H

hangyaboly2

0

Problémamegoldás hangya-kolóniával

Ebben a feladatban egy hangyaboly túlélését kell segítenünk az élelmet gyűjtögető viselkedésük leprogramozásával. A szimulációban van egy hangyaboly, ahonnan a hangyák elindulnak, és ahova az élelmet hordják, vannak ételforrások, és persze vannak hangyák is.

Általános információk

A megoldást egyetlen `zip` állományként kell feltölteni, amely tartalmazza a csomagnak megfelelő könyvtárszerkezetben az összes forráskódot. Az 5. csoport beadási helye Linux alatt az smb://nas2.inf.elte.hu/zh/java (smb://nas2.inf.elte.hu/zh/java), Windows alatt a \\nas2.inf.elte.hu\zh\java (/java/hangyaboly2/wikis/%5C%5Cnas2.inf.elte.hu%5Czh%5Cjava) mappa. A fordítás során létrejövő `class` állományokat viszont nem szabad már mellékelni! A fordításhoz a Java Standard Edition 8 használata kötelező.

A megoldásunk tesztelésére használjuk a Test.java (/java/hangyaboly2/wikis/Test.java) osztályt, melyet a gyökérmappába kell helyezni. Kommentezzük vissza azokat a részeket, amelyeket már leimplementáltunk és fordítás (`javac Test.java`) után futtassuk le (`java Test`), hogy az ellenőrzés megtörténjen.

Amennyiben a Test.java osztály már sikeresen lefut a SzimulacioFuttatas.java (/java/hangyaboly2/wikis/files.zip) osztállyal tudjuk futtatni a szimulációt. Fordítása a `javac SzimulacioFuttatas.java` , futtatása a `java SzimulacioFuttatas` paranccsal lehetséges. Ebben karakterterminálon követhetjük a hangyáinkat, ahogy az élelmet gyűjtögetik. A hangyabolyt az `O` betű jelzi, az élelmet a `K` , az egyes hangyákat a `H` vagy `+` attól függően, hogy van-e náluk étel vagy sem.

Kezdés

- Hozzuk létre a `Vilag` interfészt a `geo` csomagon belül.
- Hozzuk létre az `AntiVilagKivetel` osztályt a `geo` csomagon belül.
- Hozzuk létre a `Pont` osztályt a `geo` csomagon belül.
- Hozzuk létre a `Terep` osztályt a `geo` csomagon belül.
- Készítsük el az `Elelem` osztályt a `flora` csomagon belül.
- Hozzunk létre egy `Hangya` osztályt a `fauna` csomagon belül.
- Hozzuk létre a `Szimulacio` osztályt a `szimulacio` csomagon belül.

A terepen való pozíciók tárolása

A hangyák kétdimenziós véges méretű világban mozognak, amelyben az egyszerűség kedvéért egész számokkal jelzett pozíciókat tudnak elfoglalni.

- A `Pont` osztály egy `x` és egy `y` egész típusú adattagot tartalmaz.
- Készítsük el a konstruktort, ami két egész számot vár, és előállítja az adott számpárnak megfelelő pontot. Legyen egy másik konstruktor, amely egyetlen `Pont` -ot kap paraméterként. Amennyiben a pont egyik koordinátája negatív lenne, dobjunk egy `AntiVilagKivetel` -t.
- Hozzuk létre az `equals` metódust, amely egy másik `Pont` objektumot vár, és megvizsgálja, hogy a két pozíció azonos-e (x és y koordináta megegyezik-e).
- Írjuk meg a `toString` metódust, amely visszaadja az objektum egy szöveges leírását, pl.: `Helyzet: x=4, y=4;`
- Írjuk meg a `hashCode` metódust, amely visszaad egy egyedi, az adott pontra jellemző egész számot, pl.:

```
int hash = 23;
hash = hash * 31 + x;
hash = hash * 31 + y;
return hash;
```

- Hozzuk létre a visszatérési érték nélküli `lepesAdottIranyba` metódust, amely egy másik `Pont` objektumot vár, és az adott pozíciót annak irányába mozgatja egyel. Ha nem azonos oszlopban (x-koordináta) vannak, akkor először az oszlop felé lép. Ha azonos oszlopban vannak, akkor a sor felé lép. Ha a két pont azonos, akkor nem módosít.
- Hozzuk létre az `eszak` , `del` , `kelet` , `nyugat` metódusokat. Ezek olyan pozíciókat állítanak elő és adnak vissza, amely a jelenlegi pozíciótól közvetlenül fölfele, lefele, jobbra vagy balra vannak.

Ételforrás

Az ételforrásoknak két jellemzője van, a helye és a benne található ételdarabok mennyisége. Amikor egy hangya eljut az ételforráshoz, kivesz belőle egy egységnyit.

- Az `Elelem` osztály a `Pont` osztály leszármazottja, és egy privát egész típusú `mennyiseg` adattagot tartalmaz.
- Írjuk meg a konstruktorát, ami elkészíti a paraméterként megadott mennyiséggel és `Pont` -al megadott pozícióval rendelkező ételforrást. Amennyiben a pozíció egyik koordinátája negatív lenne, dobjunk egy `AntiVilagKivetel` -t.
- Írjuk meg a `toString` metódust, amely visszaadja az objektum egy szöveges leírását, pl.: `Helyzet: x=4, y=4; mennyiség: 2`.
- Írjuk meg az `egysegnyitCsokkent` metódust, ami eggyel csökkenti az ételmennyiséget, ha az nagyobb mint nulla és igazat ad vissza ebben az esetben, egyébként hamisat.
- Készítsük el az élelem helyzetének pontját visszaadó `helyzet` metódust.

Világ interfész

- Legyen egy egész számot visszaadó `meret` metódus, amely visszaadja a világ méretét.

AntiVilagKivetel

- Legyen egy saját kivételünk, amelyet azon ritka események jelzésére fogunk használni, amikor egy Pont egyik koordinátája negatív lenne.

Terep

- A `Terep` valósítja meg a Világot. Két rejtett adattagja van: egy egész típusú `terepMeret` valamint egy `feromonszint` hasítótábla, amelynek a kulcsa egy Pont, a hozzá tartozó érték pedig egy valós szám.
- A konstruktor egy egész értéket vár, amely megadja a keletkező világ méretét. Hozzuk létre a megadott `meret * meret` hasítótáblát, és minden így megkapott új pontban állítsuk a kezdeti feromonszintet nullára.
  - Írjuk meg a `meret` metódust, amely visszaadja a terep méretét.
  - Írjuk meg a `terepenVane` metódust, amely eldönti a kapott pontról, hogy a terepen van-e vagy sem.
  - Írjuk meg a `feromonszint` metódust, amely visszaadja a feromonszintet a megadott ponton.
  - Írjuk meg a `feromonjeloles` metódust, amely adott mező feromonmennyiségéhez hozzáadja a másik paraméter által megadott értéket a megadott ponton.
  - Írjuk meg a `feromonbomlas` metódust, amely `0.9` -el megszorozza az összes mezőn található feromonmennyiséget

Hangya osztály

- Legyen benne egyelőre egy üres, visszatérési érték és paraméter nélküli `lep` metódus.

Szimuláció

A szimulációban egy hangyaboly (kolónia), egy ételforrás, több hangya vesz részt. Az ételforrásból a hangya ki tud venni egy ételdarabkát, és el tudja vinni a bolyba.

- A `Szimulacio` osztály a `Terep` osztályból származik. Legyen benne egy objektumszintű véletlenszám-generátor `Random` típusú publikus `veletlenszamGenerator` adattag. Ezt a teszt kedvéért állítsuk indítsuk el a 0 kezdőértékkel.
- Tárolja egy egész típusú `begyujtottElelemMennyiseg` adattagban, hogy a hangyák hány ételdarabkát gyűjtöttek be.
- A konstruktora kapjon egy terepméretet, a hangyák számát, egy pontot a hangyaboly helyzetének és végül egy élelmet. Létrehoz egy `hangyak` listában annyi hangyát a bolynál, amennyire szükség van.
- Az `elelem` metódus egy pontot vár és visszatér az azon a ponton levő élelem referenciájával, ha az az adott mezőn található. Egyébként `null` -t ad vissza.
- A `szimulacioLeptetes` metódus minden hangya `lep` metódusát végrehajtja és visszatérés előtt még meghívja a `feromonbomlas` metódust.
- A `hangyabolyHelyzet` és az `elelemForras` metódusok visszaadják a kolónia helyét és a szimulációban szereplő élelemforrást.
- A `hangyak` metódus visszaadja a szimulációban szereplő hangyák listáját.

`Pont` befejezése

- Hozzuk létre a `Pont` osztályban a `szomszedosPontok` metódust. Ez megkapja a jelenlegi szimulációt és egy láncolt listában visszaadja az eszak , del , kelet , nyugat `metódusok eredményei közül azokat, amelyek a vizsgált területen belül (` terepenVane` metódus) vannak.

A hangyák programozása

A hangyák alapvetően véletlenszerűen fognak mozogni a terepen. Ha ételt találnak, akkor felszedik és visszaviszik a bolyba. Ekkor izgatottak lesznek, és feromon-nyomokat hagynak maguk után.

- Készítsük el az `Hangya` osztályt. A hangya rendelkezik helyzettel, egy `boolean` értékkel, ami jelzi, hogy van-e nála étel, egy 0 és 1 közötti `izgatottsag` értékkel, egy egész értékkel, ami számolja, hogy mennyi ideje (hány szimulációs lépés óta) van a hangya a bolyon kívül. Minden hangya tudja a saját létrehozáskor beállított sorszámát, amelyet egy osztályszintű számláló alapján kap meg. Továbbá a hangya példány rendelkezik egy referenciával a szimulációra, amiben szerepel.
- A hangya konstruktora megkapja a szimulációt és a hangyaboly helyének pontját (ami az ő kezdőhelye).
- A `Hangya` osztálynak legyen egy `VISSZAFORDULASIPONT_LEPESSZAM` osztályszintű egész értékű konstansa, ami meghatározza, hogy mennyi ideig legyen kinn a hangya, mielőtt visszaindulna a bolyba. Az értéke legyen `15` .

- A hangya pozíciója legyen lekérdezhető a `helyzet` metódussal, mely visszaad egy pontot.
- A `viszeElelmet` metódus választ ad arra, hogy a hangya visz-e élelmet.
- A `hangyabolyfeleLep` metódus a `lepesAdottIranyba` metódus segítségével a boly felé viszi a hangyát. Amennyiben van a hangyánál étel, akkor feromonnyomokat hagy maga után (`feromonjeloles` metódus). Kezdetben az izgatottsága (`izgatottsag`) 1 lesz, de minden megtett lépésben `0.9` -el szorzódik. Minden mezőn akkora feromonnyomot hagy, amekkora az izgalma.
- A `bolyong` metódus a hangyák véletlenszerű mozgását valósítja meg. A `szomszedosPontok` függvény használatával létrehozza az elérhető mezők listáját, majd megkeresi, hogy melyiken a legnagyobb a feromonkoncentráció (`feromonszint` metódus). Mivel a feromonok érzékelése bizonytalan, ezért egy 0 és 0.1 közötti értéket kell hozzáadni az összes mezőn levő feromonmennyiséghez, amit a szimuláció véletlenszám-generátorával állítunk elő.
- A `lep` metódus választ a lehetséges viselkedések között. Ha a hangyánál étel van és visszaért a bolyhoz, akkor azt lerakja a bolyban (`viszeElelmet` `false` lesz), és eggyel növeli a begyűjtött élelmiszermennyiséget (`begyujtottElelemMennyiseg`). Bármikor, ha a bolynál van, akkor lenullázza az izgatottságát, és a kint töltött időt. Ha ételt visz vagy már annyi ideje van kinn, mint a `VISSZAFORDULASIPONT_LEPESSZAM` akkor visszafelé megy a bolyhoz a `hangyabolyfeleLep` metódussal. Ha azon a mezőn van, ahol az étel van (`elelem`) és még tud belőle hozni, akkor izgatottsága 1 lesz, és innentől kezdve egy hangyaharapásnyi ételegységet szállít (`viszeElelmet` igaz lesz).
- Írjuk meg a `toString` metódust, amely visszaadja az objektum egy szöveges leírását, pl.: `Hangyaszám: 2; Helyzet: x=4, y=4; Izgatottság: 0.5314410000000002; Megtett lépések: 0; Visz-e élelmet: igen