Annexe 1B– Exercices fichiers texte en stockage interne

Quiz Kotlin

Traduisez le code Java en Kotlin :

|  |  |
| --- | --- |
| Java | Kotlin |
| double valeur = 90.5; | var valeur : Double = 90.5  var valeur = 90.5 |
| class Gestion implements View.OnClickListener | inner class Gestion: View.OnClickListener |
| System.out.println ( “bonjour” ); | println(“bonjour”) |
| public void calculerSomme ( int nbEntrees ) | public fun calculerSomme( nbEntrees :Int) : Unit |

Pratiquons les acquis de l’annexe précédente…

* À l’aide de Bloc-notes, créer un fichier texte ( .txt ) contenant un certain nombre de lignes de texte.
* Créez un nouveau projet avec Android Studio **( en Kotlin ).** Exécuter votre projet une fois ( Hello World ) afin de le gérérer sur le téléphone
* Sauvegarder le fichier texte à l’aide du Device File Explorer ( nécessité de le rafraîchir ) dans data/data/votre package/ et créez-y un dossier ( directory ) appelé files. « Uploader » le fichier texte à cet endroit

1. À l’intérieur de l’activité principale de votre projet, codez les fonctions suivantes ( **en Kotlin** ) relatives au fichier texte que vous avez précédemment créé :
2. Une fonction retournant le nombre de lignes que compte votre fichier texte
3. Une fonction retournant le nombre de caractères que compte votre fichier texte
4. Une fonction retournant le nombre de « c » que comprend votre fichier texte
5. Une fonction permettant d’écrire votre nom sur une ligne située à la fin du fichier texte
6. l’interface graphique rattachée à votre activité de manière à pouvoir afficher les réponses des questions A-B-C ainsi que qu’écrire votre nom ( question D ).

Retour ensemble

Utilisation du mot-clé use : use pour ne pas être obligé de close (fermer le fichier dont on lit les infos) sinon pm peut mettre un finaly pour mettre buffer pour close car on pourrait avoir un trow d’exception pis il vas falloir faire des try catch mes c’est trop compliquer : mot du proff permet de fermer le flux sur lequel on applique la fonction use des qu’on quitte le contenue de la fonction (peu importe la raison

Fonctions d’ordre supérieur/ de haut niveau :fonction qui prend en paramètre une autre fonction /lambda

ex. use, forEachLine : fonctions qui demandent une fonction lambda en paramètre

Qu’est-ce qu’une interface fonctionnelle ? c’est une interface qui contient une seule méthode à l’intérieur de celle-ci :

Ex : OnClickListener : une seul methode a l’interieur de la methode onClick

\*

Fonction lambda : fonction anonyme qui permet d’ecrire du code de façon concie

Syntaxe : { parametre(s) -> traitement

{traitement } : s’il ny a pas de parametre ou si je parametre n’est pas utilisé dans le traitement

Trailing lambda : si une fonction a pour paramètre une lambda comme dernier paramètre, on peu alors enlever les () et coder la fonction interne anonyme a l’intérieur

On peut tout de même capter des exceptions avec une structure try/catch mais toutes les exceptions en Kotlin sont non-contrôlées.

1. Lisez sur les [Scanners](https://developer.android.com/reference/java/util/Scanner) et tentez de trouver le nombre de **mots** de votre fichier texte en utilisant un Scanner.

Un scanner vien de la de java   
il lit un doc

Il est spécialiser a trouver un mot avec un token   
pis il prend un delimiteur par deffeaut quie un withe face ( espace )donc compte le nombre de mot utiliser un scanner

Pis il y a use delimiteur (celui -si on peux choisir quell symbole est delimiteur

## Quoi faire pour distribuer une app utilisant un fichier texte ?

On a simulé la lecture d’un fichier texte présent en mémoire interne du téléphone. **On ne peut faire cela qu’en tant que programmeur** ( en utilisant le Device File Explorer ) .

On pourrait aussi installer le fichier texte dans l’app elle-même afin de pouvoir l’installer sans avoir à placer le fichier sur le téléphone en tant que programmeur. Pour ce faire, on créer un dossier raw dans le dossier res. On y ajoute le fichier texte à utiliser à la racine de ce dossier.

Pour retrouver ce fichier :

En Java :

InputStream fis = getResources().openRawResource(R.raw.palmares);

En Kotlin :getRessources( .openRare

Et utiliser les flux de traduction et de traitement qu’on a vus, on peut lire le contenu du fichier

**\*\*\* IMPORTANT**

**Les ressources contenues dans ce répertoire raw vont être enveloppées dans notre fichier .apk destiné à être installé sur les téléphones alors on ne peut pas écrire dedans au moment de l’exécution, seulement lire**

**Pour écrire, on doit alors utiliser la mémoire interne ( comme on a fait ) ou la mémoire partagée ( comme on verra plus tard )**