L2, L3

POO **2014-2015**

Projet : Évolution

Modalités

Le projet est à faire en binôme au plus. Les soutenances se feront à deux, mais la note pourra être individualisée si le travail a été trop inégalement réparti, naturellement chacun doit être capable de répondre à toutes les questions. Si vous faites le projet seul, vous serez noté sur le même barème que si vous étiez à deux.

En guise de rapport, vous fournirez une liste des fonctionnalités que vous avez effectivement implémentées, et si elles fonctionnent ou non, ainsi qu'un diagramme des classes utilisées. Un rapport détaillé dans lequel on expliquerait *comment* les fonctionnalités sont implémentées *n'est pas* demandé.

Vous devrez fournir une javadoc et votre code devra être soigneusement commenté.

La date de rendu du projet sera fixée ultérieurement.

Il nous faudra pouvoir tester le projet durant les soutenances, il est donc préférable qu'il fonctionne sur les machines du script si vous êtes en L2 ou sur celles de l'ufr si vous êtes en L3. Ceci même si vous avez prévu d'apporter un portable pour l'occasion car il peu tomber en panne au mauvais moment.

1 Sujet

Le but de ce projet est d'implémenter une simulation d'univers cohérent composé de quelques animaux et de quelques végétaux.

Le but principal du projet n'est pas uniquement de faire une interface graphique, une partie importante de la note portera sur la conception et la qualité de la programmation du ou des jeux eux-mêmes. On s'attend à ce qu'il soit fait usage des concepts appris en cours, et en particulier de l'héritage.

Une interface graphique dont vous n'êtes pas capable d'expliquer la conception ni le fonctionnement, ne rapportera aucun point.

Vous avez le droit d'utiliser les collections de Java.

2 Conseils pour réaliser le projet

1. Prenez le temps de réfléchir à la conception du projet avant de coder.
2. Réfléchissez aussi à la manière de répartir le travail entre les deux personnes du binômes. Faites des points réguliers entre vous.
3. Vous pouvez éventuellement dans un premier temps vous contenter d'une interface texte pour pouvoir tester votre jeu et ajouter l'interface après. Si jamais vous n'arriviez pas à obtenir une interface graphique qui fonctionne, vous pouvez présenter le projet juste avec une interface texte. Vous ne pourrez pas avoir tous les points, mais vous en aurez plus que si vous présentez un projet non testable.

4. Enfin, pensez à faire des sauvegardes fréquentes, sur au moins deux supports différents et en particulier sauvez les versions testables de votre projet même s'il reste des corrections à y apporter.

3 Généralités et vocabulaire

L'univers considéré est composé de moutons, de loups, d'herbe et de sels minéraux. Les loups mangent les moutons pour survivre, les moutons mangent l'herbe dans le même but. Lorsqu'un loup ou un mouton meurt de faim ou de vieillesse, il dépose sur la case de sa mort des sels minéraux, qui serviront d'engrais à l'herbe.

On considère que les loups ne se mangent pas entre eux, et ne peuvent que mourir de vieillesse.

L'univers est dit "mort" lorsqu'il n'y a plus ni loups ni moutons vivants. Un des buts du projet est d'éviter cette situation, en équilibrant soigneusement le nombre initial de loups, moutons, herbe et sels minéraux.

4 Règles à implémenter

Univers

Un univers possède $m \cdot n$ cases, et de l'herbe sur chaque case au début de la simulation. Le nombre de moutons et de loups est fixé par l'utilisateur, et les animaux sont placés aléatoirement dans l'univers (avec un animal maximum par case).

Votre programme doit savoir gérer les cas limites, par exemple plus d'animaux que de cases dans l'univers...

Déplacement

Les moutons et les loups se déplacent d'une case par tour. Chaque animal possède donc 9 possibilités de déplacement à chaque tour.

Dans un premier temps, vous pourrez implémenter un déplacement aléatoire, avant de penser à des algorithmes plus évolués.

Fonctions vitales

Un mouton mange toute l'herbe d'une case, et vit 50 tours (s'il n'est pas mangé avant). À sa mort, si elle est naturelle, il dépose des sels minéraux sur la dernière case qu'il a fréquenté. Deux moutons sur des cases adjacentes peuvent se reproduire si leur sexe le permet : ils sont immobiles pendant un tour, puis un nouveau mouton apparaît sur une case adjacente au mouton mère [1]. Enfin, un mouton meurt s'il est mangé par un loup, s'il n'a pas mangé d'herbe pendant 5 tours (il meurt au $6^{\text{ème}}$), ou s'il est trop vieux.

Un loup mange des moutons (un mouton rassasie un loup), et doit en manger un tous les 10 tours. S'il meurt de vieillesse (au bout de 60 tours) ou de faim, il dépose des

1. Attention, toutes les cases ne sont pas forcément disponibles... Si le jeune mouton n'a pas de case libre, il ne naît pas.

sels minéraux sur la dernière case qu'il a fréquenté. Deux loups peuvent se reproduire de la même manière que les moutons.

Enfin, de l'herbe pousse un tour après que des sels minéraux aient été déposés sur une case dépourvue d'herbe.

5 Interface graphique

Pour l'interface graphique, vous pourrez utiliser les images issues de `http://opengameart.org/` pour les entités de votre univers. Si vous utilisez une autre source, veillez bien à ce que les images soient libres de droits.

6 Interface texte

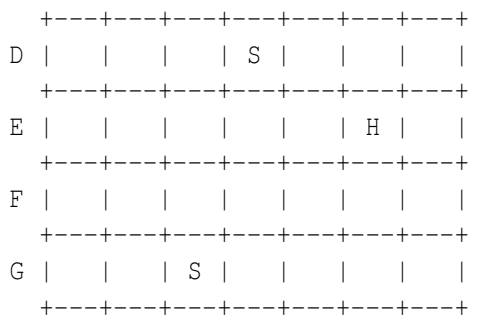
Pour l'interface texte, on pourra par exemple, représenter l'univers par un tableau 2D, utiliser des lettres ou symboles ASCII pour représenter chaque entité, et afficher une trace textuelle de ce qu'il se passe dans l'univers. Par exemple :

```
    1   2   3   4   5   6   7
  +---+---+---+---+---+---+---+
A |   | S |   |   |   |   |   |    [C4] Un mouton a servi
  +---+---+---+---+---+---+---+    de repas a un loup.
B |   |   | L |   | S |   |   |    [A2, B5, D4] De l'herbe
  +---+---+---+---+---+---+---+    repousse !
C |   |   |   | M |   |   |   |    [G3] Un loup meurt de
  +---+---+---+---+---+---+---+    vieillesse.
D |   |   |   | S |   |   |   |    [E6] Un mouton meurt de faim.
  +---+---+---+---+---+---+---+
E |   |   |   |   |   | M |   |
  +---+---+---+---+---+---+---+
F |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+
G |   |   | L |   |   |   |   |
  +---+---+---+---+---+---+---+

Tour 800 | Loups : 2 | Moutons : 2
```

Et l'univers au tour suivant :

```
    1   2   3   4   5   6   7
  +---+---+---+---+---+---+---+
A |   | H |   |   |   |   |   |
  +---+---+---+---+---+---+---+
B |   |   |   |   | H |   |   |
  +---+---+---+---+---+---+---+
C |   |   |   | L |   |   |   |
```




```
Tour 801 | Loups : 1 | Moutons : 0
```

Les plus avancés d'entre vous peuvent ajouter d'autres fonctions. Voici une liste d'idées non exhaustives :

- Ajoutez la possibilité de sauvegarder/charger un univers existant.
- Ajoutez la possibilité de mettre la simulation en pause (et de la reprendre).
- Modifiez le déplacement des loups/des moutons pour qu'il soit plus intelligent. Notamment, pensez à regarder du côté du design pattern *Strategy*.
- Modifiez votre code pour imposer un rythme de reproduction : par exemple, une louve ne peut faire un petit qu'une fois tous les n tours.
- Trouvez les meilleurs paramètres pour faire durer l'univers le plus longtemps possible !