TP WATOR

Le sujet et son traitement ont déjà été largement présentés lors du dernier cours en amphi.

Vous trouverez sur le portail :

- une archive permettant de tester le résultat que vous devez obtenir,
- les squelettes de classes et interfaces issus de l'analyse présentée cours,
- les `.class` du paquetage grid,
- la javadoc du paquetage `grid`.

Rappels

Un environnement est représenté par un tore découpé en cases carrées. Dans cet environnement évoluent, selon certaines règles, des thons et des requins. On trouve au plus un poisson par case. L'évolution se déroule ainsi : on tire aléatoirement une position dans l'environnement et le poisson qui s'y trouve (quand il y en a un) essaie de se déplacer puis éventuellement donne naissance à un autre poisson puis éventuellement meurt.
La dynamique de l'évolution consiste à voir évoluer dans le temps les populations respectives de requins et de thons.

Comportements

Thon (proie)

- Les thons peuvent se déplacer vers une des 8 cases voisines si elle est libre.
- Ils se reproduisent avec une certaine fréquence : après un déplacement réussi, la descendance apparaît sur la case libérée.

Requin (prédateur)

- Les requins peuvent se déplacer vers une des 8 cases voisines si elle est libre ou si elle est occupée par un thon. Dans ce dernier cas le requin mange le thon qui "disparaît".
- La reproduction des requins suit les mêmes règles que celle des thons (avec une fréquence a priori différente)
- Un requin meurt de faim s'il n'a pas mangé de thon avant une certaine durée. Dans ce cas il disparaît.

La gestion des "durées" pour la reproduction ou la famine des poissons se fait en considérant que, du point de vue du poisson, une unité de temps passe à chaque fois qu'il est sélectionné.

Programmation

Programmer la simulation Wator présentée en cours.
Pour avoir un aperçu du résultat attendu, récupérez l'archive `wator.jar` sur le portail (partie Documents) et exécutez[1]

```
java -jar wator.jar 30 30 40 10 2 5 3 100000 100 10 g 10
```

ou

```
java -jar wator.jar 20 20 30 15 2 5 3 100000 100 10
```

Pour pouvoir utiliser les classes de ce paquetage, n'oubliez pas que vous devez soit placer `grid.jar` dans votre variable `CLASSPATH` au moment de la compilation, soit extraire les classes de l'archive dans votre répertoire `classes` (ou les sources dans `src` et les recompiler).

[1] faire simplement `java -jar wator.jar` pour connaître la signification des paramètres

Conformément à l'analyse présentée en cours, vous devrez définir un paquetage `wator` contenant un certain nombre de classes, parmi lesquelles :

- l'interface `Fish` permettant la modélisation des poissons (un source est disponible, vous êtes libre de le modifier si vous le jugez utile)
- les classes `Tuna` et `Shark`, implémentant l'interface `Fish` et qui modéliseront les thons et les requins.
- une classe `Environment` qui implémentera l'interface `grid.Grid`.
  Le "contenu" de l'environnment sera défini par un tableau à 2 dimensions, contenant des "`Fish`". L'absence effective d'un poisson dans une "case" pourra être signalée par la valeur `null`.
  L'initialisation de l'environnement est paramétrée par des pourcentages de thons et de requins.
- une classe `Wator` permettant l'initialisation et l'exécution d'une simulation.
  La visualisation de l'environnement lors d'une simulation se fera en utilisant une implémentation de l'interface `grid.GridDisplayer` (deux implémentations utilisables sont fournies dans le paquetage `grid`).
- ... et peut être d'autres classes dont vous jugerez avoir besoin.