Annexe 1B– Exercices fichiers texte en stockage interne

Quiz Kotlin

Traduisez le code Java en Kotlin :

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

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 :

On a vu qu’on devait libérer les ressources des flux de données (close)

Nous avons vu qu’il a des problèmes potentiels s’il y a une erreur avant le close, le flux de données ne sera pas fermé

Solutions:

1.try/catch/finally (alourdit le code)

2.Try-with-resources (solution JAVA)

try(declarer / créer tous les flux que on doit fermer)

{ Code readLine,write }

Catch ()

{}

3.fonction use (Solution kotlin)

Br.use{tous ce que je fait dans ma méthode} \*\*(plus besoin de close)

Fonctions de haut – niveau : ex : use, forEachLine : fonction qui demandent une fonction lambda en paramètre.

Exemple : br.forEachLine {ligne -> compteur += ligne.length }

Fonction lambda :

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.

## 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 :

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 )**