3. beadandó feladat: Osztályszerkezet megvalósítása

Közös követelmények:

- Az alkalmazást objektumorientáltan kell megvalósítani, a leírtaknak megfelelő osztályok létrehozásával, öröklődés és polimorfizmus alkalmazásával. A feldolgozott objektumokat közös adatszerkezetbe kell szervezni.
- Az adatokat szöveges állományból olvassuk be (amelynek nevét a felhasználó adhatja), az eredményeket a konzol felületre írjuk ki. A program készüljön fel arra, ha a fájl nem található (ekkor kérje be újra a fájlnevet), üres, vagy tartalma nem megfelelő (ekkor ugorja át a nem megfelelő sorokat).
- A program használjon alprogramokat az egyes funkciók (beolvasás, kiírás, feldolgozás) megvalósítására. Az alprogramok a kommunikációhoz használjanak paraméterátadást és/vagy visszatérési értéket, ne legyen a programban globális változó.
- A dokumentáció tartalmazza a feladat leírását és elemzését, az osztályszerkezet tervét (az osztályokat és relációikat, UML osztálydiagrammal), az esetleges további programszerkezetet (az osztályon kívüli alprogramok leírását és kapcsolatait), valamint a megvalósítás leírását.

Feladatok:

1. Sugárzott növények

Egy bolygón különböző fajtájú növények élnek, minden növény tápanyagot használ. Ha egy növény tápanyaga elfogy (a mennyisége 0 lesz), a növény elpusztul. A bolygón három fajta sugárzást különböztetünk meg: alfa sugárzás, delta sugárzás, nincs sugárzás. A sugárzásra a különböző fajtájú elő növények eltérő módon reagálnak. A reakció tartalmazza a tápanyag változását, illetve a következő napi sugárzás befolyásolását. A másnapi sugárzás alakulása: ha az alfa sugárzásra beérkezett igények összege legalább hárommal meghaladja a delta sugárzás igényeinek összegét, akkor alfa sugárzás lesz; ha a delta sugárzásra igaz ugyanez, akkor delta sugárzás lesz; ha a két igény közti eltérés háromnál kisebb, akkor nincs sugárzás. Az első nap sugárzás nélküli.
Minden növény jellemzői: az egyedi neve, a rendelkezésre álló tápanyag mennyisége, hogy él-e. A szimulációban részt vevő növények fajtái a következők: puffancs, deltafa, parabokor. A következőkben megadjuk, hogy az egyes fajták miként reagálnak a különböző sugárzásokra. Először a tápanyag változik, és ha a növény ezután él, akkor befolyásolhatja a sugárzást.

- *Puffancs*: Alfa sugárzás hatására a tápanyag mennyisége kettővel nő, sugárzás mentes napon a tápanyag eggyel csökken, delta sugárzás esetén a tápanyag kettővel csökken. Minden esetben úgy befolyásolja a másnapi sugárzást, hogy 10 - tápanyag értékben növeli az alfa sugárzás bekövetkezését. Ez a fajta akkor is elpusztul, ha a tápanyag mennyisége 10 fölé emelkedik.

- *Deltafa*: Alfa sugárzás hatására a tápanyag mennyisége hárommal csökken, sugárzás nélküli napon a tápanyag eggyel csökken, delta sugárzás hatására a tápanyag néggyel nő. Ha a tápanyag mennyisége 5-nél kisebb, akkor 4 értékben növeli a delta sugárzás bekövetkezését, ha 5 és 10 közé esik, akkor 1 értékben növeli a delta sugárzás bekövetkezését, ha 10-nél több, akkor nem befolyásolja a másnapi sugárzást.

- *Parabokor*: Akár alfa, akár delta sugárzás hatására a tápanyag mennyisége eggyel nő. Sugárzás nélküli napon a tápanyag eggyel csökken. A másnapi sugárzást nem befolyásolja.

Készítsünk programot a növények viselkedésének és a sugárzás szimulálására. A program egy szövegfájlból olvassa be a szimuláció adatait. Az első sorban a szimuláció napinak száma található. A következő sorok tartalmazzák a növények adatait szóközökkel elválasztva: a növény nevét, a fajtáját és a kezdetben rendelkezésére álló tápanyag mennyiségét. A fajtát egy karakter azonosít: **a** – puffancs, **d** – deltafa, **p** – parabokor.
A program kérje be a fájl nevét, majd jelenítse meg a túlélők nevét. Ehhez valósítsuk meg a növényeket reprezentáló osztályokat, amelyek egy absztrakt lény osztály leszármazottai. A lényekhez szükséges műveletek a különböző sugárzásoknak, valamint 2 tulajdonság: él-e a lény, illetve a név lekérdezés.

Egy lehetséges bemenet:

```
10
Falánk a 7
Sudár d 5
Köpcös p 4
Nyúlánk d 3
```

2. Lények versenye

Egy többnapos versenyen lények vesznek részt. A versenyt az a lény nyeri, aki életben marad, és a legnagyobb távolságot teszi meg. Kezdetben minden lény valamennyi vízzel rendelkezik, és a megtett távolság 0. A verseny során háromféle nap lehetséges: napos, felhős és esős. Ezekre a különböző fajtájú lények eltérő módon reagálnak vízfogyasztás és haladás szempontjából. Minden lény először a rendelkezésére álló víz mennyiségét változtatja meg, ezután ha tud, mozog. Bármely

lény elpusztul, ha a vize elfogy (0 lesz az érték), ezután értelemszerűen semmilyen tevékenységre sem képes.
Minden lény jellemzői: az egyedi neve, a rendelkezésre álló víz mennyisége, a maximálisan tárolható víz mennyisége, hogy él-e, illetve az eddig megtett távolság. A versenyen részt vevő lények fajtái a következők: homokjáró, szivacs, lépegető.
A következő táblázat tartalmazza az egyes fajták jellemzőit.

Fajta	*Víz változás*			*Távolság*			*Max. víz*
	napos	*felhős*	*esős*	*napos*	*felhős*	*esős*	
homokjáró	-1	0	3	3	1	0	8
szivacs	-4	-1	6	0	1	3	20
lépegető	-2	-1	3	1	2	1	12

Az egyes lények a vízkészlet megváltoztatása során nem léphetik túl a fajtára jellemző maximális értéket, legfeljebb azt érhetik el.
Valósítsuk meg a versenyt megnyerő lényt megadó programot. A program egy szövegfájlból olvassa be a verseny adatait. Az első sorban napok jelei következnek egymás után. Az egyes jelek értelmezése: **n** – napos, **f** – felhős, **e** – esős. A következő sorok tartalmazzák a lények adatait, úgymint a nevét, a fajtáját és a kezdetben rendelkezésére álló víz mennyiségét. A fajtát egy karakter azonosít: **h** – homokjáró, **s** – szivacs, **l** – lépegető.
A program kérje be a fájl nevét, majd jelenítse meg a nyertes nevét. Ehhez valósítsuk meg a lényeket reprezentáló osztályokat, amelyek egy absztrakt lény osztály leszármazottai. A lényekhez szükséges 3 művelet a különböző napoknak, valamint 3 tulajdonság: név, él-e a lény, a név illetve a megtett távolság lekérdezése.

Egy lehetséges bemenet:

```
nffeeennf
Vandor h 4
Seta l 7
Csuszo s 12
Siklo s 10
```

3. Óvoda

Készítsünk programot, amellyel szimulálni tudjuk a következő feladatot. Egy óvodában óvónéninek kell gyerekekkel foglalkozni úgy, hogy probléma nélkül teljes az idő. Négyféle tevékenységre lehet az óvodásokat rávenni: ének, labdázás, rajz, tánc. Az egyes gyerekek eltérően reagálnak az egyes tevékenységekre, és ennek megfelelően változik az elégedettségük. Ha egy gyerek elégedettsége nullára csökken, akkor nyafogni kezd, és ha 3 vagy több gyerek nyafog egyszerre, akkor az óvónéni már nem tudja a rendet fenntartani.

A gyerekeket három csoportra lehet osztani annak alapján, hogy miként reagálnak az egyes tevékenységekre:

- *Eleven*: Labdázás esetén az elégedettsége három lesz, tánc esetén nem változik, ének és rajz esetén eggyel csökken.
- *Zenekedvelő*: Ének esetén az elégedettsége négy lesz, tánc esetén nem változik, labdázás és rajz esetén eggyel csökken.
- *Kényelmes*: Rajz esetén az elégedettsége négy lesz, ének esetén nem változik, tánc esetén eggyel, labdázás esetén kettővel csökken.

A szimuláció adatait egy szövegfájl tartalmazza a következő formátumban. Első sorban a tevékenységek jönnek sorban, számmal azonosítva: **1** – ének, **2** – labdázás, **3** – rajz, **4** – tánc. Ezután soronként adottak az egyes gyerekek adatai. A sor első eleme a gyerek viselkedési módja, ezt követi a gyerek neve, végül a gyerek kezdeti elégedettsége szerepel a sorban.
A program kérje be a fájl nevét, majd adja meg, hogy rendben telt-e a vizsgált időszak. Ehhez valósítsuk meg a gyerekeket reprezentáló osztályokat, amelyek egy absztrakt gyerek osztály leszármazottai.

Egy lehetséges bemenet:

```
1 2 4 3 1 3
Eleven Ede 1
Kenyelmes Kati 4
Zenekedvelo Zsuzsa 2
Eleven Emese 2
Eleven Emil 3
Kenyelmes Karcsi 1
Zenekedvelo Zoli 3
```

4. Kertész

Készítsünk programot a következő kertészeti szimulációhoz. Egy kertben különböző növényeket tartanak, amelyeket gondozni kell. A kertész három fajta anyag közül naponta csak egyet adhat az összes növénynek. Az anyagok fajtái: víz, tápoldat, műtrágya. A növények az egyes anyagokra eltérően reagálnak: sorvadnak; tűrik, ekkor nem nőnek; szeretik, ekkor nőnek. Egy növény kipusztulhat, ha sorvadás során a mérete nullára csökken, illetve túlburjánzásban is kipusztulhat, ha túl naggyá válik. A túlburjánzás határa növényfajtánként eltérő. Előfordul, hogy a kertész nem ad nekik tápanyagot, ekkor garantáltan sorvadnak eggyel.
A konkrét kertben három fajta növényt termesztenek: kaktuszt, virágot és bokrot, ezek jellemzői a következők:

- *Kaktusz*: Víz esetén a mérete eggyel csökken, tápoldat esetén eggyel nő, műtrágya esetén eggyel nő. A túlburjánzás határa 4.

- *Virág*: Víz esetén a mérete hárommal nő, tápoldat és műtrágya esetén eggyel csökken. A túlburjánzás határa 6.

- *Bokor*: Víz és műtrágya esetén a mérete eggyel nő, tápoldat esetén eggyel csökken. A túlburjánzás határa 10.

A szimuláció adatait egy szövegfájl tartalmazza a következő formátumban. Az első sorban szerepel a napok száma, majd az egyes napokon adott anyag fajtái: 1 – víz, 2 – tápoldat, 3 – műtrágya, illetve 0 – nincs tápanyag. Ezután soronként adottak az egyes növények adatai. A sor első eleme a növény fajtája, ezt követi a növény azonosítója, végül a növény kezdeti mérete szerepel a sorban.
A program kérje be a fájl nevét, majd írja ki az életben maradt növények azonosítóját. Ehhez valósítsuk meg a növényeket reprezentáló osztályokat, amelyek egy absztrakt növény osztály leszármazottai.

Egy lehetséges bemenet:

```
5 1 2 0 1 3 1
Virag V102 3
Bokor B223 4
Kaktusz K15 2
Kaktusz K18 1
```

5. Állatkert

Készítsünk programot, amellyel szimulálni tudjuk a következő feladatot. Egy állatkertben különböző állatokat tartanak, amelyeket etetniük kell. A gondozó három fajta étel közül naponta csak egyet adhat az összes állatnak. Az ételek fajtái: hús, zöldség, gyümölcs. Az állatok az egyes ételfajtákra eltérően reagálnak: nem eszik meg, ekkor fogynak; elfogyasztják, de nem szeretik, ekkor a súlyuk nem változik; szeretik, ekkor a súlyuk gyarapszik. Egy állat éhen pusztulhat, ha a súly nullára csökken, illetve elhízásban is kimúlhat, ha túlsúlyossá válik. A túlsúlyosság határa állatfajtánként eltérő.

A konkrét állatkertben négy fajta állatot tartanak: nyulat, farkast, sünt és medvét, ezek jellemzői a következők:

- *Nyúl*: Hús esetén a súlya eggyel csökken, zöldség esetén kettővel nő, gyümölcs esetén eggyel nő. A túlsúlyosság határa 4.

- *Farkas*: Hús esetén a súlya hárommal nő, zöldség és gyümölcs esetén eggyel csökken. A túlsúlyosság határa 6.

- *Sün*: Hús esetén a súlya eggyel csökken, zöldség esetén nem változik, gyümölcs esetén kettővel nő. A túlsúlyosság határa 5.

- *Medve*: Hús és gyümölcs esetén a súlya eggyel nő, zöldség esetén eggyel csökken. A túlsúlyosság határa 10.

A szimuláció adatait egy szövegfájl tartalmazza a következő formátumban. Az első sorban az etetés rendje szerepel, először a napok száma, majd az egyes napokon adott étel fajtái: **1** – hús, **2** – zöldség, **3** – gyümölcs. Ezután soronként adottak az egyes állatok adatai. A sor első eleme az állat fajtája, ezt követi az állat neve, végül az állat kezdeti súlya szerepel a sorban.
A program kérje be a fájl nevét, majd írja ki az életben maradt állatok nevét. Ehhez valósítsuk meg az állatokat reprezentáló osztályokat, amelyek egy absztrakt állat osztály leszármazottai.

Egy lehetséges bemenet:

```
5 1 2 1 3 1
Farkas Akela 3
Medve Balu 4
Nyul Tapsi 2
Medve Brumi 6
Sun Durum 4
```

6. Ruhaipar

Készítsünk programot, amellyel szimulálni tudjuk ruhaipari bedolgozók (továbbiakban gazdálkodók) tevékenységét. A gazdálkodók saját otthonukban kötőgépeket üzemeltetnek, így jutnak bevételhez. Egy logisztikai központ gondoskodik az alapanyag kiszállításáról és a kész termék elszállításáról. A központ egy gép egy napi termelése után gazdálkodóknál üzemelő összes gép számának és az aktív gazdálkodók számának arányában fizet hozamot az alábbiak szerint:

- ha a gépek száma nem haladja meg a gazdálkodók számát, akkor 10 egységet;
- ha a gépek száma nagyobb vagy egyenlő a gazdálkodók számának duplájával, akkor nincs hozam (0 egység);
- egyébként $10 \cdot \frac{\text{gazdálkodók száma}}{\text{gépek száma}}$ egységet.

Kezdetben minden gazdálkodónak egy gépe van, és a pénze 0 egység. Egy gép napi üzemeltetése 6 egység költséget igényel. Egy gazdálkodó napi jövedelme: a gépei által megtermelt hozamból le kell vonni a gépei üzemeltetési költségét. (A napi jövedelem lehet veszteséges is.) Ha a gazdálkodónak van elég pénze és úgy látja jónak, akkor 18 egységért újabb gépet vásárolhat, de 3 gépnél nem lehet több gépe. Lehetősége van gépet eladni a vételáron, ezzel eggyel csökkenti gépeinek számát. Egy gazdálkodó tönkremegy, ha negatív lesz a pénze.
Minden nap végén az addig tönkre nem ment gazdálkodók az aznapi jövedelmükkel megváltoztatják a pénzüket. Miután ez minden aktív gazdálkodó megtette, eldöntik

miként folytatják tevékenységüket. Három dolgot tehetnek: nem változtatnak semmit, egy új gépet vehetnek, vagy eladhatnak egy gépet. Ha a döntésük végrehajtása után vagyonuk negatív, azaz tönkrementek, akkor vissza kell adniuk a gépeiket, de a gépek árát nem kapják vissza. Ha nem mentek tönkre, akkor a következő nap már a döntésüknek megfelelően folyik a termelés.
Háromféle gazdálkodót különböztetünk meg:

- *Egyszerű*: semmit nem változtat, végig egy gépet használ.
- *Mohó*: ha lehetősége van rá (több pénze van, mint egy gép ára és háromnál kevesebb gépe van) új gépet vesz.
- *Gazdaságos*: gépet vesz, ha háromnál kevesebb gépe van, több pénze van, mint a gép ára és egy gép hozama meghaladja az üzemeltetési költséget; elad gépet, ha egynél több gépe van és a pénze negatív vagy az erőforrás jövedelme nem haladja meg az üzemeltetési költséget.

A szimuláció adatait a program egy szöveges fájlból olvassa be. A fájl első sora megadja a napok számát, majd a gazdálkodók adatai következnek soronként. Egy karakter azonosítója a gazdálkodó fajtáját, amit szóköz után a gazdálkodó neve követ. Az azonosítók: **`E`** – egyszerű, **`M`** – mohó, **`G`** – gazdaságos.
A program kérje be a fájl nevét, majd a szimuláció végén írja ki a tönkre nem ment gazdák közül a leggazdagabbat. Ehhez valósítsuk meg a gazdálkodókat reprezentáló osztályokat, amelyek egy absztrakt gazdálkodó osztály leszármazottai.

Egy lehetséges bemenet:

```
20
G Szamito
M Torekvo
E szereny
G okos
```

7. Ipari robotok

Készítsünk programot, amellyel szimulálni tudjuk ipari robotok működését. A robotok egy raktárban dolgoznak, és a feladatuk áruk helyrevitele. A robotok árammal működnek, minden nap elején adott mennyiséggel tölthetik fel az akkumulátoraikat. Ennek során az akkumulátor töltési szintje nő, de a maximális töltési kapacitást nem haladhatja meg. A feltöltés után kezdik meg az áruk szállítását, ennek során az akkumulátor töltése csökken. A robotok sorban veszik fel az árukat. Minden robot annyi árut vesz fel, amennyi a tényleges szállítási kapacitásának és a még szállításra váró áruk mennyiségének a minimuma. Az egyes robotok tényleges szállítási kapacitása nem haladhatja meg a maximális értéket, de a lehető legnagyobb olyan érték, hogy a szállítás után az akkumulátor töltési szintje nem lehet negatív.

Egy robot egy nap csak egyszer vesz fel árut. Ha egy adott nap a robotok nem tudják az összes beérkezett árut helyre vinni, akkor a maradék a következő napra marad.

A munkában háromféle robot vesz részt:

- *Mac*: Maximális töltési szintje 10, maximális szállítási kapacitása 10 egység, egy egység szállítása eggyel csökkenti a töltési szintet.
- *Eco*: Maximális töltési szintje 4, maximális szállítási kapacitása 6, két egység szállítása eggyel csökkenti a töltési szintet. (Páratlan egység szállításakor a pár nélkül maradó egység is eggyel csökkent, azaz 5 egység esetén 3 lenne a fogyasztás.)
- *Pro*: Maximális töltési szintje 12, maximális szállítási kapacitása 8. Egy egység szállítása 5 egységig eggyel csökkenti a töltési szintet, az 5 feletti egységek kettővel (Pl. 7 egység esetén 5+4=9 a fogyasztás.)

A szimuláció adatait egy szövegfájl tartalmazza a következő formátumban. Az első sorban szerepelnek az időszak napjainak adatai. Először a napok száma, majd minden napról először a töltési mennyiség, aztán a beérkező áruk egysége. Ezután soronként adottak az egyes robotok adatai. A sor első eleme a robot fajtája, ezt követi a robot neve, végül a robot akkumulátorának kezdeti töltöttsége szerepel a sorban.

A program kérje be a fájl nevét, majd a szimuláció végén írja ki, az egyes robotok mennyi árut szállítottak a vizsgált időszak alatt. Ehhez valósítsuk meg a robotokat reprezentáló osztályokat, amelyek egy absztrakt robot osztály leszármazottai.

Egy lehetséges bemenet:

```
4 3 20 2 32 5 20 3 16
Mac MM10 4
Eco E5 3
Pro PX 10
Eco E12 4
```

8. Kristálytermelés

Készítsünk programot, amellyel szimulálni tudjuk a következő feladatot. Egy gép kristályokat állít elő egy nap folyamán. A nap elején több nyíláson keresztül adhatnak be kristályokat, és ezek felhasználásával állít elő újabb kristályokat, amelyeket a nap végén szétoszt az egyes nyílások között. Minden nyílásnál egy-egy résztvevő áll, akik nem ismerik a gép működését. Minden résztvevő valamilyen stratégia alapján dönti el, hogy mennyi kristályt helyez a nyílásba a nap elején. A nap végén kiveszi a nyílásból az összes kristályt, ezekkel nő a nála lévő kristályok száma. Kezdetben minden résztvevőnek 10 kristálya van. A stratégia alapján három fajta résztvevőt különböztethetünk meg:

- *Mohó*: A nála lévő összes kristályt a nyílásba helyezi.

- *Óvatos*: A nála lévő kristályok számának a felét teszi a nyílásba.
- *Egyszerű*: Előre adott maximális érték figyelembe vételével dönt a kristályok számáról. A maximális értéket helyezi el, ha van legalább annyi kristálya, ellenkező esetben az összes kristályát felhasználja.

A gép a következő elven működik. Az összes nyílásból összegyűjti a kristályokat, és meghatározza az egy nyílásra átlagot (a). Ezután minden nyílásba termel kristályokat. Ha a nyílásba k kristályt tettek, akkor ott $2 \cdot a - (k - a)$ kristály keletkezik, ha $k \geq a$; illetve $2 \cdot k - (a - k)$ kristály képződik, ha $k < a$. Ha a kristályok száma negatív lenne, akkor a nyílásba nem kerül kristály.
A szimuláció adatait egy szövegfájl tartalmazza a következő formátumban. Az első sorban szerepel a napok száma. Ezután soronként adottak az egyes résztvevők adatait. A sor első eleme a résztvevő fajtája, ezt követi a résztvevő neve, amit egyszerű fajta esetén a maximálisan felhasznált kristályok száma követ. A fájl utolsó sora a napok számát adja meg.
A program kérje be a fájl nevét, majd jelenítse meg, hogy az adott számú nap elteltével kinek van a legtöbb kristálya (név, kristályszám). Ehhez valósítsuk meg a résztvevőket reprezentáló osztályokat, amelyek egy absztrakt résztvevő osztály leszármazottai.

Egy lehetséges bemenet:

```
10
Moho Misi
Ovatos Otto
Egyszeru Elek 10
Egyszeru Ede 6
Moho Matyas
```

9. Autóverseny

Készítsünk programot, amellyel szimulálni tudunk egy autóversenyt. A versenyzők ugyanolyan benzin üzemelésű autóval küzdenek. A versenyautók tankja kezdetben tele van (100), de ez természetesen csökken a verseny során, 5 egységgel körönként. A verseny során próbálkozhatnak előzni is, egy próbálkozás további 4 egység benzincsökkenés. Ha elég kevés a benzin, akkor kiállnak tankolni (ismét 100 lesz a benzinszint), ekkor azonban 5 hellyel hátrébb kerülnek.
A versenyzőket négy kategóriába soroljuk a vezetési stílus alapján.

- *Agresszív*: Minden második körben megpróbál előzni, de csak minden harmadik előzése sikeres. Ha a benzinszint 10 alá esik, akkor kihajt tankolni.
- *Lendületes*: Minden ötödik körben megpróbál előzni, és minden második sikeres. Ha a benzinszint 20 alá esik, akkor kihajt tankolni.
- *Veszélyes*: Minden negyedik körben megpróbál előzni, és csak minden negyedik előzése sikeres. Csak akkor megy tankolni, ha a benzinszint 5 alá esik.

- *Óvatos*: Egyáltalán nem előz. Ha a benzinszint 20 alá esik, akkor kihajt tankolni.

A verseny során természetesen történhet baleset is. Egy előzés során 4% az esélye annak, hogy kiesik a támadó, 4%, hogy kiesnek mindketten, illetve 2%, hogy tömegkarambol lesz, és még az előttük, illetve mögöttük lévők is kiesnek (így összesen négyen). Veszélyes versenyző esetén ezek a számok duplázódnak.
A verseny adatait egy szövegfájl tartalmazza a következő formátumban. Az első sorban szerepel a körök száma, illetve a versenyzők száma. Ezután soronként adottak a pilóták adatai a rajtsorrend szerint. A sor első eleme a név, majd ezt követi a kategória (1: agresszív, 2: lendületes, 3: veszélyes, 4: óvatos).
A program kérje be a fájl nevét, majd írja ki minden körre a sorrendet, illetve a baleset miatt kiesők neveit. Végül adja meg a verseny első három helyezettjét.

Egy lehetséges bemenet:

```
72 8
Farina 1
Fangio 2
Fagioli 1
Rosier 4
Ascari 3
Parsons 2
Holland 4
Bira 1
```

10. Programozók

Készítsünk programot, amellyel szimulálni tudjuk egy informatikai cég működését. Az informatikai cég programokat készít, amelyek részfeladatokból állnak. A részfeladatok bonyolultság szerint 3 kategóriába sorolhatóak (1: könnyű, 2: közepes, 3: nehéz), illetve rendelkeznek egy tervezett óraszámmal (mennyi idő alatt készülnek el). A programozókat 3 kategóriába soroljuk:

- *Junior*: A könnyű feladatokkal az előírt idő alatt birkózik meg, közepes feladatokkal dupla annyi idő alatt, nehéz feladatot nem kaphat. Egyszerre csak egy feladaton tud dolgozni.
- *Senior*: A könnyű feladatokkal az előírt idő fele alatt birkózik meg, a közepesekkel a megadott idő alatt, a nehezekkel az idő háromszorosa alatt. Egyszerre csak két feladaton tud dolgozni.
- *Architect*: A könnyű feladatokkal az előírt idő duplája alatt birkózik meg (mivel túlbonyolítja), a közepes és nehéz feladatokkal időben. Egyszerre akár három feladaton is tud dolgozni.

A szimuláció során szeretnénk megállapítani, hogy az előre eltervezett beosztás szerint mennyi ideig fog tartani a program megvalósítása, illetve melyik programozóra összesítve hány napot tölt a fejlesztéssel.

A szimuláció adatait egy szöveges fájl tartalmazza a következő formátumban. Az első sorban a programozók száma, majd a részfeladatok száma szerepel. Ezután következnek soronként a programozók nevei és típusai, majd soronként a részfeladatok típusa, tervezett ideje, illetve melyik (hányadik) programozónak szánják.
A program kérje be a fájl nevét, majd írja ki minden órára, hogy a programozók aktuálisan melyik (hányadik) részfeladat(ok)on dolgoznak (előfordulhat, hogy egy programozó már végzett, ekkor nem kell kiírni), végezetül minden programozóra a teljes munkaidőt. Ehhez valósítsuk meg a programozókat reprezentáló osztályokat, amelyek egy absztrakt programozó osztály leszármazottai.

Egy lehetséges bemenet:

```
3 8
Sára junior
Rezső senior
Ödön architect
1 5 1
2 10 3
2 2 2
3 8 3
1 2 2
1 4 1
3 7 2
2 6 3
```

11. Autóvezetők költségei

Készítsünk programot, amellyel szimulálni tudjuk autóvezetők költségeinek alakulását. Egy gépkocsi tulajdonosának fizetnie kell a jármű felelősségbiztosítását, az üzemanyag árát és a karbantartási költségét. A gépkocsik háromfélék lehetnek: benzines, gázolajos, villanyos, és egy hónapra adottak az adataik.

- Benzines gépjármű esetén:
  - a jármű rendelkezik lökettérfogattal, hengerszámmal és maximális fordulatszámmal,
  - a jármű kötelező biztosítási díja = *1500 + lökettérfogat / 2 + hengerszám / 10 egészrésze*,
  - a jármű egység-fogyasztása = *maximális fordulatszám + hengerszám * 10 + lökettérfogat*,
  - a jármű karbantartási költsége = *maximális fordulatszám + hengerszám * 10*.

- Diesel kocsi esetén:
  - a jármű rendelkezik lökettérfogattal és hengerszámmal,
  - a jármű kötelező biztosítási díja = *1700 + hengerszám / 10 egészrésze*,
  - a jármű egység-fogyasztása = *lökettérfogat*,

- a jármű karbantartási költsége = *250 + hengerszám * 7*,

- Villanyos jármű esetén:
  - a jármű rendelkezik akkumulátorkapacitással és teljesítménnyel,
  - a jármű kötelező biztosítási díja = *1000 + akkumulátorkapacitás*,
  - a jármű egység-fogyasztása = teljesítmény * 2,
  - a jármű karbantartási költsége = *500*.

Ezen adatok alapján a gépjármű teljes költsége az adott hónapokra kiszámolható:

$$\text{költség} = \text{hónapok száma} \cdot (\text{karbantartási költség} + \frac{kötelező\ biztosítás\ díja}{12}) + \frac{egység - fogyasztás \cdot megtett\ távolság}{100}$$

A szimuláció adatait egy szövegfájl tartalmazza a következő formátumban. Az első sorban szerepel a gépjárművek száma, valamint a hónapok száma. A következő sorokban az autók adatai következnek. A sor első eleme a sorszám, majd következik a vezető neve (két szó), a gyártmány (egy szó), a típus (egy szó), a jármű típusának első karaktere (b: benzines, g: gázolajos, v: villanyos), majd típusonként a megfelelő paraméterek a fenti sorrendben. Ezt követően minden hónapra a futási adatok minden autóra (ha az autó nem futott, akkor 0), azaz sorszám, valamint megtett távolság.
A program kérje be a fájl nevét, majd jelenítse meg hónaponként, valamint a teljes időtartamra vonatkozóan is) kinek volt a legtöbb a költsége (és mennyi). Ehhez valósítsuk meg a gépjárműveket reprezentáló osztályokat, amelyek egy absztrakt gépjármű osztály leszármazottai.

Egy lehetséges bemenet:

```
4 3
1 Dörmögő Dömötör Toyota Corolla b 3500 8 6500
2 Pöttöm Panna FIAT Panda v 4500 1750
3 Mek Elek Ford Focus g 1900 5
4 Tojás Tóbiás Citroen C5 b 1900 4 6300
1 125
2 38
3 1500
4 150
1 462
2 12
3 806
4 96
1 330
2 97
3 0
4 230
```