Projet de Semestre 2

Développement d'un gestionnaire de cartes Hearthstone

16 mars 2018

1 Organisation générale du projet

Le projet de S2 va être organisé en lien avec plusieurs des enseignements du semestre 2, en particulier les UEs *GPI* (J.-M. Mottu), *CO-Conception* (D. Tamzalit), *PO2* (J.-F. Remm), *IHM* (A. Lanoix) et *CO-Test* (J.-M. Mottu) : ce sera donc une application directe et une prolongation de toutes les thématiques abordées dans ces différentes UEs. Vous aurez différents rendus à effectuer tout au long du semestre ainsi que des créneaux dédiés au projet dans l'emploi du temps (entre 1h20 et 2h40 par semaine), avec, à la toute fin du semestre, la semaine 25 consacrée exclusivement au développement proprement dit et à son évaluation.

Vous travaillerez par équipe de 4 à 5 étudiants au sein d'un même groupe de TP.

Comme tout projet industriel, les exigences de vos clients risquent d'évoluer en cours de projet. De plus, ce document est évidemment incomplet : il faudra donc intéragir avec vos clients. TOUTES les questions et réponses doivent être portés dans le forum du module madoc du projet, cela inclue les réponses que vous aurez pu obtenir directement auprès d'un professeur. Le partage d'information est exigée, une information non partagée sera remise en cause.

http://madoc.univ-nantes.fr/course/view.php?id=29855

2 Thématique : Hearthstone

Votre projet va consister à développer une application de bureau permettant de gérer des cartes HEARTHSTONE, en particulier enregistrer sa collection de cartes et constituer un "deck" [1]. Vous êtes 87% à connaître HEARTHSTONE, dont 32% de joueurs réguliers, et 58% de joueurs occasionnels [2] : l'appropriation du domaine métier ne devrait donc pas être une difficulté. Attention néanmoins, il ne sera pas ici question de jouer à HEARTHSTONE, mais il s'agira bien uniquement de développer une application de gestion de cartes.

Un joueur à HEARTHSTONE possède une collection de cartes : les cartes sont soit des cartes *Serviteur*, soit des cartes *Sort* soit des cartes *Arme*. Elles ont des caractéristiques communes et des caractéristiques spécifiques. Parmi les différentes manières d'obtenir des cartes, il est possible d'en fabriquer. Pour cela, il faut "détruire" des cartes afin d'obtenir des points permettant ensuite de "créer" de nouvelles cartes. Avant de jouer une partie d'HEARTHSTONE, un joueur va constituer un deck[1], c'est-à-dire qu'il va sélectionner parmi toutes les cartes de sa collection lesquelles il va utiliser pour la partie. Il y a plusieurs sortes de decks avec des contraintes sur le nombre de cartes, les cartes possibles, etc.

Notez que l'application à développer devra (entre autre) permettre

— de créer une carte et l'ajouter à la collection,

— de détruire des cartes de sa collection,

— d'afficher/trier les cartes de la collection,

— de créer un deck, c'est-à-dire sélectionner des cartes de la collection,

— d'afficher les cartes d'un deck,

— de sauvegarder la collection de cartes et les decks créés, entre deux lancements de l'application,

— etc.

Vous trouverez de nombreuses informations détaillées sur internet à propos des différentes cartes HEARTHSTONE et la constitution de decks, en commençant par :

1. https://fr.wikipedia.org/wiki/Deck_(jeu)
2. sondage réalisé ente le 06 et le 15 février 2018 : 103 réponses

— le site officiel d'HEARTHSTONE : https://playhearthstone.com/fr-fr/

— La page wikipedia : https://fr.wikipedia.org/wiki/Hearthstone

— Des sites "communautaires" comme http://www.hearthstone-decks.com/ ou https://tools.millenium.org/hearthstone-deckbuilder/

3 Travail à réaliser

Ce projet consiste à réaliser pour vous un premier projet complet qui débute après la **phase préliminaire**, celle ci ayant été réalisée par vos enseignants et dont ce document résulte. Si l'on reprend les étapes de l'AFNOR vues en GPI, il s'agira donc d'effectuer les étapes de **conception, définition, construction**. La **mise en route** consistera à montrer pendant l'évaluation que votre système fonctionne bien, en fin de la semaine 25 dédiée au projet. Le **transfert à l'exploitation** est inutile ici. Ces étapes devront être réalisées dans l'ordre au rythme imposé par vos enseignants au fur et à mesure des cours. Pour vous y contraindre et être efficace, nous diviserons le travail en 2 phases :

1. Conception produisant le cahier des charges (module de GPI) et la modélisation (module de CO-C) avec rendu noté,
2. Définition (choix techniques, architecturaux) et Construction (programmation, tests) se feront ensuite sur la base d'un cahier des charges et une conception communs qui vous seront donnés par les enseignants.

Notez que la programmation (des algos, de l'IHM) sera guidée, il ne vous sera d'aucune utilité de commencer le développement avant que cela vous soit indiqué.

4 Planning prévisionnel

Voici une estimation du planning avec quelques jalons et contraintes temporelles qu'il faudra préciser et compléter dans le cahier des charges.

Semaine	Information
12-13	Cahier des charges - conception en séance de GPI et CO-C
14	Semaine de DS
15-17	Cahier des charges - conception en séance de projet Rendu Jeudi 26 avril, 17h50
18-19	Vacances
20-23	Etude des notions nécessaires en séances de PO2, CO-T, IHM et pendant les séances de projet
24	Semaine de DS
25	Semaine de projet, rendu, évaluation

4.1 Conception

Il s'agit de réaliser un cahier des charges incluant les spécifications UML. Ce document sera rendu et noté dans les modules de GPI et de CO-Conception.,

Cahier des charges. Il s'agit de mettre en oeuvre l'ensemble du module de GPI pour produire un cahier des charges en utilisant le format AFNOR (disponible dans le module GPI de madoc). Certains diagrammes servent à la réflexion et ne doive pas forcément apparaitre dans le cahier des charges (bête à cornes), d'autres sont indispensables. Une partie des diagrammes viennent de la Spécification UML (section suivante) et doivent être intégré dans le document. De même le diagramme de Gantt sera annexé au projet.

Spécifications UML. Il vous est demandé de spécifier :

— Le diagramme de cas d'utilisation.

— Les scénarios de chaque cas d'utilisation.

— Le diagramme de classes en précisant le détail des classes (attributs et opérations).

Attention à ne pas inclure de détails d'implémentation dans votre conception ! Rappelez-vous qu'il faut produite une modélisation correcte correspondant au contexte du projet.

4.2 Développement

Au démarrage de cette étape de construction, la définition du projet vous sera fournie pour que vous travailliez tous sur une même base.

L'ensemble du développement sera réalisé en Java.

Implémentation du modèle de données. Il s'agira de réaliser l'implémentation d'un diagramme de classes concret fourni. Ce diagramme représentera les données métier que votre application doit manipuler.

L'implémentation devra respecter des bonnes pratiques de développement, dont en particulier des nommages explicites, de la documentation du code, de la documentation dans un document annexe (à usage de l'utilisateur en particulier). Vous justifierez également vos choix d'implémentation.

Tests du modèle de données implémenté. En parallèle de l'implémentation du modèle de données, il vous faudra écrire des cas de test Junit, permettant de valider chacun des aspects du modèle de données à implémenter.

Vous testerez votre implémentation, mais également des implémentations réalisées par une autre équipe de votre groupe de TP. Vous remarquerez bien ici l'importance que le code soit interfaçable et puisse être utilisée par une autre équipe de développement sans que vous y ayiez ajouté des contraintes.

Nous aurons également écrit des cas de test de notre coté, et votre implémentation sera également tester par nos soins.

Implémentation d'une IHM. Vous développerez une IHM en Java/Swing qui manipulera le modèle de données précédemment implémenté pour réaliser les fonctionnalités attendues.

5 Evaluation

Tous les aspects du projet seront évalués :

— rédaction du cahier des charges,

— réalisation des spécifications UMLs,

— implémentation correcte du modèle de données fourni

— qualité et pertinence des tests réalisés / rapports de bugs fournis aux autres groupes.

— développement d'une IHM

— présentation du travail réalisé sous la forme d'un "pitch" [3] en fin de semaine de projet.

3. https://fr.wikipedia.org/wiki/Elevator_pitch