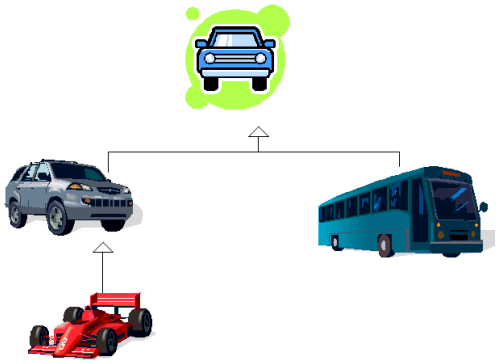
Készítsen osztálydiagramot!

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!

8. feladat

Feladatunk a gépjárművekkel foglalkozik.
Egy **gépjármű** jellemző tulajdonságai:

- színe, rendszáma, üzemanyagszintje, fogyasztása, stb.
- indítás(), gyorsítás(), fékezés(), tankolás(), stb.



Feladatok:

- Készítse el a GepJarmu osztályt.
- Készítse el a SzemelyGepKocsi, Busz és SportKocsi osztályokat
  (értelmes változókkal és metódusokkal).
- Készítsen főprogramot (legyen kreatív).

9. feladat

Készítsen osztályt a Caesar-titkosításhoz! (A Caesar-titkosítás során a karakterek ASCII kódjaihoz egy egész számot adunk, így kapunk egy új karaktert. Az ABC-t körbe-körbe kezeli, tehát a Z után újra az A jön.)

Az osztálynak két metódusa van, egy kodol() és egy dekodol(). Az alap ABC az angol ABC kisbetűi. Az egyéb karaktereket (pl. szóköz, számjegyek) az egyszerűség kedvéért nem változtatjuk meg.

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 kódolással mentsen, ha ansi kódolással menti, akkor valószínűleg nem tudja beolvasni a fájlt.
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.