1. Önt egy bolt **árukészlet-nyilvántartó rendszerének** a **fejlesztésével** bízzuk meg. Feladata a **megadott szempontrendszer alapján** olyan **osztályhierarchia** kialakítása, amellyel megvalósítható ez az árukészlet-nyilvántartás.

   Az osztályhierarchiában a szükséges osztályok szerkezetét ön határozhatja meg.

   Tetszőleges számú és funkciójú saját kivételosztályt is definiálhat a feladathoz. Az egyes **tevékenységekhez** definiáljon **külön metódusokat**!

   A fejlesztés során vegye figyelembe az alábbiakat:

   a. Az árukészlet-nyilvántartó rendszerben **különféle termékeket kell nyilvántartani**. A termékeket **két csoportra** osztjuk: **élelmiszerekre** (rövidítés: E) és **tartós fogyasztási cikkekre** (rövidítés: T).

      Minden terméknek van **neve** (sztring típusú), **mennyiségi egysége** (sztring típusú) és **egységára** (valós típusú). Az élelmiszereknek ezen kívül **szavatossági idejük** is van (egész szám, napokban kifejezve), a tartós fogyasztási cikkeknek nincs ilyen jellemzőjük.

   b. A bolt aktuális árukészlete egy **szöveges állományban** van tárolva, az állomány minden sorában **egy-egy termék leírásával** a következő formátumban:

      [T|E];<terméknév>;<ár>;<egység>[;<szavatosság>];<mennyiség>

      Például:

      E;kígyóuborka;129.0;darab;7;15
      E;kígyóuborka;129.0;darab;3;5
      T;16 cm-es fazék;1299.0;darab;5
      E;élő ponty;799.0;kg;20;10
      T;súlyzókészlet;5990.0;szett;5
      E;2.8%-os tej;189.0;liter;10;20
      E;2.8%-os tej;189.0;liter;5;10

c. A bolt árukészletét
    i. **be** kell tudni **olvasni** egy **megadott nevű szöveges állományból**,
    ii. a beolvasott adatokkal dolgozni a d. pontban meghatározott módon, illetve
    iii. egy **megadott nevű szöveges állományba** kell tudni **írni**.
d. Az árukészlet kezelésével kapcsolatban a következő tevékenységek megvalósítását kell elvégezni:
    i. Az **aktuális árukészlet bővítése** egy **termékkollekcióval**, új áruk érkeztetése.

       Ilyenkor a termékkollekció elemeivel értelemszerűen kell bővíteni az árukészletet, azaz **ha volt már** az adott jellemzőkkel rendelkező termék, akkor **csak a darabszámát** kell növelni, **ha nem volt**, akkor **fel kell venni** az árukészletbe (ilyenkor a mennyiség értéke alapértelmezésként 1 legyen).

    ii. **Termék eladása** az árukészletből. Ez kétféleképpen történhet:
        1. A vásárló **pontosan leírja** a terméket, annak **minden jellemzőjével**, és **megadja** a vásárolni kívánt **mennyiséget** is.

           Ilyenkor

           a. ha **nincs** a vásárló által megjelölt termék az árukészletben, akkor ezt **egy figyelmeztető üzenettel paraméterezett kivétel dobásával** jelezni kell;
           b. ha **van ilyen** termék az árukészletben, de **nincs belőle megfelelő mennyiség**, akkor **el kell adni** a vásárlónak a készleten lévő mennyiséget, majd azt, hogy nem volt teljes körű a kiszolgálás, **egy figyelmeztető üzenettel és a hiányzó mennyiség értékével paraméterezett kivétel dobásával** jelezni kell;
           c. minden egyéb esetben **ki kell szolgálni** a vásárlót, és **csökkenteni kell** a termék **mennyiségét** az árukészletben.
        2. A vásárló **csak a termék nevét** adja meg.

Ilyenkor

a. ha **nincs** a vásárló által megjelölt termék az árukészletben, akkor ezt **egy figyelmeztető üzenettel paraméterezett kivétel dobásával** jelezni kell;
b. **tartós fogyasztási cikk** esetén a termék **mennyiségét 1-gyel csökkenteni** kell;
c. **élelmiszer** esetén **ki kell szolgálni** a vásárlót az adott termék **legkisebb szavatossági idővel** rendelkező tételéből, annak **mennyiségét 1-gyel csökkentve**.

iii. Technikai jellegű, **árukészlet-tisztító** műveletként a **0 mennyiségű** termékek **törlése** az árukészletből.
iv. **Akciós ár beállítása** a pontos jellemzőkkel **megadott termékeknél**.

A programfejlesztéshez jó munkát kíván **a bolt vezetősége**.

2. Az előbbi feladathoz készítsen egy interaktív **grafikus felületet,** amelyben lehet a beérkező árukat is kezelni, és az eladást is le tudjuk bonyolítani. (Ehhez végezhet módosításokat a 2. feladat kódjában, ha szükségét érzi. Ebben az esetben, az eredeti kódot küldje el 2. feladatként(projektként), és e módosított kódot 3. feladat részeként).

3. (ZH feladat volt)

a) Írj egy interfészt **ReakcioKepes** névvel a **kemia** csomagba, ami deklarálja a logikai értéket visszaadó reakciobaLephet nevű metódust! A metódusnak legyen paraméter nélküli, illetve **ReakcioKepes** objektumot váró változata is! A paraméter nélküli változat akkor adjon vissza igaz értéket, ha az objektum egy másik objektummal reakció létrehozására képes állapotban van, a paraméteres változat pedig csak abban az esetben térjen vissza igaz értékkel, ha a kérdéses objektum a paraméterben lévővel reakcióba tud lépni!

a) A **kemia** csomagon belül hozd létre a kémiai elemek tárolására képes **KemiaiElem** osztályt, amely implementálja a **ReakcioKepes** interfészt, de nem példányosítható! Az osztály adattagjainak a tárolt elem vegyjelét, rendszámát, valamint főcsoportját kell tartalmaznia! Az osztályváltozókat lásd el megfelelő láthatósággal, és készíts

konstruktor(oka)t, továbbá legalább egy **getter()** és **setter()** metódust!

b) Hozd létre a **KemiaiElem** osztállyal egy csomagban lévő, és abból származó **NemesGaz** és **NemFem** osztályokat! Egyik osztályból se lehessen további osztályt származtatni, valamint a **NemesGaz** osztály konstruktora dobjon kivételt, ha a példányosítás során a főcsoportot tároló tagváltozó beállításáért felelős paraméter értéke nem egyenlő 8-cal! Két elem akkor tud egymással reakcióba lépni, ha a főcsoportjaik (vegyértékelektronjaik) számának összege 8-cal osztva nem ad maradékot. Mivel a nemesgázok külső elektronhéja mindig teli van, nem képesek reakcióra lépni.

c) A **kemia.vegyuletek** csomagba készítsd el azt a **ReakcioKepes** interfészt megvalósító **Vegyulet** osztályt, ami az objektumpéldány által reprezentált vegyületben található elemeket és a közöttük fennálló mennyiségi arányokat egy Map-ben tárolja! Definiálj egy metódust, ami egy 2 elemű logikai értékekből álló tömbbel tér vissza, amelynek az első eleme akkor igaz, ha a vegyület szerves (található benne szén atom), második eleme pedig akkor igaz, ha a vegyület szénhidrát (szénatomon kívül hidrogén és oxigén található még benne, méghozzá 2:1 arányban).

d) Írj futtatható metódust az osztályhoz, ami a parancssori paraméterekben megkapott elérési útvonalakon található szöveges fájlokat beolvasva **Vegyulet** objektumokat hoz létre! Egy-egy bemeneti fájlban egy vegyület definíciója szerepel a következők szerint: minden sor a vegyületet egy alkotóelemét és a vegyületen belüli mennyiségét tartalmazza tabulátorral tagolva. Az alkotóelemekkel kapcsolatos információk az alábbi sorrendben követik egymást: vegyjel, rendszám, főcsoport. Ha a vegyjel a „He”, „Ne”, „Ar”, „Kr”, „Xe”, „Rn” sztringeket tartalmazó *konstans tömb* elemei között szerepel, úgy **NemesGaz** objektumot hozz létre, (egyébként **NemFem**-et). A fájlkezelés vagy a példányosítás során keletkező hibákat kezeld megfelelően, az egyes fájlok beolvasása végén pedig írasd ki a létrehozott vegyületet a toString() metódus felüldefiniálásának segítségével!

Példa input fájlok:

viz.txt

H	1	1	2
O	8	6	1

szolocukor.txt

C	6	4	6
H	1	1	12
O	8	6	6

Mérlegelés

Írjon programot, amelynek első parancssori argumentuma egy szöveges állomány neve! A szöveges állomány cégek adatait tartalmazza, soronként egyet-egyet, a következő formában:

cégnév:*telephely*:*év*:*éves_árbevétel*

A cégeket a nevük és a telephelyük azonosítja: ha két különböző bejegyzésben megegyezik a cégnév és a telephely, akkor ugyanarról a cégről van szó, ebben az esetben az év biztosan különbözni fog a két sorban.

A program cégenként összegezze az árbevételeket, majd írja a standard kimenetre a cégek nevét és telephelyét, soronként egyet-egyet, az összesített árbevételek értéke szerint csökkenő sorrendben, a példa kimenetben látható formában! Azoknak a cégeknek a nevét, akiknél megegyezik ez az érték, aszerint írja a standard kimenetre, hogy hány évben tettek szert erre a bevételre: előrébb kerüljenek azok, akik (darabszámra) kevesebb évben gyűjtötték össze az összbevételüket! Ha ez a mutató azonos lenne két vagy több cég esetén, akkor őket a neveik lexikografikusan növekvő sorrendje szerint jelenítse meg a kimeneten! Ha pedig a cégnevek is megegyeznének, akkor ezek a cégek tetszőleges sorrendben kerülhetnek kiírásra.

Példa állomány (sample.txt)

```
A:Debrecen:2014:10000
B:Budapest:2013:20000
A:Debrecen:2012:10000
A:Budapest:2013:10000
```

letöltés szöveges állományként

Parancssori argumentumok

```
sample.txt
```

letöltés szöveges állományként

A futtatás eredménye a standard kimeneten

```
B (Budapest): 20000
A (Debrecen): 20000
A (Budapest): 10000
```

letöltés szöveges állományként

Postai küldemények

Írjon programot, amelynek első parancssori argumentuma egy szöveges állomány neve! A szöveges állomány postai csomagok kézbesítési adatait tartalmazza, soronként egyet-egyet, a következő formában:

címzett_neve:*irányítószám*:*település*:*utca*:*házszám*:*csomag_értéke*

Elképzelhető, hogy egy kézbesítési helyre több csomagot is szállítani kell, ekkor értelemszerűen többször is előfordul ugyanaz a kézbesítési cím az állományban (sőt, még akár a kiszállított csomagok értéke is megegyezhet). A csomagok értéke egy mindig nemnegatív egész szám.

A program irányítószámonként összegezze a csomagok értékeit, majd írja a standard kimenetre az irányítószámokat és a hozzájuk tartozó összértéket az irányítószámok növekvő sorrendjében, majd minden irányítószám alatt sorolja fel az ide tartozó címzett(ek) adatait! Ha egynél több adatsor tartozna egy irányítószámhoz, akkor azokat a következő szempontok preferenciasorrendje szerint rendezve jelenítse meg a kimeneten: (1) a települések neve szerint ábécésorrendben, (2) az utcák neve szerint ábécésorrendben, (3) a házszám szerint növekvő sorrendben, (4) a címzett neve szerint ábécésorrendben és végül (5) a csomag értéke szerint csökkenő sorrendben.

A kimenet pontos formátumát lásd a példa kimenetben!

Példa állomány (sample.txt)

```
Gipsz Jakab:4027:Debrecen:Böszörményi út:12:30000
Teszt Elek:4031:Debrecen:Kishegyesi út:31:20000
Menő Jenő:4028:Debrecen:Kassai út:26:10000
Béna Béla:4027:Debrecen:Sinai Miklós utca:10:50000
```

letöltés szöveges állományként

Parancssori argumentumok

```
sample.txt
```

letöltés szöveges állományként

A futtatás eredménye a standard kimeneten

```
4027: 80000 Ft
Debrecen, Böszörményi út 12., Gipsz Jakab, 30000 Ft
Debrecen, Sinai Miklós utca 10., Béna Béla, 50000 Ft
4028: 10000 Ft
Debrecen, Kassai út 26., Menő Jenő, 10000 Ft
4031: 20000 Ft
Debrecen, Kishegyesi út 31., Teszt Elek, 20000 Ft
```

letöltés szöveges állományként

Oroszlánüvöltés (Java)

Írjon programot, amelynek első parancssori argumentuma egy szöveges állomány neve! A szöveges állomány egy vadállatok hangjait rögzítő rendszer napi adatainak blokkjait tartalmazza. A blokkokban soronként egy-egy hangrögzítő állomás két jellemzőjét tároljuk: az állomások nevét és a hangrögzítések pillanataiban beazonosított élőlények (legtöbbször kisebb-nagyobb állatok) azonosítóit, egymástól egy ':' (kettőspont) karakterrel elválasztva:

állomásazonosító:megfigyelések_sztringje

A rendszert úgy konstruálták, hogy az állomások a méréseket mindig azonos időpillanatban, egymással összehangolva végzik, így a hangrögzítések darabszáma egy blokkon belül minden állomásnál azonos. Az adott pillanatban felismert állatot egy betű jelzi a megfigyeléseket leíró sztringben. A különböző állatokat az angol ábécé különböző nagybetűi jelölik, az oroszlánt például az 'O' (nagy O) betű. Azokat a pillanatokat, amikor egy hangrögzítő állomás nem ismerte fel a megfigyelt állatot, a sztringben egy '.' (pont) karakter jelzi. A blokkok végét egy olyan sor jelzi, amelyben kizárólag csak az „END" szó szerepel.

A program minden blokk esetén határozza meg és írja a standard kimenetre külön sorban, hogy a nap folyamán hány olyan pillanat volt, amikor egy oroszlán akkorát üvöltött, hogy azt mindegyik hangrögzítő állomás be tudta azonosítani!

Példa állomány (sample.txt)

```
1:A...BO..A.
1999:.O...O.B.A
2016:.O..OOA...
END
A:OABOO..OOOA.
Cs:O..OOB.OO.AB
Dzs:OB.BO.AOOOA.
Z:O..AO..OOOAB
END
```

letöltés szöveges állományként

Parancssori argumentumok

```
sample.txt
```

letöltés szöveges állományként

A futtatás eredménye a standard kimeneten

```
1
4
```

Kéktúra (Java)

Írjon programot, amely nyilvántartást készít a kéktúrázókról, a hazánk leghosszabb összefüggő, körmentes turistaútvonalát (27 szakaszban összesen 1160 km) bejáró kirándulókról! Az egyes kirándulók adatai az első parancssori argumentumban megadott nevű szöveges állományban vannak leírva a következő formában:

turista_neve:szakasz_egyik_vége-szakasz_másik_vége:szakasz_hossza

A sorban található sztringek mindegyike tetszőleges karaktereket tartalmazhat a pontosvessző és a kötőjel karakterek kivételével, e két karakter kizárólag a többi sztring elválasztására szolgál. A szakaszokat a két végük jellemzi, ezek sorrendje azonban tetszőlegesen felcserélhető, a Bódvaszilas–Putnok szakasz például megegyezik a Putnok–Bódvaszilas szakasszal. A *szakasz_hossza* minden esetben egy egész szám, a szakasz hosszát adja meg kilométerekben számolva. A turistákat a nevük azonosítja. Ha az állományban több olyan sor is van, amely azonos névvel kezdődik, akkor azok a sorok ugyanahhoz a turistához tartoznak.

A program gyűjtse össze azokat a turistákat, és írja a standard kimenetre a nevüket lexikografikusan növekvő sorba rendezve, soronként egyet-egyet, akik a kéktúra útvonalának egy összefüggő részét teljesítették (és nem több különálló, egymással nem kapcsolódó részét)! A kéktúra egyetlen szakaszát önmagában összefüggőnek tekintjük. Több szakaszt akkor tekintünk összefüggőnek, ha egymás után illeszthetők oly módon, hogy két, egymást közvetlenül követő szakasz végpontjai közül az első szakasz valamelyik végpontja megegyezik a rákövetkező szakasz valamelyik végpontjával.

Példa állomány (sample.txt)

```
Peter:Matrahaza-Sirok:22
Nikolett:Nograd-Becske:60
Aron:Sumeg-Sarvar:72
Peter:Sirok-Szarvasko:20
Gabor:Becske-Nograd:60
Nikolett:Huvosvolgy-Piliscsaba:22
Peter:Szarvasko-Putnok:62
Gabor:Nagymaros-Nograd:40
Nikolett:Sarvar-Sumeg:72
```

letöltés szöveges állományként

Parancssori argumentumok

```
sample.txt
```

letöltés szöveges állományként

A futtatás eredménye a standard kimeneten

```
Aron
Gabor
Peter
```

letöltés szöveges állományként

Téli fagyi

Adott egy szöveges állomány, melynek a sorai egy-egy fagylaltrendelés adatait tartalmazzák a következő formában:

<megrendelő>;*<íz>*[,*<íz>*]...

ahol a *megrendelő* és az *ízek* sztringek, az *ízek* akár azonosak is lehetnek egy rendelésen belül. A sztringek egyike sem tartalmaz sem vesszőt, sem pontosvesszőt karaktert, e két karakter csak a sor egyes elemeinek az elválasztására szolgál.

Példa szöveges állomány

```
Renata;eper,citrom,csokolade
Laci;csokolade,csokolade,malna,sargabarack
Johnny Firpo;karamella,tutti frutti,rumos dio,kave,pisztacia
Csilla;citrom,malna
```

letöltés szöveges állományként

Írjon programot, amely első parancssori argumentumként megkapja az előbb említett állománynak a nevét! A program a standard bemenetről a példa bemenetben megadott formában olvasson be egymástól különböző fagylaltízeket az egységáraikkal együtt, majd írja ki a standard kimenetre a példa kimenetnél megadott formában, hogy hány rendelést tud teljesíteni, illetve ezekből mekkora bevételre tehet szert a cukrászda! Egy rendelést akkor tud teljesíteni a cukrászda, ha a rendelésben felsorolt összes íznek ismeri az árát.

Példa bemenet

```
eper;120
csokolade;100
malna;120
citrom;110
```

letöltés szöveges állományként

A példa bemenethez tartozó kimenet

```
2 rendeles
560 Ft
```

letöltés szöveges állományként

Antikvár könyvek

Írjon programot, amely az első parancssori argumentumaként megadott szöveges állományból könyvek adatait olvassa be állományvégjelig! Egy könyv jellemzői: (1) a címe, (2) a szerzői (vigyázat, több is lehet, de az is előfordulhat, hogy egy sincs!) és (3) a kiadási éve. Ezek a következő formában szerepelnek egy-egy feldolgozandó sorban:

[[*<szerző>*;]...*<szerző>*;]*<cím>*;*<kiadási_év>*

ahol a *szerzők* és a *cím* sztringek, a *kiadási_év* pedig egy pozitív egész szám. A sztringek egyike sem tartalmaz pontosvesszőt karaktert, a pontosvesszők karakterek csak a sor egyes elemeinek az elválasztására szolgálnak. Két könyvet akkor tekintünk azonosnak, ha a kiadási évük kivételével minden adatuk (még a szerzőiknek a sorrendje is!) megegyezik egymással.

A programja írja a standard kimenetre minden könyv legrégebbi kiadásának az adatait a példa kimenetben megadott módon, a könyveket kiadási év szerint növekvő sorrendbe rendezve! Ha több olyan könyv is lenne, amelyet ugyanabban az évben adtak ki, akkor ezek a könyvek a szerzők száma szerint csökkenő sorrendben jelenjenek meg a kimeneten! Ha még ez a tulajdonsága is megegyezne kettő vagy több könyvnek, akkor címeik szerint rakja őket ábécérendbe! További szempontokat nem kell a rendezésnél figyelembe vennie.

Példa szöveges állomány

```
Gardonyi Geza;Egri csillagok;2011
Biblia;2010
Vamos Miklos;A csillagok vilaga;2010
Gardonyi Geza;Egri csillagok;2002
Ramez Elmasri;Shamkant B. Navathe;Fundamentals of Database Systems;2010
Agatha Christie;Halal a Niluson;2010
```

letöltés szöveges állományként

A példa szöveges állományhoz tartozó kimenet

```
Gardonyi Geza: Egri csillagok (2002)
Ramez Elmasri, Shamkant B. Navathe: Fundamentals of Database Systems (2010)
Vamos Miklos: A csillagok vilaga (2010)
Agatha Christie: Halal a Niluson (2010)
Biblia (2010)
```

letöltés szöveges állományként