Document de synthèse

Organisation et répartition du travail

Le travail s'est tout d'abord réparti en deux parties distinctes indispensables :
En premier lieu, ce que nous voulions faire.
Cette partie-là fut assez rapidement concluante, fort heureusement. Et malgré quelques modifications ponctuelles due aux difficultés rencontrées lors de l'implémentation qui nous ont contraints à revoir à la baisse certaines de nos ambitions, notamment de faire une version web, les idées directrices ont été maintenues tout au long du projet.
Lucas faisant assez souvent de l'escalade, il était celui sachant le mieux ce que l'application devrait contenir et bien que nous ayons tous contribué à définir les besoins de l'utilisateur et discuté de l'optimisation de leur satisfaction, en conformité avec un cahier des charges qui était dans notre cas bien détaillé avant toute séance délibérative, il a donc été celui qui, de loin, a le plus décisivement défini le cap que nous suivrions au cours du projet.

Puis, dans un second temps, nous avons cherché à réaliser concrètement ce que nous avions imaginé. Cette phase aura été beaucoup plus complexe, et peut se décrire en trois catégories que sont l'analyse préalable au développement, le développement lui-même et les phases intermédiaires, dont le rapport d'analyse et le présent document de synthèse sont les seuls produits et leur rédaction une préoccupation accrue de ces phases n'excédant pas quelques jours.

Lors de la phase d'analyse, nous avons découvert Umbrello, un logiciel de modélisation UML avec lequel nous avons fait nos diagrammes UML.

Quant à la phase de développement, elle peut se diviser en trois sous-parties correspondant au contenu du package principal pour la première, celui du package cli pour la deuxième et à notre tentative de réaliser un UI pour la troisième.
Cédric s'est occupé de faire la CLI ainsi que les classes du package principal dont l'implémentation était plutôt difficile tandis que Lucas s'est occupé de l'UI et des classes dont l'implémentation était pratiquement plus accessible.

Ce qui a été réalisé

Nous avons défini tout ce qui pouvait se trouver dans une salle d'escalade et avons créé la possibilité pour les utilisateurs de l'application de connaître en permanence l'état de la salle mais aussi de pouvoir échanger via l'application avec les autres membres sur l'univers de l'escalade, grâce aux annonces et aux événements.
L'application permet aussi à l'utilisateur d'enregistrer et de consulter sa progression afin de l'aider dans sa pratique de l'escalade.
Quand un grimpeur enregistre une grimpe, il doit donner son avis sur le niveau de la voie.
Il peut dire si la difficulté devrait être augmentée, diminuée ou si elle lui semble adéquate.
Cet outil permet ensuite au gestionnaire de modifier le niveau de certaines voies si une trop grande divergence est observée entre son avis initial et celui de l'ensemble des grimpeurs.
Malgré l'échec de construction d'une interface graphique, nous sommes parvenu à une application fonctionnelle avec laquelle l'utilisateur peut interagir en lignes de commandes.

Structure du code

Le code se divise en trois packages.
Un premier package nommé principal, qui contient toutes les ressources, humaines comme matérielles pouvant exister dans une salle d'escalade.
La classe fondamentale de ce package est la classe abstraite activité dont toutes les autres classes retiennent des fonctionnalités et dépendent donc plus ou moins directement.
Cette classe fondamentale s'exprime concrètement par deux autres classes qui en sont les héritières ; les classes voie et grimpeur qui représentent respectivement les ressources matérielles et humaines de la salle.
L'ensemble des autres classes de ce package sont soit des classes héritant directement d'une des deux classes voie et grimpeur soit des classes servant à définir les différentes interactions entre les voies et les grimpeurs ou entre les grimpeurs.
La classe MainBoard fait toutefois quelque peu exception à cette règle dans le package principal puisqu'il s'agit de la classe regroupant tout ce qui se trouve dans la salle d'escalade.
Un autre nommé io qui gère les sauvegardes nécessaires au bon fonctionnement de l'application.
Enfin, un dernier package nommé cli qui permet une interaction entre l'application et l'utilisateur.

Exécution du programme

Ll s'agit d'un invite de commande qui récupère les commandes, puis la chaîne de caractères obtenue est transformée en commande grâce au constructeur de la classe command puis la commande est exécutée en appelant la method exec() de la classe commande.
Le point d'entrée dans la version de test c'est io.Test.main() qui appelle login() sur le MainBoard créé et ensuite login() appelle startconsole().

Ce qui n'a pas pu l'être et pourquoi

Nous aurions aimé pouvoir faire une interface graphique, ce que nous avons d'ailleurs ardemment tenté de concrétiser mais qui s'est surtout, eu égard du projet, révélé une perte d'un temps précieux.

Intérêt pédagogique

Nous avons appris à utiliser Github. Nous avons dû créer des branches séparées et comprendre comment Github permettait de coordonner les contributions de plusieurs personnes différentes, ce qui nous a causé quelques difficultés par moment mais nous sommes désormais en mesure d'exploiter Github efficacement.

Nous avons appris à maîtriser Umbrello, un logiciel de modélisation UML, permettant de mettre en pratique nos enseignements d'analyse informatique.

De par les pistes explorées pour améliorer notre code tout au long du projet, nous avons pu nous essayer à de nombreuses fonctionnalités qu'offre le langage Java, principalement celles concernant les graphismes qui n'avaient pas été abordées en cours, et donc améliorer nos compétences en Java tout en renforçant notre acquisition des concepts orienté objet.