**Rapport de Projet JAVA**

Hermel Jean-Alexis et GHYS Robin  
  
Sujet : Western

Introduction :  
  
 Nous avons choisi de prendre le sujet sur le Western car nous avons trouvé que c’était un sujet très ouvert et possiblement plus créatif que les autres, mais aussi parce que nous avions un petite base dans le cours pour nous lancer et voir ce qu’il était possible de faire grâce au langage JAVA.

Pour débuter nous avons d’abord, avant de coder, mit en place ce que nous voulions qu’il y ait dans notre projet sur papier. Le but étant de savoir où l’on va lorsque qu’on commence à écrire du code et de ne pas avancer à l’aveugle. Ainsi nous avons réalisé un premier diagramme de classes (disponible en PDF), peaufiné au fil du temps, qui nous a été très utile pour nous repérer à travers nos différentes classes. Le Western que nous avons créé reflète l’histoire classique qui se déroule dans les films de western, avec des personnages de type cowboy, brigand, banquier, barman, indien… Ainsi il est possible de générer quelques histoires en fonction des différents choix qui s’offrent à l’utilisateur comme nous allons l’expliquer.

Sujet et Description :

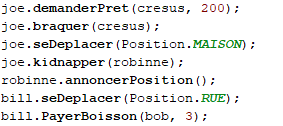
Le but du projet est de créer un environnement de classes dans lequel il va être possible de générer des histoires selon les décisions prises par l’appel de méthodes de ces différentes classes.  
Notre projet s’articule autour d’un genre de village Western, où des lieux vont être créés (Classe Enumération), sur cette base de lieux, nous allons construire des personnages de différents types. Certains auront un lieu attitré, d’autre non. Un personnage verra sa classe héritée d’une ou plusieurs autre classe, en fonction de s’il possède des propriétés communes avec d’autres personnages. Exemple avec le personnage indien qui hérite de la classe Homme (arme, tirer, force…) qui elle-même hérite de la classe Humain (nom, prénom, Position, âge...).

Nous avons fait le choix de nous restreindre dans le nombre et les différences entre les personnages, ce malgré nos nombreuses idées, car sinon le temps nécessaire au projet n’aurait pas suffi pour avoir un programme avec un fonctionnement correct. Finalement nous avons 7 types de personnages pouvant interagir dans 7 lieux différents.

Les personnages pourront être : Banquier, Cowboy, Barman, Indien, Dame, Brigand, Sherif.

Les lieux sont : Le campement, la rue, la prison, la maison, le désert, le bar, la banque.

Un personnage va être décomposé en plusieurs classes (héritées), dont certaines abstraites. Comme dit précédemment les caractéristiques partagées par différents personnages sont réunies dans des classes mère (cf. exemple humain>homme>civil…) afin de factoriser le code. Le barman et le banquier partagent une interface commune, la classe du barman est composée avec une classe Boisson pour pouvoir gérer et entretenir un stock. Enfin les lieux sont regroupés dans une classe Enumération pour y accéder facilement dans le code.

Le principe de notre projet va être d’instancier des personnages avec leurs caractéristiques propres (cf. diagramme) dans le *main*, pour ensuite pouvoir leur faire effectuer des actions et interagir entre eux. Ceci va avoir la forme ci-contre, pour connaître les options possible pour chaque personnage créé, une méthode *help(personnage)* renvoie chaque action disponible.

Notre programme est fait de sorte que l’utilisateur ne puisse pas faire ce qu’il veut avec toutes les méthodes, afin de garder un aspect cohérent et logique à la génération de l’histoire. Par exemple si l’utilisateur souhaite commander une boisson, il devra être vivant (donc avoir au minima gagné un précédent combat), et se trouver au bar. Pour cela il devra s’y déplacer et discuter avec le barman en particulier. Nous avons voulu impliquer l’utilisateur dans les choix s’offrant à lui plutôt que simplement appeler une méthode « d’affichage d’information » lorsqu’il effectue une action avec un personnage.

Ainsi, nous avons dans quelques classes et méthodes introduit le hasard, qui fait varier et augmente considérablement les issues des histoires possibles. Il est envisageable que si, par exemple, le brigand décide de combattre le cowboy, celui-ci réplique et tue le brigand. Et ce à la place d’un méthode qui tuerait atomiquement le personnage qui se fait tirer dessus. Notre programme est donc plus vivant et permet une plus grand durée d’utilisation.

Parmi les différentes méthodes et classes disponibles, certaines sont au cœur du projet et ce sont celles-ci qui vont augmenter les interactions et scénarios possibles :

- La fonction *tirer (Homme)*, agit avec un système de force (0-10) des personnages de type Homme, l’issue du *combat (Homme1, Homme2)* va être décidée par le hasard, pondérée par la force de chaque personnage. L’utilisateur va pouvoir découvrir qu’il ne peut pas faire ce qu’il veut et doit respecter des conditions pour s’il veut faire combattre des personnages (ex. le cowboy est gentil et ne tire donc que sur les méchants).

- La fonction *négocier (Homme)*, est en lien avec la fonction *tirer* car elle va avoir des effets principalement sur le Cowboy et le Brigand, pouvant négocier avec un l’Indien pour augmenter un peu leur force. Des variantes existent aussi si le cowboy négocie avec le brigand pour en devenir un, ou si le brigand essaie avec le shérif il va en prison.

- Le Barman, qui va avoir un vrai stock de boissons avec quantité et prix va pouvoir en proposer et servir les clients qui lui en demandent via *demanderBoisson (Civil, Boisson).* C’est grâce au système d’argent que nous avons mit en place pour les personnages de type Civil, nous allons le détailler par la suite.

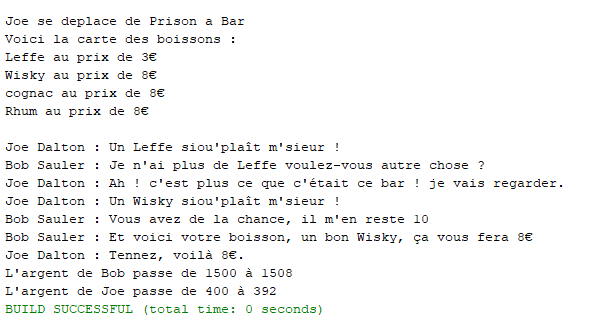
- Un système de « crimes » que va pouvoir commettre le brigand qui vont pouvoir être déjoués/contrés par le cowboy ou le shérif, comme exemple, la Dame va se faire enlever, le cowboy va pouvoir la libérer et le shérif emprisonner le brigand.

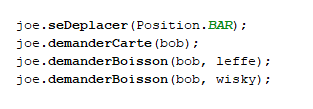
- Un système de classe interface pour les personnages commerçants (Banquier et Barman), qui va déclencher une réaction du personnage à la suite d’un braquage par le brigand. Ceux-ci auront une certaine chance de vider leur réserve d’alcool et peut être de tomber en dépression à la suite de ça.

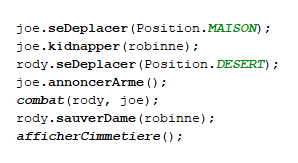
- Enfin un système d’argent pour les personnages de Type Civil. Ceux-ci auront un somme d’argent de base dès leur instanciation, cette somme pourra varier selon ce qu’ils décideront de faire, par exemple tous pourront choisir de donner de l’argent à un autre personnage via *donnerArgent(Civil,argent)* et auront la possibilité de faire un prêt au banquier (donc en se rendant à la banque). Ce prêt pourra être accordé si celui qui le demande rempli quelques conditions sur la somme qu’il demande. Par exemple un personnage ne peut pas demander plus de 45% de ce qu’il a déjà. Ce système rajoute un plus au programme en augmentant les interactions possibles entre personnages, c’est le cas avec les commande de boissons au Barman.

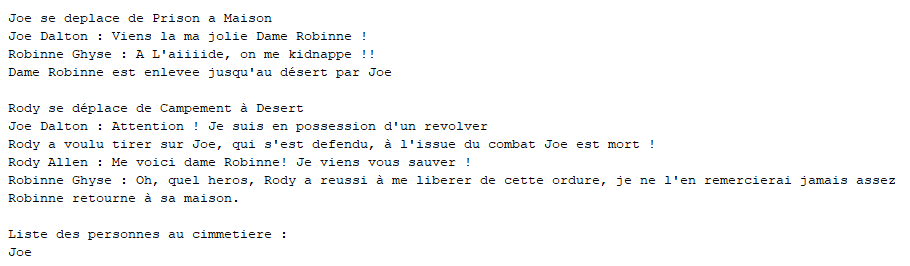
Il y a également d’autres méthodes permettant de faire quelques petites choses comme jouer d’un instrument pour le Barman, se présenter (pour tous les personnages), ou encore s’échapper de prison pour le Brigand.

Toutes ces classes mettent en œuvre un environnement dans lequel l’utilisateur va pouvoir bouger et agir, il reste très basique mais pose les bases d’un jeu vidéo RPG. Nous avons mis en place un système de sauvegarde partie, qui va enregistrer dans un fichier toutes les répliques et commentaire qui se sont déroulés dans le scénario.

Voici un aperçu de ce que le programme renvoi comme scénario :







Travail effectué :

Concernant nous deux : nous avons élaboré le plan du projet et son diagramme de classes ensemble pour être sûr que chacun soit d’accord sur la conduite à adopter. Ensuite, lorsque nous avons commencé à coder nous nous sommes réparti le travail en deux concernant l’écriture des *getters* et *setters* de chaque classe, chose assez répétitive. Il en va de même pour l’écriture des constructeurs, nous avons pour ça décider de l’ordre des paramètres de ceux-ci pour s’y retrouver plus facilement. Enfin nous avons mit en place le système de force pour avoir des combats qui ne se soient pas toujours prédéterminés, puis déterminé les conditions que l’utilisateur devait remplir s’il voulait utiliser certaines actions d’un personnages. Comme exemple cité précédemment, se trouver au même endroit pour interagir et être en vie.

Concernant Jean-Alexis :

J’ai pu réaliser le système d’argent et tout ce qui va s’y allie, c’est-à-dire l’écriture des méthodes de la classe Barman et Banquier, le système de prêts, la classe Boisson, la classe PeutBoireBar. Mon objectif était que cette gestion d’argent des personnages se rapproche de ce qu’il peut se faire dans les jeux vidéo mais en restant assez simple. Nous aurions pu développer d’autres méthodes similaire d’achats/vente mais elles n’auraient rien apporté de plus au contenu du projet en termes de richesse de code.

J’ai eu l’occasion de concevoir, une fois l’idée mise en place, le système de combats entre personnages, celui reste basique mais est fonctionnel, par exemple les personnages ont une arme mais la nature de celle-ci n’influe pas sur le résultat du combat. La force et le hasard décideront de l’issue. Pour ajouter des fonctionnalités à cette méthode *tirer* j’ai également conçu la méthode *combat* et *afficherCimmetière* pour simplifier la lecture du jeu à l’utilisateur.

Par la suite j’ai réalisé l’interface *SestFaitFaucher* qui intervient lors d’un braquage du brigand, le but ici était d’utiliser le principe des interface pour appliquer un même comportement à des classes qui n’ont pas le même héritage. Ici, après avoir écrit les méthodes concernant le braquage (*seFaireBraquer*, *braquer*), j’ai codé celles conséquentes au braquage, *videReserveAlcool* et *depression* toujours avec le hasard qui vient augmenter les possibilités de scénario. Par exemple le fait de vider la réserve d’alcool peut entrainer une dépression du personnage (avec plus de chance s’il s’est fait braquer juste avant).

Enfin j’ai rajouté des exceptions dans le code pour prévenir l’utilisateur d’une mauvaise saisie, notamment pour un âge ou une somme d’argent négative. Ceci avec l’utilisation de blocs *Try* et *Catch*.

Concernant Robin :

robin : negocier, dame, brigant, sherif, systeme de lieu, ecriture fichier texte, override, négocier (ajout force…)

Difficultés rencontrées :

Concernant les difficultés que nous avons rencontrées, les premières furent dès la conception du diagramme de classes, nous avions nos personnages en tête mais il fallu bien réfléchir à comment factoriser, via des héritages, tous les attributs commun à chacun. Nous avons réglé ce problème en créant des classes telles que Humain, Homme, le seul bémol reste que la femme ne peut pas bénéficier des attributs et méthodes de la classe Civil dans notre système actuel (voir diagramme).

Vient ensuite un petit soucis concernant l’usage de l’attribut force, comment rendre notre projet moins prédictible et plus axé sur des « rebondissements ». Nous avons donc pour cela fait appel au hasard qui prend une grande place dans notre simulation de Western car c’est grâce à lui que les scénarios sont plus variés qu’ils ne le seraient s’ils n’étaient basés que sur un simple modèle action/conséquence.  
  
 Ensuite nous aurions aimé concevoir une méthode permettant d’instancier un type de personnage choisi avec son nom, du type *creerCowboy(‘rody’,24,’Revolver’…)* mais le problème, non résolu, est que dans ce cas, on ne peut pas donner le nom du personnage à l’objet lui-même, tel qu’on le fait lorsqu’on en créer un dans le main. Par exemple *bill = new Cowboy (‘bill’…)* ne fonctionnera pas, on devra au mieux se contenter d’un numéro dans une liste de personnages, ce qui rend l’utilisation des méthodes moins compréhensible. On ne pourrait plus faire *bill.negocier(bob)* mais *persos[1].negocier(persos[2])* qui n’est pas ce que l’on souhaite.

Au terme du projet, nous voulions que l’utilisateur puisse, via une interface type *scanner* pouvoir créer un personnage en rentrant ses caractéristique, mais nous nous sommes heurtés au problème de la *reflection* en java et du *invok* que nous n’avons pas sur gérer finalement.

Enfin un dernier petit problème fut la façon dont on allait ajouter un stock de boissons au barman. Au début nous avions comme idée de créer une ArrayList de ArrayList mais on ne peut pas faire de Tuple en Java comparé à Python. C’est pourquoi nous avons créer une classe Boisson, possédant des attributs comme quantité, type, prix, pour ajouter une boisson il suffit de l’instancier dans le main et de l’ajouter au Barman. Cette solution permet une meilleure flexibilité et utilise le principe même de Java à savoir créer des classes d’objets.

Conclusion :

Pour conclure, nous sommes contents du résultat obtenu, et des scénarios en tant que tels produits par notre jeu. Ce projet fut pour nous l’occasion de revoir l’ensemble du cours de Java et de le mettre en application, avec plus ou moins de difficultés. Le fait de d’avoir choisi un sujet avec plus de libertés nous a forcé à mettre de la rigueur dans ce que l’ont faisait dès le début. C’est grâce à cela que nous n’avons pas vraiment eu de gros soucis de code une fois que nous avons commencé à nous plonger dans l’écriture. Les concepts de classe abstraite et d’héritage sont au cœur de notre projet Western et donc bien acquises. Il a parfois été nécessaire de contourner des choix de jeu que nous avions du fait de la difficulté ou l’infaisabilité des idées mais dans l’ensemble nous avons réussi à respecter le cahier des charges fixé au début, surtout concernant la partie combats et argent. Pour terminer, nous sommes conscient que le jeu n’est pas vraiment jouable comme on un jeu normal mais c’est justement un des points d’amélioration que nous avons en tête si on a l’occasion de reprendre le projet.