NFA032 – Compte-rendu de projet et mode d'emploi

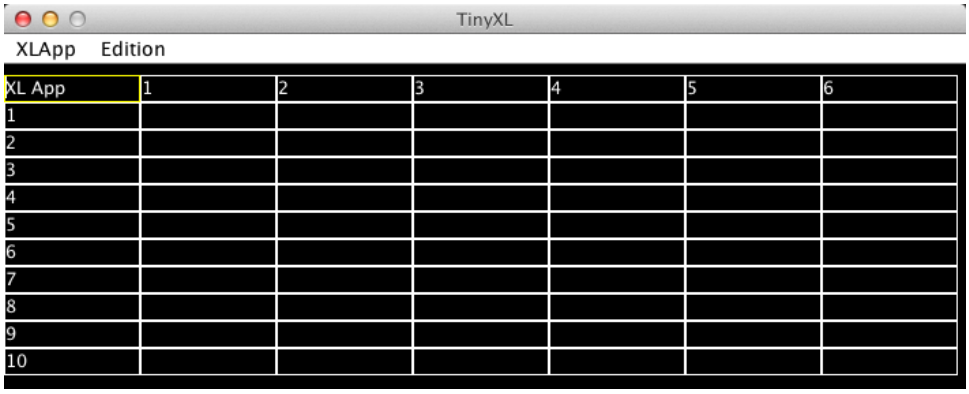
"TinyXL"

Structure du projet :

- **TinyXL** : la classe contenant le "main". Instancie **TinyXLCore**.
    - **TinyXLCore** : la classe principale. Contient la fenêtre et instancie toutes les autres classes nécéssaires, dont 11 **Line** de 7 **Cell** (pour les 10 lignes de 6 cellules requises par le projet).
        - **TinyXLMenu** : la classe qui gère les menus associés à **TinyXLCore**.
        - **CopyPaster** : une classe abstraite qui gère le copier/coller.
        - **Line** : une ligne de 7 cellules.
            - **Cell** : la classe des Cellules, qui contient la plupart des fonctions du programme relatives aux calculs.
- **Argument** : une classe abstraite pour toutes les variables relatives aux calculs :
    - **Operation** : sous-classe pour les opérations Somme et Moyenne,
    - **Value** : sous-classe pour les valeurs simples,
    - **Operator** : sous-classe pour les opérateurs (+ - / =),
    - **Position** : sous-classe pour les références relatives ou absolues à d'autres cellules.
- **Printable** : une interface qui donne à tous les Arguments une méthode pour s'écrire en String.
- **TerribleMistake** : la classe des exceptions.
- **PositionTest** : une classe de test JUnit4 pour Position.

Execution : le programme instancie **TinyXLCore**, qui s'occupe d'instancier toutes les cellules. Celles-ci contiennent la plupart des méthodes concernant les calculs. Les deux méthodes principales sont **registerFormula**, qui parse le contenu d'une **Cell** en plusieurs **Arguments**, et **resolveFormula**, qui s'occupe d'afficher les résultats.

Mode d'emploi :



Le programme permet l'exécution de la totalité des fonctions demandées par l'énoncé du projet.

Il y a cependant quelques petites différences :

- les **coordonnées des cellules** sont obtenues par l'appel à la ligne (y) et la cellule (x).
- les **références absolues à d'autres cellules** sont appellées par :

$(y, x)

- les **références relatives à d'autres cellules** sont appellées par :

!(y, x)

- les Sommes sont appelées par :

Som((celluleDébutY, celluleDébutX),(celluleFinY, celluleFinX))

- Les Moyennes sont appelées par :

Moy((celluleDébutY, celluleDébutX),(celluleFinY, celluleFinX))

<u>Utilisation du programme :</u>

Le programme s'ouvre en **mode Edition.** Dans ce mode, vous pouvez entrer toutes les formules et valeurs que vous voulez.
Pour obtenir le calcul et les résultats, vous devez passer en **mode Valeurs.** Ceci se fait en utilisant le menu **XLApp.**
Le menu XLApp contient également les fonctions **d'importation (Importer …) et de sauvegarde (Sauvegarder …)** de l'application.

Vous trouverez par ailleurs inclus un fichier texte de test, **documenttest.txt**.

Vous pouvez **copier les cellules et les coller.** Pour ceci, vous devez passer en **mode Edition**. Cliquez ensuite sur la cellule à copier : son contour s'affichera en **jaune.** Vous pouvez alors utiliser le **menu**

Edition, et sélectionner **Copier.** Sélectionnez ensuite la cellule de destination en cliquant dessus, et retournez dans le **menu Edition** et sélectionnez **Coller.** Les valeurs relatives seront modifiées.

Historique

Le projet a été effectué en même temps que deux autres :

• TinyMage, le lecteur d'images PPM pour NFA035,

• MonsterMayhem, un jeu vidéo type Asteroids réalisé sans framework, réalisé pour NFA019 (Java : mise en pratique) qui était à rendre le 20 juin.

Je l'ai commencé très peu de temps après que vous l'ayiez mis en ligne. A cette époque, nous n'avions pas vu les interfaces graphiques, mais je m'y étais déjà frotté avec MonsterMayhem. Nous n'avons pas eu de cours en NFA019 : lorsque j'ai demandé où est-ce que j'étais censé apprendre les interfaces, le professeur m'a plus ou moins dit de me débrouiller. J'ai donc appris avec la JavaDoc et Internet.

Ce qui a marché jusqu'à un certain point, mais votre cours était beaucoup plus clair et m'a bien aidé à comprendre. Cependant, je n'ai pas eu le temps de retaper l'interface graphique de TinyXL, ce qui explique comment elle est faite. Chaque cellule est montée sur le LayeredPane de la JFrame principale de TinyXLCore. Elles sont positionnées avec une boucle et une incrémentation en x et en y. C'est entre autres pour cela que la fenêtre n'est pas redimensionnable. Avec le recul, j'aurais probablement utilisé un **GridLayout** mais à l'époque, je ne connaissais pas les Layout, j'ai donc fait comme ça.

La première version n'avait pas de classe abstraite, et ne contenait pas les travaux optionnels. Ce sont eux qui m'ont poussé à faire **Argument** et **registerFormula** qui "parse" la formule en **Arguments**.

Vous avez dit que pour ce projet, vous vouliez voir des **Collections**, mais on ne les avait pas encore vues quand j'ai commencé ; et l'énoncé demandait un tableur statique de 10 lignes et 6 cellules, donc je n'en ai pas eu besoin au départ.
Vous trouverez cependant de nombreux ArrayList permettant de récupérer et de gérer de nombreuses informations dans le programme, et la formule, qui elle est parsée en LinkedList d'Arguments (ce qui était, j'imagine, la principale chose à faire).
Si vous souhaitez voir d'autres utilisations des collections, je vous propose de télécharger le projet pour NFA019, MonsterMayhem à l'adresse suivante :

www.mynameiswar.com/MonsterMayhem.zip

Et de jeter un œil à la classe HighScores.java, qui contient une implémentation d'un très joli TreeMap, que vous pouvez voir en lançant le projet (qui ne marche cependant pas sous Windows 10 aux dernières nouvelles, mais qui fonctionne sous Linux et macOS, et probablement en l'important dans Eclipse), et en appuyant sur la touche H. N'hésitez pas à m'envoyer vos meilleurs scores !

Merci pour cette année ! Je crois que l'an prochain, j'attaque les choses sérieuses avec NFA006. Je vais essayer de vous demander comme professeur, je trouve très équilibré l'alternance entre cours et exos comme ça. Voilà bon été à vous !