Licence MASS – INF F3 – TP n° 3

2012–2013

Le but de ce TP est d'écrire une classe, d'exposer un certain nombre de ses méthodes et de la faire intéragir avec le reste de l'application. Pour ce faire, vous allez créer une arène de combattants puis vous créerez une interface graphique capable de la manipuler.

1 Les combattants

Chaque combattant est unique. Il possède un nom, un nombre de points de vie et une force de frappe. Une jauge de santé peut aussi lui être associé pour représenter son état pendant un combat. Pour créer des combats avec un certain intérêt, les combattants ne peuvent pas dépasser une force de frappe de 8. Et la somme de leurs points de vie totaux et de leur force de frappe ne doit pas dépasser 20.

Tous les combattants sont capables de se battre contre les autres combattants. Mais celui qui est attaqué peut alors se défendre. Lorsqu'un combattant en attaque un autre, il lui retranche autant de points de vie que sa force de frappe. Si l'attaqué n'est pas mort, il se défend en ôtant des points de vie à l'attaquant à auteur de la moitié de sa force de frappe.

Pour faciliter son interaction avec le reste de l'application, un combattant est capable de donner son nom et possède une représentation sous forme de chaîne de caractères (méthode `toString()`) qui reprend les informations que vous jugerez intéressantes.

Écrivez une classe Combattant qui implémente ces contraintes.

2 L'arène

L'arène est l'endroit où les combattants s'affrontent en fonction de règles pré-établies. La règle la plus simple étant celle du « chacun pour soi ».

À sa création, l'arène est vide mais limitée en capacité. Seuls 8 combattants peuvent y entrer. Tant que des places sont disponibles, il est possible de créer de nouveaux combattants dans une arène. Il est aussi possible de retirer un combattant à tout moment.

À tout moment, il est possible de déclencher un combat dans une arène. Les combattants agissant alors en fonction des règles du combat jusqu'à ce qu'il n'en reste plus qu'un (ou un groupe).

Écrivez une classe Arene qui implémente ces contraintes. Vous devez créer une méthode qui lance un combat de type « chacun pour soi » mais vous pouvez en créer d'autres pour d'autres règles (en équipe, par exemple).

3 Les combats et les tests

Même si cela ne correspond pas à ce qui est demandé *in fine* dans ce TP, vous pouvez utiliser une méthode `main` pour tester vos combattants et vos arènes. Chaque `main` étant limité à ce que la classe connait du reste de l'application, faites attention à ce que vous tentez de tester et

respectez le principe d'encapsulation plutôt que de faire des modifications « juste pour que le `main` marche ».

Concernant les combats, vous pouvez définir des stratégies déterministes du genre « chaque combattant attaque toujours le suivant dans la liste » ou vous pouvez user du `Math.random()` à foison. À vous de voir ce qui vous convient le mieux.

4 L'interface graphique

Pour profiter au mieux de l'utilisation de l'aspect orienté-objet de Java, l'interface ne devra contenir, comme variable ajoutée manuellement, qu'une arène.

Pour vous en servir, vous utiliserez sa méthode de création de combattants et sa ou ses méthodes de déclenchement de combat que vous avez définis en section 2. Afin de simplifier la gestion de l'interface graphique, les différentes phases du combat seront affichées dans la console.

Un exemple d'interface est donné en annexe. Si vous voulez vous en inspirer et utiliser une zone qui va afficher les combattants créés au fur et à mesure de leur création, vous pouvez vous inspirer du code suivant à associer à l'`actionPerformed` du bouton « Ajouter » :

```
private void addPlayer(String name, String hp, String strength) {
  final int index = arena.addPlayer(name, hp, strength);
  final JButton button = new JButton(""+arena.getPlayer(index));
  getJPanel1().add(button);
  button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
      arena.removePlayer(index);
      getJPanel1().remove(button);
      getJPanel1().validate();
      getJPanel1().repaint();
    }
  });
  getJPanel1().validate();
  getJPanel1().repaint();
}
```

Cet extrait de code fait les suppositions suivantes :

- La classe Arene a été instanciée dans la variable arena et possède les méthodes addPlayer, getPlayer et removePlayer dont les signatures peuvent êtres déduites.
- Le panel dans lequel ajouter les boutons est accessible via la méthode getJPanel1(). Par défaut, cela signifie qu'il a été créé avec le nom jPanel1.

5 Et après

Une fois que l'interface graphique permet de créer des combattants et de lancer des combats, rajoutez les éléments suivants :

1. Permettre aux combattants de se soigner. Cette action remplace une attaque et restaure 4 points de vie.
2. Permettre aux combattants d'attaquer à 1.5× leur force habituelle lorsqu'ils n'ont plus qu'un point de vie.
3. Permettre de lancer un combat par équipe dans l'arène. Chaque équipe contient la moitiée des combattants de l'arène. La première moitié forme l'équipe une et la seconde moitié l'équipe deux.

4. Permettre à un combattant d'en défendre un autre. Cette action, utilisable lors d'un combat par équipe, permet de répartir les dégâts d'une attaque sur les deux défenseurs. Lors de la riposte, chacun des deux défenseurs inflige des dégats à l'attaquant.

A Exemple d'interface

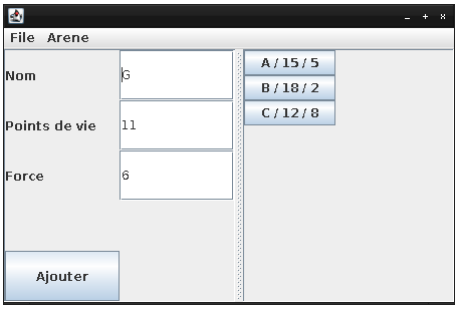

FIGURE 1 – Création des combattants

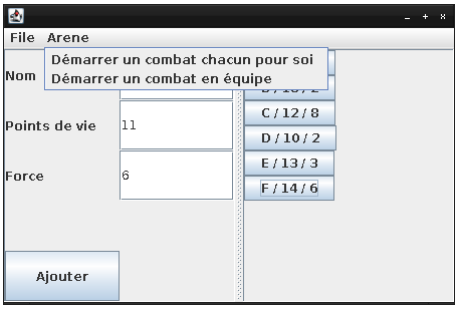

FIGURE 2 – Lancement d'un combat

B Exemple de combat

```
Les challengers sont :
A / 15 / 5
B / 18 / 2
C / 12 / 8
D / 10 / 2
E / 13 / 3
F / 14 / 6
null
null
D attaque C et lui inflige 2 blessures.
C se défend contre D et lui inflige 4 blessures.
C attaque A et lui inflige 8 blessures.
A se défend contre C et lui inflige 2 blessures.
E attaque D et lui inflige 3 blessures.
D se défend contre E et lui inflige 1 blessures.
B attaque C et lui inflige 2 blessures.
C se défend contre B et lui inflige 4 blessures.
F attaque C et lui inflige 6 blessures.
C est mort.
A attaque F et lui inflige 5 blessures.
F se défend contre A et lui inflige 3 blessures.
E attaque D et lui inflige 3 blessures.
D est mort.
A attaque F et lui inflige 5 blessures.
F se défend contre A et lui inflige 3 blessures.
F attaque E et lui inflige 6 blessures.
E se défend contre F et lui inflige 1 blessures.
B attaque F et lui inflige 2 blessures.
F se défend contre B et lui inflige 3 blessures.
A attaque B et lui inflige 5 blessures.
```

```
B se défend contre A et lui inflige 1 blessures.
A est mort.
F attaque E et lui inflige 6 blessures.
E est mort.
B attaque F et lui inflige 2 blessures.
F est mort.
B est le survivant.
```