Partie 4 : Accès aux données : traitement d'un fichier Json

Lors de ce TP nous allons utiliser beaucoup de procédés déjà utilisés.
L'objectif de ce travail est de récupérer les données du catalogue (produits, catégories et promotions) à partir d'un fichier au format Json stocké sur un serveur distant, d'enregistrer ces données dans des fichiers locaux et d'utiliser ces fichiers dans le reste de l'application.
Nous allons effectuer ce travail en deux étapes :

- Sérialisation des classes
- Traitement d'un fichier Json local

I. Sérialisation des classes du modèle

Nous devons pouvoir sérialiser toutes les classes du modèle.

a. Traitement des classes Catégorie, produit et Promotion.

- La classe Catégorie doit implémenter l'interface Serializable.

```
import java.io.Serializable;

public class Categorie implements Serializable {
    private String id;
```

- Faire de même pour les classes Produit et Promotion.

b. Préparation du Catalogue

i. La Classe Catalogue

La classe catalogue va nous permettre de répertorier toutes les informations nécessaires à l'application des tarifs des produits.

- Il nous faut donc ajouter les listes de promotions et les catégories de l'application.
- La classe doit implémenter l'interface Serializable.

- Ajouter les deux attributs supplémentaires. (Penser à les instancier dans le constructeur)
- Ajouter l'interface Serializable.

ii. La Classe CatalogueRepository

- Renommer la méthode RecupererLeCatalogue() en RecupererLeCatalogueTest()
- Ajouter les deux méthodes afin de pouvoir sérialiser la classe

```
public static Catalogue RecupererLeCatalogue(Context context)  {
    return (Catalogue) Serializer.deSerialize(nomFichier,context);

}

public static void EnregistrerLeCatalogue(Catalogue leCatalogue, Context  context){
    Serializer.serialize(nomFichier, leCatalogue,  context);
}
```

II. Traitement d'un fichier Json local

a. Présentation du JSON

Le **JSON** (*JavaScript Object Notation*) est le second type de fichier très populaire lors de l'utilisation de webservice. Afin de parser ce fichier, nous allons utiliser la classe **JSONObject**.
JSON est un format léger d'échange de données. Il est facile à écrire et à lire, facilement analysable et totalement indépendant de tout langage, ce qui en fait un langage d'échange de données idéal.
Il base sa structure sur deux éléments :

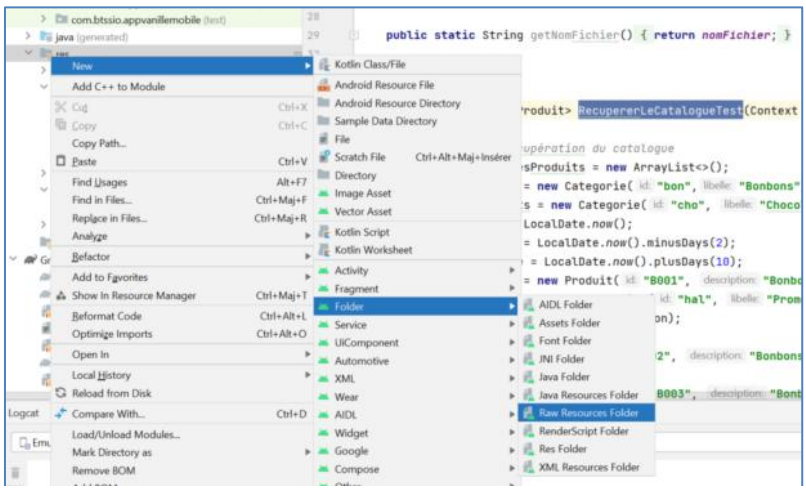
- Une collection de couples clé/valeur.
- Une liste de valeurs ordonnées.

➢ Récupérer le fichier donnees_catalogue.json et ouvrir celui-ci, en déduire la structure d'un produit. (Regarder les liens entre catégorie, promotion et produit)

b. Copie du fichier Json dans le projet

Nous allons pour l'instant travailler avec un fichier local, pour cela nous devons l'ajouter au projet dans les ressources.

➢ Clic droit sur res – New – folder – Raw Ressources Folder



➢ Clic sur finish.
➢ Copier le fichier .Json dans le dossier.

Celui-ci est désormais accessible via son identifiant R.raw.donnees_catalogue.

c. Lecture du fichier

➢ Créer une nouvelle classe LectureFichierJson abstraite dans le package outils.
➢ Recopier les deux méthodes suivantes et commenter !

```
public abstract class LectureFichierJson {



    public static String getDonneesJson(Context context) throws IOException {
        return readText(context, R.raw.donnees_catalogue);
    }

    private static String readText(Context context, int resId) throws IOException {
        InputStream is = context.getResources().openRawResource(resId);
        BufferedReader br= new BufferedReader(new InputStreamReader(is));
        StringBuilder sb= new StringBuilder();
        String s= null;
        while((  s = br.readLine())!=null) {
            sb.append(s);
            sb.append("\n");
        }
        return sb.toString();
    }
}
```

Nous pouvons désormais avoir à notre disposition le contenu du fichier sous forme d'une chaine de caractères.

III. Traitement du « texte » Json

- Créer une nouvelle classe TraitementJson abstraite dans le package outils.
- Ajouter la méthode suivante :

```
public static Catalogue LectureDonneesDepuisJson(String jstring, Context context) {
return null;
}
```

Afin de tester cette méthode (il faudrait faire des tests unitaires) mais nous allons l'appeler dans l'activité MainActivity.

- Ajouter les lignes suivantes à votre méthode onCreate() de l'activité main.

```
try {
    String texte =  LectureFichierJson.getDonneesJson( context: this);
    catalogueEnCours = TraitementJson.LectureDonneesDepuisJson(texte,  context: this);
} catch (JSONException | IOException e) {
    e.printStackTrace();
}
```

Nous allons récupérer les catégories, puis les promotions et enfin les produits.

a. Traitement des catégories

Parser un fichier JSON s'effectue en suivant les étapes décrites ci-dessous :

- Obtenir le contenu du fichier **JSON** cible.
- Créer un objet de type **JSONObject** à partir du contenu du fichier JSON.
- Récupérer les différentes balises à l'aide des méthodes (**getJSONObject** : récupération du contenu d'une balise et **getJSONArray** : récupération de la liste d'éléments contenus dans une balise).
- Récupérer les différentes valeurs contenues dans la liste (balise **chapitre**) : identifiant, nom et description.

➢ Recopier la méthode LectureDonneesDepuisJson() et commmenter.

```
public static Catalogue LectureDonneesDepuisJson(String jstring, Context context) throws JSONException {

    //déclaration de la variable retour
    Catalogue leCatalogue;
    //variables de lecture
    Categorie categorieLue;
    //
    leCatalogue = new Catalogue( id: "1", titre: "catalogue en cours", LocalDate.now());
    //
    JSONObject jsonObjet = new JSONObject(jstring);
    JSONArray jCategories = jsonObjet.getJSONArray( name: "categories");
     //traitement des catégories
    for (int i =0; i< jCategories.length();i++){
        categorieLue = new Categorie(jCategories.getJSONObject(i).getString( name: "id"),
                jCategories.getJSONObject(i).getString( name: "description"));
        leCatalogue.getLesCategories().add(categorieLue);
        Log.i( tag: "ajout",categorieLue.toString());
    }

    CatalogueRepository.EnregistrerLeCatalogue(leCatalogue,context);

    return leCatalogue;
}
```

- Lancer l'activité MainActivity afin de tester. Les catégories s'affichent elles dans votre fenêtre logCat ?

b. Traitement des promotions

- Traiter les promotions. A vous de jouer !

Pour vous aider :

- Déclarer une variable de lecture.
- Créer un JSONArray jPromotions.
- Parcourir le tableau en créant les promotions et en les ajoutant à la liste des promotions du catalogue.
- Pour transformer une date Json en LocalDate il faut parser celle-ci.
- Exemples :

```
debutPromo = LocalDate.parse(jPromotions.getJSONObject(i).getString( name: "debutPromo"));
finPromo = LocalDate.parse(jPromotions.getJSONObject(i).getString( name: "finPromo"));
```

- Tester !

c. Traitement des produits

Deux difficultés s'offrent à nous. Les produits appartiennent à une catégorie et peuvent avoir une promotion (système dérivé afin de simplifier).

Traiter les produits en mettant ses difficultés de côté. Pour ce faire créer les produits avec une catégorie à « null » et ne pas ajouter de promotion.

- A vous de jouer ! (Comme pour les promotions et catégories)
- Tester

Tout fonctionne ? Parfait.

d. Traitement de la catégorie du produit

- Regarder la structure d'un produit , que signifie d'après vous le 2 dans l'exemple ci-dessous ?

```
{
    "PDT_id": "CH01",
    "description": "Chocolats Pralinés lot 1Kg",
    "prix": 17.0,
    "image": "images/chocolats/choco1.png",
    "categorie": 2,
    "promo": 1
},
```

Le fichier json a été créé de telle sorte que pour chaque produit, l'on récupère l'indice de la catégorie dans la liste lesCategories précédemment créée.

- Ajouter la catégorie du produit lors de la création du produit.

Pour vous aider :

```
categorieLue = leCatalogue.getLesCategories().get(jProduits.getJSONObject(i).getInt( name: "categorie"));
```

e. Traitement de la promotion du produit

Le même principe est utilisé pour la promotion. Mais tous les produits ne sont pas en promotion. Il faut tester l'existence de la promotion avant de pouvoir la traiter.

```
if (!jProduits.getJSONObject(i).isNull( name: "promo")) {
    int position = jProduits.getJSONObject(i).getInt( name: "promo");
      promoLue = leCatalogue.getLesPromos().get( position);
       produitLu.getLesPromos().add(promoLue);
    }
```

- Ajouter le traitement des promotions et tester !

IV. Gestion de l'application

a. Récupération du catalogue

- Modifier l'activité ListProduitsActivity afin de récupérer le catalogue.
- Tester.

b. Logique de l'application

Dans l'activité MainActivity ajouter un bouton qui lance l'activité ListProduitsActivity.

- A vous de jouer !