Structures de données et algos fondamentaux (prog1bis)

Sujet 4 (TP) : Combat d'orques

1 Introduction

Nous allons développer un programme de jeu qui consiste à gérer des orques (guerriers) qui s'affrontent dans des arènes.

Nous proposons plusieurs versions du jeu de plus en plus élaborées, en donnant des pistes d'extension. N'hésitez-pas à partir sur d'autres choix.

En plus des classes codant le jeu, nous utiliserons les classes `Ut` et `EE` (ensemble d'entiers). Nous vous conseillons d'utiliser une version correcte/corrigée de cette classe, notamment pour les méthodes `ajoutElt`, `ajoutPratique` et `retraitPratique`. Ajouter également à la classe `EE` les méthodes suivantes qui pourront être utiles.

```
public int retraitEltAleatoirement() {
   // Pré-requis : ensemble this est non vide
   // Résultat/action : enlève un élément de this (aléatoirement) et le renvoie
   int i = Ut.randomMinMax (0, this.cardinal - 1);
   int select = retraitPratique(i);
   return select;
}
public int selectionEltAleatoirement() {
     // Pré-requis : ensemble this est non vide
     // Résultat : un élément quelconque de this choisi aléatoirement
     int i = Ut.randomMinMax (0, this.cardinal - 1);
     return this.ensTab[i];
}
public int selectionElt() {
    // Pré-requis : ensemble this est non vide
    // Résultat : un élément quelconque de this (le dernier, par commodité)
    return this.ensTab[this.cardinal - 1];
}
```

2 Version de base

Le jeu comprend trois classes principales : `Orque`, `Arene`, `MainCombat`.

La classe Orque

Dans sa version de base, la classe `Orque` comprend deux variables d'instance (attributs) : un numéro d'identification `id` et un objet `arene` de la classe `Arene` où l'orque va combattre.

La classe comprend également un *répertoire* de tous les orques créés/nés, géré par deux *variables de classe* :

- un entier `Orque.nbOrques` donnant le nombre d'orques vivants ou morts créés/nés à un moment (nous supposons que ce nombre ne dépassera pas 1000).
- un tableau `Orque.tabOrques` d'objets de la classe `Orque` contenant l'ensemble des orques vivants ou morts créés à un moment ; on écrira :
  `private static Orque [] tabOrques = new Orque [1000];`

Tous les orques créés (dans n'importe quelle arène) se trouvent dans le sous-tableau `Orque.tabOrques[0..Orque.nbOrques-1]`, et l'identifiant d'un orque est son indice dans ce sous-tableau. La variable `Orque.nbOrques` donne donc aussi l'identifiant du prochain orque qui sera créé.

Remarque

Les variables de classe (statiques) sont initialisées au chargement de la *classe* (par la JVM) et ne sont donc pas initialisées dans les constructeurs (d'objets).

Coder une méthode de construction qui prend en paramètre une arène (déjà créée), initialise les variables d'instance et met à jour les variables de classe.

Il faut aussi prévoir une méthode de combat avec un autre orque retournant l'identifiant du gagnant. Dans cette version naïve, le gagnant sera choisi au hasard.

Indication : La méthode de classe `Ut.randomMinMax(int min, int max)` permet de renvoyer un nombre entier pseudo-aléatoire compris entre `min` et `max` (lire le code dans `Ut.java`).

Pour vous aider, prévoir aussi une méthode `int getId()` qui permet de connaître l'identifiant d'un orque et une méthode duale `static Orque getOrqueById (int ident)` qui permet d'obtenir un objet `Orque` à partir de son identifiant.

La classe Arene

Dans cette première version, la classe `Arene` comprend essentiellement comme attribut un ensemble d'entiers `ensOrques` (de type `EE`) contenant les identifiants des orques encore vivants de l'arène.

Le constructeur de `Arene` prend en paramètre un nombre d'orques `nbo`. Il crée (construit) `nbo` orques qui combattront dans cette arène.

Par exemple, si une arène `arene1` est créée avec 10 orques, le sous-tableau `Orque.tabOrques [0..9]` contient ces 10 orques dont les identifiants sont les entiers de 0 à 9, `Orque.nbOrques` est égal à 10 et `arene1.ensOrques` contient les entiers de 0 à 9.

Si une deuxième arène `arene2` est créée avec 15 orques, le sous-tableau `Orque.tabOrques[10..24]` contient ces 15 orques dont les identifiants sont les entiers de 10 à 24, `Orque.nbOrques` est égal à 25 et `arene2.ensOrques` contient les entiers de 10 à 24.

La méthode `bataille` gère les combats entre les orques de l'ensemble `this.ensOrques`. Elle procède itérativement à des duels à mort entre deux orques jusqu'à ce qu'il n'en reste plus qu'un. Pour un duel donné, deux éléments de `this.ensOrques` sont sélectionnés au hasard et retirés de l'ensemble. Après le duel, le gagnant est remis dans `this.ensOrques`.

Par exemple, après l'exécution d'une bataille dans `arene1`, `arene1.ensOrques` ne contient plus que l'identifiant de l'orque gagnant ; restent inchangées les variables de classe `Orque.tabOrques` et `Orque.nbOrques`.

Le programme principal : la classe MainCombat

La procédure principale construit des arènes (qui créent elles-mêmes des ensembles d'orques combattants) et appelle la méthode `bataille` sur ces arènes.

Bonus

Une fois cette première version codée, on pourra rendre la simulation plus réaliste en améliorant deux points principaux :

- améliorer les duels en ajoutant et prenant en compte des caractéristiques des orques : poids, points de vie (pdv, 0 signifiant la mort de l'orque), aggressivité, etc., et un équipement en armes.
- placer les orques spatialement : les orques sont placés sur l'arène (qui est un carré de taille de côté donné) en une position donnée (abscisse et ordonnée), et se déplacent pour combattre.

3 Des duels plus réalistes (bonus)

Les suggestions suivantes sont données à titre indicatif. Ne pas hésiter à faire d'autres choix.

Caractéristiques des orques et des armes

Les caractéristiques des orques peuvent rendre les combats plus réalistes. On peut les choisir aléatoirement (dans un intervalle donné) lors de leur création pour disposer d'individus différents. On peut envisager les attributs `pdv`, un entier qui diminue au cours des combats (0 signe l'arrêt de mort de l'orque), `poids`, qui modifie les dégâts causés par l'arme, et `aggressivite` (ou `dexterite`), qui modifie la chance de toucher l'adversaire. Enfin, l'attribut `armes` de type `EE` qui donne l'équipement en armes de l'orque, représenté par l'ensemble des identifiants des armes (voir ci-dessous).

Un duel entre un orque et son adversaire peut se dérouler comme suit. Ils prennent chacun une de leurs armes (au hasard). Tant que les deux adversaires sont en vie, un des duettistes cherche à blesser l'autre, à tour de rôle. Sa probabilité de toucher l'adversaire dépend des caractéristiques de son arme et de sa dextérité. En cas de touche, les dégâts causés dépendent de l'arme et du poids de l'attaquant.

Une classe Arme

On pourra définir une arme par son `type` (une chaîne de caractères comme "hache", "epee", "lance", "fléau"), les `degats` causés, c'est-à-dire le nombre de pdv que l'arme peut retirer à l'adversaire à chaque touche, et un nombre entre 0 et 100 `probaTouche`, une « probabilité » de toucher l'adversaire.

On pourra définir une variable de la classe `Arme` nommée `tabArmes` qui est un tableau de tous les objets `Arme` existants : un objet pour l'épée, un autre pour la hache, etc. Noter qu'il n'y a dans le tableau qu'un exemplaire de chaque type d'armes, si bien qu'en pratique toutes les épées utilisées par les orques auront les mêmes caractéristiques (dégâts et probabilité de toucher).

A la création d'un orque, on pourra lui associer un ensemble `armes` de la classe `EE`, les entiers représentant les indices dans le tableau `Arme.tabArmes`.

Prévoir une méthode permettant de récupérer l'objet arme (et donc ses caractéristiques) à partir de son identifiant (l'indice dans le tableau).

4 Placement dans l'arène (bonus)

On peut aussi considèrer que les orques ont une position (x, y) dans l'arène et se déplacent.

On augmente le réalisme des combats en faisant évoluer les coordonnées des orques. Par exemple, tous les orques se déplacent d'une case entre chaque tour de jeu (aléatoirement), et les orques à une certaine distance l'un de l'autre s'affrontent en duel...

Interface graphique

Dans les versions précédentes, on se contente de simples traces textuelles pour montrer l'évolution de la bataille, par exemple : combat entre orque 6 et orque 3, victoire de l'orque 6, etc.

Pour améliorer l'esthétique du jeu, on pourra étudier et utiliser les classes présentes dans le répertoire `graphisme/` obtenu après avoir « dézippé » le fichier d'archive `UTILE/graphismeUpdate.zip` sous Moodle.