

AVANTHIKA NANDAKUMARAN & THEODORE COMAN

LYCEE MARTIN LUTHER KING 2015-2016

PROJET ISN

HAGALAB

UN JEU DONT VOUS ETES LES HEROS

Table des matières

[INTRODUCTION 2](#_Toc452698195)

[PORTABILITE 3](#_Toc452698196)

[SCM : GIT 3](#_Toc452698197)

[BASE DE DONNEES 4](#_Toc452698198)

[MUSIQUE 5](#_Toc452698199)

[IHM 6](#_Toc452698200)

[DEPENDANCES 6](#_Toc452698201)

[LICENCES 7](#_Toc452698202)

# INTRODUCTION

Cette année, pour notre projet d’ISN, nous avons décidé de concevoir un jeu dont vous êtes le héros, c’est-à-dire un jeu narratif qui met le joueur dans une situation donnée et qui lui propose plusieurs choix d’action. L’histoire évolue en fonction des choix du joueur.

L’idée de concevoir un jeu de ce genre est venue à Théodore pendant qu’il rangeait sa bibliothèque, en tombant sur un livre-jeu. Après avoir fait part de son idée à Avanthika, cette dernière partagea son enthousiasme.

Le but de notre projet est de créer un jeu dont vous êtes le héros, avec une interface graphique, et des musiques.

Au début de notre projet, nous avons envisagé d’utiliser simplement le langage de programmation Java et l’environnement de travail Eclipse pour arriver à nos fins. Sauf que nous avons rapidement abandonné cette idée car la structure du programme deviendrait trop labyrinthique. En effet, avoir recours à du hardcode devient tout simplement illisible lorsque le nombre de possibilités s’élève.

Nous avons donc choisi d’utiliser une base de données pour « stocker » les différents choix possibles du joueur et les différentes conséquences découlant de ses choix.

Le premier problème qui s’est posé lorsque nous avons commencé à travailler fut le suivant : comment pouvoir travailler parallèlement de manière efficace et comment mettre en commun notre travail. Pour travailler ensemble depuis notre domicile nous avons utilisé l’appel Skype et le partage d’écran. Pour mettre en commun nos avancées dans le projet nous avons utilisé le logiciel de gestion de version GIT.

Nous avons commencé le projet par choisir le thème du jeu dont vous êtes le héros. Nous avons convenu que l’intrigue se déroulera à l’intérieur d’un bâtiment étrange, composé de plusieurs étages avec des laboratoires de recherche au sous-sol et que le but du joueur est de s’en échapper.

Après cela, nous avons ébauché l’histoire du jeu, puis nous nous sommes réparti les tâches. Avanthika s’est occupée de l’interface graphique et de l’insertion du script dans la base de données. Pour ma part, je me suis occupé de la création et de la connexion à la base de données ainsi que de la musique. Nous avons travaillé ensemble le script du jeu.

# PORTABILITE

Une des difficultés initiales du projet a été le travail sur plusieurs PC différents (à l’école et à domicile). Il est vrai que l’on peut garder la trace des fichiers de code modifiés et les copier sur une clef USB à la fin du travail sur le PC 1 pour les remettre par la suite sur le PC 2, mais c’est fastidieux et sujet aux erreurs.

La volonté d’utiliser des fonctionnalités présentes dans Java 8 (que nous n’avons finalement pas utilisées…) nous a conduit à chercher un environnement portable. Nous avons découvert le site <http://portableapps.com/> et nous avons pu créer une clef USB permettant de travailler en autonomie sur n’importe quel PC. Nous avons installé sur la clef USB les éléments suivants :

* Le SDK Java 1.8 (32 bits qui s’exécute à la fois sur Windows 64 et 32 bits)
* Eclipse Luna 32 bits
* Notepad++
* Git
* 7-Zip
* Window Builder, un plug-in Eclipse permettant de créer des IHM en mode visuel

Le projet Eclipse peut être transféré entre deux clefs USB contenant les éléments énumérés ci-dessus sans problème. Si l’on ferme Eclipse sur la première clef USB, on copie le projet et on ouvre Eclipse depuis la deuxième clef tout fonctionne sans problème. Pour pouvoir avoir ce comportement, Eclipse utilise le JDK installé sur la clef.

# SCM : GIT

Nous nous sommes rapidement confrontés au problème de sauvegarde d’un fichier en vue de le réintégrer plus tard dans un autre environnement. Il fallait que l’on décide à l’avance sur quels fichiers chacun d’entre nous pouvait travailler et il fallait faire souvent (surtout en cas d’oubli de sauvegarde/intégration des modifications) des comparaisons de taille et de date de modification pour les fichiers.

Il nous fallait un SCM (Source Code Management, ou encore un logiciel de contrôle de versions). Nous avons choisi Git pour plusieurs raisons :

* Il est « à la mode », il est devenu le SCM le plus utilisé
* Il est très puissant mais également assez simple d’utilisation si on se résume à une utilisation basique
* Il existe une version portable disponible sur portableapps.com
* Il y a GitHub qui permet de travailler en équipe gratuitement (si on choisit de créer un projet public)
* L’aide en ligne est très bien faite

Après quelques recherches, nous avons trouvé comment ajouter une version initiale à GitHub (https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/). Une fois le projet sur GitHub, il est possible de le cloner sur la deuxième clef USB (ou sur tout autre disque dur) et de travailler facilement :

1. On travaille sur le projet sur la clef USB 1
2. On ouvre Git Bash sur la même clef et on navigue à la racine du projet
3. On fait git add pour rajouter nos changements à la staging area
4. On fait un git commit
5. On fait un git push
6. Depuis la clef USB 2, on fait un git fetch suivi d’un git pull pour récupérer les modifications effectuées sur la clef USB 1

# BASE DE DONNEES

La base de données utilisée est HSQLDB, choisie à cause de sa légèreté extrême (aucune installation nécessaire, téléchargement rapide, bibliothèque qui pèse moins de 2 Mo à rajouter aux dépendances). Créer une base de données en HSQLDB est très simple et la base est portable (par copie de tous les fichiers créés).

Nous avons lu des tutoriels sur les bases de données et avons décidé d’utiliser un strict minimum pour commencer, à savoir les requêtes SELECT (pour naviguer d’un état vers un autre) et UPDATE (pour mettre à jour la base de données depuis l’IHM).

Une connexion (java.sql.Connection) à la base est ouverte lorsque l’application démarre. La connexion est fermée lorsque l’application est fermée (au déclenchement de l’événement WindowClosing). La classe Db regroupe les accès à la base de données ; pour une plus grande facilité d’utilisation, les méthodes de cette classe sont statiques. A chaque fois que l’on veut exécuter une requête, la « recette » est la suivante :

* On écrit l’ordre SQL à exécuter (un String)
* On demande un objet java.sql.PreparedStatement à la connexion, en lui passant l’ordre SQL
* On précise à l’objet PreparedStatement le type et la valeur des paramètres. Ces paramètres apparaissent en tant que points d’intérrogation dans le String représentant la requête SQL. Des méthodes de type setAAA(int position, AAA value) permettent de préciser que le « ? » numéro « position » est de type AAA et qu’on lui associe la valeur « value ».
* On demande au PreparedStatement d’exécuter la requête (executeQuery() pour les SELECT, executeUpdate() pour les INSERT, UPDATE ou DELETE)
* Si nous avons à faire avec une requête SQL de type SELECT, la méthode executeQuery() retourne un objet de type java.sql.ResultSet. Nous faisons l’extraction des données retournées dans une boucle tant que la méthode hasNext() de l’objet ResultSet retourne true.
* Enfin, le PreparedStatement et le ResultSet sont systématiquement fermés dans la partie catch du bloc try..catch (ou dans un try-with-resources, plus compact).

Nous n’avons pas eu à utiliser la notion de transaction : en effet, toute base HSQLDB est créée par défaut avec un commit automatique en place (Autocommit).

# MUSIQUE

Les morceaux de musique du projet sont originaux : nous avons utilisé le logiciel Sibelius et, en cours de MAO, le logiciel Cubase pour les composer.

Nous avons initialement essayé de voir comment rendre la musique en Java, mais c’était très compliqué (notamment la gestion d’un thread séparé, différent de celui dans lequel l’IHM s’exécute…) et il fallait travailler obligatoirement avec des fichiers wav volumineux. Des recherches nous ont permis de trouver une bibliothèque appelée TinyMusic (<https://github.com/kevincennis/TinyMusic>) qui :

* Simplifie le codage (il n’est plus nécessaire de penser au multi-threading…)
* Sait lire d’autres formats que les volumineux wav (exemple : le format ogg). Du coup, nous avons converti en ligne (<http://audio.online-convert.com/convert-to-ogg>) nos fichiers de musique wav au format ogg.

# IHM

Nous avons essayé de suivre les recommandations de codage correct (openclassrooms, developpez.com, …). Notamment, nous nous sommes efforcés de :

* Avoir un code lisible (noms significatifs pour les classes, les variables, les méthodes…)
* Découper le projet correctement : comme il y a une base de données, nous avons créé un package dédié pour contenir les lectures/écritures base de données ; un autre package contient des POJO (Plain Simple Java Objects) : des classes qui sont utilisées pour véhiculer les informations entre la base et l’IHM ; un dernier package contient l’IHM elle-même.
* De ne pas avoir de code dupliqué dans le projet (si la même action est réalisée depuis plusieurs endroits différents, on crée une méthode qui va être appelée plusieurs fois, éventuellement avec des arguments différents)

La gestion des évènements joue un rôle important dans l’IHM. Nous avons appris comment ajouter des listeners à des composants Swing pour implémenter différentes actions liées au clicks de la souris. Nous avons également utilisé les Generics que Java met à notre disposition : en effet, lorsque le besoin s’est présenté, nous avons utilisé plutôt une java.util.List<String> qu’une List tout court pour désigner une liste qui contiendra exclusivement des String.

# DEPENDANCES

Nous avons vu que Eclipse propose de créer des projets de type Maven, mais nous avons trouvé que c’était trop compliqué pour notre projet. Nous avons actuellement uniquement deux dépendances : HSQLDB (un fichier jar) et TinyMusic (4 fichiers jar). Les cinq fichiers ont été copiés dans un répertoire « lib ». Ces dépendances sont spécifiées au niveau des propriétés du projet (Classpath) et les fichiers jar nécessaires ont été mis sous Git.

# LICENCES

Tous les outils nécessaires pour mener à bien notre projet (à l’exception de Microsoft Word pour l’écriture de ce document, du système d’exploitation Windows et des logiciels Sibelius et Cubase qui ont permis l’écriture de la musique) sont des [logiciels libres](https://fr.wikipedia.org/wiki/Logiciel_libre) [open source.](https://fr.wikipedia.org/wiki/Open_source)