

Projet Java / POO / UML

CONTEXTE DU PROJET

Il s'agit de réécrire un vieux jeu (1984) Boulder Dash Peter Liepa et Chris Gray.

Le principe du jeu consiste à ramasser un nombre défini de diamants pour ouvrir un passage vers le niveau suivant. Le personnage, surnommé Rockford, doit creuser la terre pour se frayer un chemin. Il devra faire attention à ne pas se faire écraser par un rocher ou un diamant mais également ne pas se faire toucher par les ennemis2.

Certains niveaux n'ont pas de diamants. Le joueur doit alors les générer lui-même en faisant tomber un rocher/diamant sur un ennemi. Il devra préparer le terrain pour y arriver. Lorsqu'on élimine un ennemi de cette façon, celui-ci explose et fait apparaître des diamants qui à leur tour, tomberont peut-être sur d'autres ennemis créant ainsi une réaction en chaîne. Le joueur peut également utiliser cette technique pour casser les murs et franchir des zones bloquées.

Le mieux est que vous essayiez le jeu pour mieux le comprendre. Pour cela il vous suffit d'aller sur le site http://www.retrogames.cz/play_232-NES.php (pas au Cesi car étant donné qu'il s'agit d'un jeu cela sera bloqué par le proxy, mais une fois chez vous n'hésitez pas à redécouvrir ce jeu mythique).



FONCTIONNALITES DEMANDEES

Il ne s'agit pas de reproduire à l'identique cet abandonware. Aussi, pour accéder aux différents niveaux du jeu, l'utilisateur doit déplacer le personnage sur une carte. Toute cette partie n'est pas demandée.

De plus, le jeu intègre un mode 2 joueurs. 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.

L'ensemble des sprites est présent dans les ressources au format PNG (16x16 pixels).

Projet Java / POO / UML

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 contract.

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.

Le plugin SureFire préconfiguré afin de produire la documentation des résultats des test (<https://maven.apache.org/surefire/maven-surefire-plugin/>).

UN CODE SOURCE FONCTIONNEL

Le code fourni en exemple produit 3 interrogations de la base de données et les affiche dans une popup.

Une connexion à la base données MySql. Le DP Singleton est déjà implémenté. Les requêtes ne sont pas stockées dans le code Java mais dans des procédures stockées.

Une JavaDoc de tout le code fourni.

Un test Junit déjà codé pour le test de la classe Example.

DES DIAGRAMMES UML DU PROJET

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

CONTRAINTE S

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

Aucun Framework graphique autre que Swing n'est autorisé.

Début du projet : Mercredi 14 Juin.

Fin du projet : Lundi 26 Juin.

Projet Java / POO / UML

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 – VENDREDI 23 JUIN – 17H00

- 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 – LUNDI 26 JUIN

L'évaluation de votre soutenance n'est pas prise en compte dans l'UE Conception d'applications orientées objets.

EVALUATIONS

Ce projet est couvert par trois évaluations :

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

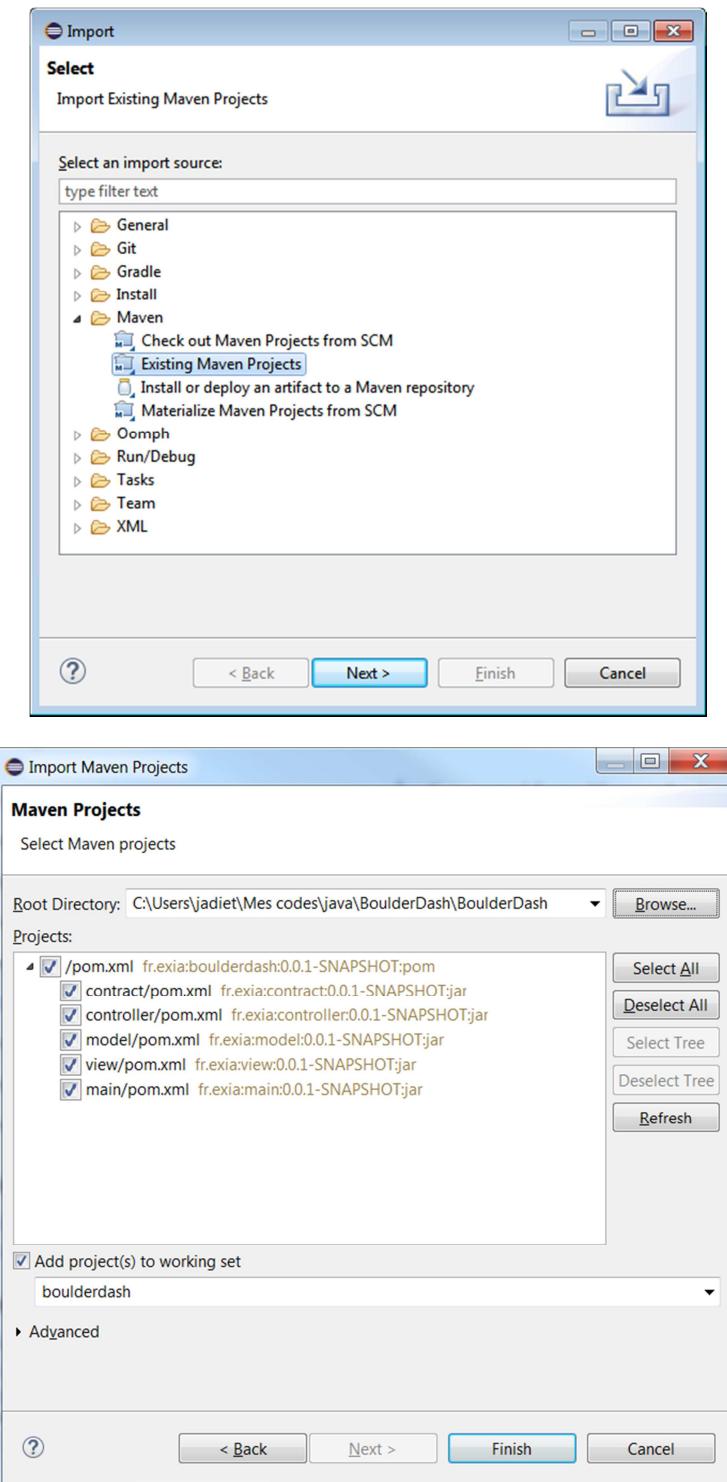
Projet Java / POO / UML

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, importez le projet dans un nouveau workspace.



Téléchargez le script SQL de création de la base de données et exécutez le dans PhpMyAdmin.

Lancez ensuite un :

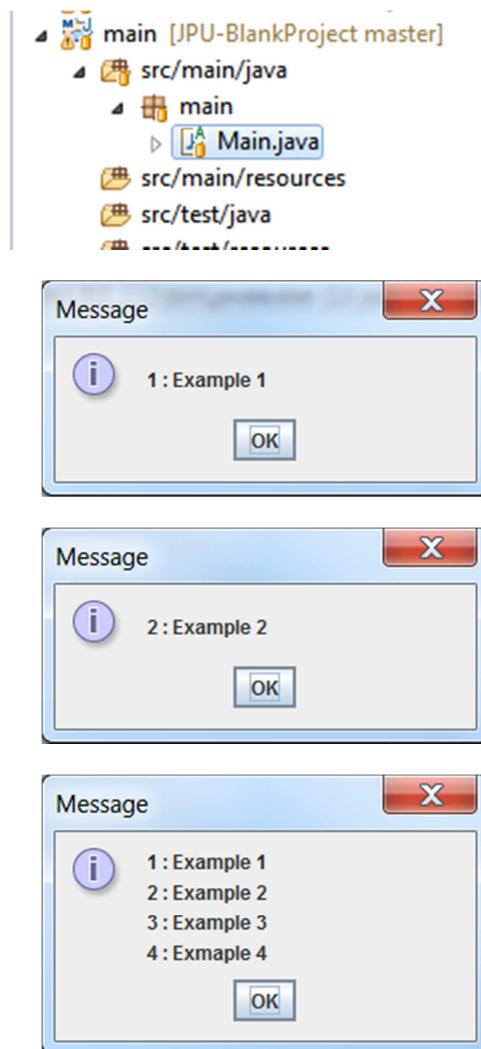
Projet Java / POO / UML

- update project
- build compile
- build package
- build install
- build site
- build site:stage

Si vous avez des erreurs lors de ces différents build, c'est très certainement que je JDK n'est pas correctement configuré dans eclipse.

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.