

Antoine de ROQUEMAUREL
Dario OLIBET
Groupe 1.1

Projet de programmation

Partie Java

Avant-propos

Ce dossier comporte l'organisation et les étapes de développement d'un projet d'Arènes en Java.

Il à été conçu par Antoine de ROQUEMAUREL et Dario OLIBET dans le cadre du module *Projet de programmation* de la L3 Informatique de l'université Toulouse III – Paul Sabatier.

Contenu de l'archive du projet

L'archive que vous avez reçus était organisée comme ceci :

rapport.pdf Le présent rapport que vous êtes en train de lire

v1Arene/ Contient le dossier avec la première version de l'application : notre propre arènes avec les règles que nous avons implémentées.

v1Arene/doc/ Contient la documentation de la première version du projet. Celle-ci à été générée à l'aide de *Doxygen*, elle est disponible en HTML ou en PDF (générée avec \LaTeX).

L'utilisation de *Doxygen* contrairement à *Javadoc* permet d'avoir une génération en PDF, mais également d'avoir l'apparition des diagrammes de classes ce qui facilite la compréhension du problème.

V2Commun/ Contient le dossier avec la version contenant notre stratégie pour l'arène commune distante.

Le projet à été développé en Java6.

Table des matières

1	Organisation du travail d'équipe	5
1.1	Un outil de gestion de projet : Redmine	5
1.2	Un logiciel de versionnement : Git	6
2	Première version : local	7
2.1	Règles du jeu	7
2.1.1	Combattant	7
2.1.2	Caractéristiques	7
2.1.3	Combat	7
2.1.4	Équipements	8
2.2	Conception	8
3	Seconde Version : commun	10
3.1	Stratégie	10
4	Les difficultés rencontrés	11
A	Liste des tâches	12
B	Table des figures	12
C	Liste des tableaux	13

1

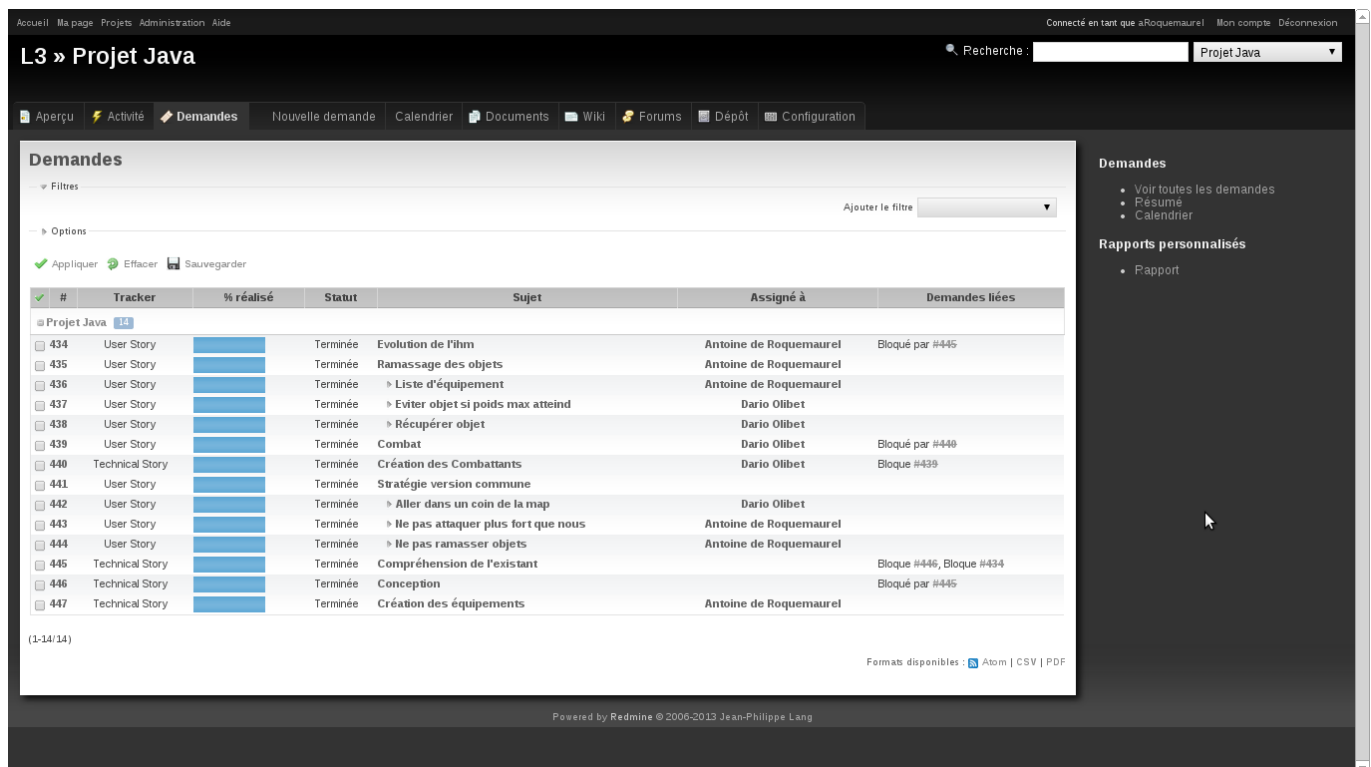
Organisation du travail d'équipe

Pour ce projet, nous étions deux à travailler dessus, ainsi nous avons utilisé plusieurs techniques afin de se coordonner et de limiter les problèmes. Ceci n'est pas notre premier projet ensemble, notre travail en fut simplifié.

1.1 Un outil de gestion de projet : Redmine

Pour le projet, nous avons utiliser *Redmine*, une plateforme web de gestion de projet. Elle nous a permis de simplifier le travail, et de ne rien oublier.

En effet, nous pouvons créer des tâches, signaler qu'elles sont en cours/terminées/en tests, leur donner des dates limites, les affecter à une personne etc... Ainsi lorsque l'un de nous commençait une tâche, il le signalait sur le *redmine*, ce qui permettait de tenir au courant son binôme de ses actions et de l'avancée du projet.



#	Tracker	% réalisé	Statut	Sujet	Assigné à	Demandes liées
Projet Java 14						
434	User Story		Terminée	Evolution de l'hm	Antoine de Roquemaurel	Bloqué par #445
435	User Story		Terminée	Ramassage des objets	Antoine de Roquemaurel	
436	User Story		Terminée	» Liste d'équipement	Antoine de Roquemaurel	
437	User Story		Terminée	» Eviter objet si poids max atteint	Dario Olibet	
438	User Story		Terminée	» Récupérer objet	Dario Olibet	
439	User Story		Terminée	Combat	Dario Olibet	Bloqué par #440
440	Technical Story		Terminée	Création des Combattants	Dario Olibet	Bloque #439
441	User Story		Terminée	Stratégie version commune		
442	User Story		Terminée	» Aller dans un coin de la map	Dario Olibet	
443	User Story		Terminée	» Ne pas attaquer plus fort que nous	Antoine de Roquemaurel	
444	User Story		Terminée	» Ne pas ramasser objets	Antoine de Roquemaurel	
445	Technical Story		Terminée	Compréhension de l'existant		Bloque #446, Bloque #434
446	Technical Story		Terminée	Conception		Bloqué par #445
447	Technical Story		Terminée	Création des équipements	Antoine de Roquemaurel	

FIGURE 1.1 – Liste des tâches sous Redmine

1.2 Un logiciel de versionnement : Git

Afin de limiter les problèmes du travail collaboratif, nous avons utilisé un logiciel de versionnement Git. Il a deux intérêts, tout d'abord, nous pouvons travailler à deux en parallèle sur le projet sans se soucier de fusionner notre travail¹.

D'autre part, tous les logs étant enregistrés, nous pouvons savoir qui à fait quoi et quel jour, cela permet de voir également l'avancée du projet.

Enfin, toutes les modification sont stockées sur le serveur, ainsi en cas de problème, il est très facile de revenir à la version précédente ou même de comparer deux versions afin de voir les changements et de comprendre rapidement pourquoi une fonctionnalité à régressé.

1. À condition de ne pas travailler sur deux lignes de code identiques

2

Première version : local

2.1 Règles du jeu

Nous avons décidé d'implanter des règles basé sur d'une part les combattants et d'autre part les équipements. Afin de gagner une partie le combattant doit être le dernier en vie dans l'arène.

2.1.1 Combattant

Chaque personnage dispose de caractéristiques, ces caractéristiques doivent respecter une règle d'équilibre. Si cette règle n'est pas respectée, le joueur n'est pas accepté sur le serveur : $\frac{vie}{10} + Attaque + Défense + Vitesse = 10$.

2.1.2 Caractéristiques

Nombre de point de vie Chaque personnage dispose d'un certain nombre de pv¹.

Attaque/Force La force d'un personnage

Vitesse/esquive Chance pour un personnage d'éviter une attaque.

Nombre d'objets Nombre d'objets qu'un personnage peut porter.

Le nombre d'objets qu'un personnage peut porter est choisi arbitrairement et compris entre 1 et 5 et ne rentre pas dans la règle d'équilibre.

		Vie	Attaque	Défense	Vitesse	Nb objet	Description
Capitaine	Luffy	30	3	2	2	3	Équilibré
Barde	Brook	10	2	3	4	2	Peu résistant mais ayant une bonne défense et une bonne fuite
Spadassin	Zorro	20	4	1	3	1	Préfère un combat rapide tout en essayant d'éviter un maximum d'attaques
Cyborg	Franky	20	3	4	1	4	Défensif et fort, mais très lent
Mascotte	Chopper	20	1	3	4	5	Bonne défense et bonne fuite.

TABLE 2.1 – Les différentes classes de personnage créées

2.1.3 Combat

Lorsqu'un combat à lieu, deux règles sont mises en place.

1. Point de vie

La première est basée sur l'esquive : la vitesse multipliée par 10 est le pourcentage de chances d'esquiver une attaque, lorsqu'une attaque est esquivée, aucun dégât n'est subi.

La deuxième est basée sur le ratio entre l'attaque et la défense, si l'attaque de l'attaquant est supérieure à la défense du défenseur nous avons : $attaque - defense = dégâts$ qui sont infligés au défenseur. Dans le cas où son attaque est inférieure à la défense, un seul point de dégât est causé.

2.1.4 Équipements

Un équipement peut donner un ou plusieurs bonus d'attaque, défense et vitesse. Tout équipement à une durée de vie qui correspond aux nombre de tours qu'un personnage l'utilise, ce nombre de tours est choisis arbitrairement entre 1 et 5².

Nos équipements sont tous équilibrés. Ils donnent un bonus total de 2 et un malus total de 1, donc $bonusAttaque + bonusDefense + bonusVitesse = 1$. Chaque personnage peut récupérer n'importe quel équipement mais chacun d'eux à une préférence selon sa classe afin d'être le plus équilibré possible.

		Attaque	Défense	Vitesse	Durée	Description
Chapeau de paille	Luffy	1	-1	1	2	
Canne	Brook	2	-1	0	1	
Sabre	Zorro	2	0	-1	4	
Plastron	Franky	0	2	-1	5	
Bottes	Chopper	-1	0	2	3	

TABLE 2.2 – Les différents équipements créés

2.2 Conception

Afin de concevoir au mieux l'application, nous avons tout d'abord étudié le code source existant. Une fois que nous avons compris son fonctionnement, nous avons définis nos équipements et nos personnes en fonction d'un **Element**.

Ci-dessous le diagramme de classe :

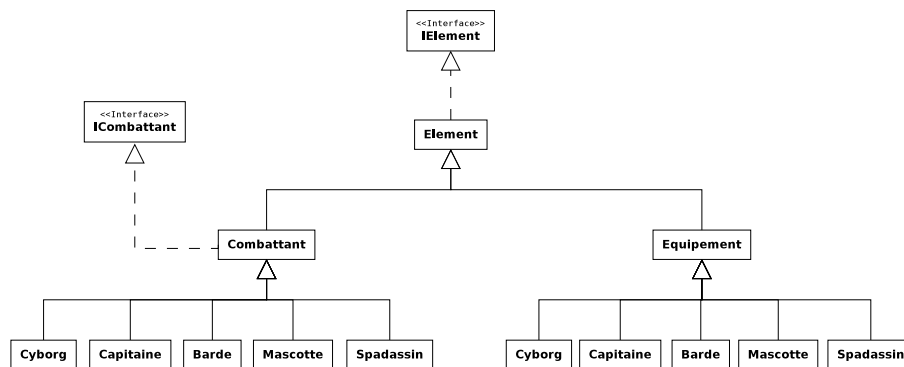


FIGURE 2.1 – Diagramme de classe des Element

Une fois ces classes créés, il fallait implémenter les règles du jeu, pour cela nous avons modifiés le serveur afin que les personnages puissent :

2. Nous avons essayer d'avoir une certaine logique, en fonction du type de l'équipement. Par exemple un plastron aura une plus grande durée qu'une canne

- Ramasser des objets, s'il en a la possibilité (vérification du poids)
- Implémentation des duels
- Vérification du respect de la règle d'équilibre
- Ajout du port d'objets dans le calculs des statistiques

Nous avons également modifié un peu l'IHM afin de distinguer la différence entre un objet et un personnage : les objets sont représenté par un carré alors que les combattants par des ronds.

3

Seconde Version : commun

3.1 Stratégie

Notre personnage, Franky a pour caractéristiques une attaque de 65 et une défense de 35. Il ne possède donc aucune esquive et aucune possibilité d'avoir un équipement.

Nous avons adapté notre objet ainsi que notre stratégie par rapport à ces caractéristiques. L'objet est une bombe, ayant un poids léger pour contrer les personnages ne prenant aucun équipement faisant un poids de 0. Ses bonus/-malus sont : 2 de force, 0 de défense, 0 d'esquive et un malus de 100 en vie. Ayant un bonus de 2 en force son poids était donc de $2 \times \frac{3}{4} = 1.5$ arrondi à 1. Nous nous débarrassons ainsi d'un possible ennemi, qui, après une certaine durée, pourrait avoir des statistiques pouvant nous battre.

La stratégie de notre combattant est, dans une première partie, de se diriger dans le coin le plus proche de l'arène et rester dans un coin de 30×30 pendant 2 minutes. Une fois que ce temps est écoulé notre personnage décide d'errer dans la totalité de l'arène en cherchant le combat, maintenant que les personnes se sont entretuées. Si il voit un adversaire ayant plus de vie que lui, il fuit en espérant que quelqu'un le rendra plus faible pour pouvoir gagner plus tard.

4

Les difficultés rencontrés

Notre difficulté principale fut la compréhension du système de **Remote**. En effet une **Arene** ne possédait qu'un entier comme guise de référence, la question étant comment pouvoir récupérer un **Element**, ou même mieux un **Combattant** à partir de cet entier. C'est en cherchant dans le projet que nous avons trouvé la solution : La récupération de la **Remote**, puis le cast vers une **Console**, récupération d'un **Element** avec la fonction `getElement` pour finalement caster de nouveau si le besoin s'en faisant sentir.

A

Liste des tâches

Projet Java - Demandes

#	Tracker	% réalisé	Statut	Sujet	Assigné à	Demandes liées
Projet Java (14)						
434	User Story	100	Terminée	Evolution de l'ihm	Antoine de Roquemaurel	Bloqué par #445
435	User Story	100	Terminée	Ramassage des objets	Antoine de Roquemaurel	
436	User Story	100	Terminée	Liste d'équipement	Antoine de Roquemaurel	
437	User Story	100	Terminée	Eviter objet si poids max atteint	Dario Olibet	
438	User Story	100	Terminée	Récupérer objet	Dario Olibet	
439	User Story	100	Terminée	Combat	Dario Olibet	Bloqué par #440
440	Technical Story	100	Terminée	Création des Combattants	Dario Olibet	Bloque #439
441	User Story	100	Terminée	Stratégie version commune		
442	User Story	100	Terminée	Aller dans un coin de la map	Dario Olibet	
443	User Story	100	Terminée	Ne pas attaquer plus fort que nous	Antoine de Roquemaurel	
444	User Story	100	Terminée	Ne pas ramasser objets	Antoine de Roquemaurel	
445	Technical Story	100	Terminée	Compréhension de l'existant		Bloque #446, Bloque #434
446	Technical Story	100	Terminée	Conception		Bloqué par #445
447	Technical Story	100	Terminée	Création des équipements	Antoine de Roquemaurel	

FIGURE A.1 – Liste des tâches Redmine

B

Table des figures

1.1	Liste des tâches sous Redmine	5
2.1	Diagramme de classe des <code>Element</code>	8
A.1	Liste des tâches Redmine	12

C

Liste des tableaux

2.1	Les différentes classes de personnage créées	7
2.2	Les différents équipements créés	8