Benjamin BEAUCAMP
Muruo WANG

TP6 – Programmation Objet

Introduction

Dans le TP précédent, nous avons amélioré la gestion des entités sur le plateau de jeu et permis à un joueur humain de créer un personnage . Dans ce TP, nous allons donner la possibilité au joueur humain de se déplacer et de combattre. Nous allons aussi implémenter un système de sauvegarde dans le jeu pour arrêter une partie puis la reprendre au même point. blabla

1) Retour sur le TP précédent

Dans le TP précédent, nous avons créé un attribut **listePositionsOccupees** dans **World** afin de connaître les positions inaccessibles lors d'un déplacement d'une créature. Cette implémentation avait le défaut de nous obliger à modifier la liste à chaque déplacement d'un personnage, ce qui n'était pas pratique. Nous avons choisi de remplacer cet attribut par une méthode **getPositionsOccupees()** qui renvoie la liste des positions occupées. Elle a le défaut d'itérer sur toutes les créatures et tous les personnages à chaque appel, alors qu'en général une seule modification est faite. Cependant, cela nous éviter à quel appel à **deplacer** de devoir rechercher l'ancienne position, de la supprimer et d'ajouter la nouvelle position. De plus, notre nombre de créatures n'est pas très grand donc la solution est acceptable. Comme nous allons seulement itérer sur cette liste (et pas faire de modifications), nous avons choisi d'utiliser une ArrayList au vu des résultats obtenus dans le TP4.

Nous avons aussi changé la signature de la fonction **deplacer**. Elle passe de `deplacer(int dx, int dy, LinkedList<Point2D> positionsOccupees, int dimension)` à `deplacer(Point2D nouvellePosition, ArrayList<Point2D> positionsOccupees, int dimension)`
Cela nous permet de déplacer plus facilement un joueur humain et nous permet d'utiliser directement les objets que nous avons créés plutôt que de passer par des intermédiaires (dx et dy).

Enfin, nous avons ajouté de l'aléatoire lors de l'initialisation des caractéristiques d'une créature. Les caractéristiques étaient auparavant des constantes (mais qui dépendaient quand même de la classe, par exemple un guerrier avait initialement plus de points de vie qu'un archer), mais nous avons maintenant ajouté une partie aléatoire à ces valeurs.

2) Mise à niveau

a) Tour de jeu d'un joueur humain

Les tours de jeu de World se basent principalement sur les tours de jeu des joueurs humains. Nous avons donc créé une méthode **tourDeJeuJoueurHumain(Joueur j)** qui permet de faire jouer un tour à un joueur humain. Il a la possibilité de se déplacer ou de combattre.

Pour écrire cette méthode, nous avons eu besoin des méthodes suivantes :

- boolean **positionInBounds**(Point2D p)

Cette méthode indique si p est à une position valide sur le plateau, c'est à dire que x et y sont entre 0 et la dimension du monde - 1.

- ArrayList<Point2D> **deplacementsPossibles**(Point2D p)

Cette méthode renvoie la liste des positions atteignables à partir du point p. Les positions considérées sont toutes les cases adjacentes à p. Elle se base sur le résultat de l'appel à **getPositionsOccupees()**.

- ArrayList<Creature> **combatsPossibles**(Joueur j)

Comme **deplacementsPossibles**, cette méthode renvoie la liste des créatures à proximité que le joueur peut attaquer.

Illustration du combat au corps à corps :

```
Test de la méthode tourDeJeuJoueurHumain
Test pour un combattant au corps à corps :
Joueur 0, entrez vos choix.
Quel classe désirez-vous jouer ?
Choix possibles : Archer, Guerrier, Mage
guerrier
Entrez un nom pour votre personnage :
grosBil
Nom : grosBil
Points de vie : 120
Position : [42 ; 7]
Points d'attaque : 20
Pourcentages d'attaque : 70
Points de parade : 5
Pourcentages de parade : 60
Points de mana : 5
Dégâts de magie : 10
Pourcentage de magie : 30
Pourcentage de résistance à la magie : 30
Distance d'attaque maximale : 1
Voulez-vous vous déplacer (1) ou combattre (2) ?
Entrez le numéro correspondant à votre choix :
2
Les créatures à proximité :
0 : [42 ; 8] Points de vie : 100
Choisissez une créature que vous souhaitez attaquer (0 à 0) :
0
Jet d'attaque : 31. Pourcentage d'attaque : 70
Attaque au corps à corps réussie
Jet de parade : 78. Pourcentage de parade : 60
Parade échouée !
20 dégâts sont infligés
```

Illustration du déplacement : grosBil est en (42, 7) (voir plus haut) et choisis de se déplacer en (41, 7).

```
Voulez-vous vous déplacer (1) ou combattre (2) ?
Entrez le numéro correspondant à votre choix :
1
Positions possibles :[[43 ; 7], [43 ; 8], [41 ; 8], [41 ; 7], [42 ; 6], [43 ; 6]]
Choisissez une position que vous souhaitez atteindre (0 à 5) :
3
Nom : grosBil
Points de vie : 120
Position : [41 ; 7]
Points d'attaque : 20
Pourcentages d'attaque : 70
Points de parade : 5
Pourcentages de parade : 60
Points de mana : 5
Dégâts de magie : 10
Pourcentage de magie : 30
Pourcentage de résistance à la magie : 30
Distance d'attaque maximale : 1
```

Illustration du combat à distance :

```
Test pour un combattant à distance :
Joueur 0, entrez vos choix.
Quel classe désirez-vous jouer ?
Choix possibles : Archer, Guerrier, Mage
Archer
Entrez un nom pour votre personnage :
robin
Nom : robin
Points de vie : 100
Position : [22 ; 35]
Points d'attaque : 15
Pourcentages d'attaque : 70
Points de parade : 5
Pourcentages de parade : 60
Points de mana : 5
Dégâts de magie : 10
Pourcentage de magie : 30
Pourcentage de résistance à la magie : 30
Distance d'attaque maximale : 5
Nombre de flèches  5
Voulez-vous vous déplacer (1) ou combattre (2) ?
Entrez le numéro correspondant à votre choix :
2
Les créatures à proximité :
0 : [22 ; 38] Points de vie : 100
1 : [20 ; 33] Points de vie : 100
2 : [18 ; 37] Points de vie : 100
Choisissez une créature que vous souhaitez attaquer (0 à 2) :
1
Jet d'attaque : 20. Pourcentage d'attaque : 70
Attaque à distance réussie
Jet de parade : 5. Pourcentage de parade : 60
Parade réussie !
10 dégâts sont infligés
```

b) Tour de jeu dans World

Dans **World**, un tour de jeu consiste à faire jouer un tour à chaque joueur puis à mettre à jour la nourriture de chaque personnage du jeu.

c) Points restant à améliorer

- Nous n'avons pas encore eu le temps d'implémenter le ramassage des potions par un personnage.
- La génération des créatures n'est pas très pratique (actuellement : une boucle par type d'entité)

3) Enregistrement/chargement de sauvegardes en mode texte

Nous avons utilisé une convention un peu différente de celle proposée dans l'énoncé. La première ligne correspond à la taille du monde (supposé carré) et les positions des entités sont de la forme "x,y" afin de récupérer un Point2D en un seul token.

a) Chargement d'une partie

Dans la classe **ChargementPartie**, on a défini un attribuer BufferedReader fichier pour être capable de lire les fichiers plusieurs de fois.
Pour entrer les données on a passé dans Buffer, on a choisi d'écrire une méthode statique par classe qui renvoie un objet de cette même classe. Cela nous permet d'appeler le constructeur avec tous les paramètres grâce aux valeurs lues avec le tokenizer. Pour appeler cette méthode statique dans le chargement de la partie, on utilise la réflexivité : on commence par lire le premier mot de la ligne qui est le nom de la classe. Puis on crée un objet de cette classe et on cherche la méthode **fromString(String params)**. Enfin, on invoque cette méthode avec en paramètres la ligne que l'on vient de lire, et suivant le type de l'objet créé on l'ajoute à la liste de créatures, d'objets ou de joueurs.
Nous avons choisi cette méthode parce qu'on n'a pas besoin de réécrire les attribuer dans le constructeur pour chaque sous-classe. De plus, si on crée une nouvelle classe, on a juste à implémenter fromString et toString (pour la sauvegarde d'une partie). Ainsi, toutes les méthodes sont centralisées dans une classe, au lieu d'aller changer les méthodes de chargement et de sauvegarde.

b) Sauvegarde d'une partie

Nous avons défini la méthode **toString()** dans chaque classe pour écrire les données de chaque **Creature** et **Objet** dans un fichier. Il nous suffit ensuite d'itérer sur les listes de créatures, d'objets et de joueurs et de les écrire dans le fichier de sauvegarde.

c) Illustration de l'enregistrement et de la sauvegarde

Chargement

On a le fichier de sauvegarde(en haut) et on obtient le monde (en bas) :

```
tailleMonde 40
Mage merlin 50 50 20 25 85 80 2 20 7 2 15,2
Guerrier grosBill 250 0 80 60 0 10 0 1 5 8 12,12
Archer 75 robin 0 50 60 0 25 10 0 10 5 20 11,11
Paysan peon 25 0 0 30 0 0 1 0 1 5 3,6
Loup 30 30 30 20  5 12,20
Lapin 30 20 20 40 10 23,23
NuageToxique nuage 5,5 5
Soin 100 soin 20,12
Mana 100 mana 1,19
Joueur Guerrier bob 150 0 90 50 0 10 20 0 1 5 12,17
```

```
Taille du monde : 40

Nombre de créatures  : 6
Première créature de la liste :
Nom : merlin
Points de vie : 80
Position : 15,2
Points d'attaque : 7
Pourcentages d'attaque : 2
Points de parade : 2
Pourcentages de parade : 20
Points de mana : 50
Dégâts de magie : 25
Pourcentage de magie : 50
Pourcentage de résistance à la magie : 20
Distance d'attaque maximale : 85

Nombre de joueurs : 1
Premier joueur de la liste :
Le joueur possède le personnage suivant :
Nom : bob
Points de vie : 10
Position : 12,17
Points d'attaque : 1
Pourcentages d'attaque : 20
Points de parade : 5
Pourcentages de parade : 0
Points de mana : 150
Dégâts de magie : 50
Pourcentage de magie : 0
Pourcentage de résistance à la magie : 90
Distance d'attaque maximale : 0

Nombre d'objets : 3
Premier objet de la liste :
nuage se trouve en 5,5
```

Sauvegarde

A partir du monde (à gauche), on obtient le fichier de sauvegarde (à droite) :

```
Taille du monde : 50

Nombre de créatures  : 16
Première créature de la liste :
Nom : Archer
Points de vie : 113
Position : 8,28
Points d'attaque : 16
Pourcentages d'attaque : 61
Points de parade : 8
Pourcentages de parade : 64
Points de mana : 3
Dégâts de magie : 10
Pourcentage de magie : 29
Pourcentage de résistance à la magie : 31
Distance d'attaque maximale : 5
Nombre de flèches  5

Nombre de joueurs : 1
Premier joueur de la liste :
Le joueur possède le personnage suivant :
Nom : robin
Points de vie : 100
Position : 43,27
Points d'attaque : 19
Pourcentages d'attaque : 68
Points de parade : 11
Pourcentages de parade : 61
Points de mana : 3
Dégâts de magie : 10
Pourcentage de magie : 30
Pourcentage de résistance à la magie : 31
Distance d'attaque maximale : 5
Nombre de flèches  5

Nombre d'objets : 10
Premier objet de la liste :
Soin se trouve en 11,34
Elle rend 20 points de vie
-----------------------------------------------
```

```
tailleMonde 50
Archer 5 Archer 3 29 31 10 5 113 61 64 16 8 8,28
Archer 5 Archer 4 25 38 13 5 124 60 59 19 8 49,11
Guerrier Guerrier 4 29 36 11 1 138 60 57 27 8 6,6
Guerrier Guerrier 5 25 30 13 1 136 60 60 30 10 48,8
Paysan Paysan 4 29 31 13 1 107 63 59 16 9 6,15
Paysan Paysan 4 26 34 14 1 112 68 59 18 8 5,31
Paysan Paysan 4 29 32 13 1 101 64 63 15 12 14,28
Paysan Paysan 3 25 31 10 1 114 61 56 17 8 13,4
Mage Mage 7 77 32 28 3 103 67 55 18 6 2,36
Mage Mage 7 78 30 29 3 116 65 63 19 7 2,25
Mage Mage 7 81 34 32 3 124 67 59 17 7 26,30
Mage Mage 7 84 36 32 3 112 61 64 16 6 12,45
Loup 54 45 50 18 6 48,29
Loup 41 53 47 16 5 43,39
Loup 42 58 52 18 4 25,9
Loup 40 49 47 16 5 25,19
Soin 20 Soin 11,34
Soin 20 Soin 11,26
Soin 20 Soin 27,20
Soin 20 Soin 24,35
Mana 5 Mana 13,26
NuageToxique NuageToxique 6,25 4
NuageToxique NuageToxique 43,46 4
NuageToxique NuageToxique 23,21 4
DegAttMalus 2 3 Objet 37,15
DegAttMalus 2 3 Objet 38,16
Joueur Archer 5 robin 3 30 31 10 5 100 68 61 19 11 43,27
```

Sauvegarde et chargement :

Au prochain TP nous illustrerons le bon fonctionnement des deux méthodes dans le cas où on crée un monde, on joue quelques tours puis on sauvegarde la partie et on la reprend après à partir du fichier de sauvegarde.

Conclusion

Dans ce TP, nous utilisons Git pour faciliter l'échange du code(avant nous avons utilisé Google Drive pour partager les codes mais ce n'est pas pratique). Nous avons fini l'implémentation des fonctions permettant à un joueur humain de jouer et nous avons implémenté le chargement et la sauvegarde de parties. Pour le moment, la sauvegarde de nourriture n'est pas encore implémentée (car nous n'avons pas encore permis à un personnage d'en ramasser). Une difficulté que nous avons rencontré est la création d'une classe à partir de son nom en chaîne de caractères. Au début, nous avions juste déclaré le nom de classe alors qu'il faut aussi ajouter déclarer le package pour le trouver. Le code est

le suivant:

```
Class classeElementJeu = Class.forName("org.centrale.projet.objet." + premierMot);
```