Programozási Technológia – 1. beadandó (1. feladat)
Abdurasitov Alekszandr
A49MZV

Lények versenye

Feladatleírás

Egy többnapos versenyen lények vesznek részt. ***Ki nyeri a versenyt, azaz melyik lény teszi meg a legnagyobb távolságot úgy, hogy közben életben marad?*** Kezdetben minden lény valamennyi vízzel rendelkezik, és a megtett távolsága 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 (string), a rendelkezésre álló víz mennyisége (egész), a maximálisan tárolható víz mennyisége (egész), hogy él-e (logikai), illetve az eddig megtett távolság (egész). A versenyen részt vevő lények fajtái a következők: `homokjáró`, `szivacs`, `lépegető`.
A következő táblázat tartalmazza az egyes fajták jellemzőit.

fajta	víz változtatás			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.

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! (Feltehetjük, hogy a fájl formátuma helyes.) Egy lehetséges bemenet:

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

(Javaslat: Hozza létre az absztrakt lény osztályt, amelyből származtatja a homokjáró, szivacs és lépegető osztályokat. A konstruktor paramétere legyen a név és a kezdeti víz mennyisége. Vezessen be három műveletet a napoknak (napos, felhős, esős) amelyek a vízfogyasztást az élet vizsgálatával együtt, illetve mozgatják a lényt. A végeredmény meghatározásához kell még 3 művelet: él-e a lény, a név illetve a megtett távolság lekérdezése.)

Programozási Technológia – 1. beadandó (1. feladat)
Abdurasitov Alekszandr
A49MZV

Feladat elemzése

A feladatban megjelölt **lényeket** egy-egy különálló objektumokként tudjuk kezelni, melyeket származtatni tudjuk egyenesen a **lény absztrakt** osztályából. Minden egyes lény máshogy reagál az időjárásra, ez alapján minden egyes lénynek kezelnie kell a különböző napokat. Ezenfelül minden egyes lénynek van egy saját tulajdonsága, hogy mekkora mennyiségű vizet képes eltárolni.

Ezt az alábbi táblázatban tudjuk szemléltetni:

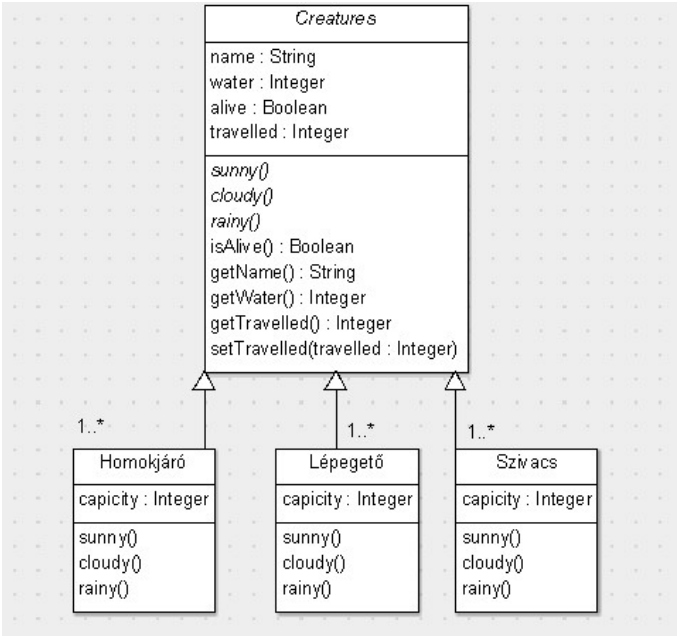
fajta	víz változtatás			távolság			max.víz
	napos	felhős	esős	napos	felhős	esős	
homokjáró	-1	0	3	3	1	0	8
szivacs	-4	-1	6	0	1	3	20
lépegető	-2	-1	3	1	2	1	12

Megvalósítási terv

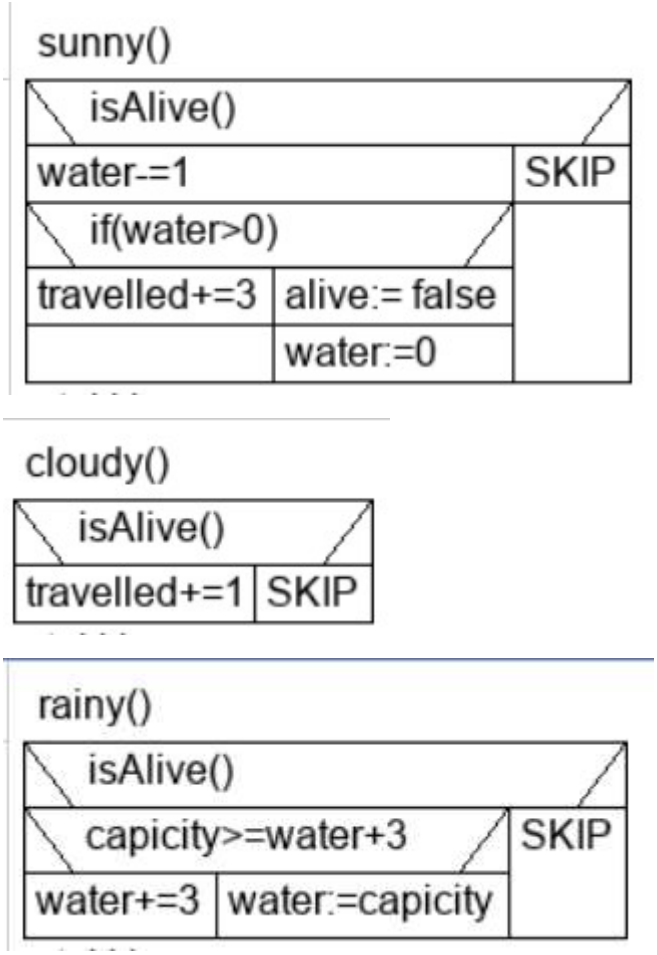
A lények reprezentálásához szükség lesz egy **Creatures osztályra,** melyből tudjuk **származtatni** a 3 lényfajtát. Ahhoz, hogy a lények létezhessenek, ahhoz származtatnunk kell őket a szülőosztályukból a **Creatures** osztályból.

A **Creatures osztály** fogja majd létrehozni az absztrakt metódusokat, melyek a különböző időjárásra adott reakciót fogja definiálni. Ezeket az absztrakt metódusokat fogják a különböző lények különféleképpen definiálni. Ezen kívül a **Creatures** osztályban lesz megtalálható az összes **getter** és **setter**, ami szükséges lehet a főprogramban.

Ezek alapján az osztálydiagramm az alábbiak szerint fog kinézni:

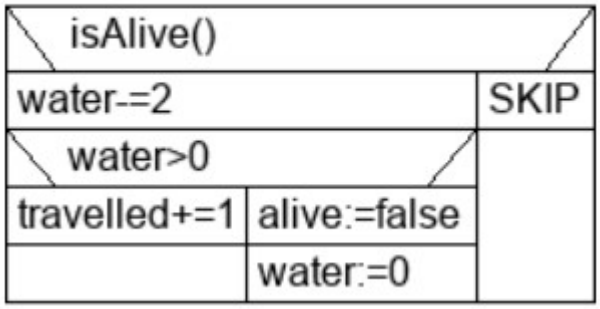


A homokjáró osztály metódusai

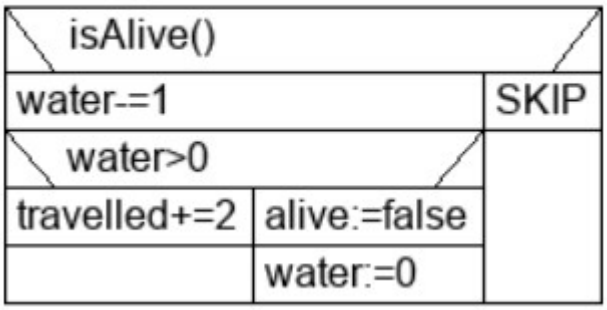


A lépegető osztály metódusai

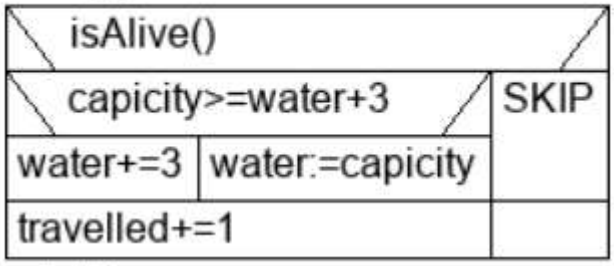
sunny()



cloudy()

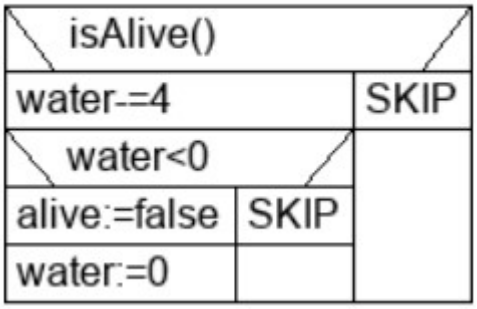


rainy()

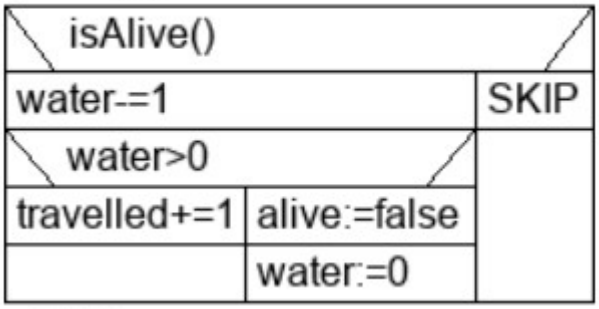


A szivacs osztály metódusai

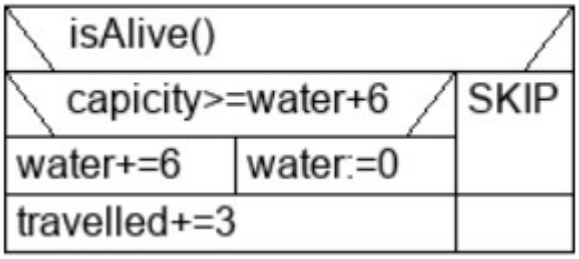
sunny()



cloudy()



rainy()



Főprogram függvényei

Főprogramban **több statikus főfüggvény van**, melyek kezelni tudják a beérkező adatokat, ezekből képesek szimulációt létrehozni, illetve a szimuláció eredményét kiértékelni és azt kiíratni a felhasználó képernyőjére.

A főprogram 7 legfontosabb statikus függvényei:

1. menu(): **Int**
2. checkInputFile(fileName: **String): Boolean**
3. choiceHandlerer(result: **String**): **Boolean**
4. startSimulation(creatures: **ArrayList<creatures>**, days: **String**, deatails: **Boolean): ArrayList<Creatures>**
5. simulationAnalyze(remainedCreatures: **ArrayList<creatures>**)
6. findLongestDistance(remainedCreatures: **ArrayList<creatures>**): **Int**
7. howManyWinners(remainedCreatures: **ArrayList<creatures>**, longestDistance: **Int**): **Int**

A főprogram legfontosabb statikus függvényeinek kifejtése:

menu()

A felhasználó számára létrehozott függvény, mely megkönnyíti a program használatát.
A felhasználó 4 lehetőség közül választhat:

1. Tetszőleges input fájl futtatása
2. Példa input fájl futtatása
3. Tesztek futtatása
4. Kilépés a programból

checkInputFile(fileName: **String): Boolean**

Amennyiben a felhasználó az első vagy a második lehetőséget választotta a fájlon elvégzünk egy hibakeresését, melyért ez a statikus metódus felel.
Ebben a metódusban leellenőrizzük az alábbiakat:

- Létezik-e a fájl?
- Üres-e a fájl?
- Típushelyes-e a fájl?
- Megfelelő adatok szerepelnek-e a fájlban?
- Szabály szerint követik-e egymást az adatok?

choiceHandlerer(result: **String**): **Boolean**

Az alábbi statikus függvény felel azért, hogy a felhasználó által választott menüpontot megfelelően kezelje, és a megfelelő metódusok fussanak le.

startSimulation(creatures: **ArrayList<creatures>**, days: **String**, deatails: **Boolean**)**: ArrayList<Creatures>**

A főprogram legfontosabb statikus függvénye. Ez a függvény indítja el a szimulációt a ***days*** paraméter **hossza**, illetve **tartalma alapján.** Minden egyes ciklus egy-egy nap a ***days*** Stringben. Amelyik nap betűjele következik, azt a napot hívjuk meg minden egyes lényen, amelyik életben van / maradt.

A ciklus addig megy, amíg vagy egy lény marad életben, vagy ***days*** String elemein végig mentünk.

details Booelan paraméter meghatározza, hogy a felhasználó számára kiírassuk-e a szimuláció részleteit minden egyes napról. Ebben az esetben a felhasználó minden egyes nap időjárásáról, a lények vízkészleteiről, megtett távolságukról és a nevükről kaphat információt a konzolon.

A visszatérési érték a ciklusok alatt módosított lények gyűjteménye lesz, melyet továbbadunk a szimulációt analizáló statikus metódusnak, mely ki fogja írni a felhasználónak a győztest vagy a győzteseket.

simulationAnalyze(remainedCreatures: **ArrayList<creatures>**)

Ez a statikus metódus felel azért, hogy a *startSimulation*-ből kapott módosított lények gyűjteményét kiértékelje, és végül kiírassa azt a felhasználó számára a konzolra.

Ez a metódus felhasználja a *findLongestDistance* és a *howManyWinners* metódusok végeredményeit.

findLongestDistance(remainedCreatures: **ArrayList<creatures>**): **Int**

A paraméterül megkapott módosított lények gyűjteményeit felhasználva egy maximumkeresés programozási tételt indít, és visszaadja a leghosszabb megtett távot.

howManyWinners(remainedCreatures: **ArrayList<creatures>**, longestDistance: **Int**): **Int**

A paraméterül megkapott módosított lények gyűjteményében, a leghosszabb megtett táv alapján egy megszámlálás programozási tételt indít a gyűjteményben, és visszatér a győztesek számosságával.

Tesztesetek

Az automatikus teszteseteket 4 fő esetre bontottam.

1. Szükséges ellenőrizni a fájl helyességét, ezen belül, hogy a fájl létezik-e, megfelelően van kitöltve-e, adatok típusában és értékében megfelelnek-e a szabályoknak.

2. Szükséges ellenőrizni továbbá, hogy a leghosszabb táv kiszámító algoritmus megfelelően működik-e a módosított lények gyűjteményeiben.

3. Szükséges ellenőrizni továbbá, hogy a győztesek összeszámoló algoritmus megfelelően működik-e a módosított lények gyűjteményeiben a megtett legnagyobb táv alapján.

4. Szükséges végül ellenőrizni, hogy a főprogram helyes adatok esetén megfelelően működik-e.

Ha a program ebben a 4. esetben az elvártaknak megfelelően működik, feltételezhetjük, hogy bármely bementek esetén működni fog és képes lesz elvégezni a szimulációt, vagy képes lesz dobni megfelelő hibaüzenetet és ezáltal leállítani a programot.

Programozási Technológia – 1. beadandó (1. feladat)
Abdurasitov Alekszandr
A49MZV

1) **Teszteset: Fájlhelyesség tesztelése**
    - **a.** Ha nem létező file-t adunk meg.
        - **i.** Hibaüzenet és programleállás
    - **b.** Ha üres file-t adunk meg.
        - **i.** Hibaüzenet és programleállás.
    - **c.** Ha kevés adat szerepel a file-ban.
        - **i.** Hibaüzenet és programleállás.
    - **d.** Ha az adatok típus hibásak (pl. szám helyett string, stb…).
        - **i.** Hibaüzenet és programleállás.
    - **e.** Ha rossz adatokat adunk meg (pl. lények száma kevesebb, mint 1, stb…)
        - **i.** Hibaüzenet és programleállás.
    - **f.** Ha minden adat megfelel az előírtaknak.
        - **i.** Program tovább fut.

2) **Teszteset: Leghosszabb távot kiszámító algoritmus tesztelése.**
    - **a.** Ebben a tesztesetben létrehozzuk manuálisan a lények gyűjteményét, majd minden egyes lénynek módosítjuk a megtett távolságát, ha ebben az esetben megtalálja a legnagyobb megtett távot, akkor következtethetünk az algoritmus helyességére.

3) **Teszteset: Győztesek megszámlálása algoritmus tesztelése.**
    - **a.** Az első esetben az ellenőrizzük, hogy megtalálja-e azt a két lényt, akik ugyanannyi távot tettek meg és még élnek is.
    - **b.** A második esetben minden lény ugyanakkor távolságot tett meg, így vissza kell adnia, hogy mindegyik lény nyert.
    - **c.** A harmadik esetben csak egy lény nyert sok lény közül, ha ez alapján is az eredmény, hogy 1 győztes van, abban az esetben következtethetünk az algoritmus helyességére.

4) **Teszteset: Program helyességének tesztelése.**
    - **a.** Első esetben a példa tesztfájl alapján futtatjuk le az összes legfontosabb statikus metódust, ha ez alapján kijön az előre megírt és papíron kiszámolt eredmények, abban az esetben a statikus metódusok (startSimulation, simulationAnalyze) megfelelően működnek.
    - **b.** Második esetben általam készített teszt fájl alapján futtatjuk le (melynek neve: t9.txt) az összes legfontosabb statikus metódust, ha ez alapján kijön az előre megírt és papíron kiszámolt eredmények, abban az esetben a statikus metódusok (startSimulation, simulationAnalyze) megfelelően működnek.