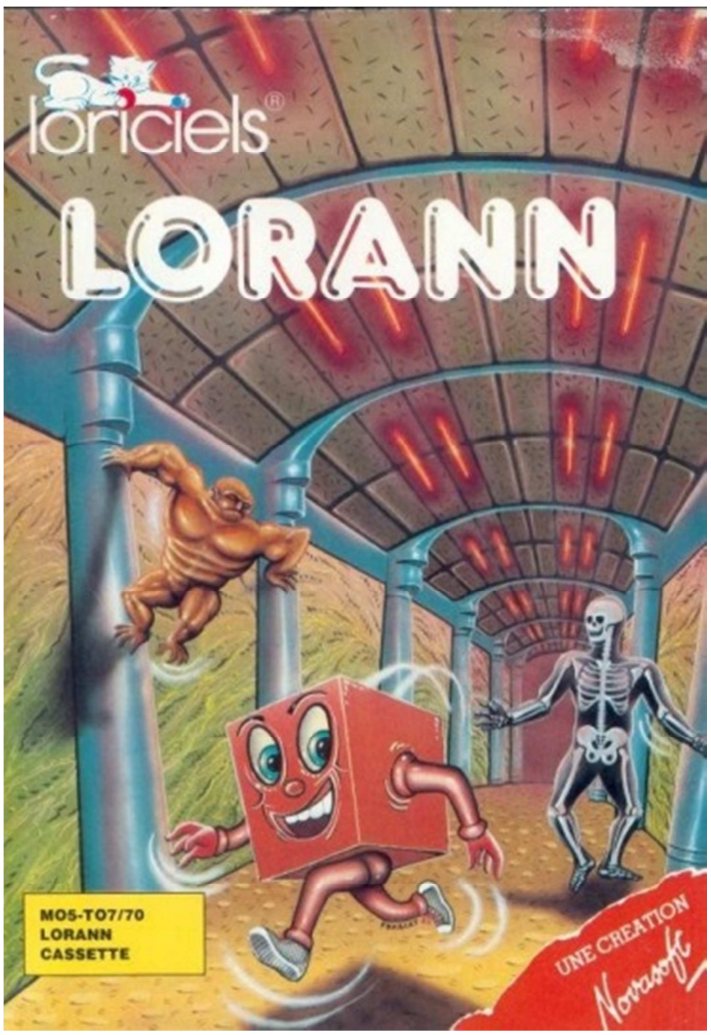


Lorann, Maître du Sortilège, le Monde de Nova-Ann a besoin de ta force et de ta ruse !

Libère les cent-une cryptes possédées par le Masque de Nékron, en touchant la Bulle Energie qui se trouve dans chacune d'elles. Rapporte les éphémères Idoles de la Vie à la Caverne, chacune te vaudra deux Vies supplémentaires. Ramasse les fabuleux trésors qui t'attendent et ta richesse sera immense ! Evite les attaques des quatre épouvantables Démons créés par l'infâme Nékron, en utilisant habilement ton très puissant Sortilège multicolore qui obéit à ta volonté.

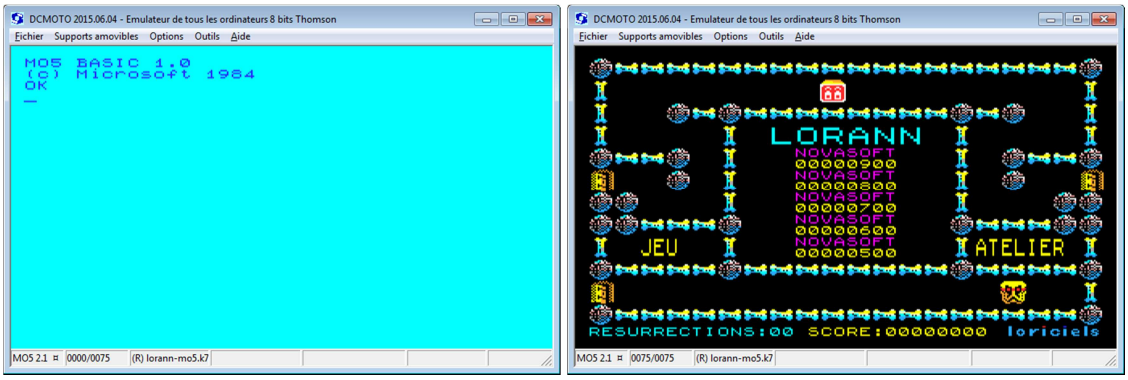
La gloire t'attend, Lorann... Que le Sortilège te protège !

CONTEXTE DU PROJET

Il s'agit de réécrire un vieux jeu (1985) LORANN de Loriciels. Ce jeu a été initialement édité pour MO5 et TO7 (des noms qui n'évoquent sans doute rien pour vous, mais qui pour les informaticiens barbus représentent une partie de leur enfance et leur découverte de l'informatique)

LORANN est un jeu de type PacMan, à savoir qu'un personnage évolue de case en case en essayant de ramasser un maximum d'objets sans se faire rattraper par 4 monstres dans 101 niveaux. Pas de super pacgom, mais une boule de feu que le personnage peut lancer pour tuer ses agresseurs. La difficulté étant qu'il ne dispose que d'une seule boule de feu. Le seul moyen d'en obtenir une nouvelle est de tuer un monstre ou de ramasser la précédente. Rien d'extraordinaire donc, mais en 1985, ce jeu a fourni à plusieurs enfants (dont je fais partie) de nombreuses heures de plaisir.

Le mieux est que vous essayiez le jeu pour mieux le comprendre. Pour cela il vous suffit d'installer un émulateur Thomson et de lancer la ROM du jeu. Les 2 sont disponibles dans les ressources.



http://dcmoto.free.fr/emulateur/index.html

http://dcmoto.free.fr/programmes/lorann/index.html

FONCTIONNALITES DEMANDEES

Il ne s'agit pas de reproduire à l'identique cet abandonware. Aussi, pour accéder aux différents niveaux du jeu (au nombre de 101), l'utilisateur doit déplacer le personnage (zone jeu) dans une salle permettant l'accès aux niveaux. Toute cette partie n'est pas demandée.

De plus, le jeu intègre un éditeur de niveau (zone atelier) incluant des contrôles permettant de valider la jouabilité et la viabilité du niveau. Cette fonctionnalité n'est, elle non plus, pas demandée.

Il s'agit de réaliser quelques niveaux (5 suffiront) accessibles par paramétrage dans le code. C'est-à-dire pour être plus explicite, qu'il n'est pas nécessaire que vous implémentiez une fonctionnalité permettant de changer de niveau au sein même du jeu. L'accès à un niveau différent pourra se faire via un paramètre dans votre code, un fichier de configuration ou un enregistrement dans votre base de données.

Les niveaux devront impérativement être stockés dans une base de données.

Beaucoup d'éléments graphiques sont présents dans le jeu. Certains représentent des éléments de décors des niveaux, d'autres des items à ramasser ou éviter et certains des éléments mobiles. Il est inutile de tous les implémenter.

Seuls les éléments suivant du jeu sont attendus :

	Lorann, le personnage
	Le sortilège multicolore
	La bulle d'énergie
	La porte de sortie du niveau
	Les éléments infranchissables de décors
	Les 4 épouvantables démons
	Les bourses permettant de gagner des points

L'ensemble des sprites est présent dans les ressources au format PNG (32x32 pixels).

ARCHITECTURE DE DEPART FOURNIE

Une architecture de base vous est fournie afin que vous puissiez vous concentrer sur la conception du jeu en lui-même.

Cette base contient :

UN PROJET MAVEN SOUS ECLIPSE

5 modules avec toutes les dépendances préconfigurées

- `controller`
- `model`
- `view`
- `contract`
- `main`

Le plugin Junit préconfiguré ainsi qu'un test d'exemple implémenté sur une des classes du module `model`.

Le plugin JXR préconfiguré afin de produire une documentation des fichiers sources (http://maven.apache.org/jxr/maven-jxr-plugin/).

Le plugin Shade préconfiguré permettant une génération d'un Uber-Jar (http://maven.apache.org/plugins/maven-shade-plugin/). *Ceci permet de ne générer qu'un seul JAR contenant l'ensemble du projet.*

Le plugin JavaDoc préconfiguré.

UN CODE SOURCE FONCTIONNEL

Le code fourni en exemple produit un HelloWorld dans une `JFrame`. En fonction de la touche enfoncée par l'utilisateur (D, G, F ou I) le texte affiché est différent (Allemand, Anglais, Français ou Indonésien).

Le DP Observer déjà intégré dans les modules `view` et `model`.

Une connexion à la base données MySql paramétrable dans un fichier de configuration (`model\main\src\ressources\model.properties`). Le DP Singleton est déjà implémenté.

Une classe `DAOEntity` abstraite permettant de gérer de nouvelles tables, ainsi qu'un exemple d'utilisation (`DAOHelloWorld`) utilisant des procédures stockées.

Une énumération des fonctions présentes dans le module `controller`, ainsi qu'une classe statique dans le module `view` pour assurer la conversion ordre utilisateur reçu => ordre envoyé au module `controller`.

Une JavaDoc de tout le code fourni.

DES DIAGRAMMES UML DU PROJET

- Un diagramme de composants
- Un diagramme de packages
- Un diagramme de classes par package (5 en tout)

CONTRAINTES

L'utilisation de Java, Maven, Git et Junit est obligatoire.

Aucun Framework graphique autre que Swing n'est autorisé.

Début du projet : Lundi 13 Juin.

Fin du projet : Mercredi 22 Juin.

L'équipe de développement sera constituée de 4 membres maximum.

Un chef de projet sera désigné. Il aura la responsabilité de :

- la bonne répartition de la charge entre les membres de l'équipe
- les livrables (délai et contenu)
- la prise de rendez-vous facultatifs (mais conseillés) avec votre tuteur

Aucune requête SQL ne devra être présente dans le code Java. L'intégralité des appels devra se faire via des procédures stockées.

LIVRABLES

DOCUMENTATIONS – MARDI 21 JUIN – 09H00

- JavaDoc complet de votre projet (tests compris)
- JXR complet de votre projet (tests compris)
- Rapport SureFire de votre projet
- Diagramme de composants
- Diagramme de packages
- Diagramme de classes (un par package)
- Diagramme de séquence (autant que vous en jugerez utiles pour comprendre et expliquer le fonctionnement de votre programme)
- Un rapport GIT permettant d'identifier la production de chacun des membres de l'équipe.
- Tous les autres documents que vous jugerez nécessaires (MCD, procédures stockées, autres diagrammes, commentaires, …)

Toute la documentation ainsi que le code devra être écrit en anglais.

DEPOT GIT CONTENANT LES SOURCES DU PROJET - LUNDI 20 JUIN – 17H00

Un accès sera ouvert en tant que lecteur sur le dépôt GIT de votre équipe.

SOUTENANCE – MERCREDI 22 JUIN

La présentation orale de ce projet se fera en anglais et durera 15 minutes. Cette présentation devra être fonctionnelle et aucunement technique.

L'intégralité de votre support de présentation (PPT ou autre) si vous décidez d'en utilisez un, devra aussi être en anglais.

A l'issu vous serez interrogés pendant 15 minutes en français sur les parties techniques et fonctionnelles de votre projet.

EVALUATIONS

Ce projet est couvert par trois évaluations :

- Une note individuelle (Oui / Non)
- Deux notes de groupe (A / B / C / A)
    - Cahier de tests (le JavaDoc, le JXR de vos tests et le rapport SureFire)
    - Projet (Fonctionnel/Conception/Code)

CONSEILS

L'architecture de départ qui vous est fournie n'est absolument pas obligatoire. Si vous préférez utiliser la vôtre, vous le pouvez. Cependant l'intégralité des plugins Maven nécessaires à la génération automatique des documents demandés est déjà installée, aussi vous risquez de perdre un temps précieux à le faire vous-même.

La journée du mardi 21 doit être en majeure partie consacrée à la préparation de la présentation du lendemain.

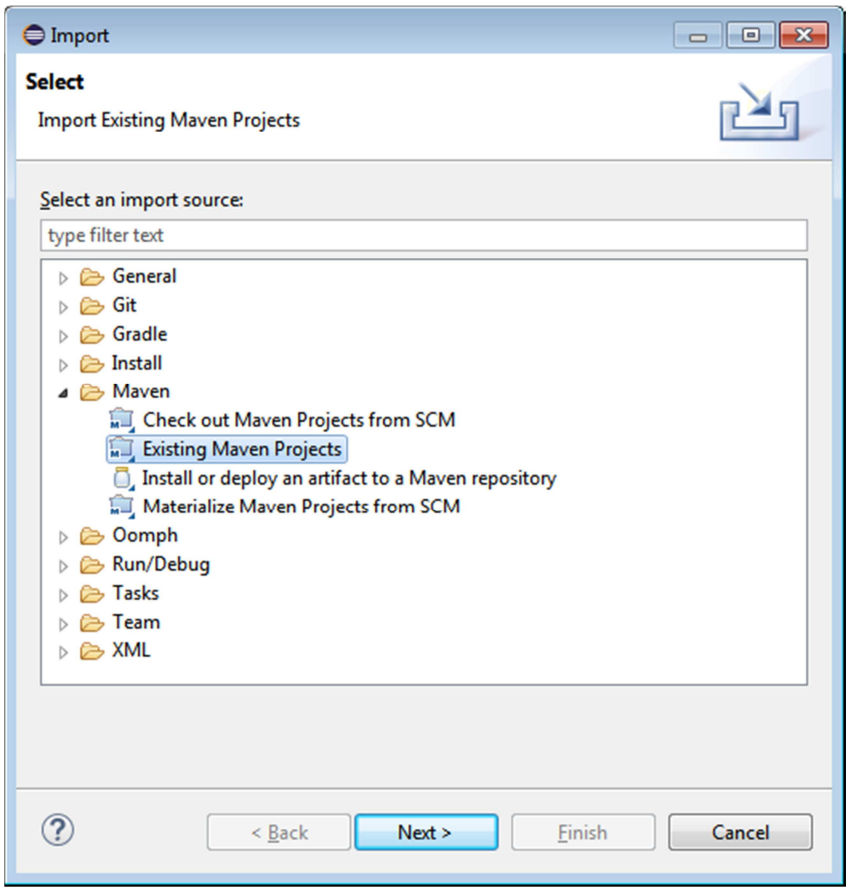
La bonne réussite de ce projet passe par une phase de conception importante. Ne sous-estimez pas cependant le temps nécessaire à l'écriture de votre code. Fixez-vous dès le début une deadline de démarrage de la phase de réalisation.

Faire du TDD semble être aussi un bon choix, vous permettant de garantir la livraison d'un projet minimal fonctionnel et testé. Il est toujours plus difficile d'écrire les tests à postériori.

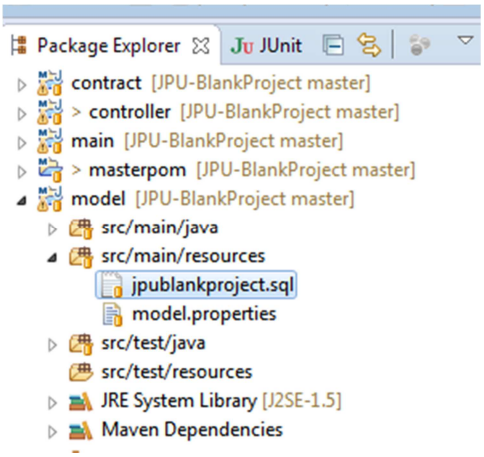
COMMENT INSTALLER L'ARCHITECTURE DE DEPART ?

Décompressez le zip présent dans les ressources.

Dans Eclipse Mars 2, importez le projet dans un nouveau workspace.



Dans le module `main`, ouvrez la ressource `jpublankproject.sql` et exécutez le script SQL dans MySql. Inutile de créer la base de données, le script se charge de tout.



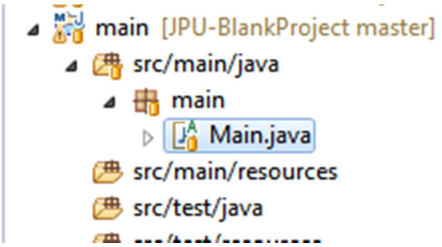
Toujours dans le module `main`, ouvrez la ressources `model.properties` et modifiez les paramètres de connexion à la base de données MySql si nécessaire. Vous avez peut-être par exemple modifié le port d'écoute de votre serveur ou le login et mot de passe.

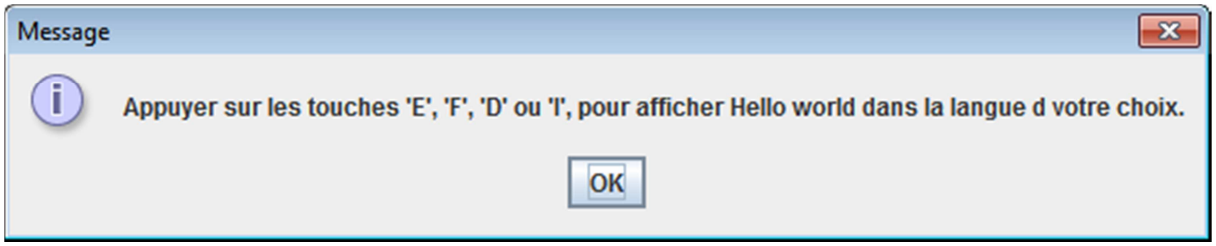
```
#DB Connection Properties
url=jdbc:mysql://localhost/jpublankproject?autoReconnect=true&useSSL=false
login=root
password=
```

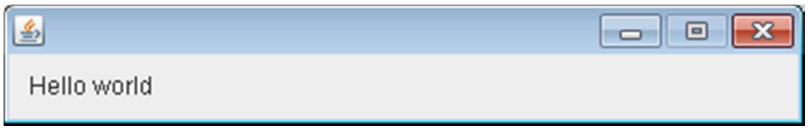
Lancez ensuite un `build package` puis un `build site`.

Vous pourrez ensuite parcourir l'ensemble des documentations.

Lancez le programme en faisant un run sur la classe `Main` du module `main`.







Voilà c'est terminé, tout est installé.

Il ne vous reste plus qu'à renommer le projet et configurer votre dépôt Git pour que vos collègues puissent travailler.