DOSSIER IUT CRAFT

ETUDE DE CONCEPTION:

Nous voulions avec notre projet faciliter la vie des joueurs Minecraft avec une application permettant de rechercher des informations tels que les recettes de crafts ou de directement consulter le wiki.

Ce que nous avions en tête au début du projet était une application permettant de faire des recherches même avec quelques lettres pour pouvoir trier les résultats, récupérer la recette des items sélectionnés et pouvoir être redirigés sur le site internet du wiki.

Pour ce faire nous avons décidé pour le développement de notre application d'utiliser du JAVA jdk 19 ainsi que des fichiers ressource comme des fichiers .json, .wav, .gif, etc.

Nous avions un mois pour développer toutes les fonctionnalités cités plutôt ainsi que le dossier que vous lisez et que les diagrammes de cas d'utilisation et de classes.

EXPLICATION DU PROJET

IUT Craft et une application outils au jeu culte développer par le studio Mojang, Minecraft pour sa version (1.12) ses principaux objectifs sont :

-Permettre de rechercher un bloc et avoir son craft (façon de le fabriquer) lorsque le bloc existe bien.

-Permettre de filtrer sa recherche sur des blocs constructibles schématiquement ou sans position précise.

-permettre d'afficher la liste dans un ordre alphabétique ou inverse

-Permettre la passerelle avec un wiki officiel du jeu .

PROBLÈMES TECHNIQUES :

Au niveau des problème que nous avons rencontré il y'a eu premièrement :

-L'inclusion du fichier .json car certaines bibliothèques n'était pas trouvées et l'IDE ne nous proposait pas l'installation des dépendances associées, ce qui nous a fait perdre un petit peu de temps en début de projets.

-Le deuxième problème que l'on a rencontré c'est avec notre première utilisation du JavaFX sur le projet car il fonctionner que pour le créateur du projet mais pas pour l'ensemble de l'équipe mais ce problème a était assez rapidement régler en se rendent compte qu'il y avait plusieurs version du JDK 19 nous avons pu installer la même afin de pouvoir push et commit sur le même projet.

-Le troisième gros problème que nous avons rencontré est la création d'une nouvelle fenêtre pour la recette du bloc choisi. N'étant pas très avancé en terme de javaFX, nous avons eu du mal à

comprendre comme ouvrir une nouvelle fenêtre nous permettant d'afficher les informations voulues du bloc sélectionné.

-Le quatrième gros problème que l'on a rencontré, c'est la création de test javaFx n'arrivant pas a configurer le projet et par manque de temps nous avons décidé de passer outre et de nous contenter de faire des tests Junit

ÉCART AVEC LES PRÉVISIONS :

Au départ nous avions prévu une application terminal qui nous afficherait la liste de tous nos bloc et l'utilisateur pourrait choisir son bloc pour voir sont craft dont le bloc serait rentrer grâce à un scanner il y aurait eu bien sûr un algorithme de tri alphabétique lors de l'affichage de la liste des blocs et aussi nous aurions inclut des exceptions si le bloc n'existe pas et un thread qui permet de mettre une musique de fond tant que l'application et en cours.

Et donc les écart avec les prévisions sont assez positif car nous avons amélioré nos prévisions :

- Nous avons utilisé une interface graphique avec JavaFx qui nous permet d'avoir des boutons pour nos actions et surtout d'avoir un réel style pour l'application comparé au terminal.

- Nous avons rajouté un thread qui permet d'ouvrir une page web selon la recherche de votre bloc uniquement si vous faites le choix dans les filtres de se renseigner sur ce bloc dans le wiki du Jeux.

- Nous avons utilisé un design pattern décorateur qui nous permet de décorer notre bloc Minecraft afin de déterminer si les blocs ont un schéma précis pour être créer ou non.

- Nous avons effectué des test automatique

- Nous avons ajouté des filtre de recherche pour ce qui sera cherché dans la barre de recherche sera en raccord avec l'attente de l'utilisateur les filtres sont (craft schématique,craft non schématique,ordre alphabétique,ordre inverse)

- Nous avons créé une nouvelle fenêtre affichant le nom du bloc, les ingrédients nécessaires et si il a une recette ou non. S'il en possède une, on affiche sa recette si non on ne l'affiche pas.

BILAN TECHNIQUE:

Lors de ce projet nous devion respecter un cahier des charges fonctionnelles donnée au préalable tout en ayant un certain degrés de liberté voici ci dessous une explication de ce que l'on a pu respecter ou pas et pourquoi :

-Les classes abstraites : nous n'en avons qu'une seule qui est un décorateur (Design Pattern) qui implémente une interface afin de décorer (ajouter une fonctionnalité) sans changer l'objet Bloc.

- Interfaces : Nous avons utilisé une interface pour avoir une généricité sur les éléments présents dans l'objet Bloc afin d'utiliser le décorateur.

- Collection : Nous avons utilisé plusieurs collections notamment des Array List contenant des String qui nous permettait de stocker les éléments de notre fichier .json pour ensuite pouvoir les exploiter , nous avons aussi utilisé des HashMap qui nous on était très utile car un fichier .json fonctionne avec le fait d'avoir une clé et une valeur.

-Itérateur : Nous avons utilisé des itérateur notamment pour parcourir le contenue de nos collection donc les listes ça nous permettait aussi notamment de pouvoir voir les élément de notre json pour y placer nos conditions.

-Exception : Nous avons utilisé des exception qui était dans tous les cas obligatoire lorsque l'on va parser notre .json car il va aller chercher dans le fichier et si il n'existe pas il lève une exception , nous avons aussi mis des exception si une condition vis à vis de l'utilisation de l'application n'était pas respecter.

-Threads: Nous avons créé deux classes qui étendent la classe threads qui nous permettent d'ouvrir et afficher une page Web et l'autre de lancer une musique de fond une fois l'application lancée.

-Généricité : Nous n'avons pas entièrement respecté la généricité car notre interface ne contient que les méthodes de Bloc car il n'y a pas d'autre classe qui possède les mêmes propriétés.

-Algorithme de tri : Nous avions fait deux algorithmes de tri qui nous permettent d'avoir notre liste soit dans l'ordre alphabétique soit l'inverse de l'ordre alphabétique on a donc pu créer des filtre pour l'application à partir de cela.

Les Test/Test FXML : Nous avons réussi à tester nos différentes classes mais nous n'avons pas réussi a faires des test FXML du à des problème de bibliothèque que avions rencontrés.

MESURE D'AMÉLIORATION :

Pour Améliorer notre projet on pourrait :

-Rendre le code de la classe controller de l'application plus simple a la lecture (essayer de le refactorer) surtout pour les conditions privilégiées des Switch case au if,else .

-Rendre notre page JavaFX responsive comme ça elle sera compatible comme nous le souhaitions sur tout type d'écran.

-Améliorer le rendu graphique et esthétique de notre application pour qu'elle soit plus impactante a l'oeil pour donner envie d'être utilisée.

-Améliorer les tests pour faire des tests incluant javaFx et apprendre à bien configurer le projet.