Projet LO02
C'est du brutal !

Il vous est proposé dans ce projet de concevoir avec UML et développer en Java une version logicielle d'un nouveau jeu intitulé *C'est du brutal !*

1. Règles du jeu

Les règles du jeu vous sont données en annexe**.** Elles servent de document de référence pour l'expression des besoins du logiciel développé.

Au-delà, on considère aussi les besoins suivants :

- L'ensemble de l'application sera intégré dans une interface graphique. Des propositions d'interfaces avec leur code Java seront disponibles sur Moodle LO02.
- Avant la conception et le développement de l'interface graphique, une interface rudimentaire en ligne de commande permettra de tester le moteur du jeu. Cette interface devra être conservée lors de l'évaluation fonctionnelle et la remise des fichiers.
- L'architecture retenue devra veiller au respect des règles de la conception orientée objet. Elle devra ainsi être (i) modulaire en identifiant des composants indépendants liés entre eux par des relations, (ii) extensible en permettant l'ajout de nouvelles fonctionnalités à ce jeu et (iii) intégrer des patrons de conception vus en cours tels que Singleton, Strategy, MVC.

2. Les 3 phases de votre projet

Le projet est découpé en trois phases qui conduiront à des soutenances effectuées lors d'une séance de TP dédiée. Les trois phases du projet sont :

1. La modélisation UML initiale (livrable 1).
2. Le développement du moteur du jeu, utilisable en lignes de commandes (livrable 2).
3. Le développement de l'interface graphique et la remise du code source documenté (livrable 3).

Pour chaque phase, les consignes à suivre vous sont données ci-après.

2.1. Phase 1 : modélisation UML

La modélisation UML proposée sera exposée sous la forme d'une présentation avec quelques transparents et devra suivre la structure suivante :

1. Introduction : présentation du projet tel que vous l'avez compris (pas de copier/coller de l'énoncé) et annonce du plan du document
2. Diagramme de cas d'utilisation : identifier l'ensemble des cas d'utilisation du système, leurs relations et les expliquer.
3. Diagramme de classes : décrire et expliquer chaque élément de conception du diagramme. Vous noterez qu'il est inutile de modéliser les interfaces de commande du jeu (en lignes de commandes ou graphique) ; on ne représentera que le cœur de l'application.

4. Diagramme de séquences : proposer un diagramme de séquence pour le déroulement d'un tour de jeu.
5. Conclusion : Identifier les aspects sûrs de la modélisation et ceux dont vous pensez que le développement pourrait induire une modification.

2.2. Phase 2 : cœur de l'application et interface en lignes de commandes

Cette phase, validée par une seconde soutenance orale, consistera à faire une première démonstration du développement. A cette étape, le moteur du jeu devra être fonctionnel et utilisable en lignes de commandes. Mais aucune mise en forme particulière du code produit n'est demandée (mise en packages, documentation, mise au propre du code, …). De même, aucun support de présentation ne devra être préparé pour cette seconde soutenance.

Il est possible d'aborder la programmation de ce jeu de deux manières.

1. *Une programmation itérative (séquentielle) où l'ensemble des protagonistes évoluent à tour de rôle.*
2. *Une programmation parallèle (multithreading) où l'ensemble des protagonistes évoluent en "même temps". Il s'agit ici d'une forme de programmation légèrement plus compliquée que la première pour laquelle un bonus sera appliqué dans l'évaluation. Evidemment, choisir la première forme de programmation n'enlève en rien à la réussite du projet.*

2.3. Phase 3 : projet complet et documentation

Cette dernière phase consistera à mettre en œuvre la version finale du projet ainsi que toute la documentation qui l'accompagne. Le travail de développement intégrera une interface graphique au cœur de l'application développé précédemment en mettant en œuvre le patron de conception MVC.
Il n'est pas demandé de développer une interface graphique avec un esthétisme poussé digne d'une application commercialisée.

Une soutenance orale permettra de valider les aspects fonctionnels de l'application et une remise des fichiers du projet permettra d'évaluer le code, son organisation, sa structure et sa documentation.

Evaluation fonctionnelle

Elle sera effectuée par le biais d'une soutenance qui aura lieu durant votre dernière séance de TP. Elle s'effectue par binôme et dure 15 minutes durant lesquelles vous présenterez l'état fonctionnel de votre application par le biais d'une démonstration. Une fois la démonstration passée, tout développement supplémentaire (correction de bugs, ajout de fonctionnalités, …) est inutile car l'état fonctionnel de votre projet sera évalué à ce moment. Suite à cela, quelques questions relatives au code que vous avez produit vous seront posées.
Remarque : Pour cette évaluation fonctionnelle, il n'est pas demandé de mettre au propre le code présenté (respect des conventions d'écriture, commentaire Javadoc, suppression des blocs de codes inutiles mis en commentaires, …). Par contre, cela devra être fait pour la remise des fichiers.

Remise des fichiers du projet

Les fichiers du projet à remettre seront déposés sur Moodle, dans un espace dédié à cet effet, et placés dans une archive ZIP dont le nom du fichier doit suivre la convention de nommage suivante :

Volfoni_Naudin_Projet_LO02_A22.zip, où Volfoni et Naudin correspondent aux noms du binôme.

L'archive devra comporter les dossiers suivants :

- src : avec l'ensemble des sources du projet ;
- classes : avec l'ensemble des classes compilées du projet ;
- doc : avec la javadoc du projet, le diagramme de classes final et un résumé de l'état actuel de l'application.

Enfin, les consignes à observer pour la mise en forme du code développé dans le cadre du projet sont les suivantes :

- L'ensemble du code doit être documenté par le biais de commentaires javadoc. Il ne faudra pas se contenter des balises standard mais décrire précisément la fonction de chaque élément de code (classe, méthode, …) en respectant les conventions de documentation pour la production logicielle vues durant le semestre.
- Le code devra être propre et suivre les conventions d'écriture spécifiées par Oracle (nommage, indentation, blocs, casse, …). Aucun code obsolète, placé dans un commentaire, ne devra figurer dans les fichiers sources.
- **Le diagramme de classes final. On détaillera ici et justifiera les changements entre la version initiale du diagramme de classes et la version finale, effectivement implémentée dans le code. Pour ce faire, on présentera les deux diagrammes.**
- L'état actuel de l'application : Cette partie donnera de manière précise l'état de l'application en regard du présent cahier des charges. On y indiquera clairement ce qui a été implémenté et ce qu'il ne l'est pas, ce qui fonctionne et ce qui reste bogué.

3. Consignes générales

3.1. Calendrier

Semaine LO 02	**Evénement**
2 (12/09) et 3 (19/09)	Distribution des projets et **constitution des binômes**
6 (10/10) et 7 (17/10)	Soutenance phase 1
13 (12/12)	Soutenance phase 2
14 (02/01)	Soutenance phase 3
Vendredi 13/01 à 23:55 (deadline)	Remise des fichiers sur Moodle

3.2. Autres consignes

- Pour l'ensemble des livrables du projet, aucun retard ne sera accepté.
- Vos intervenants de TP sont vos interlocuteurs exclusifs pour toute question relative au projet.

C'est du brutal ![1]

Contexte & Protagonistes

Le jeu *C'est du brutal !* se joue à deux joueurs et a pour décor l'université de technologie de Montauban. Chaque joueur représente un des sept programmes de l'UTM (ISI, RT, A2I, GI, GM, MTE, MM) et possède un effectif de 20 étudiants. A l'aide de son équipe, chaque joueur devra essayer de contrôler une majorité de zones d'influence.

Ces zones d'influence sont

1. La bibliothèque
2. Le Bureau des Etudiants
3. Le Quartier Administratif
4. Les Halles industrielles
5. La Halle Sportive

Chaque étudiant possède les caractéristiques suivantes :

- **Crédits ECTS** : ce nombre de crédits est initialement de 30. Lorsque la valeur atteint 0 ou moins, l'étudiant sort définitivement de la bataille.

- **Dextérité** : la dextérité initialement affectée à 0 est comprise entre 0 et 10. Elle augmente les chances de "toucher" son adversaire en lançant son gobelet (*dit le gobi*) lors d'une attaque, ou d'esquiver lorsqu'il est attaqué. Un point correspond à 3% de chance supplémentaire d'atteindre sa cible ou d'esquiver une attaque. La dextérité sert également à la réussite d'un soin porté à un frère d'arme. Un point correspond à 6% de chance supplémentaire de réussir le soin.

- **Force** : la force initialement affectée à 0 est comprise entre 0 et 10. Elle augmente les dégâts que peut infliger un étudiant à son adversaire de 10% par point affecté. Ainsi, un étudiant avec une force de 2 "frappera" avec 20% de force en plus. Il fera 20% de dégâts en plus.

- **Résistance** : la résistance initialement affectée à 0 est comprise entre 0 et 10. Elle permet de diminuer les dégâts reçus de 5% par point affecté. Ainsi, un étudiant avec une résistance de 2 "absorbera" 10% des dégâts infligés.

- **Constitution** : la constitution initialement à 0 est comprise entre 0 et 30. Elle permet d'augmenter la constitution d'un étudiant en lui donnant des crédits ECTS supplémentaires. Ainsi, 10 points de constitution font augmenter les crédits ECTS initiaux à 40 (au lieu de 30).

- **Initiative :** l'initiative initialement affectée à 0 est comprise entre 0 et 10. Sur une zone de combat, l'étudiant qui a la plus forte initiative effectue son action, puis c'est au tour de l'étudiant qui a la seconde meilleure initiative etc… A titre d'exemple, les trois premiers étudiants à effectuer leur action peuvent appartenir à la même équipe.

[1] *Le moment culture… film des années 60*

- **Stratégie** : Chaque étudiant possède une stratégie qui peut être défensive, offensive ou aléatoire.

Chaque étudiant possède les actions suivantes :

- **Soigner un combattant allié**. Cela consiste à choisir un allié sur la même zone de combat ayant le moins de crédits ECTS. Pour le soigné, le nombre de crédits gagnés dépend de la dextérité du soignant et de la constitution du soigné.
  Un tirage aléatoire permet de savoir si le soin est réussi. Soit $x \in [0{,}100]$ un nombre tiré aléatoirement. Si $x \in [0\ ,\ 20 + 6 * dextérité\ du\ soignant]$ alors le soin est réussi. Dans un tel cas, le soigné engrange $E(y * (10 + consitution\ du\ soigné))$ crédits ECTS avec $y \in]0, 0.6]$ un nombre tiré aléatoirement et $E(z)$ la partie entière de z. Le gain obtenu ne peut en aucun cas dépasser la valeur $(30 + constitution\ du\ soigné)$

- **Attaquer frontalement**. Sur sa zone de combat, l'étudiant lance son *gobi* à l'ennemi qui a le moins de crédits ECTS.
  Un tirage aléatoire permet de savoir si l'attaque est réussie. Soit $x \in [0{,}100]$ un nombre tiré aléatoirement. Si $x \in [0\ ,\ 40 + 3 * dextérité\ de\ l'attaquant]$ alors l'attaque est réussie. Dans un tel cas, l'attaqué perd $E(y * (1 + coefficient\ dégât) * dégât\ de\ référence)$ crédits ECTS avec $y \in]0, 1]$ un nombre tiré aléatoirement et $E(z)$ la partie entière de z. Le dégât de référence vaut 10 et $coefficient\ dégât = \max(0, \min(100, 10 * force\ de\ l'attaquant - 5 * résistance\ de\ l'attaqué))$

- Soigner un partenaire constitue une stratégie défensive alors qu'attaquer frontalement est une stratégie offensive. Une stratégie aléatoire consiste pour un étudiant à évoluer d'une stratégie à une autre de manière aléatoire.

Les 20 protagonistes de chaque équipe sont hiérarchisés de la manière suivante :

- 15 ***étudiants*** avec les caractéristiques initiales présentées précédemment.
- 4 **étudiants *d'élite*** avec des caractéristiques initiales augmentées (Force +1, Dextérité +1, Résistance +1, Constitution +5, Initiative +1)
- 1 ***Maitre du gobi*** avec des caractéristiques initiales augmentées (Force +2, Dextérité +2, Résistance +2, Constitution +10, Initiative +2)

Déroulement du jeu

Etape 1 – Paramétrage des troupes

En début de partie, chaque joueur possède 400 points à distribuer à chacun de ses 20 combattants en les affectant aux caractéristiques Force, Dextérité, Résistance, Constitution, Initiative. Le joueur choisit également 5 combattants qui seront des réservistes (un réserviste peut être un étudiant, un étudiant d'élite ou le maître du gobi), les 15 autres se retrouveront donc sur le champ de bataille dès le début des hostilités. Cette étape est bien entendue cachée pour l'adversaire. Chaque étudiant aura également une stratégie de combat.

Etape 2 – Affectation des troupes sur le champ de bataille

A cette étape, chaque joueur décide de répartir ses 15 combattants sur les 5 zones de combat. Il répartit à sa convenance ses troupes : il peut par exemple choisir de ne mettre qu'un seul combattant sur une zone (un sacrifice) et d'en mettre 5 sur une autre zone. Cette étape est bien entendue cachée pour l'adversaire.

Chaque joueur doit affecter au moins un combattant à chaque zone.

Etape 3 – Mêlée

La bataille est lancée et les 5 zones de combats font rage. Au fur et à mesure, les crédits ECTS des étudiants peuvent augmenter ou diminuer.

> *D'un point de vue algorithmique, les étudiants d'un champ de bataille sont stockés dans une liste selon la meilleure initiative. Le premier de la liste commet son action puis repasse à la fin de cette même liste.*

Une mêlée s'achève avec le contrôle d'une des zones par un des joueurs (les adversaires ont été terrassés sur cette zone). Une trêve est donc déclarée et tous les combats sur les 5 zones sont arrêtés. Trêve pendant laquelle chaque joueur peut sournoisement faire certains mouvements de troupes à l'insu de son adversaire ; voir ci-après.

Etape 4 – Trêve & mouvements de troupes

A cette étape les 2 joueurs peuvent faire les actions suivantes de manière cachée :

- Affecter 1 ou plusieurs réservistes sur les zones de combats non contrôlées
- Si un joueur contrôle déjà une zone de combat, il peut redéployer ses combattant valides qui se trouvent sur cette zone vers d'autres zones de combats. Mais, un combattant doit rester sur la zone contrôlée pour maintenir l'ordre et l'influence du programme. Attention, un combattant qui est redéployé ne regagne pas de crédits ECTS. Il repart au combat avec ses blessures. Un combattant redéployé peut se voir attribuer une nouvelle stratégie.
- On ne peut pas redéployer de combattants d'une zone dont le combat n'est pas fini.
- Le joueur peut connaître le nombre de crédits ECTS par zone de combat.

Etape 5 – Cycle Mêlées – Trêves & mouvements de troupes

Les étapes 3 et 4 se répètent jusqu'à ce qu'un joueur contrôle au moins 3 zones et est donc déclaré vainqueur.