Rapport de projet

Application d'aide au déroulement d'une partie
De jeu de rôle papier.

Alexandre Fray *Alexia Huc-Lhuillery* *Basile Gros*
Hamza Hathoute *Kamal Hammi* *Léane Hauteville*
Lucas Goussard

Groupe GH-1

Département Sciences du Numérique – Première Année

2020-2021

Sommaire

I - Rappel du sujet

Nous avons décidé de traiter pour ce projet la conception d'une interface utile aux jeux de rôle. Cette interface permet pour un maître du jeu de créer et de modifier les fiches de chacun des joueurs, ainsi que d'enregistrer une fiche d'un personnage sous la forme d'une image, et de sauvegarder les fiches pour reprendre une partie plus tard.

II - Fonctionnalités implantés

1 - Interfaces maître du jeu et fiche personnage

A l'aide de la bibliothèque Java Swing, nous avons réalisé trois interfaces graphiques. L'une permet au maître du jeu de créer les différentes fiches de personnages avec leurs différentes caractéristiques (nom, rôle, capacité, inventaire, vie). La deuxième interface concerne la fiche personnage. Nous avons mis en place une image du personnage par défaut lorsqu'une fiche est créé que le joueur peut modifier et également imprimer. La dernière interface permet d'utiliser un dé dont on peut choisir le nombre de faces. La structure de ces interfaces a été travaillée durant les trois itérations pour les interfaces du maître du jeu et de la fiche personnage, et à l'itération 3 pour celle du dé.

2 - Boutons de modification et de validation des fiches

Les fiches peuvent être modifiées grâce à des boutons et des zones de texte.

Sur l'interface Maître du Jeu, il est possible de rentrer un nom et un rôle pour chaque fiche avec une zone de texte. La vie peut être modifiée en changeant la valeur prise par la barre de progression (itération 1).

Une capacité peut être ajoutée en notant son nom dans la zone de texte du cadre capacité puis en modifiant ensuite sa valeur après avoir confirmé l'ajout. Pour l'inventaire, l'ajout d'un élément se fait

de la même manière que pour une capacité et une fois un élément ajouté sa valeur est modifiable avec des boutons + et - qui permettent d'ajouter ou de retirer une unité de l'élément (itération 2). De plus, il est possible d'ajouter ou de supprimer une statistique pour un joueur, puis d'y ajouter des données avec des valeurs correspondantes de la même manière que pour les capacités. Le bouton sauvegarder permet d'enregistrer les modifications apportées sur les données du joueur, et le bouton réinitialiser réinitialise ces données (itération 3).

Le bouton imprimer fiche permet enfin de créer la fiche du personnage, les valeurs des éléments composants la fiche ne sont alors plus modifiable. Le joueur a aussi la possibilité d'imprimer sa fiche sous forme d'une image jpeg (itération 1).

Il est possible pour le Maître du Jeu de sauvegarder le jeu en cours grâce au bouton prévu à cet effet. Il dispose d'un menu principal qui lui permet d'ajouter ou de supprimer un jouer, d'ouvrir la fiche permettant de modifier les données d'un des joueurs inscrits dans la liste, ou d'afficher la fiche d'un joueur. (itération 3).

Le dé dispose aussi d'un bouton afin de le lancer et d'une zone de texte pour modifier le nombre de faces que l'on souhaite sur le dé (itération 3).

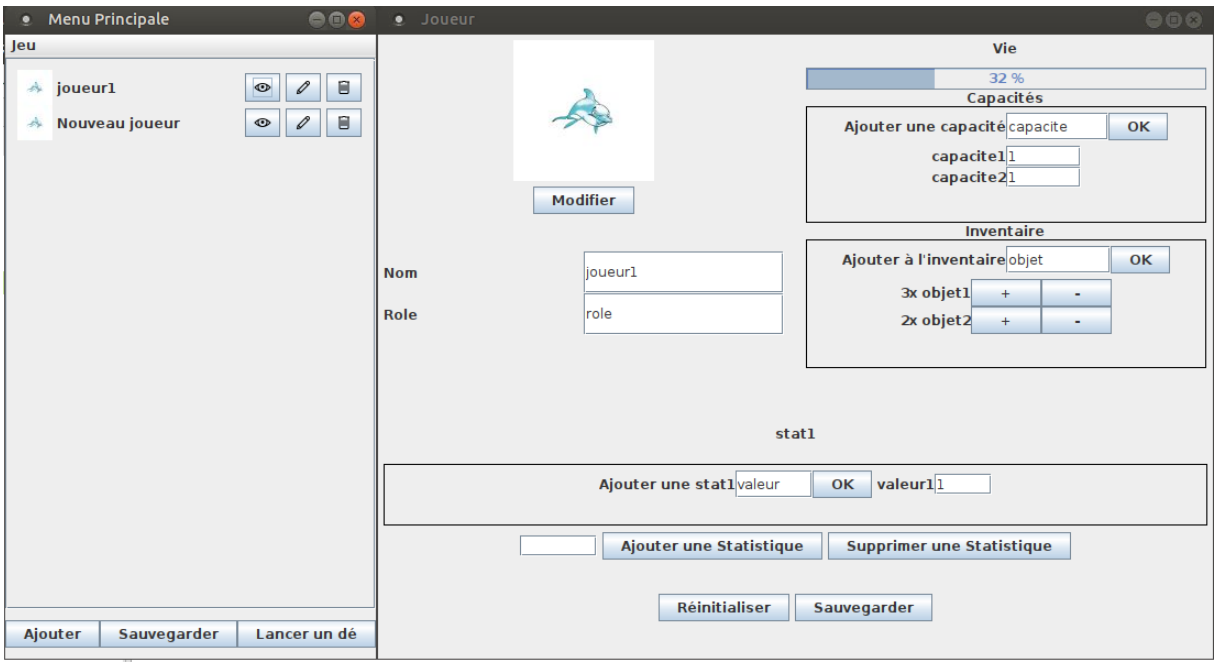

Figure 1 – Interface Maître du jeu

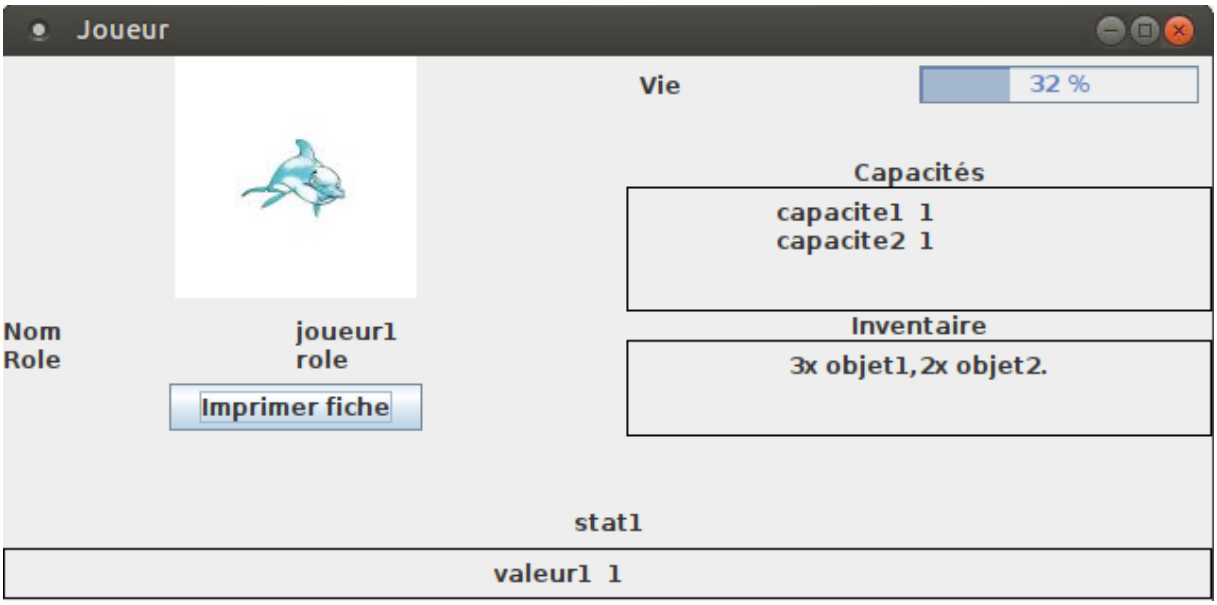

Figure 2 – Interface fiche personnage

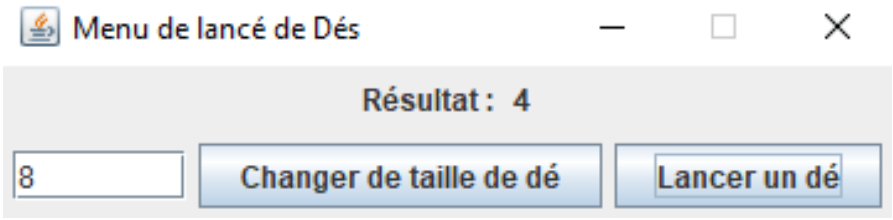

Figure 3 – Interface dé

Un entier est demandé obligatoirement pour la valeur d'une capacité, et une fiche personnage doit obligatoirement avoir un nom. Si une valeur incorrecte est rentrée pour les capacités, le programme affiche une fenêtre avec le message d'erreur.

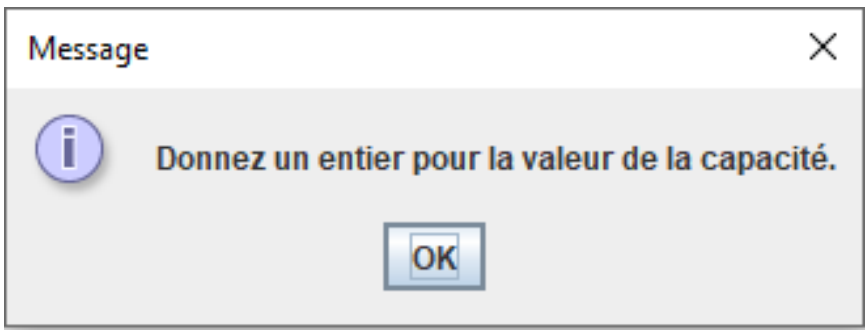

Figure 4 – Message d'erreur

3 - Test et compilation

Un paquetage composé de test JUnit permet de tester les fonctionnalités implantées pour les capacités, l'inventaire et la fiche personnage (itération 2).

Afin de compiler plus facilement les programmes pour une première utilisation, il est mis à disposition un fichier Makefile, qui permet aussi de créer une archive du dossier (itération 3).

III - Découpage de l'application

L'application est décomposée en 6 paquetages, et un dossier de ressources pour les images ajoutées aux interfaces.

Le paquetage **Jeu** contient trois autres paquetages permettant la modélisation des objets présents dans les interfaces du maître du jeu et des fiches personnage, les fichiers Swing pour la création des interfaces des fiches personnages, du maître du jeu et du menu de sélection, un fichier utilisé pour redimensionner les images télécharger pour ajouter une photo à une fiche, et le fichier exécutable de l'application.

Dans le paquetage **Jeu.Interfaces** on retrouve les interfaces utilisées pour les fiches personnages et maître du jeu, qui permettent de créer l'inventaire, les capacités et les statistiques liées aux personnages.

Le paquetage **Jeu.Modeles** contient les fichiers d'implantation des interfaces précédentes, ainsi que les modèles des fiches maître du jeu et personnage.

Le paquetage **Jeu.Vues** contient les vues des menus utilisés, et de la vue du bouton et de la zone de texte permettant de rajouter une capacité ou un élément à l'inventaire.

Dans le paquetage **De** sont regroupés une interface utilisée pour créer un dé, le modèle, le fichier Swing de l'interface du dé.

Le dernier paquetage **Test** regroupe les tests JUnit faits pour tester les fonctionnalités utilisées sur les capacités, l'inventaire et les fiches personnages.

IV - Architecture et diagramme UML

Le diagramme UML suivant présente l'architecture générale du programme, avec les différentes classes utilisées dans ce projet ainsi que leurs relations.

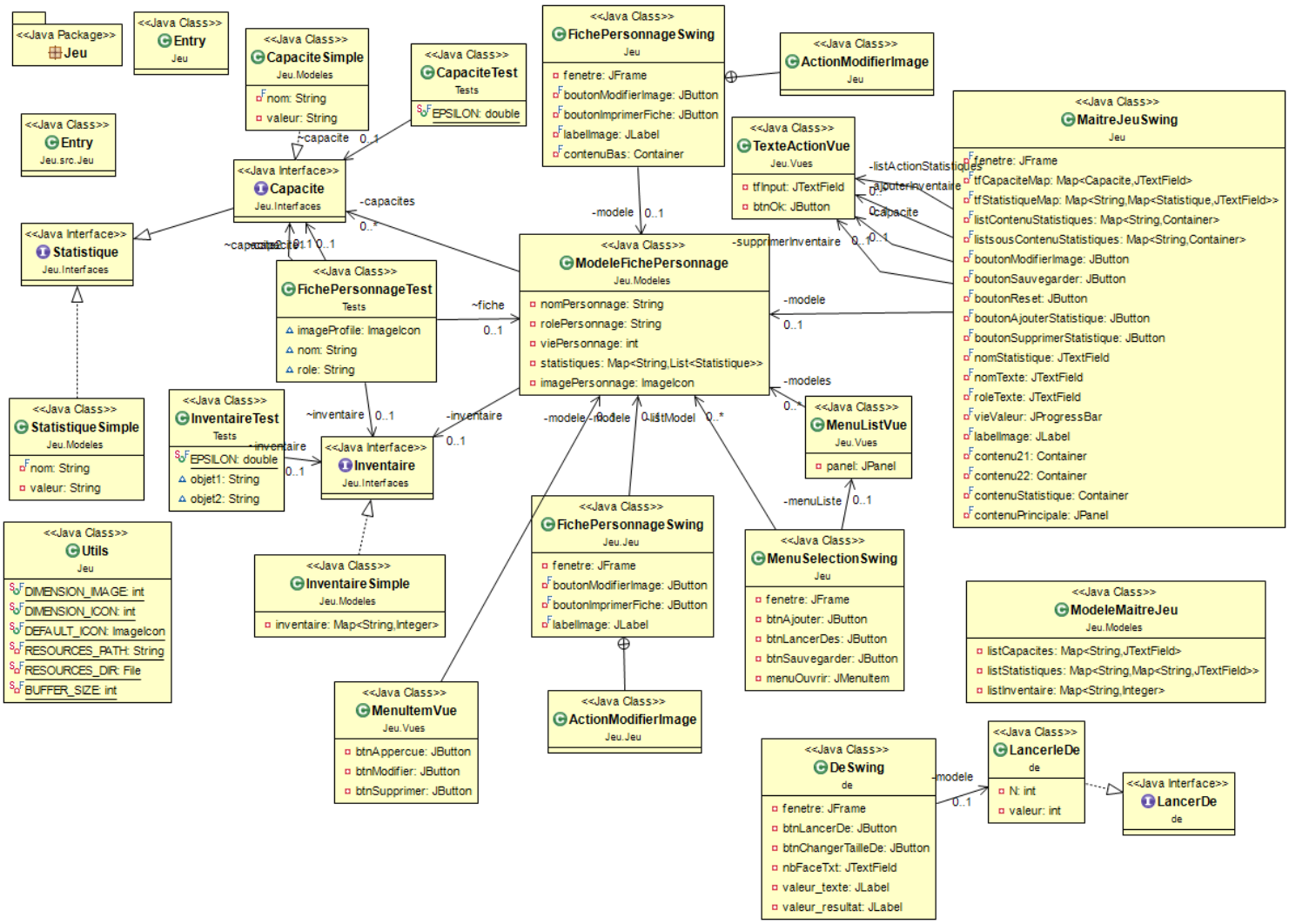

Figure 5 – Diagramme UML

La classe MaitreJeuSwing constitue la classe majeure de notre programme. Elle comporte l'ensemble des composants principaux et nécessaires pour faire fonctionner l'interface swing. Cette dernière s'appuie sur deux classes. L'une est le modèle pour le maître du jeu et l'autre concerne la FichePersonnage, puisque chacun possède des droits spécifiques. En effet, un personnage ne peut pas modifier son inventaire ou s'ajouter des capacités. Pour la classe Utils, elle contient l'ensemble des fonctions nécessaires pour gérer une carte au

format jpg, notamment pour sauvegarder une carte. Le maître du jeu est le seul à pouvoir modifier les valeurs inscrites sur les fiches, il dispose aussi de la liste de tous les joueurs présents. Les classes principales de notre application sont donc celles qui sont liées au maître du jeu, elles permettent la modification des autres objets disponibles notamment les composantes des fiches personnages, comme l'inventaire ou les capacités. Les interfaces utilisées dans le paquetage Jeu pour les capacités, l'inventaire et les statistiques ont été créées afin de faciliter un hypothétique développement futur de l'application.

V - Conception de l'application

1 - Première itération

Lors de la première itération, nous nous sommes concentrés sur la création d'une seule fiche de personnage fonctionnelle. Celle-ci répondait aux User stories :

- En tant que joueur de jeu de rôle, j'ai besoin d'avoir un inventaire modifiable.
  (Valeur d'effort : 25, valeur métier : excitant pour l'utilisateur, Must be done)
- En tant que joueur de jeu de rôle, j'ai besoin d'avoir une liste de capacités, composées d'un nom et d'une valeur, modifiable.
  (Valeur d'effort : 25, valeur métier : excitant pour l'utilisateur, Must be done)
- En tant que joueur de jeu de rôle, je souhaite pouvoir modifier sur ma fiche de personnage ma vie de façon dynamique.
  (Valeur d'effort : 5, valeur métier : intéressant pour l'utilisateur, Should be done)
- En tant que joueur de jeu de rôle, je souhaite pouvoir ajouter et modifier une image de profil à ma fiche de personnage.
  (Valeur d'effort : 15, valeur métier : intéressant pour l'utilisateur, Should be done)

- En tant que joueur de jeu de rôle, je souhaite à la fin de ma partie conserver une image de ma fiche de personnage que jeu puisse imprimer.
  (Valeur d'effort : 35, valeur métier : intéressant pour l'utilisateur, Could be done)

Nous avons commis l'erreur dans la première itération d'avoir davantage à l'esprit ce que nous ferions après avoir répondu à ces user stories, c'est pour cela que nous avons créé deux interfaces : l'une où tout est modifiable, pour le maître du jeu, et une où, après que le maître du jeu ait généré la fiche, le joueur peut seulement modifier l'image. Mais à ce stade, ce double niveau était redondant et plus inutile qu'autre chose.

2 - Deuxième itération

Lors de la deuxième itération, content du fonctionnement de la première fiche en général, nous nous sommes concentrés sur l'ergonomie de l'application, en nous basant notamment sur des retours faits par le reste du groupe de TD GH lors d'une séance en présentiel de méthodes agiles.

Ainsi, les gestions des capacités et de l'inventaire ont été séparés, et une valeur numérique a été mise sur la barre de vie. Pour pallier le manque d'organisation du code de la première itération, nous avons créé des programmes de test pour s'assurer d'une continuité dans le fonctionnement du code et pour prévenir de futurs problèmes, puis nous avons réarrangé le code en différents paquetages par type de classes : les interfaces, les modèles, les vues, et les programmes utilitaires et d'affichages.

Nous avons aussi modifié les rôles des deux interfaces créées, avec l'interface de maître du jeu devenant un menu de toutes les fiches créées avec un accès pour les modifier et l'interface du joueur la modification des fiches de personnage :

- En tant que joueur, je veux pouvoir créer et gérer plusieurs fiches de personnage en même temps.

(Valeur d'effort : 45, valeur métier : excitant pour l'utilisateur, Could be done)

Nous avons aussi commencé à voir comment permettre à l'utilisateur de créer lui-même des catégories comme les capacités ou l'inventaire.

3 - Troisième itération

Pour la troisième itération, nous avons traité les trois user stories suivantes qui nous sont apparus intéressantes pendant la conception et les itérations précédentes :

- En tant que joueur, je veux sauvegarder mes fiches et les charger lors d'une prochaine session.
  (Valeur d'effort : 25, valeur métier : excitant pour l'utilisateur, Should be done)
- En tant que joueur, je veux pouvoir ajouter mes propres catégories de statistiques semblables aux capacités pour mieux m'organiser.
  (Valeur d'effort : 20, valeur métier : excitant pour l'utilisateur, Could be done).
- En tant que joueur de jeu de rôle, je veux pouvoir lancer des dés avec un nombre de faces de mon choix.
  (Valeur d'effort : 10, valeur métier : intéressant pour l'utilisateur, Should be done).

De plus, un Makefile permettant de compiler et de lancer l'application, ainsi que de nettoyer les fichiers de compilation a été écrit pour permettre une utilisation plus simple pour l'utilisateur. Il nous a aussi servi à créer l'archive .jar de rendu.

Enfin, avec le rendu final et l'oral de projet arrivant, nous avons dévoué une plus grande partie de notre temps à l'explication et la documentation de notre travail.

VI - Organisation de l’équipe

L’organisation du travail s’est faite en regardant l’aise et les compétences de chacun pour les différents domaines nécessaires pour les user stories.

Alexandre Fray a programmé le lancer de dé, et aider à résoudre des problèmes sur l’application.

Alexia Huc-Lhuillery et Léane Hauteville ont beaucoup travaillées ensemble et se sont occupées de la partie ergonomique de l’application, notamment les messages d’erreur en cas de problème (aucun nom entré dans la fiche de personnage par exemple) et la refonte de la deuxième itération. Elles ont aussi créé les diagrammes UML et écrit le squelette des rapports.

Basile Gros a créé les fichiers de test et Makefile pour aider à la programmation de l’application et a aidé à l’écriture des rapports. Il a organisé la partie de jeu de rôle de présentation à l'équipe.

Hamza Hathoute a fait tout ce qui touche aux fichiers externes : sauvegarde d’une image, importation d’une image de profil et sauvegarde des fiches, ainsi que le menu des fiches.

Kamal Hammi a fait une bonne partie des interfaces : l’ensemble de l’interface initiale (il ne devait en faire qu’une partie mais il y a eu un problème de communication au sein de l’équipe), et la création des catégories de statistiques pour les itérations 2 et 3.

Lucas Goussard a aidé pour la partie ergonomique de l’interface, notamment de la barre de vie et l’intégration du lancer de dé. Il a aussi rédigé une bonne partie des rapports de fin d’itération.