Feladatleírás

A feladatban egy hangyabolyban hangyák közlekednek a bolyón belül táplálékot keresve. Az elkészítendő szimuláció során a hangyák élelmet keresve bolyonganak, szaporodnak és ha nem találnak bizonyos időn belül táplálékot, elpusztulnak.

A feladat megoldása során a megfelelő nyelvi eszközök igénybevételével szükséges az erőforrások elérését szabályozni, elkerülve holtpontok létrejöttét.

Az alábbi osztályokat szükséges elkészíteni. Ezek az alapértelmezett (névtelen) csomagba kerüljenek.

1. **Nest osztály**:
    - A boly egy területet szimulál, amelynek határain túl a program futása során a hangyák nem léphetnek. Ezt egy mátrix segítségével fogjuk reprezentálni. A mátrix elemei lehetnek `Object` típusúak létrehozáskor.
    - Az alapértelmezett hossz és szélesség legyen 30, de ez legyen konfigurálható.
    - Alapértelmezetten 10 hangyát, 10 élelemforrást és 10 feromont hozunk létre a program futásának elején.
    - Miután megtörtént a kezdeti inicializáció, konstans időnként (alapértelmezett érték 200 ms) helyezzen egy feromont és egy élelmet a program a mátrix egy-egy véletlenszerű mezőjére, ami még üres.
2. **Ant osztály**:
    - Minden hangya saját szálon fut, a `toString()` által visszaadott érték legyen `A`.
    - A hangyák eltárolják, hogy mennyi lépésre elegendő élelmet vettek magukhoz. Kezdetben ez az érték 50. Ha a program futása során az érték elérné a 0-t, akkor a hangya elpusztul.
    - A hangyák egy véletlenszerű mezőn kezdenek.
    - Két lépés között egy konstans ideig várnak (alapértelmezetten 200 ms).
    - Egy hangya véletlenszerűen választja ki mozgásának irányát, bár olyan mezőre nem léphet, amin már hangya áll. Ha olyan mezőre lép, amelyen élelem van, akkor 20-szal növelje meg a még megtehető lépések számát, majd lépjen arra a mezőre, ahol korábban az élelem volt. Ha olyan mezőre lépne, amelyen feromon van, akkor szaporodik és egy véletlenszerű mezőn új hangya jön létre, majd megteszi a lépést a mezőre, ahol korábban a feromon volt.
    - A hangya osztály lépései során használt szinkronizáció ne az egész mátrixot foglalja le. Ehelyett csupán a hangya körüli mezőket használja szinkronizációs erőforrásként olyan sorrendben és módon, ami nem okoz holtpontot.

2. **Ant osztály**:

   - Minden hangya saját szálon fut, a `toString()` által visszaadott érték legyen `A`.
   - A hangyák eltárolják, hogy mennyi lépésre elegendő élelmet vettek magukhoz. Kezdetben ez az érték 50. Ha a program futása során az érték elérné a 0-t, akkor a hangya elpusztul.
   - A hangyák egy véletlenszerű mezőn kezdenek.
   - Két lépés között egy konstans ideig várnak (alapértelmezetten 200 ms).
   - Egy hangya véletlenszerűen választja ki mozgásának irányát, bár olyan mezőre nem léphet, amin már hangya áll. Ha olyan mezőre lép, amelyen élelem van, akkor 20-szal növelje meg a még megtehető lépések számát, majd lépjen arra a mezőre, ahol korábban az élelem volt. Ha olyan mezőre lépne, amelyen feromon van, akkor szaporodik és egy véletlenszerű mezőn új hangya jön létre, majd megteszi a lépést a mezőre, ahol korábban a feromon volt.
   - A hangya osztály lépései során használt szinkronizáció ne az egész mátrixot foglalja le. Ehelyett csupán a hangya körüli mezőket használja szinkronizációs erőforrásként olyan sorrendben és módon, ami nem okoz holtpontot.

3. **Nutrition osztály**

   - Saját szálon futó osztály, amelyet el lehet helyezni a mátrix valamelyik mezőjén. A szöveges reprezentációja `N` legyen.

4. **Pheromone osztály**

   - Saját szálon futó osztály, amelyet el lehet helyezni a mátrix valamelyik mezőjén. A szöveges reprezentációja `P` legyen.

5. **Empty osztály**

   - Az osztáy szerepe mindössze egy üres mező reprezentálása a bolyban. Szöveges reprezentációja egy szóköz karakter. Kezdetben a mátrix összes olyan mezője legyen egy `Empty` példány, ahol nincs hangya, élelem vagy feromon. Ha futás során egy mező megüresedik, egy `Empty` objektum foglalja el azt a helyet.

6. **Wall osztály**

   - Az ilyen típusú objektumok alkotják a boly falát. A szöveges reprezentáció ebben az esetben egy `#` legyen.