Fakarava

Dans le lagon de Fakarava, les requins gris chassent les mérous ...
(librement inspiré du documentaire 700 requins dans la nuit)

A22 / P21

Merci à tous les lecteurs attentifs ...

Table des matières

1 Présentation du projet

Dans la présentation qui suit, il y a quelques constantes numériques notées en majuscule gras. Ces constantes sont fixées une fois pour toutes au lancement d'une simulation (méthode `init`*).*

Le lagon de Fakarava est modélisée par une grille carrée contenant $\mathbf{N} \times \mathbf{N}$ cellules.

Chaque **poisson** a un *nom* (en fait le nom de son espèce comme « mérou, requin gris, poisson-perroquet, murène, requin marteau, etc. ») , un *poids* (un réel), un *âge* et un *numéro* qui l'identifie de manière unique. Chaque poisson occupe une *position* (une cellule) dans la lagune donnée par un couple d'entiers entre 0 et $\mathbf{N}-1$ (exemple $(3,4)$). Il se déplace aléatoirement dans le lagon toutes les unités de temps (une unité de temps = 12 heures = 1 journée ou 1 nuit). La longueur d'un déplacement est d'au plus 1 (elle peut être nulle).

On distingue deux types de poissons, les prédateurs et les proies.

— Les **proies** ont une *vivacité* (un entier) qui dépend de l'individu. Dans tous les cas, cette vivacité est divisée par deux la nuit.

— Les **prédateurs** se nourrissent de proies. Leur poids diminue de 1 unité à chaque unité de temps mais augmente ensuite éventuellement du poids de la proie consommée.

Les poissons se reproduisent par clonage [1] , toutes les **PREDATOR_CLONE_TIME** unités de temps pour les prédateurs et toutes les **PREY_CLONE_TIME** unités de temps pour les proies.

Les prédateurs chassent leurs proies en des endroits bien déterminés, des cellules particulières du bord du lagon qui communiquent avec l'océan : les **passes** . On peut créer autant de passes qu'on veut dans la limite de la capacité du bord de la grille mais d'une part, on ne peut pas créer deux passes dans des cellules adjacentes et d'autre part, plus il y a de passes, moins elles sont efficaces pour la chasse car la force du courant — facteur positif pour la chasse — diminue :

force du courant = **MAX_CURRENT_STRENGTH**/nb de passes.

1. ~~Ce sont les valeurs **instantanées** des attributs qui sont dupliquées.~~

Une fois créée, une passe ne peut pas être supprimée. Chaque unité de temps, les passes (à l'exclusion des autres cellules) sont le théâtre de scènes de chasse : chaque prédateur présent dans la passe attaque à tour de rôle, du plus gros au moins gros, la proie survivante la moins vive [2] (si une telle proie existe). La probabilité de survie d'une proie dépend de son agilité, de la vitesse du courant dans la passe et du poids du prédateur :

$$p = (\ \max(0,\ \text{vivacite de la proie}/\text{poids du predateur} - \text{force du courant}/100.0)\ ).$$

Lorsqu'une horde (plusieurs prédateurs) attaque une proie dans une passe, tout prédateur qui rate sa proie mord au hasard l'un des autres prédateurs de la horde. Le poids du blessé est alors diminué d'une quantité q dépendant du poids de l'attaquant :

$$q = \text{poids de l'attaquant}/\textbf{BITE\_FACTOR}.$$

Des scientifiques, spécialistes de biologie marine, plongent régulièrement dans le lagon (jamais la nuit). Leur but est d'obtenir le maximum d'informations sur les requins présents à Fakarava.

Ces **plongeurs** ont bien sûr un *identifiant*, un *nom* et un *laboratoire* (en fait juste le nom de la ville de leur laboratoire) et savent se présenter : "Plongeur ⟨id⟩ : ⟨nom⟩ (⟨labo⟩)". Ils consignent dans un *fichier de log* toutes leurs actions. Ils font au maximum une action par jour — mais, ils sont tout à fait capables de gérer une TODO-liste.

— Ils peuvent équiper les passes de caméras (au maximum une par passe). Une caméra transmet au plongeur qui l'a posée tout un ensemble d'informations :
  * identifiant de la passe ;
  * descriptions des poissons présents dans la passe : id, nom, poids [3], age et, pour les proies, vivacité ;
  * description de la scène de chasse éventuelle : qui a chassé qui ? Avec succès ou pas ? Qui a mordu qui ?

  Ces informations sont consignées par le plongeur dans son fichier de log à la fin de chaque unité de temps.
— Les plongeurs peuvent aussi poser des émetteurs [4] sur les prédateurs présents dans une passe (au plus un émetteur par prédateur). Ainsi, à partir de la prochaine unité de temps (c'est-à-dire, dès la nuit suivante) un plongeur qui a posé des émetteurs peut ajouter les informations transmises chaque unité de temps par ceux-ci (id, nom, poids, age, position) à son fichier de log.
— On doit pouvoir retirer un plongeur du lagon lorsqu'il retourne dans son laboratoire. Dans ce cas, il repart avec tout son matériel, caméras et émetteurs compris.

Pour la gestion du temps, on supposera que les poissons et les plongeurs héritent d'une interface **Clock** comportant une seule méthode `ticktock()` dont l'appel met les pendules chronobiologiques à l'heure chez les humains comme chez les poissons. Le challenge est de trouver un jeu de paramètres qui permettent d'obtenir avec une bonne probabilité un nombre de tic-tac le plus grand possible. C'est pourquoi votre application retournera juste avant sa fermeture le nombre de tic-tac exécutés. De plus, vous munirez l'application d'une option -v ou –verbose qui aura pour effet d'afficher la description de tous les poissons à chaque unité de temps. Cela vous permettra de vérifier ce qui se passe dans la lagune pendant la phase de développement de l'application.

L'application se termine soit lorsque tous les plongeurs ont quitté l'atoll de Fakarava, soit lorsqu'il n'y a plus de proies (les requins quittent le lagon, ou meurent de faim), soit lorsque la densité de proies devient trop importante (pression trop forte sur l'écosystème du lagon), c'est-à-dire lorsque le nombre de proies est supérieur à **MAX_DENSITY** fois le nombre de cellules du lagon.

2. En cas d'ex-æquo, on peut utiliser n'importe quel poisson qui satisfait au critère (« plus gros » ou « moins vif »).
3. Il s'agit du poids avant l'action de chasse éventuelle.
4. Poser une caméra et poser des transmetteurs sont des actions différentes même si les instructions sont données en même temps et pour la même passe.

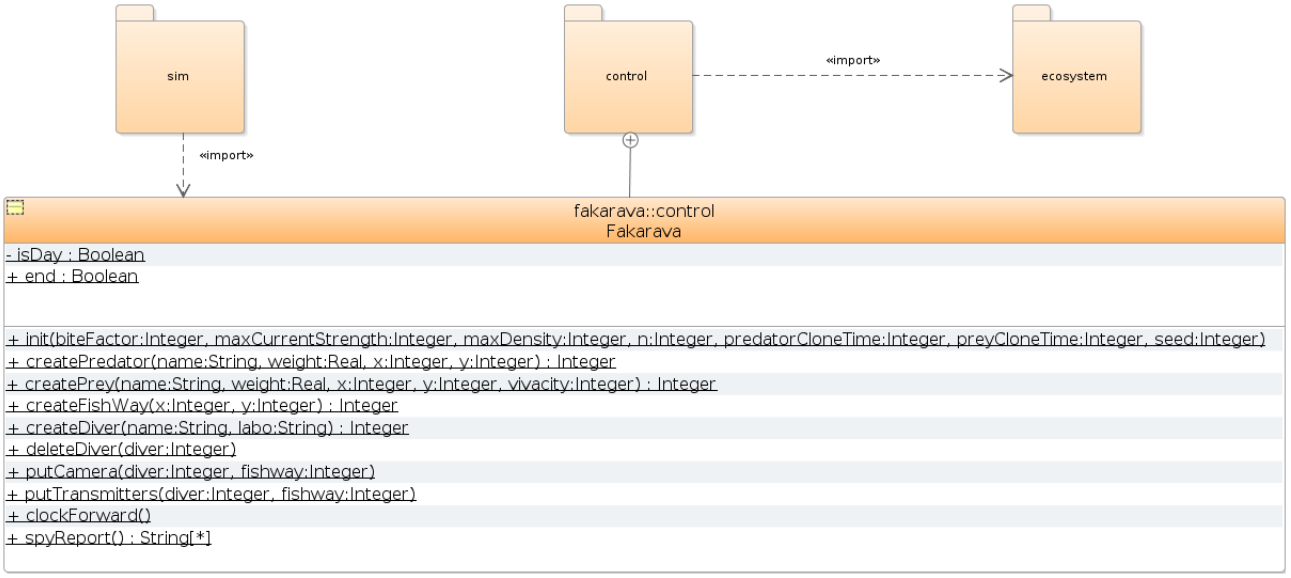
2 Packaging — Interface — Hasard

— Les classes métiers seront écrites dans un paquet `fakarava.ecosystem`, vos gestionnaires éventuels dans un paquet `fakarava.control` et le scénario de la simulation dans un paquet `fakarava.sim`.

— Afin d'uniformiser l'accès a vos classes métier (ce qui nous permettra de vous proposer des tests de validation), vous écrirez une classe `Fakarava` dans le paquet `fakarava.control`. Cette classe contiendra les attributs et méthodes statiques suivantes :

```
private static boolean isDay;
public static boolean end;
public static void init( int biteFactor, int maxCurrentStrength, int maxDensity,
        int n, int predatorCloneTime, int preyCloneTime, Long seed)
public static int createPredator(String name, double weight, int x, int y)
public static int createPrey(String name, int weight, int x, int y, int dayVivacity)
public static int createFishway(int x, int y)
public static int createDiver(String name, String labo)
public static void deleteDiver(int diver)
public static void putCamera(int diver, int fishway)
public static void putTransmitters(int diver, int fishway)
public static void clockForward()
public static String[] spyReport()
```

La dernière méthode ne sera utilisée que pour l'option `--verbose`.



— Enfin, pour obtenir un comportement identique et reproductible pour toutes vos applications,

* l'ordonnancement des tâches se fera comme dans l'exemple de l'annexe B : clonage → déplacement → prédation ;
* vous utiliserez pour tous vos tirages aléatoires la classe `Random` fournie sur Moodle. Cette classe spécialise la classe `java.util.Random` en fournissant trois méthodes `move(x:int, y:int, gridSize:int):int[2]`, `selection(p:double):boolean` et `who(k:int):int` permettant respectivement de déplacer un point dans le lagon en accord avec le cahier des charges, de tirer aléatoirement l'exécution ou non d'un événement de probabilité p et de choisir aléatoirement un individu parmi k individus. Lors d'un test, si on donne à l'attribut `mySeed` de cette classe une valeur donnée, différente de `null`, toutes les exécutions du test donneront les mêmes résultats. ~~Ce résultat ne dépendra que de la suite des appels aux trois méthodes `move`, `selection` et `who`, indépendamment de la valeur de leurs paramètres.~~

3 Déroulement du projet

- Un projet par binôme (ou monôme/trinôme : justifier)
- Le développement du projet se fera de façon itérative :

  iteration 1 on oublie les plongeurs et la prédation ; il reste des poissons qui se déplacent, se reproduisent et finissent par mourir (surpeuplement).

  iteration 2 on oublie toujours les plongeurs mais on tient compte de la prédation. Il faut alors créer les passes à poisson et ajouter éventuellement quelques opérations aux prédateurs.

  iteration 3 on ajoute les plongeurs (ce qui nécessite peut-être d'ajouter des membres aux autres classes).
- Lorsque vous faites une modification qui impacte le DCA et/ou un DCC déjà rendu (la phase de codage vous a amené à ajouter une méthode, ou modifier sa signature), il vous est demandé de le noter sur un **journal du projet** en indiquant la date, la modification effectuée et la raison de cette modification et d'inclure dans votre rendu les nouvelles versions de ces diagrammes.
- Échéancier des dépots (voir avec votre enseignant) :
  - Semaine 1 ~~(13 mai)~~
    - Diagramme des classes d'analyse de l'ensemble du projet : classes, attributs, relations.
    - Diagramme des classes de conception pour l'itération 1
  - Semaine 2 ~~(20 mai)~~
    - code de l'itération n° 1
    - Diagramme des classes de conception pour l'itération 2
  - Semaine 3 ~~(27 mai)~~
    - code de l'itération n° 2
    - Diagramme des classes de conception pour l'itération 3
  - Semaine 4 ~~(3 juin)~~
    - code de l'itération n° 3
- La note du projet tiendra compte des critères suivants :
  - respect des conventions de nommage JAVA ;
  - code commenté ;
  - respect des principes de la conception orientée objet (encapsulation, séparation des responsabilités, etc.) ;
  - qualité des diagrammes : types, visibilités, distinction abstrait/concret, distinction individuel/collectif/commun, relations munies de noms, de rôles et de multiplicités à chaque extrémités.

L'épreuve finale en A22 consistera en un **TP noté individuel** au cours duquel vous devrez apporter une modification à votre application et déposer le code modifié et les nouveaux diagrammes de classe.

Annexes

A Exemple de simulation

```
int biteFactor = 10,
    maxCurrentStrength = 20,
    maxDensity = 3,
    n = 4,
    predatorCloneTime = 10,
    preyCloneTime = 5;
Fakarava.init(biteFactor, maxCurrentStrength, maxDensity, n, predatorCloneTime,
    preyCloneTime, null);

int jojo = Fakarava.createPrey("Mérou", 3.0, 1, 2, 99);

int lola = Fakarava.createPrey("Poisson Lune", 1.0, 0, 1, 90);
int alfred = Fakarava.createPrey("Poisson Perroquet", 2.0, 0, 3, 51);
int bertha = Fakarava.createPredator("Requin Marteau", 100.0, 1, 2);
int adolphe = Fakarava.createPredator("Requin Gris", 75.0, 0, 2);
int p1 = Fakarava.createFishWay(0, 2);
int enzo = Fakarava.createDiver("Enzo", "Tahiti");
Fakarava.clockForward();

Fakarava.putCamera(enzo, p1);
Fakarava.putTransmitters(enzo, p1);
long time = 1;

while(! Fakarava.end){
    Fakarava.clockForward();
    time++;
    if (verbose)
        System.out.println(Arrays.toString(Fakarava.spyReport())+"\n\n");
}
System.out.println(time);
```

B Exemple de déroulement conforme de la simulation

Constantes :

BITE_FACTOR 10
MAX CURRENT STRENGTH 20
MAX DENSITY 3
N 4
PREDATOR_CLONE_TIME 10
PREY_CLONE_TIME 5

Déroulement :

- Début // $t = 0$, il fait jour.
  - Créations
    - ◇ Proie : jojo (Mérou, poids 3.0, vivacitéJour 99) en cellule (1,2),
    - ◇ Proie : lola (Poisson Lune, poids 1.0, vivaciteJour 90) en (0,1),
    - ◇ Proie : alfred (Poisson Perroquet, poids 2.0, vivacitéJour 51) en (0,3),
    - ◇ Prédateur : bertha (Requin Marteau, poids 100.0) en (1,2),
    - ◇ Prédateur : adolphe (Requin Gris, poids 75.0) en (0,2),
    - ◇ Passe : p1 en cellule (0,2),
    - ◇ Plongeur : enzo (nom Enzo, labo Tahiti).
- `ticktock()` // $t = 1$, il fait nuit.
  - Les prédateurs perdent tous 1 unité de poids (disons, le kg).
  - Tous les poissons se déplacent dans une cellule adjacente. Le hasard fait qu'ils se retrouvent tous dans la passe p1 (adolphe, qui était déjà dans la passe, a tiré au sort la cellule (-1,2) qui se trouve hors de la lagune, il reste donc dans la passe).
  - Le prédateur le plus gros, bertha, attaque la proie la moins vive, alfred :
    - ◇ il fait nuit donc la vivacité d'alfred est 51/2, soit 25,
    - ◇ la poids de bertha est 99.0,
    - ◇ la force du courant est 20/1=20 ;
    - ◇ ainsi, $p = \max(0, 25/99.0 - 20/100.0) \approx 0.052$;
    - ◇ l'appel à la méthode `selection` de `Random` avec l'argument p renvoie `false` : alfred ne survit pas. Il est mangé par bertha dont le poids augmente alors de 2kg (le poids d'alfred).
  - Le prédateur adolphe (deuxième et dernier prédateur de la passe P1) attaque la proie vivante la moins vivace, lola :
    - ◇ il fait nuit donc la vivacité de lola est 90/2, soit 45 ;
    - ◇ la poids d'adolphe est 74.0 ;
    - ◇ la force du courant est 20 ;
    - ◇ $p = \max(0, 45/74.0 - 20/100.0) \approx 0.41$;
    - ◇ l'appel à la méthode `selection` de `Random` avec l'argument p renvoie `true` : adolphe rate sa proie ;
    - ◇ adolphe mord bertha :
      - ○ le poids d'adolphe est 74,
      - ○ la constante BITE_FACTOR vaut 10,

      ⇒ bertha perd 74/10 kg. Son poids est maintenant de 100 -1 + 2 - 7.4 = 93.6 kg.
  - Le plongeur enzo doit aller poser une caméra en p1. Il fera cela quand il fera jour. Il le note sur sa TODO-liste.
  - Le plongeur enzo doit aller poser des émetteurs sur les prédateurs en p1. Il le note sur sa TODO-liste.
- `ticktock()` // $t = 2$, il fait jour
  - Les prédateurs perdent tous 1 kg : bertha → 92.6kg, adolphe → 73kg.
  - Les poissons se déplacent vers une cellule adjacente :
    jojo va en (0, 3), bertha reste en (0, 2) (bertha a tiré un déplacement vers l'ouest, hors du lagon), lola va en (1, 2) et adolphe va (0, 3).
  - Il n'y a pas de scène de chasse puisque le prédateur bertha est seul dans l'unique passe du lagon.
  - enzo pose une caméra dans la passe p1. Il met à jour sa TODO-liste
- `ticktock()` // $t = 3$, il fait nuit.

  - Les prédateurs perdent tous 1 kg : bertha → 91.6kg, adolphe → 72kg.
  - Les poissons se déplacent vers une cellule adjacente :
    jojo va en (1, 3), bertha va en (1, 2) , lola va en (0, 2) et adolphe va (0, 2).
  - lola et adolphe sont dans la passe p1 ; adolphe dévore lola (il gagne 1kg).
  - enzo a tout vu et tout enregistré dans son fichier de log.
- `ticktock()` // $t = 4$, il fait jour.
  - Les prédateurs perdent tous 1 kg : bertha → 90.6kg, adolphe → 72kg.
  - Les poissons se déplacent vers une cellule adjacente :
    jojo va en (1, 2), bertha va en (0, 2) et adolphe va (0, 1).
  - enzo plonge dans la passe p1 et pose des émetteurs sur les prédateurs présents dans la passe : bertha est désormais trackée par enzo. La TODO-liste d'enzo est vide
  - enzo consigne dans son fichier de log les informations transmises par la caméra : présence de bertha.
- `ticktock()` // $t = 5$, il fait nuit.
  - Les prédateurs perdent tous 1 kg : bertha → 89.6kg, adolphe → 71kg.
  - Les proies se reproduisent par clonage :
    - ◇ jeffFilsDeJojo (Mérou, poids=3.0, vivacité=99) en (1,2).
  - Les poissons se déplacent vers une cellule adjacente :
    jojo va en (0, 2), jeffFilsDeJojo va en (0,2), bertha va en (0, 1) et adolphe va (0, 2).
  - adolphe attaque jojo : succès.
  - enzo consigne dans son fichier de log la présence de jojo et adolphe dans la passe p1. Il note aussi que adolphe a chassé jojo : succès et que bertha est actuellement en (0,1).
- `ticktock()` // $t = 6$, il fait jour.
  - Les prédateurs perdent 1 kg.
  - Les poissons se déplacent vers une cellule adjacente :
    jeffFilsDeJojo reste en (0,2), bertha va en (0, 2) et adolphe va (0, 3).
  - bertha attaque jeffFilsDeJojo : succès
  - enzo remplit son fichier de log.
  - Il n'y a plus de proies : la simulation se termine.