1. feladatcsoport: Objektumorientált konzol alkalmazás

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, az eredményeket a konzol felületre írjuk ki. A program készüljön fel arra, ha a fájl nem található, vagy tartalma nem megfelelő. A hibakezeléshez használjon kivételkezelést.
- A dokumentációnak tartalmaznia kell a feladat elemzését, a felhasználói felület leírását, a specifikációt (fájlformátumokat), a program osztályainak rövid leírását (UML osztálydiagrammal), valamint a végállapot teszteseteit.

Feladatok:

1. Alakzatok kezelése

Egy szöveges állomány háromféle síkidom adatait (négyzetnél az alapot, téglalapnál az alapot és az oldalt, rombusznál az alapot és az alapok által bezárt szöget) tartalmazza.
Minden sorban egy-egy síkidom adatai találhatók. A sor első számjegye a síkidom fajtájára utal (0, ha négyzet; 1, ha téglalap; 2, ha rombusz), ezután szóközökkel elválasztva a síkidom fajtájától függően egy vagy két valós szám. Definiáljuk külön-külön az egyes síkidomok-típusok osztályait úgy, hogy a négyzetét az alább megadott absztrakt osztályból származtatjuk, a másik kettőt pedig a négyzet osztályából.

```
abstract class Alakzat{
   public abstract String getNev();
   public final double getTerulet() {
      return getAlap() * getMagassag();
   }
   public final double getKerulet() {
      return 2 * (getAlap() + getOldal());
   }
   public abstract double getAlap();
   public double getOldal() { return getAlap(); }
   public double getMagassag() { return getOldal(); }
}
```

Készítsünk olyan programot, amely a szöveges állomány alapján létrehozza a megfelelő síkidom-objektumokat, és azokat területük szerint is, és kerületük szerint is növekedő sorrendben tárolja két lista segítségével. Ehhez valósítsunk meg alkalmas rendező algoritmust. A felhasználónak legyen lehetősége megadni a fájlnevet, valamint listázni a kívánt sorrendben a tartalmat (név és terület, vagy név és kerület).

Példa: **1. bemenet**, **2. bemenet**

2. Helyiség adatok kezelése

Egy szöveges állomány egy lakás helységeinek adatait (lakószobáknál az oldalhosszakat, az ajtók számát és az ablakok számát, terasz esetén az alapterületet, egyéb helységek esetén az alapterületet, az ajtók számát és az ablakok számát) tartalmazza. Minden sorban egy-egy helység adatai találhatók. A sor első számjegye a helység fajtájára utal (1, ha szoba; 2, ha terasz, 3 ha egyéb helység), ezután szóközökkel elválasztva a helység fajtájától függően egy vagy két valós szám, majd (a terasz kivételével) az ajtók és ablakok számát megadó két egész. Definiáljuk külön-külön az egyes helység-típusok osztályait, amelyek az alábbi interfész leszármazottai.

```
interface Helyiseg{
   String getNev();
   double getTerulet();
   int getAblakokSzama();
   int getAjtokSzama();
}
```

Készítsünk olyan programot, amely a szöveges állomány alapján létrehozza a megfelelő helyiség-objektumokat. A felhasználónak legyen lehetősége megadni a fájlnevet, valamint lekérdezni a következő adatokat:

- teljes terület, teljes belterület (terasz nélkül),
- összes ablak száma, összes ajtó száma,
- adott helyiség összes adatainak listája.

Példa: **1. bemenet**, **2. bemenet**

3. 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 az induló lények száma szerepel. A következő sorok tartalmazzák a lények adatait szóközökkel elválasztva: a lény nevét, a fajtáját és a kezdetben rendelkezésére álló víz mennyiségét. A fajtát egy karakter azonosít: **`h`** - homokjáró, **`s`** - szivacs, **`l`** - lépegető.
A lényeket leíró részt követő sorban a verseny napjai szerepelnek egy karaktersorozatban. Az egyes jelek értelmezése: **`n`** - napos, **`f`** - felhős, **`e`** - esős.
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.

Példa: **1. bemenet**, **2. bemenet**

4. 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 növények száma szerepel. 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 növényeket leíró részt követő sorban a szimuláció napjainak száma adott egész számként.
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.

Példa: **1. bemenet**, **2. bemenet**

5. Autóvezetők költségei

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*.

Készítünk programot, amely kiszámítja az egyes autók költségeit egy megadott időtávra.

A program bemenete a következő: az első sorban 2 szám található: n db autót figyelünk meg m hónapig, ezután n sorban a gépkocsik adatai következnek, egy gépkocsihoz tartozó sor tartalma szóközökkel elválasztva:

- autó gyártmánya és típusa,
- a megtett út mennyisége a teljes időtartamra,
- a gépkocsi típusa (0: benzines, 1: dieseles, 2: villanyos), majd
- típustól függően 2 vagy 3 szám a fent leírt adatok megadására.

A bemenet feldolgozása során a program hozza létre a megfelelő gépkocsi objektumokat, amelyek egy absztrakt osztály leszármazottai, majd számolja ki, hogy a megadott m hónapban mennyit költenek az általuk birtokolt gépkocsikra. (Az egységnyi fogyasztás 100 km megtett útra vonatkozik.) Ezt az összeget írja ki a program minden autótípus mellé.

Példa: **1. bemenet**, **2. bemenet**

6. Teremfoglalás kezelő

Készítsünk programot, amely elősegíti termek foglalását az oktatók számára. A rendszer háromféle termet tart nyilván, amelyek a következő adatokkal rendelkeznek:

- *Előadó terem* (0-s kód): terem neve, az ülőhelyek száma, van-e beépített projektor (0: nincs, 1: van)
- *Szeminárium terem* (1-es kód): terem neve, az ülőhelyek száma, valamint a tábla típusa (0: krétás, 1: filces).
- *Gépterem* (2-es kód): terem neve, az ülőhelyek száma, számítógépek száma.

Ezen felül minden teremre kiszámolható a kapacitása a következőeknek megfelelően:

- *Előadó terem*: ha van projektor, és azt igénybe veszik, akkor az ülőhelyek számának 115%-a, mivel ekkor úgyse fognak bejönni a hallgatók órára, különben a ülőhelyek száma.
- *Szeminárium terem*: ha filces a tábla, az ülőhelyek száma, ha krétás, akkor az ülőhelyek száma -6 fő, mert az első sorba senki sem ül a szálló krétapor miatt.
- *Gépterem*: a számítógépek számának 90%-a (mert a többi biztos rossz) +10 fő (akik úgyis laptoppal járnak), de maximálisan az ülőhelyek száma.

A program az adatokat szöveges fájlból olvassa be, amelynek minden sora egy terem adatait tartalmazza a megadott sorrendben. Legyen lehetőség a felhasználónak megadni, mekkora kapacitású termet akar foglalni, géptermet szeretne-e, legyen-e projektor (gépteremben mindig van, szemináriumteremben nincs), illetve filces táblát szeretne-e

(gépteremben mindig filces a tábla, előadóban sosem), és a program ekkor listázza ki azokat a termeket, amelyek az igénynek eleget tesznek, vagy írja ki, hogy nincs a keresésnek megfelelő terem.

Példa: **1. bemenet**, **2. bemenet**

7. Testek tömegszámítása

Készítsünk programot, amely különböző típusú testek tömegét tudjuk kiszámítani. A program háromféle test adatait tudja kezelni:

- Hengeres márványoszlop (0): sűrűség, magasság, átmérő;
- Tölgyfa egészben, gyökér nélkül (1): sűrűség, törzs hossza, törzs sugara, fa kora, zsugorodási tényező;
- Raktári polc (2): sűrűség, teljes magasság, hosszúság, mélység, polcok száma, polcok vastagsága.

A program az adatokat szöveged fájlból olvassa be, ahol az első sor a testek száma, majd ezt követik soronként egy test kódja (0,1, vagy 2) és az adatai a megadott sorrendben. Amennyiben egy sor szerkezete nem megfelelő, akkor ezt a program jelezze, de a további sorokat dolgozza fel. A testeket osztályok segítségével valósítsuk meg a következő absztrakt osztályból származtatva:

```
abstract class Test{
   public abstract String getNev();
   public abstract double getTerfogat();
   public abstract double getSuruseg();
   public final double getTomeg() {
      return getTerfogat() * getSuruseg();
   }
}
```

A tömeg kiszámításához használjuk a következő képleteket:

- tömeg: $m = V \cdot \rho$ (V: térfogat, ρ: sűrűség)
- henger térfogata: $V = \pi \cdot r^2 \cdot h$ (r: henger sugara, h: henger magassága)
- fa térfogata: $V = V_1$, ahol $V_n = \begin{cases} \pi \cdot r^2 \cdot h \cdot d^n + \frac{1}{d+n} \cdot V_{n+1}, ha\ n < k \\ \pi \cdot r^2 \cdot h \cdot d^n, ha\ n = k \end{cases}$ (r: törzs sugara, h: törzs magassága, k: fa kora, d: zsugorodási tényező)
- polc térfogata: $V = l \cdot d \cdot t \cdot n$ (l: hossz, d: mélység, t: vastagság, n: polcok száma)

Példa: **1. bemenet**, **2. bemenet**

8. Matematikai számítások

Készítsünk programot, amely segítségével matematikai műveleteket tudunk elvégezni és azok eredményét kiírni a képernyőre. A program négyféle műveletet támogat, amelyek kódjai és operandusai egy szöveges fájlban találhatóak:

- Faktoriális számítás (0-ás kód), 1 operandusú
- Legkisebb közös többszörös számítása (1-es kód), 2 operandusú
- Fibonacci sorozat n-edik tagja (2-es kód), 1 operandusú
- Vektorok skaláris szorzata (3-as kód), 2 operandusú

Vektorok esetén a vektor hossza nem adott előre, így azt a sorban található számok számából kell kikövetkeztetni.

A program olvassa be a fájlt, majd írja ki a képernyőre a műveletek nevét, operandusait, valamint eredményét soronként. Amennyiben egy sorban nincs elég operandus, vagy hibás a sor szerkezete, úgy a hibát jelezze, de a többi műveletet a fájlból végezze el. A megvalósításnál a műveleteket készítsük el osztályok formájában, amelyek az alábbi interfészt valósítják meg:

```
interface Muvelet{
   String getNev();
   String[] getOperandusok();
   double Kiszamol();
}
```

Példa: **1. bemenet**, **2. bemenet**