IUT de Montpellier – Département informatique – 1ère année

Programmation 1 bis

TP : Combat d'orques

1 Introduction

Nous allons développer un programme de jeu comprenant plusieurs classes. Le jeu consiste à gérer un ensemble d'orques qui vont s'affronter dans une arène.

Nous proposons plusieurs versions du jeu de plus en plus élaborées pour donner des idées de programmes possibles. N'hésitez-pas à partir sur d'autres choix.

2 Version de base

Le jeu comprend trois classes principales (`Orque`, `Arene`, `MainCombat`), en plus de classes utiles (`Ut` et `EE`).

La classe Orque

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

Coder une méthode de construction qui prend en paramètre un numéro d'identification de l'orque et une arène (déjà créée).

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 : pour ce tirage aléatoire comme pour les suivants, on trouvera dans la classe `Ut` (« Utile ») la méthode :

```
public static int randomMinMax(int min, int max) {
    Random rand = new Random();
    int res = rand.nextInt(max - min + 1) + min;
    assert min <= res && res <= max : "tirage aleatoire hors des bornes";
    return res;
}
```

(Pour que les assertions soient prises en compte par java, il faut ajouter l'option `-ea` à l'exécution.)

La classe Arene

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

La classe comprend également deux variables de classe :

– un tableau `Arene.tabOrques` d'instances de la classe `Orque` contenant les orques vivants ou morts créés dans toutes les arènes ; on écrira :
`private static Orque [] tabOrques = new Orque [1000] ;`
– un entier `Arene.nbOrques` donnant le nombre d'orques vivants ou morts créés dans toutes les arènes.

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

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 `Arene.tabOrques [0..9]` contient ces 10 orques dont les identifiants sont les entiers de 0 à 9, `Arene.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 `Arene.tabOrques[10..24]` contient ces 15 orques dont les identifiants sont les entiers de 10 à 24, `Arene.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`.

Pour faciliter cette sélection, on pourra ajouter à la classe `EE` la méthode :

```
public int selectEltAleatoirement() {
    // Pre-requis : ensemble this est non vide
    // Resultat/action : enleve un element de this (aleatoirement)
    //                   et le retourne
    int i = Ut.randomMinMax(0, this.cardinal - 1);
    int select = retraitPratique(i);
    return select;
}
```

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

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.

3 Des combats plus réalistes

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, taille, pdv (points de vie, 0 signifiant la mort de l'orque), aggressivité 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.

Une classe Arme et un tableau d'armes possibles

On pourra définir une arme par son type (une chaîne de caractères comme "hache", "epee", etc), le nombre de pdv maximum que l'arme peut retirer à l'adversaire à chaque touche, et une « probabilité » de toucher l'adversaire (un nombre entre 0 et 100).

On pourra prévoir un constructeur qui construit aléatoirement une arme dans une liste donnée (par exemple, hache, épée, lance, fléau).

Un moyen de disposer de toutes les armes disponibles sur une arène est de définir une variable de classe nommée `tabArmes` qui est un tableau d'objets de classe `Arme`. A la création d'un orque, on pourra lui associer un ensemble `armes` de classe `EE`, les entiers représentant les indices dans le tableau `Arene.tabArmes`.

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

Placement dans l'arène

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.

Si on gère une position courante des orques, cela peut augmenter le réalisme des combats. Par exemple, tous les orques se déplacent entre chaque tour de bataille, et les orques à une certaine distance l'un de l'autre s'affrontent en duel...

4 Interface graphique

Dans les versions précédentes, on se contente de simples traces textuelles pour montrer l'évolution de la bataille : 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/` que l'on peut construire en dézippant le fichier d'archive `PROG1bis/UTILE/graphisme.zip` de l'ENT.