Beküldési határidő: ZH- időpontja!

1. Feladat

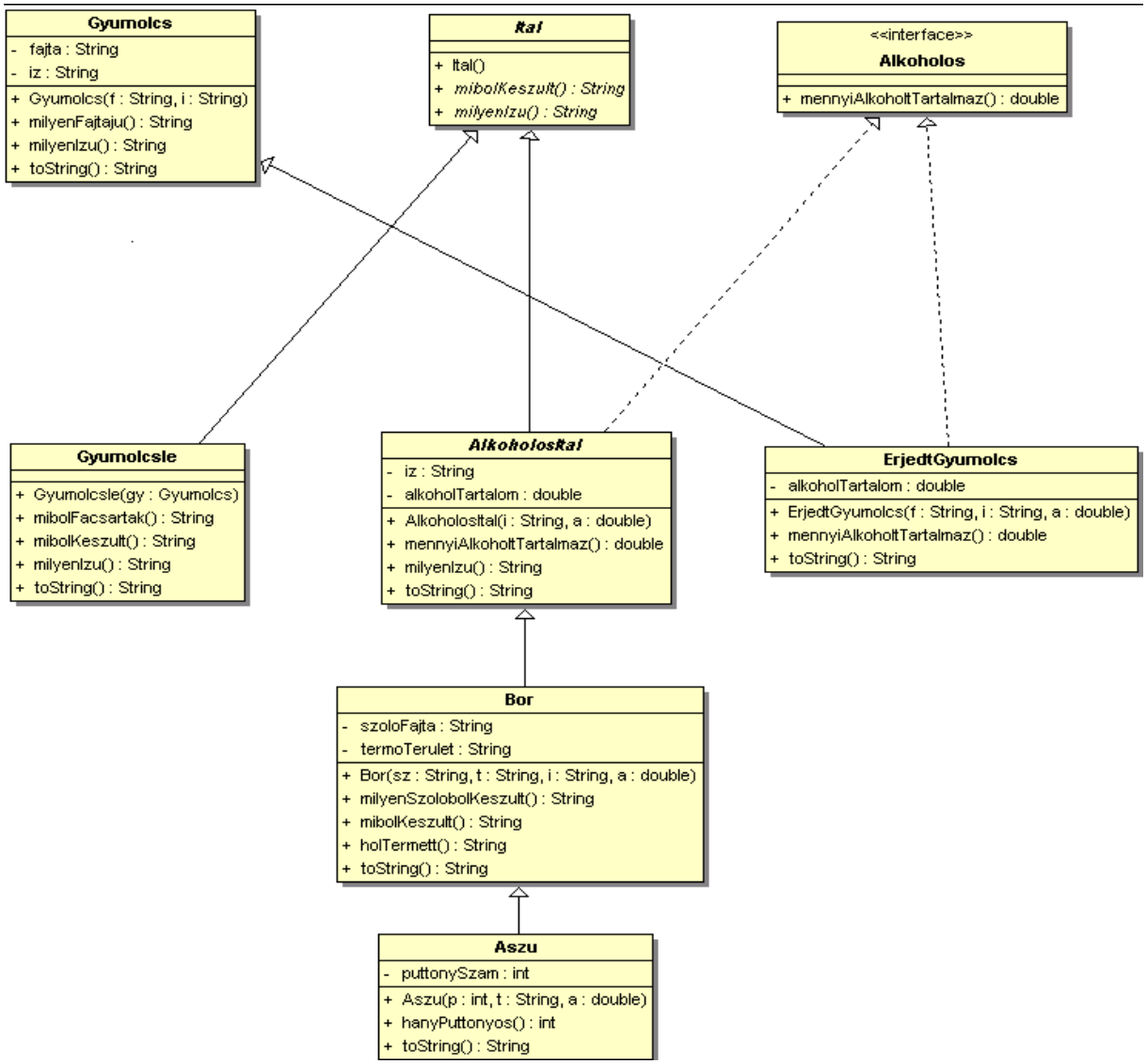
Implementálja az alábbi osztálydiagramnak megfelelő osztályokat.

A gyümölcs osztály leszármazott osztálya: ErjedtGyumolcs, ennek a konstruktora beállítja rendre a fajtát, ízt és alkoholtartalmat.

Az Ital nevű osztály abstract osztály(ha megfigyeled, dőlt betűkkel vannak benne a miből készült és milyen ízű metódusok, azaz ezek abstactak). Ebből leszármaztatjuk a Gyumolcsle és AlkoholosItal(szintén abstract) osztályokat. Az utóbbit tovább származtatjuk: ebből lesz a Bor osztály, aminek a gyereke az Aszu.

Van egy interfészünk is: Alkoholos néven. Ezt az AlkoholosItal és az ErjedtGyumolcs osztályok implementálják. Az implementáció mindkét estben az illető osztály alkoholTartalom értékének a visszaadását jelenti(azaz return alkoholTartalom)

Írjon egy Main osztályt is, amiben létrehoz egy Bor típusú tömböt, amiben több féle bor van, köztük Aszú típusú is, nem csak Bor típusú. Írjunk egy olyan függvényt, amely paraméterként megkapja ezt a tömböt, és visszaadja azon Aszú tömböt, amiben csak 5 puttonyos aszúk vannak.



2. Tekintse az alábbi Java kódot:

```
package ital;

public class Ital {
   protected String név;
   protected String kiszerelés;
   protected int ár;
}
```

   a. Egészítse ki az osztályt egy konstruktorral, amelynek segítségével mindhárom adattagjának kezdőérték adható!

b. Írja meg az adattagokhoz tartozó lekérdező metódusokat!
c. Definiálja felül a toString() metódust úgy, hogy az ital adatait az alábbi formában adja vissza (idézőjelek nélkül): „<név>, <kiszerelés>, <ár> Ft” (például „Coca-Cola, 5 dl, 150 Ft”)!
d. Definiálja felül az equals() metódust úgy, hogy két ital csak akkor legyen egyenlő, ha a nevük és a kiszerelésük megegyezik!
e. Egészítse ki az Ital osztály definícióját úgy, hogy példányai az áruk alapján összehasonlíthatók legyenek! A feladat megoldásához implementálja a Comparable vagy a Comparable<T> interfészt!
f. Származtasson az Ital osztályból egy nyilvános láthatóságú SzeszesItal osztályt, amelyet szintén az ital csomagban helyezzen el! A szeszesitalok az ital tulajdonságain kívül még egy valós értékű (más osztályból nem látható), csak lekérdezhető alkoholTartalomtulajdonsággal is rendelkeznek.
g. Egészítse ki a SzeszesItal osztályt egy konstruktorral, amelynek segítségével mind a négy adattagjának kezdőérték adható!
h. Definiálja felül az Ital osztályból örökölt toString() metódust úgy, hogy a szeszesital adatait az alábbi formátumban adja vissza (idézőjelek nélkül): „<alkoholTartalom>% alkoholtartalmú <név>, <kiszerelés>, <ár> Ft” (például „11.5% alkoholtartalmú Kékfrankos, 0.75 l, 1500 Ft”)! A megoldásban használja fel az örökölt toString() metódust!

i. Írjon a Dolgozat osztályban egy statikus metódust, amely paraméterként megkap egy italokat tartalmazó olyan tömböt, amelyben a szeszesitalok alkoholtartalma különböző. A metódus adja vissza a tömbben található három legmagasabb alkoholtartalmú szeszesitalt! Ha nincs ennyi szeszesital a tömbben, akkor a metódus null referenciát adjon vissza!
j. A főprogramban illusztrálja az előző feladatban megírt metódus működését! Definiáljon egy italok tárolására alkalmas tömböt, töltse fel különféle italokkal és szeszesitalokkal, hívja meg vele az előbb megírt metódust, majd — feltételezve, hogy nem null referenciát szolgáltatott a metódus —, írassa ki a képernyőre a visszakapott italokat!

k. Rendezze a fent definiált tömböt, majd írja ki a rendezett tömböt a képernyőre.

3. Tekintse a következő interfészt:
   Írja át java generikus formába(:használjon templétet <T>)

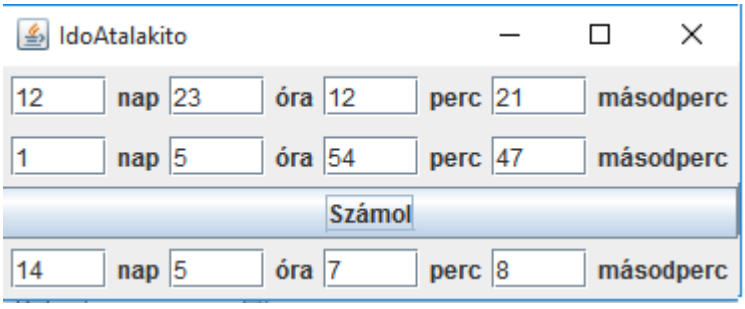
```
public interface Rendezheto {
 public boolean egyenlo(Object o);
 public boolean nagyobbMint(Object o);
 public boolean kisebbMint(Object o);
 }
```

   Adott a következő osztály:

```
public class Tea implements Rendezheto {
 public String nev;
 public int ár;
 public Tea(String s,int j) {
  nev=s;
  ar=j;
  }
}
```

   Egészítse ki a Tea osztályt az interfész implementációjával.

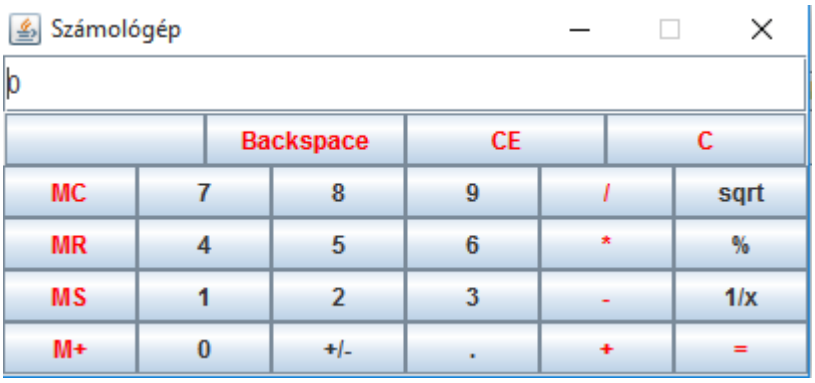
4. Hozza létre az alábbi grafikus felületet! Az idő átalakító alkalmazásunk első két sora két időpontot kap, majd a számol gomb lenyomása után kiszámolja ezek összegét. A két időpont nap, óra, perc, másodperc alakú. A számolás végén ugyan ilyen formátumú időpont kerül kiíratásra a negyedik sorba. Ügyeljünk arra, hogy a huszonnegyedik óra után a napok száma növekszik, míg a hatvanat túllépve az órák, percek, másodpercek váltanak.



5. Készítse el a Windows rendszerbe beépített számológéphez hasonló grafikus felületet a minta alapján. A gombok közül elég, ha csak az összeadás művelet működik tetszőleges számú szám esetén.

   Megjegyzés: A gombokat nem muszáj színezni, lehet minden fekete. Ha valaki mégis meg szeretné próbálni, html ismereteit fitogtatva tudja megtenni. Például a CE gomb esetén (mindenre így megy, csak a CE-t kell cserélni):

   JButton ce = new JButton("<html><body>CE</body></html>");



6.

a) Írj egy interfészt **Pontozhato** néven az **egyetem** csomagba, amely deklarál egy ***megfelelt*** nevű függvényt, amelynek nincs paramétere, és visszatérési értéke boolean.

b) Írj egy osztályt **Dolgozat** néven a **zh** csomagba, amely konstruktora a dolgozat pontszámát (egész szám) várja paraméterül. Ez a ***Dolgozat*** osztály megvalósítja a ***Pontozhato*** interfészt! A ***megfelelt*** függvény implementációja ellenőrzi, hogy az adott objektum pontszáma nagyobb-e mint 35. Ha igen akkor igazzal tér vissza, ha nem akkor hamissal. Ha a pontszám -1, akkor lépjen ki hamissal és írja ki a „Nem írt” üzenetet. Legyen a ***Dolgozat*** osztály String-é alakítható, továbbá legyen get/set metódus az adattaghoz.

c) Írj egy osztályt **Hallgato** néven, amely rendelkezzen egy **mennyitKeszult** (egész [0-5] között) és egy **sokatTanul** (boolean) adattaggal. A ***Hallgato*** osztálynak legyen egy konstruktora, amellyel beállíthatjuk, hogy mennyit készült, és hogy sokat tanul-e. Az osztálynak legyen még egy **dolgozatotIr** függvénye, amely nem vár bemeneti

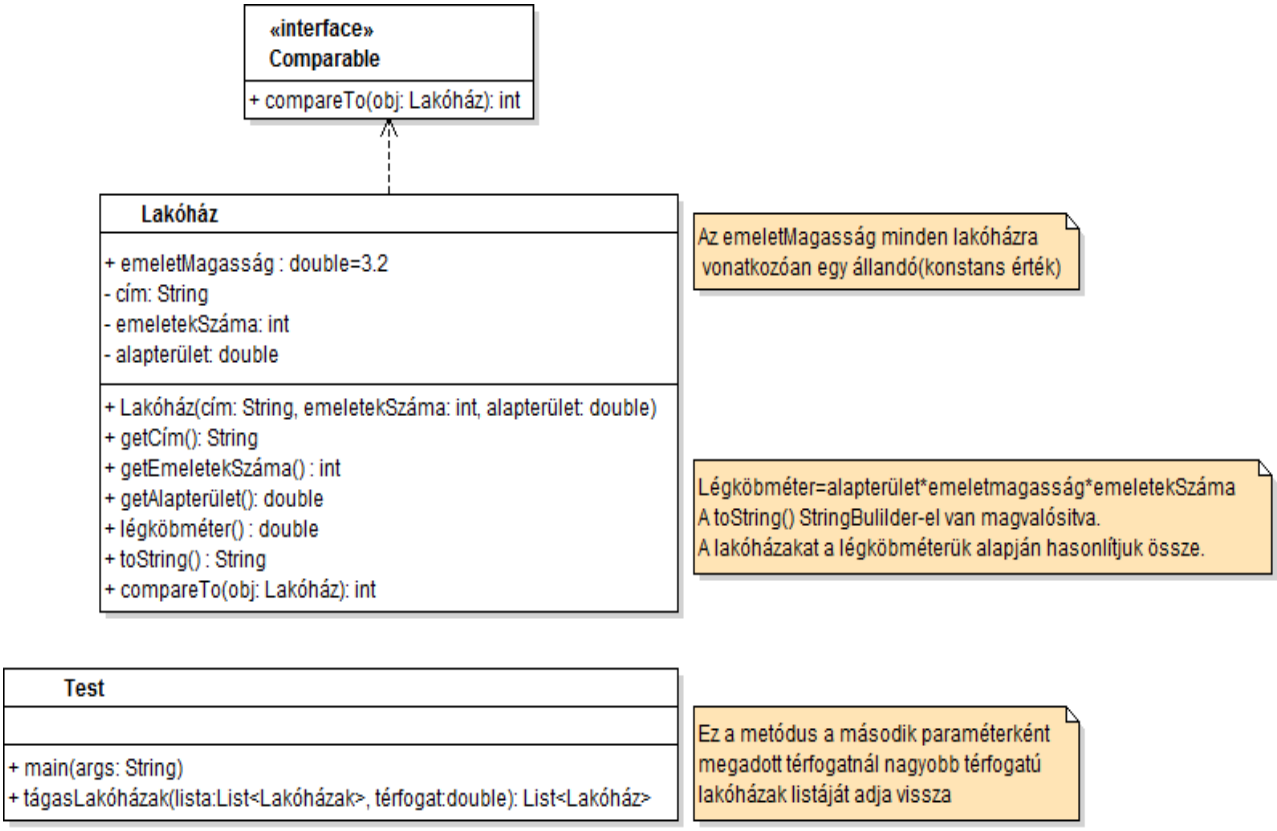
paramétert, és egy ***Dolgozat*** típusú objektumot ad vissza. A dolgozat alakulása a következő legyen:

- Ha az adott ***hallgato*** sokat tanul, akkor generáljunk egy véletlen számot 4 és 7 között, majd szorozzuk ezt meg a **mennyitKeszult** adattagjával, így kapjuk a dolgozat pontszámát,
- Ha az adott ***hallgato*** nem tanul sokat, akkor generáljunk egy véletlen számot 0 és 5 között, majd szorozzuk ezt meg a **mennyitKeszult** adattagjával, így kapjuk a dolgozat pontszámát. A kiszámolt pontszámmal hozzunk létre egy dolgozatot, majd ezzel térjünk vissza.
- Egy ***hallgato*** 2% eséllyel nem megy el egy dolgozatra, ekkor a dolgozat pontszáma -1, és egy ”Nem írt” String legyen.

d) Írj futtatható osztályt, amely a standard inputról olvas. A bemenet első sora megadja hány hallgató objektumot hozzunk létre. A többi sor egy számot, és egy ”igaz” vagy ”hamis” szöveget tartalmaz (az igaz true-nak a hamis false-nak felel meg). Ezekre az adatokra hívjuk meg a hallgató konstruktorát. Helyezzük ezeket egy tömbbe mely csak hallgató típusú objektumokat tárol, majd az összes tömbben szereplő hallgatóra hajtsuk végre a következőt:

e) Írassunk meg vele két dolgozatot a **dolgozatotIr** segítségével, és a másodiknak hívjuk meg a ***megfelelt*** függvényét. Amennyiben az adott hallgató mindkét dolgozata megfelelt, írjuk ki a ”megfelelt” sort, ellenkező esetben a ”nem felelt meg” sort. Ha valamelyik dolgozat „Nem írt” üzenettel „tér vissza”, akkor a hiba üzenetét is a kimenetre írja. A hibás formátumú sorokat hagyjuk figyelmen kívül.

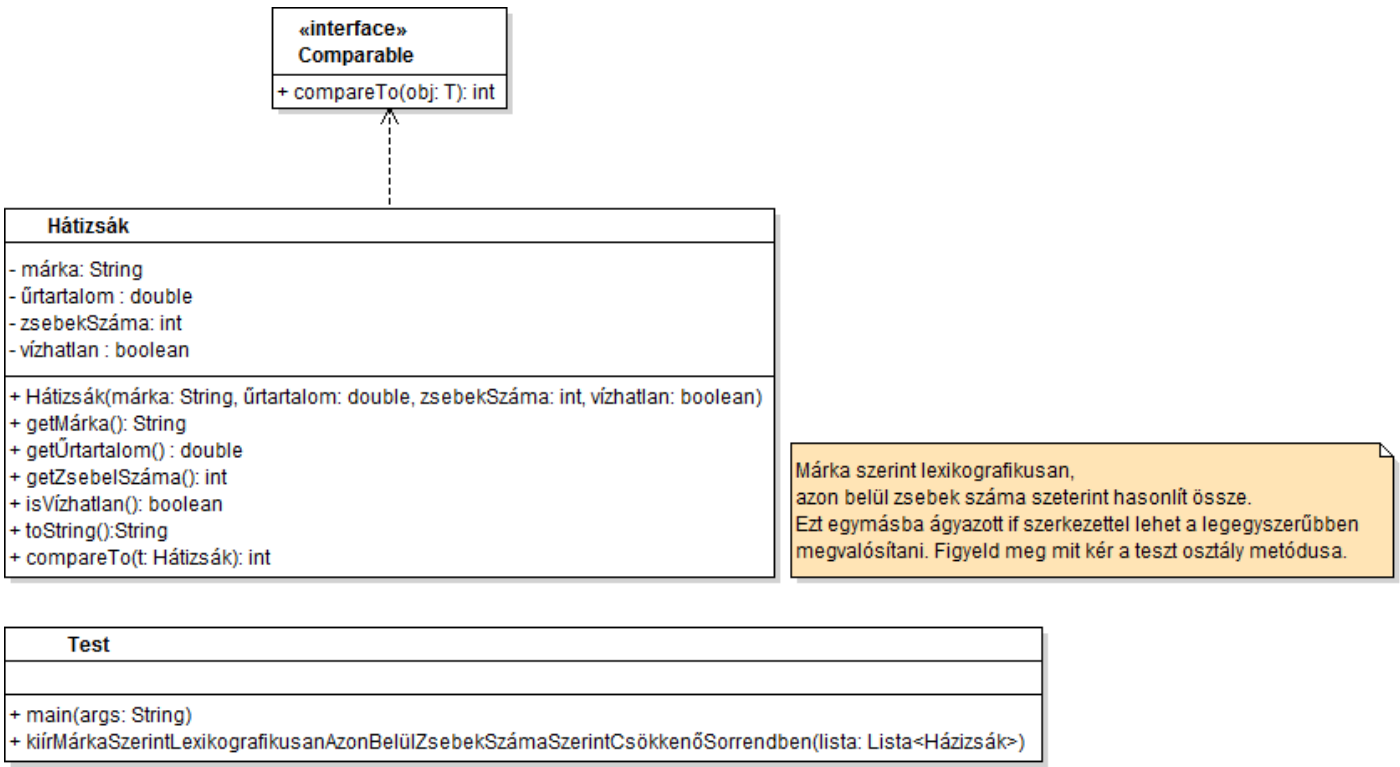
7.



Hozzuk létre azt a legalább 10 lakóházból álló listát, amivel a metódust tesztelni tudjuk.

Írassuk a végeredményként kapott lista elemeit ***rendezve***!

9.



Tesztelje is a megfelelő (legalább 10 elemű) listával a fenti kiír metódust.

10.

```
public class Autó {
   private String rendszám;
   private int teljesítmény;
   private boolean automata;
}
```

a) Egészítse ki az osztályt egy konstruktorral, amelynek segítségével mindhárom adattagjának kezdőérték adható!

b) Írja meg a három adattaghoz tartozó lekérdező metódusokat!

c) Definiálja felül az Object osztályból örökölt toString() metódust úgy, hogy az autó mindhárom adattagját egy sorban adja vissza!

d) Definiálja felül az Object osztályból örökölt equals() metódust úgy, hogy két autó csak akkor legyen egyenlő, ha a rendszámuk megegyezik!

e) Egészítse ki az Autó osztály definícióját úgy, hogy a példányai teljesítményük alapján összehasonlíthatók legyenek!

f) Származtasson az Autó osztályból egy Teherautó osztályt! A teherautók az autó tulajdonságain kívül még egy egész értékű, kg-ban értelmezett teherbírás nevű mezővel rendelkeznek.

g) Egészítse ki a Teherautó osztályt egy konstruktorral, amelynek segítségével mind a négy adattagjának kezdőérték adható!

h) Definiálja felül az Autó osztályból örökölt toString() metódust úgy, hogy a teherautó mind a négy adattagját egy sorban adja vissza!

i) Készítsen egy Main nevű osztályt, amely a főprogramot tartalmazza! A főprogramban hozzon létre egy Autó típusú objektumot, amelynek a rendszámát beolvassuk a standard inputról, a teljesítménye 100 lóerő és automataváltós! Írja ki a képernyőre a létrehozott objektumot!

j) Egészítse ki a főprogramot egy olyan kódrészlettel, amely beolvassa a billentyűzetről 2 autó és 2 teherautó adatait (ilyen sorrendben soronként egyet-egyet), és elhelyezi őket egy négyelemű tömbben! Az autók és a teherautók egyes adatait szóköz választja el egymástól. Például:

```
ABC123 80 true
```

```
XYZ876 55 false
TIR666 130 true 25000
PQR111 125 false 20000
```

k) Egészítse ki az Autó osztályt egy olyan metódussal, amely igazzal tér vissza, ha az autó rendszáma szabályos, azaz pontosan 6 karakter hosszúságú, amelyből az első három (angol) betű, a második három pedig (decimális) számjegy!

l) Írjon a Main osztályban egy statikus metódust, amely paraméterként megkap egy autókat tartalmazó tömböt, és visszaad egy teherautókat tartalmazó listát, amely azokat a tömbbeli Teherautó típusú objektumokat tartalmazza, amelyeknek a teherbírása meghaladja a 20 tonnát!

m) Írjon a Main osztályban egy statikus metódust, amely paraméterként megkap egy autókat tartalmazó listát, és képernyőre írja a három legnagyobb teljesítményű autót! Ha a lista háromnál kevesebb elemet tartalmaz, akkor írja ki mindet!

n) Egészítse ki az Autó osztály definícióját olyan lehetőséggel, amelynek segítségével bármelyik osztályból lekérdezhető, hogy hány autót hoztak eddig forgalomba (azaz hány példánya létezik az Autó osztálynak)!

o) Egészítse ki az Autó osztály definícióját olyan lehetőséggel, amelynek segítségével „extrákat” (nem szériafelszereléseket) tárolhatunk minden autónál! Tegye lehetővé, hogy egy autó új extrát kapjon, illetve hogy egy létező extra törölhető legyen! Egy új autó semmilyen extrával nem rendelkezik.

11. Adott az alábbi Java osztály:

```
public class Személy
{
    protected String név;
    protected int életkor;
    private boolean férfi;
}
```

a. Egészítse ki az osztályt egy konstruktorral, amelynek segítségével mindhárom adattagjának kezdőérték adható!

b. Írja meg a három adattaghoz tartozó lekérdező és beállító metódusokat!

c. Definiálja felül az Object osztályból örökölt toString() metódust úgy, hogy a személy mindhárom adattagját egy sztringgé összefűzve, egymástól egy-egy szóköz karakterrel elválasztva adja vissza!

d. Egészítse ki a Személy osztály definícióját úgy, hogy a példányai életkoruk alapján összehasonlíthatók legyenek!

e. Származtasson a Személy osztályból egy Hallgató osztályt! A hallgatók a személy tulajdonságain kívül még egy valós értékű átlagtulajdonsággal rendelkeznek. Egészítse ki az osztályt egy konstruktorral, amelynek segítségével mind a négy adattagjának kezdőérték adható!

f. Definiálja felül a Személy osztályból örökölt toString() metódust úgy, hogy a hallgató négy adattagját külön sorokba írva, de egy sztringgé összefűzve adja vissza!

g. Származtasson a Személy osztályból egy Oktató osztályt! Az oktatók a személy tulajdonságain kívül még egy String típusú tanszék tulajdonsággal rendelkeznek. Egészítse ki az osztályt egy konstruktorral, amelynek segítségével mind a négy adattagjának kezdőérték adható!

h. Készítsen egy Main nevű osztályt, amely a főprogramot tartalmazza! A főprogramban hozzon létre egy Személy típusú objektumot, amely egy Mona Liza nevű 20 éves nőt reprezentál! Írja ki a képernyőre a létrehozott objektumot!

i. Egészítse ki a főprogramot egy olyan kódrészlettel, amely beolvassa a billentyűzetről 2 hallgató és 2 oktató adatait (ilyen sorrendben, soronként egyet-egyet), és elhelyezi őket egy négyelemű tömbben! A hallgatók és az oktatók egyes adatait szóköz választja el egymástól.

   Például:

```
Gabriella 23 false 3.5
Gábor 22 true 4.2
Péter 36 true Grafika
Andrea 32 false Számítástudomány
```

j. Egészítse ki a Hallgató osztályt egy olyan metódussal, amely igazzal tér vissza, ha a hallgató jó képességű, azaz az átlaga legalább 4!

k. Írjon a Main osztályban egy statikus metódust, amely paraméterként megkap egy személyeket tartalmazó **halmazt**, és visszatér a jó képességű hallgatók életkorainak az átlagával!

l. Egészítse ki a Hallgató osztályt olyan lehetőséggel, amelynek segítségével bármely más osztályból beállítható, de nem lekérdezhető, hogy milyen átlag felett tekintünk

egy hallgatót jó képességűnek! Ha szükséges, írja újra a az előzőekben megírt metódust!

m. Készítsen egy Egyetem osztályt, amely tetszőleges számú hallgató és oktató tárolására alkalmas (Help: nyilván itt listát kell használni)! Legyen lehetőség új hallgató és új oktató felvételére, valamint létező hallgató, illetve létező oktató törlésére (Help: például add() és remove() metódusok használata a lista esetén)!

n. Egészítse ki az Egyetem osztályt egy olyan metódussal, amely képernyőre írja a három legfiatalabb hallgatóját! Ha háromnál kevesebb hallgatója van az egyetemnek, akkor a metódus írja ki az összes hallgatót!

o. Egészítse ki az Egyetem osztályt egy olyan metódussal, amely az IT tanszék női oktatóinak neve végéhez hozzáfűz egy felkiáltójelet!

Karácsonyi hozzávalók

A nagymama karácsony alkalmából ünnepi ebédet készít a család számára. Előkeresi a kedvenc ételek receptjeit, és soronként összeírja az egyes ételek hozzávalóit és a szükséges mennyiségeket, minden sorban egy pontosvesszővel elválasztva a hozzávaló nevét a mennyiségtől. Az ön feladata, hogy írjon egy programot, amely a standard bemenetről beolvassa a nagymama által összeírt hozzávalókat, összegzi az egyes hozzávalókból kért mennyiségeket, majd a mennyiségek alapján csökkenő sorrendben kiírja az összegzett adatokat a standard kimenetre a példa kimenetben látható formában! Ha több hozzávalóból is azonos mennyiségre lenne szükség, akkor őket a hozzávalók neve alapján ábécérendben listázza ki!

Példa bemenet

```
paradicsom;4
paprika;3
zeller;5
paradicsom;4
paprika;5
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
paprika;8
paradicsom;8
zeller;5
```

letöltés szöveges állományként

Osztálypénz

Írjon programot, amely a standard bemenetről egy általános iskolai osztály tanulóinak osztálypénz-befizetéseit olvassa be, majd azokat az egyes gyerekek által befizetett pénzek összege alapján csökkenő sorba rendezi, és soronként kiírja a standard kimenetre! Ha azonos összeget fizetett be több gyerek is, akkor a rájuk vonatkozó adatokat a nevek ábécérendjében kell a standard kimenetre írni! Az egyszerűség kedvéért feltételezheti, hogy az osztályban nincs két azonos nevű gyerek. A gyerekek nevét az általuk befizetett összegektől mind a bemeneten, mind a kimeneten egy pontosvessző karakter választja el egymástól.

Példa bemenet

```
Pisti;500
Feri;1000
Peti;200
Feri;300
Pisti;800
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
Feri;1300
Pisti;1300
Peti;200
```

letöltés szöveges állományként

Zárthelyi eredmények

Írjon programot, amely a standard bemenetről a másodéves mérnökinformatikus hallgatók zárthelyi eredményeit olvassa be, majd azokat az egyes hallgatók által szerzett pontok összege alapján csökkenő sorba rendezi, és soronként kiírja a standard kimenetre! Ha azonos pontszámot ért el több hallgató is, akkor a rájuk vonatkozó adatokat egyedi azonosítójuk szerint ábécérendben kell a standard kimenetre írni! A beolvasott adatokat a példa bemenetnek, a kiírtakat a példa kimenetnek megfelelően formázza!

Példa bemenet

```
Edit;18
Alex;9
Csilla;13
Kelemen;20
Edit;3
Kelemen;1
Csilla;7
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
Edit: 21 pont
Kelemen: 21 pont
Csilla: 20 pont
Alex: 9 pont
```

letöltés szöveges állományként

Parkoló

Írjon programot, amely a standard bemenetről parkolók adatait olvassa be! Egy parkoló jellemzői: (1) a címe, (2) a területe (valós számként, négyzetméterben kifejezve), (3) a kapacitása (hogy legfeljebb hány autó fér el benne, egész számként) és (4) a benne parkoló autók rendszáma. Ez a következő formában szerepel egy-egy feldolgozandó sorban:

<cím>;<terület>;<kapacitás>[;*<rendszám>*]...

ahol a *cím* egy sztring, a *terület* egy valós szám, a *kapacitás* egy egész szám, a *rendszám*ok pedig ismét sztringek. A sztringek egyike sem tartalmaz pontosvesszőt karaktert, a pontosvessző karakterek csak a sor egyes elemeinek az elválasztására szolgálnak.

A programja írja a standard kimenetre azoknak a parkolóknak a címét és a parkoló szabad helyeinek a számát a példa kimenetben megadott módon a szabad helyek száma szerint csökkenő sorrendbe rendezve, ahol még legalább három autó számára van szabad hely! Ha több parkolóban is azonos mennyiségű szabad hely lenne, akkor őket a címük szerinti ábécérendben írja a standard kimenetre a program!

Példa bemenet

```
Debrecen;300.25;20
Budapest;500.0;30;ABC123;CXK962
Miskolc;140.5;2;CGI299
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
Budapest: 28 szabad hely
Debrecen: 20 szabad hely
```

letöltés szöveges állományként

Bankkártyaszám

A bankkártyaszám egy 16 tagból álló számsor, melynek első tagja az iparág-azonosító. Így például a 3-mal kezdődnek bizonyos hitelkártyák, mint például az American Express, 4-gyel, illetve 5-tel az általános banki tranzakcióknál és vásárlásoknál használt kártyák, mint a Visa vagy a Mastercard. Az első hat szám (beleértve az előbb említett legelsőt is) a kibocsátónak az azonosítója, míg a rákövetkező, 7–15. pozícióban szereplő, véletlenszerűen generált számok a tulajdonos azonosítására alkalmasak. A számsor utolsó tagja a Luhn-algoritmus kiegészítő, ellenőrző tagja.

A Luhn-algoritmussal lehet tesztelni – többek között – egy bankkártyaszám érvényességét. Az algoritmus lépései a következők:

1. A legjobboldalibb számjegytől (amely, ugye, az ellenőrző számjegy) bal felé haladva kétszerezzük meg minden második számjegy értékét! Ha a kétszeres szorzat értéke nagyobb 9-nél, akkor összegezzük a szorzat számjegyeit, és ezzel helyettesítsük az eredeti számjegyet! (Ez utóbbi értéket úgy is megkaphatjuk, ha a 9-nél nagyobb számból kivonunk 9-et.)
2. Adjuk össze az átalakított számsorozat számjegyeit!
3. Ha az összeg 0-ra végződik, azaz osztható 10-zel, akkor a számsorozat érvényes, egyébként érvénytelen.

Példa számítás egy érvényes bankkártyaszámra

Bankkártyaszám	5	4	2	5	9	0	3	1	4	2	6	5	7	3	5	3	
Megduplázott páros pozíciók	10	4	4	5	18	0	6	1	8	2	12	5	14	3	10	3	
Számjegyek összege	1	4	4	5	9	0	6	1	8	2	3	5	5	3	1	3	=60

Írjon programot, amely a standard bemenetről soronként egy-egy 16 karakter hosszúságú, kizárólag számjegy karakterekből álló sztringet olvas be állományvégjelig! A program minden egyes beolvasott sztringről döntse el, hogy az érvényes bankkártyaszám-e, és ha igen, akkor egy „YES”, egyébként pedig egy „NO” szót tartalmazó sort írjon a standard kimenetre!

Példa bemenet

```
5425903142657353
5425903142657354
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
YES
NO
```

letöltés szöveges állományként

Vízgyűjtők

Írjon programot, amely meghatározza és a standard kimenetre írja a parancssori argumentumaiként megadott vízgyűjtőkbe (akár közvetlenül, akár közvetetten, más vízgyűjtőkön keresztül) befolyó vízgyűjtők számát! A program a standard bemenetről a következő formában olvassa a feldolgozandó adatokat:

vízgyűjtő[;*vízgyűjtő*]

Abban az esetben, ha a sor csak egyetlen vízgyűjtő nevét tartalmazza, akkor egy olyan vízgyűjtőről van szó, amely más vízgyűjtők vizét összegyűjtheti ugyan, de ő maga nem folyik bele egyetlen más vízgyűjtőbe sem. Minden más esetben az elsőként megadott vízgyűjtő belefolyik a másodikként megadott vízgyűjtőbe.

A kimenet egyes soraiba a parancssori argumentumokban megadott vízgyűjtők nevét és az általuk – közvetlenül vagy közvetetten – összegyűjtött vízgyűjtők számát kell írni a példa kimenetben megadott formában.

Példa bemenet

```
Tisza;Duna
Duna;Fekete-tenger
Sió csatorna;Duna
Zala;Balaton
Csendes-óceán
Bodrog;Tisza
Don;Fekete-tenger
Rába;Duna
Dnyeper;Fekete-tenger
```

letöltés szöveges állományként

Parancssori argumentumok

```
Duna Amazonas Fekete-tenger Zala
```

letöltés szöveges állományként

A futtatás eredménye a standard kimeneten

```
Duna: 4
Amazonas: 0
Fekete-tenger: 7
Zala: 0
```

letöltés szöveges állományként

Alakul a molekula

Írjon programot, amely a standard bemenet első néhány sorából kémiai elemek vegyjelét és nevét olvassa be! Az elemek vegyjelét és nevét a következő formában tartalmazzák a bemenet sorai:

vegyjel: *elemnév*

Az elemek felsorolásának a végét egy mínuszjelet tartalmazó sor jelzi. A további sorokban molekulák nevei és az őket alkotó atomok felsorolása szerepel az alábbi formában:

molekulanév: *vegyjel*[, *vegyjel*]...

A programjának minden molekula esetén egy listát kell a standard kimenetre írnia, amely összegzi a molekulát alkotó elemeket a példa kimenetben látható formában. A molekulákat nevük szerint lexikografikus sorrendben kell felsorolni a kimeneten. Egy-egy molekula elemeinek a listázásakor előre kerüljenek azok az elemek, amelyekből a legtöbb atomot tartalmazza a molekula! Azokat az elemeket, amelyekből azonos darabszámú atomot tartalmaz a molekula, neveik lexikografikus sorrendjében adja meg!

Példa bemenet

```
H:hidrogén
O:oxigén
Na:nátrium
Cl:klór
-
víz:H,O,H
konyhasó:Na,Cl
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
konyhasó:
1 klór
1 nátrium
víz:
2 hidrogén
1 oxigén
```

letöltés szöveges állományként

Jégkorongmérkőzések (Java)

Írjon programot, amely a standard bemenetről jégkorongmérkőzések adatait olvassa be, és azok alapján elkészíti a csapatok rangsorolását! A jégkorongmérkőzések adatai a következő formában adottak:

csapatA–*csapatB*: *gólA1*–*gólB1*, *gólA2*–*gólB2*, *gólA3*–*gólB3*[, *gólAh*–*gólBh*[, *gólAsz*–*gólBsz*]]

A jégkorongmérkőzések két csapatát jelöli a *csapatA* és a *csapatB* sztring. A következő három gólpár (*gólA1*–*gólB1*, *gólA2*–*gólB2* és *gólA3*–*gólB3*) azt mutatja meg, hogy hány gólt ütöttek a csapatok az egyes harmadokban (a jégkorongmérkőzések három harmadból állnak). Ha a három harmad után döntetlen lenne az állás, akkor hosszabbítás következik (ennek az eredményét jelzi *gólAh*–*gólBh*), míg ha ezután is döntetlen az állás, akkor szétlövéssel (*gólAsz*–*gólBsz*) döntik el, hogy ki nyeri a mérkőzést. A szétlövés eredménye sohasem lehet döntetlen.

Ha a küzdelem már három harmad után eldől, és nincsen szükség hosszabbításra, akkor a győztes csapat 3 pontot, a vesztes 0 pontot kap. Ha csak hosszabbítás vagy szétlövés után dől el a küzdelem, akkor a győztes csapat 2, míg a vesztes 1 pontot szerez.

A programja összegezze az egyes csapatok által szerzett pontokat, és ezek alapján állítsa csökkenő sorrendbe őket! Ha két csapatnak azonos lenne a pontszáma, akkor az a csapat szerepeljen előrébb, amelyik több gólt ütött! Ha ebben a tekintetben is megegyezne két csapat helyezése, akkor őket nevük alapján állítsa lexikografikusan növekvő sorrendbe!

A program írja a standard kimenetre a csapatok adatait a példa kimenetben megadott formában, soronként egy-egy csapat adataival!

Példa bemenet

```
Sweden-Latvia:1-0,0-0,0-1,1-0
Czech Republic-Russia:1-0,1-0,1-0
Latvia-Czech Republic:0-2,1-0,2-1,0-0,0-1
Kazakhstan-Russia:3-3,0-1,1-2
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
Czech Republic: 5 points (7 scored goals)
Russia: 3 points (6 scored goals)
Latvia: 2 points (4 scored goals)
Sweden: 2 points (2 scored goals)
Kazakhstan: 0 points (4 scored goals)
```

letöltés szöveges állományként

Mozisztárok (Java)

Írjon programot, amely a standard bemenetről jól ismert mozifilmek adatait olvassa be, és azok alapján elkészíti a benne szereplő színészek rangsorolását! A mozifilmek adatai a következő formában adottak:

filmcím (*évszám*): *színésznév*[, *színésznév*]...

A film címe és a színészek nevei sztringek, míg az évszám egy négyjegyű pozitív egész szám, amelyre $1900 \leq$ *évszám* ≤ 2016 teljesül.

A programja számolja össze, hogy az egyes színészek hány filmben szerepeltek, és írja ki a neveiket szerepléseik száma szerint csökkenő sorrendben! Ha két vagy több színész is azonos számú filmben szerepelt volna, akkor őket nevük szerint lexikografikusan növekvő sorrendben írja a kimenetre!

A színészek neve utáni sorokban jelenítse meg azoknak a filmeknek az adatait is, amelyekben játszottak! A filmek adatait a példa kimenetben megadott formában írja a standard kimenetre: először a film elkészítésének az évszámát, majd attól egy kettősponttal és egy szóköz karakterrel elválasztva a film címét! Ha egy színészhez több film is tartozna, akkor őket elkészítésük sorrendjében írja a kimenetre! Ha egy színész több filmben is szerepelt volna ugyanabban az évben, akkor a filmeket címük szerint lexikografikusan növekvő sorrendben szerepeltesse a kimeneten (lásd a példa kimenetet)!

Példa bemenet

```
A tegla (2006):Leonardo DiCaprio,Matt Damon,Jack Nicholson,Mark Wahlberg,Martin Sheen
Eredet (2010):Leonardo DiCaprio,Ken Watanabe,Joseph Gordon-Levitt,Marion Cotillard
Ted (2012):Mark Wahlberg,Mila Kunis,Seth MacFarlane
Kapj el, ha tudsz (2002):Leonardo DiCaprio,Tom Hanks,Christopher Walken
Angyalok es demonok (2009):Tom Hanks,Ewan McGregor,Ayelet Zurer
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
Leonardo DiCaprio
2002: Kapj el, ha tudsz
2006: A tegla
2010: Eredet
Mark Walhberg
2006: A tegla
2012: Ted
Tom Hanks
2002: Kapj el, ha tudsz
2009: Angyalok es demonok
Ayelet Zurer
2009: Angyalok es demonok
Christopher Walken
2002: Kapj el, ha tudsz
Ewan McGregor
2009: Angyalok es demonok
Jack Nicholson
2006: A tegla
Joseph Gordon-Levitt
2010: Eredet
Ken Watanabe
2010: Eredet
Marion Cotillard
2010: Eredet
Martin Sheen
2006: A tegla
Matt Damon
2006: A tegla
Mila Kunis
2012: Ted
Seth MacFarlane
2012: Ted
```

letöltés szöveges állományként

Mozisztárok (Java)

Írjon programot, amely a standard bemenetről jól ismert színészek adatait olvassa be, és azok alapján elkészíti azoknak a filmeknek a rangsorolását, amelyekben szerepeltek! A színészek adatai a következő formában adottak:

név (*születési_év*):*filmcím*[;*filmcím*]...

A színész adataitól a filmeket ':' (kettőspont), a filmeket egymástól ';' (pontosvessző) karakterek választják el. E két karakter sehol máshol nem bukkan fel az egyes sorokban. A színész neve és a filmek címei sztringek (mindegyik tartalmazhat tetszőleges számú szóköz karaktert), míg a *születési_év* egy négyjegyű pozitív egész szám, amelyre $1900 \leq$ *születési_év* ≤ 2016 teljesül.

A programja számolja össze, hogy az egyes filmekben hány színész szerepelt, és írja ki a filmek címeit a bennük szereplő színészek száma szerint csökkenő sorrendben! Ha két vagy több filmben is azonos számú színész szerepelt volna, akkor azokat címük szerint lexikografikusan növekvő sorrendben írja a kimenetre! A filmcímek utáni sorokban jelenítse meg a bennük szereplő színészeknek az adatait is! A színészek adatait a példa kimenetben megadott formában írja a standard kimenetre: először a színész születésének az évszámát, majd attól egy kettősponttal és egy szóköz karakterrel elválasztva a színész nevét! Ha egy filmben több színész is játszott volna, akkor őket születésük időrendjében írja a kimenetre! Ha több színész is ugyanabban az évben született volna, akkor őket nevük lexikografikusan növekvő sorrendjében szerepeltesse a kimeneten (lásd a példa kimenetet)!

Példa bemenet

```
Leonardo DiCaprio (1974):A tegla;Eredet;Kapj el, ha tudsz
Matt Damon (1970):A tegla
Jack Nicholson (1937):A tegla
Mark Wahlberg (1971):A tegla;Ted
Martin Sheen (1940):A tegla
Ken Watanabe (1959):Eredet
Joseph Gordon-Levitt (1981):Eredet
Marion Cotillard (1975):Eredet
Mila Kunis (1983):Ted
Seth MacFarlane (1973):Ted
Tom Hanks (1956):Kapj el, ha tudsz;Angyalok es demonok
Christopher Walken (1943):Kapj el, ha tudsz
Ewan McGregor (1971):Angyalok es demonok
Ayelet Zurer (1969):Angyalok es demonok
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
A tegla
1937: Jack Nicholson
1940: Martin Sheen
1970: Matt Damon
1971: Mark Walhberg
1974: Leonardo DiCaprio
Eredet
1959: Ken Watanabe
1974: Leonardo DiCaprio
1975: Marion Cotillard
1981: Joseph Gordon-Levitt
Angyalok es demonok
1956: Tom Hanks
1969: Ayelet Zurer
1971: Ewan McGregor
Kapj el, ha tudsz
1943: Christopher Walken
1956: Tom Hanks
1974: Leonardo DiCaprio
Ted
1971: Mark Walhberg
1973: Seth MacFarlane
1983: Mila Kunis
```

letöltés szöveges állományként