

Kotlin

Les fondamentaux
du développement
d'applications **Android**



informatique technique

Fichiers complémentaires
à télécharger




Collection

epsilon

Anthony COSSON

Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence de l'ouvrage **EPKOT** dans la zone de recherche
et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement. 

Avant-propos

Chapitre 1 Présentation de Kotlin

- 1. Historique 21
- 2. Philosophie 22
- 3. Avantages 23
- 4. Environnement de développement 23
 - 4.1 Présentation 23
 - 4.2 Premiers pas avec IntelliJ IDEA 24
- 5. Premier exemple Kotlin 25

Chapitre 2 Kotlin : les fondamentaux

- 1. La fonction main. 29
- 2. Les variables 30
 - 2.1 Présentation 30
 - 2.2 Les variables immutables 31
 - 2.3 Les variables mutables 32
 - 2.4 Les constantes 32
 - 2.5 Les types de base 33
 - 2.5.1 Les types entiers 33
 - 2.5.2 Les types décimaux 34
 - 2.5.3 Le type booléen 35
 - 2.5.4 Le type caractère 35

2.6	Les valeurs numériques littérales	36
2.6.1	Long et Float	36
2.6.2	Écritures binaire, octale et hexadécimale	37
2.6.3	Utilisation des underscores	38
2.7	Les chaînes de caractères	38
2.8	Les types de base Nullable	41
2.9	Les types Any et Any?	43
2.10	Le type Unit	43
2.11	Le type Nothing	44
2.12	Les tableaux	45
2.12.1	Création d'un tableau	45
2.12.2	Utilisation courante	46
2.13	Conversions des types de base	50
2.14	Portée des variables	51
3.	Les opérateurs	51
3.1	Les opérateurs arithmétiques	51
3.2	Les opérateurs de bits	52
3.3	Les opérateurs unaires	52
3.4	Les opérateurs d'affectation	53
3.5	Les opérateurs de comparaison	53
3.6	Les opérateurs logiques	54
4.	Les structures de contrôle	54
4.1	Les conditionnelles	54
4.1.1	L'expression if	55
4.1.2	L'expression when	56
4.2	Les répétitives	57
4.2.1	La boucle for	57
4.2.2	La boucle while	60
4.2.3	La boucle do while	61
4.2.4	L'interruption d'une boucle	62

- 5. Les fonctions 62
 - 5.1 Syntaxe 62
 - 5.2 Usage 63
 - 5.3 Les paramètres 64
 - 5.3.1 Syntaxe 64
 - 5.3.2 Immutable 65
 - 5.3.3 Paramètre par défaut 66
 - 5.3.4 Argument nommé 66
 - 5.3.5 Paramètre variadique 68
 - 5.4 Les fonctions contenant une expression simple 69
 - 5.5 Le type de retour 70
 - 5.5.1 Explicite 70
 - 5.5.2 Implicite 70
 - 5.6 Fonction récursive 71
 - 5.7 Portée des fonctions 71
 - 5.7.1 Fonction hors d'une classe 72
 - 5.7.2 Fonction dans une classe 72
 - 5.7.3 Fonction dans une fonction 72
- 6. Les packages 73
 - 6.1 Présentation 73
 - 6.1.1 Conventions et bonnes pratiques 74
 - 6.1.2 Déclaration d'un package 74
 - 6.1.3 Importation d'un package 74
- 7. Démonstration 76

Chapitre 3**Kotlin : la programmation orientée objet**

1. Introduction	79
2. Les classes	79
2.1 Présentation	79
2.2 Syntaxe	80
2.3 Déclaration d'une classe	80
2.4 Création et utilisation d'instances d'une classe	81
2.5 Les modificateurs de classe	81
2.5.1 Modificateur public	81
2.5.2 Modificateur internal	81
2.5.3 Modificateur protected	82
2.5.4 Modificateur private	82
2.5.5 Modificateur final	82
2.5.6 Modificateur open	82
2.5.7 Modificateur abstract	82
2.5.8 Pseudo modificateur nested (imbriqué)	82
2.5.9 Modificateur inner (interne)	83
2.5.10 Modificateur sealed (fermé)	83
2.6 Les constructeurs	83
2.6.1 Constructeur principal	83
2.6.2 Bloc d'initialisation	85
2.6.3 Constructeurs secondaires	87
2.7 Les champs et les propriétés	89
2.7.1 Présentation	89
2.7.2 Syntaxe	89
2.7.3 Création de propriétés via le constructeur	89
2.7.4 Création de propriétés dans la classe	90
2.7.5 Création de propriétés personnalisées	91
2.8 Les modificateurs de propriétés	94
2.8.1 Modificateur public	94
2.8.2 Modificateur private	95
2.8.3 Modificateur protected	95

2.8.4	Modificateur internal	95
2.8.5	Modificateur open	95
2.8.6	Modificateur override	95
2.8.7	Modificateur final	95
2.9	Les fonctions de classes	96
2.9.1	Syntaxe	96
2.9.2	Usage	96
2.9.3	Les modificateurs de fonctions	97
2.9.4	Les fonctions d'extensions	97
2.9.5	Notation infixée	100
3.	Les classes objet	102
3.1	Présentation	102
3.2	Syntaxe	102
3.3	Usage	102
4.	L'héritage	106
4.1	Présentation	106
4.2	Les bases de l'héritage	107
4.2.1	Présentation	107
4.2.2	Syntaxe	107
4.2.3	Héritage simple	107
4.2.4	Héritage et constructeur principal	108
4.2.5	Héritage et constructeur secondaire	108
4.2.6	Super	109
4.3	Redéfinition de propriétés et de fonctions	111
4.3.1	Différence entre surcharge et redéfinition	111
4.3.2	Redéfinition de propriétés	112
4.3.3	Redéfinition de fonctions	112
5.	Les classes abstraites	113
5.1	Présentation	113
5.2	Syntaxe	113
5.3	Usage	113

6.	Les classes scellées	114
6.1	Présentation	114
6.2	Syntaxe	114
6.3	Usage	114
7.	Les interfaces	114
7.1	Présentation	114
7.2	Syntaxe	115
7.3	Usage	115
8.	Les classes internes	116
8.1	Présentation	116
8.2	Syntaxe	116
8.3	Usage	116
9.	Les classes imbriquées	117
9.1	Présentation	117
9.2	Syntaxe	117
9.3	Usage	117
10.	Les classes anonymes	118
10.1	Présentation	118
10.2	Syntaxe avec les object expressions	118
10.3	Usage avec les object expressions	118
11.	Singleton	119
11.1	Présentation	119
11.2	Syntaxe	119
11.3	Usage	119
12.	Fonctions statiques	120
12.1	Présentation	120
12.2	Syntaxe	120
12.3	Usage	120
13.	Enum classe	121
13.1	Présentation	121
13.2	Déclarations	121

- 14. Généricité 122
 - 14.1 Présentation 122
 - 14.2 Syntaxe 122
 - 14.3 Usage 123
- 15. La gestion des erreurs d'exécution 124
 - 15.1 Présentation 124
 - 15.2 Syntaxe 124
 - 15.3 Usage 125
- 16. Les collections 126
 - 16.1 Présentation 126
 - 16.2 Syntaxe 127
 - 16.2.1 Collection de type List 127
 - 16.2.2 Collection de type MutableList 127
 - 16.2.3 Collection de type Set 127
 - 16.2.4 Collection de type MutableSet 127
 - 16.2.5 Collection de type Map 128
 - 16.2.6 Collection de type MutableMap 128
 - 16.3 Usage 128
 - 16.3.1 Collection de type List 128
 - 16.3.2 Collection de type MutableList 128
 - 16.3.3 Collection de type Set 129
 - 16.3.4 Collection de type MutableSet 129
 - 16.3.5 Collection de type Map 129
 - 16.3.6 Collection de type MutableMap 130
 - 16.4 Fonctions d'agrégation 130
 - 16.4.1 Présentation 130
 - 16.4.2 Usage 131
 - 16.5 Fonctions de sélection 135
 - 16.5.1 Présentation 135
 - 16.5.2 Usage 136

17. Les expressions lambda	141
17.1 Présentation	141
17.2 Syntaxe	141
17.3 Usage	142
17.4 Stocker le traitement dans une variable	143

Chapitre 4

Android : présentation de l'environnement

1. Présentation	145
1.1 Introduction	145
1.2 La plateforme Android	146
1.3 Processus de construction d'une application	148
2. Environnement de développement	148
2.1 Introduction	148
2.2 SDK Android	149
2.3 Android Studio	149
2.3.1 Présentation	149
2.3.2 Principaux outils	150
2.4 Gradle	155
2.4.1 Présentation	155
2.4.2 Gestion des dépendances	155
3. Application Android	156
3.1 Création d'un nouveau projet	156
3.2 Structure d'une application	160
3.2.1 manifests	160
3.2.2 java	161
3.2.3 res	162
3.2.4 Gradle Scripts	162
3.3 Exécution d'une application	162
3.3.1 Sur émulateur	163
3.3.2 Sur téléphone	168

Chapitre 5

Android : les fondamentaux

1. Les composantes principales	171
1.1 Activité	171
1.2 Intention	171
1.3 Fragment	172
1.4 Service	172
1.5 Content provider	172
1.6 BroadcastReceiver	172
1.7 Intent filter	172
1.8 Manifeste	172
1.9 Context	173
2. Activité	173
2.1 Présentation	173
2.2 Pile de gestion	173
2.3 Création d'une activité	174
2.4 Contrôleur	174
2.5 Cycle de vie	175
2.6 Usage	177
3. Vue liée à l'activité	179
3.1 Présentation	179
3.2 Fonctionnement de l'interface utilisateur	179
3.3 Les conteneurs de type ViewGroup	180
3.3.1 Présentation	180
3.3.2 GridLayout	181
3.3.3 LinearLayout	183
3.3.4 RelativeLayout	184
3.3.5 ConstraintLayout	189
3.4 Les composants graphiques de type View	198
3.5 Démonstration	201

4.	Gestion des événements	204
4.1	Gestion des événements via XML	204
4.2	Gestion des événements via Kotlin	206
4.2.1	Présentation	206
4.2.2	Usage historique	207
4.2.3	Usage optimisé.	208
5.	Gestion des ressources	210
5.1	Présentation	210
5.1.1	Stocker les ressources	210
5.1.2	Accès aux ressources	211
6.	Manifeste	213
6.1	Présentation	213
6.2	Les balises	213
6.3	Exemple.	215
7.	Gestion des droits.	216
7.1	Présentation	216
7.2	Fonctionnement	216
7.3	Syntaxe	217
7.4	Usage.	217
7.5	Complément.	219
8.	Les intentions	220
8.1	Présentation	220
8.2	Intent explicite	220
8.2.1	Présentation.	220
8.2.2	Syntaxe.	220
8.2.3	Exemple	221
8.2.4	Transfert de données scalaires	221
8.2.5	Transfert de données parcelables	221
8.2.6	Démonstration	223
8.2.7	Usage simplifié.	226

- 8.3 Intent implicite 226
 - 8.3.1 Présentation 226
 - 8.3.2 Syntaxe 227
 - 8.3.3 Usage 228
 - 8.3.4 Démonstration 228
 - 8.3.5 Usage simplifié 232
- 8.4 Intent-filter 232
 - 8.4.1 Présentation 232
 - 8.4.2 Syntaxe 232
 - 8.4.3 Usage 234

Chapitre 6

Android : les éléments incontournables

- 1. Les listes 235
 - 1.1 Présentation 235
 - 1.2 ListView 236
 - 1.2.1 Présentation 236
 - 1.2.2 Définir une ListView 236
 - 1.2.3 Définir le style des éléments de la liste 237
 - 1.2.4 Créer une classe adapter 238
 - 1.2.5 Lier l'adapter à la ListView 239
 - 1.3 RecyclerView 240
 - 1.3.1 Présentation 240
 - 1.3.2 Dépendance 241
 - 1.3.3 Définir un RecyclerView 241
 - 1.3.4 Définir le style des éléments de la liste 242
 - 1.3.5 Créer une classe Adapter 243
 - 1.3.6 Lier l'adapter au RecyclerView 245
 - 1.4 CardView 246
 - 1.4.1 Présentation 246
 - 1.4.2 Dépendance 246
 - 1.4.3 Usage 246

2.	Les menus	248
2.1	Présentation	248
2.2	ActionBar	248
2.2.1	Présentation	248
2.2.2	Créer la structure du menu	249
2.2.3	Lier le menu à l'activité	250
2.2.4	Définir les actions de chaque bouton	250
2.3	NavigationDrawer	252
2.3.1	Présentation	252
2.3.2	Mise en place	252
2.3.3	Dépendance	252
2.3.4	Utilisation de l'assistant	253
2.3.5	Personnalisation	254
3.	Les pop-up	262
3.1	Toast	262
3.1.1	Présentation	262
3.1.2	Usage historique	262
3.1.3	Usage optimisé	262
3.1.4	Démonstration	263
3.2	AlertDialog	264
3.2.1	Présentation	264
3.2.2	Dépendance	264
3.2.3	Usage	265
3.2.4	Démonstration	265
4.	Les fragments	266
4.1	Présentation	266
4.2	Cycle de vie	266
4.3	Structure	269
4.3.1	Fragment indépendant	269
4.3.2	Fragment communiquant avec l'activité	269
4.3.3	Activité communiquant avec le fragment	269
4.4	La librairie de support V4	269

- 4.5 Création.....270
 - 4.5.1 Fragment indépendant270
 - 4.5.2 Fragment communiquant avec l'activité274
 - 4.5.3 Activité communiquant avec le fragment277
- 5. Démonstration281

Chapitre 7
Android : la persistance des données

- 1. Présentation293
- 2. SharedPreferences293
 - 2.1 Présentation293
 - 2.2 Fonctionnement294
 - 2.3 Récupérer un objet de type SharedPreferences.....294
 - 2.3.1 Interactivité294
 - 2.3.2 Intra activité295
 - 2.4 Enregistrer dans un fichier de préférences.....295
 - 2.5 Lire dans un fichier de préférences295
 - 2.6 Démonstration296
- 3. Fichier interne.....298
 - 3.1 Présentation298
 - 3.2 Écrire dans un fichier interne.....298
 - 3.3 Lire dans un fichier interne299
 - 3.4 Mettre en cache un fichier interne299
 - 3.5 Quelques fonctions.....299
 - 3.6 Démonstration300
- 4. Fichier externe.....302
 - 4.1 Présentation302
 - 4.2 Permissions303
 - 4.3 Vérifier la disponibilité d'un espace de stockage.....303
 - 4.4 Écrire dans un fichier externe privé.....304
 - 4.5 Lire un fichier externe privé.....304

4.6	Écrire un fichier externe public	305
4.7	Lire un fichier externe public	305
4.8	Démonstration	306
5.	Base de données avec SQLiteOpenHelper	309
5.1	Présentation	309
5.2	Procédure	310
5.3	Mise en place	310
5.3.1	Schéma de la base de données	310
5.3.2	Modèle	310
5.3.3	Contract	311
5.3.4	BddHelper	311
5.3.5	DAO	312

Chapitre 8

Android : la programmation concurrente

1.	Présentation	317
2.	Problématique	318
2.1	Présentation	318
2.2	Démonstration	318
3.	Les threads	321
3.1	Présentation	321
3.2	Syntaxe	321
3.3	Usage	322
3.4	Démonstration	322
3.5	Problématique	325
3.6	runOnUiThread	325
3.6.1	Présentation	325
3.6.2	Démonstration	325
4.	Les tâches asynchrones	328
4.1	Présentation	328
4.2	Démonstration	329

- 5. Les Handlers 334
 - 5.1 Présentation 334
 - 5.2 Démonstration 334
- 6. Les coroutines 341

Chapitre 9

Android : quelques librairies incontournables

- 1. Retrofit 343
 - 1.1 Présentation 343
 - 1.2 Procédure 343
 - 1.3 Démonstration 344
 - 1.4 Permission 344
 - 1.5 Dépendances 345
 - 1.6 Création d'un objet modèle 345
 - 1.7 Création d'une interface de communication 346
 - 1.8 Utilisation 347
- 2. Anko 349
 - 2.1 Présentation 349
 - 2.2 Installation 350
 - 2.3 Anko Commons 351
 - 2.3.1 Présentation 351
 - 2.3.2 Intent 351
 - 2.3.3 Les boîtes de dialogues 354
 - 2.3.4 Les logs 357
 - 2.3.5 Les toasts 359
 - 2.4 Anko SQLite 359
 - 2.4.1 Présentation 359
 - 2.4.2 Accès et définition de la base de données 360
 - 2.4.3 Insertion de données 362
 - 2.4.4 Mise à jour de données 363
 - 2.4.5 Suppression de données 363
 - 2.4.6 Extraction de données 364

2.5	Anko Layouts	365
2.5.1	Présentation	365
2.5.2	Définition rapide des propriétés essentielles	367
2.5.3	Définition rapide du thème des éléments graphiques	368
2.5.4	Modification des propriétés des éléments graphiques	368
2.5.5	Modification des propriétés layout des éléments graphiques	368
2.5.6	UI Wrapper	369
2.6	Conclusion	369
3.	Room	370
3.1	Présentation	370
3.2	Procédure	370
3.3	Création de la base de données	371
3.4	Dépendances	371
3.5	Création d'une classe de type Entity	371
3.6	Création d'une interface de type DAO	373
3.7	Création d'une classe de type Database	374
3.8	Récupération d'une instance de la base de données	374
3.9	Utilisation	375
3.9.1	Insertion	375
3.9.2	Lecture	375
3.9.3	Modification	376
3.9.4	Suppression	376
3.10	Conclusion	376

Chapitre 10

Android : les éléments avancés

1.	Création et utilisation d'un content provider	377
1.1	Présentation	377
1.2	Structure	378
1.3	Identification	378
1.4	Créer un content provider	379

1.5	Création : définir les fonctionnalités d'un content provider . . .	379
1.5.1	Lire des données via un content provider	379
1.5.2	Insérer des données via un content provider	381
1.5.3	Mettre à jour des données via un content provider . . .	382
1.5.4	Supprimer des données via un content provider	384
1.6	Création : définir les permissions d'un content provider	385
1.6.1	Présentation	385
1.6.2	Usage	386
1.7	Utiliser un content provider	386
1.7.1	Connaître l'URI du content provider	386
1.7.2	Récupérer un objet de type ContentResolver fourni par le Context.	387
1.7.3	Définir les permissions requises par le content provider	388
1.7.4	Usage	388
1.8	Utilisation : accès aux contacts de votre téléphone	390
1.8.1	Présentation	390
1.8.2	Fonctionnement	390
1.8.3	La table ContactsContract.RawContacts	390
1.8.4	La table ContactsContract.Data	391
1.8.5	La classe ContactsContract.CommonDataKinds	392
1.8.6	La table ContactsContract.Contacts	393
1.8.7	Permission	395
1.8.8	Usage	396
1.9	Démonstration : création d'un content provider	398
1.9.1	Présentation	398
1.9.2	Sources	399
2.	Utilisation de données JSON	410
2.1	Présentation	410
2.2	Formats	410
2.3	Transformer des données JSON en objets Kotlin.	411
2.3.1	Manière historique	411

3. Utilisation de services distants	414
3.1 Présentation	414
3.2 Permission	414
3.3 Consommer un web service	414
3.4 Démonstration	415
4. Les WebViews	419
4.1 Présentation	419
4.2 Déclaration dans l'interface	419
4.3 Charger une page web dans une WebView	420
4.3.1 Accès à des pages web distantes	420
4.3.2 Accès à des pages web locales	423
4.4 Appel de fonctions Kotlin avec du JavaScript	427
4.4.1 Présentation	427
4.4.2 Usage	427
5. Les récepteurs d'événement	432
5.1 Présentation	432
5.2 Création d'un Broadcast Receiver	432
5.3 Abonnement à des événements	433
5.4 Diffusion d'un événement	434
5.5 Démonstration	434

Chapitre 11

Android : tester l'application

1. Logcat	437
1.1 Présentation	437
1.2 Vue d'ensemble	438
2. Tests unitaires avec JUnit	439
2.1 Présentation	439
2.2 Procédure	440
2.3 Démonstration	440

- 3. Test de l'interface avec Espresso 444
 - 3.1 Présentation 444
 - 3.2 Procédure de mise en place..... 445
 - 3.3 Dépendances..... 445
 - 3.4 Définir un test..... 446
 - 3.4.1 Définir une classe de test 446
 - 3.4.2 Définir l'activité à tester dans la classe de test..... 446
 - 3.4.3 Définir un cas de test..... 447
 - 3.5 Démonstration 447

- Index 451



Chapitre 6

Android : les éléments incontournables

1. Les listes

1.1 Présentation

Afficher une liste pour présenter des informations est incontournable dans le développement d'une application. Il y a deux formes de listes fréquemment utilisées :

- ListView : toutes les données sont chargées dès la création de la liste.
- RecyclerView : seulement les données visualisées sont chargées.

Il est aujourd'hui souvent recommandé d'utiliser les RecyclerView pour afficher des listes de données. Les ListView sont présentées dans un but pédagogique étant donné que c'est la façon de faire historique et idéale pour débiter.

Les CardView seront aussi abordées dans ce chapitre, car elles sont indissociables des listes. Elles permettent de perfectionner le style des listes.

1.2 ListView

1.2.1 Présentation

Pour créer une liste à l'aide des `ListView` il est nécessaire de passer par quatre étapes :

- Étape 1 : définir un élément de type `ListView` dans l'IHM d'une activité ou d'un fragment.
- Étape 2 : définir le style des lignes d'informations affichées dans la liste. Ceci se fait dans un fichier de type `layout`.
- Étape 3 : définir une classe adapter qui permettra de lier les données au style. L'adapter va créer autant d'éléments `View` que de données.
- Étape 4 : lier l'adapter à la `ListView`.

1.2.2 Définir une ListView

La balise `ListView` permet de définir la liste. Elle est associée à l'identifiant `la_liste_view`, cet identifiant est utilisé dans le code de l'activité dans le but de récupérer un objet représentant la `ListView`. L'objet sera ensuite utilisé pour y lier l'adapter.

■ Remarque

Les exemples sont réalisés dans un nouveau projet dont l'activité principale se nomme `MainActivity`.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:
android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="fr.acos.demolistviewwithkotlin.MainActivity">

    <ListView
        android:id="@+id/la_liste_view"
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent">
</ListView>
```

```
</android.support.constraint.ConstraintLayout>
```

Cet exemple présente comment définir une *ListView* dans un *layout*. En gras, les éléments intéressants et relatifs à la *ListView*.

1.2.3 Définir le style des éléments de la liste

Le fichier `style_dune_ligne.xml` contenant le style des éléments de la liste se situe dans le dossier `res/layout`. Chaque élément de la liste affiche un nom et un prénom grâce aux vues de type `TextView`.

style_dune_ligne.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <TextView
        android:id="@+id/tv_nom"
        android:text="XXXXX"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_margin="10dp"
    />

    <TextView
        android:id="@+id/tv_prenom"
        android:text="YYYYY"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_margin="10dp"
    />

</android.support.constraint.ConstraintLayout>
```

Cet exemple présente comment définir le style de chaque item/élément de la ListView. En gras, les éléments intéressants et relatifs à la présentation.

1.2.4 Créer une classe adapter

La classe `Personne` permet de créer des objets qui seront affichés dans la liste. On définit cette classe dans le fichier `MainActivity.kt`. La classe `PersonneAdapter` permet de lier les données (Liste d'objets de type `Personne`) au style (défini dans `style_dune_ligne.xml`). La classe `PersonneAdapter` est définie dans le fichier `MainActivity.kt`.

Classe `Personne` et Classe `PersonneAdapter` (fichier `MainActivity.kt`)

```
/**
 * Classe BO permettant de stocker les données à afficher
 */
data class Personne(val id: Long, val nom: String, val prenom: String)

/**
 * Classe adapter permettant de charger les données dans la liste
 */
class PersonneAdapter(context: Context?, resource: Int, objects:
MutableList<Personne>?) : ArrayAdapter<Personne>(context,
resource, objects)
{
    //Variable permettant de stocker l'identifiant du layout de
    //style d'une ligne de liste
    private val ma_ligne: Int;

    //Au moment de l'initialisation de l'objet adapter, on récupère
    //l'identifiant du layout passé en paramètre
    //pour le fournir à la variable ma_ligne.
    init
    {
        ma_ligne = resource;
    }

    /**
     * Fonction appelée autant de fois qu'il y a d'objets à afficher
     * dans la liste.
     *
     * @param position index de l'élément de la liste à afficher
     * @param convertView Vue représentant le layout
     * style_dune_ligne.xml
     * @param parent Parent de la liste
     */
}
```

```
    */
    override fun getView(position: Int, convertView: View?, parent:
ViewGroup?): View
    {
        //Au premier appel on charge le fichier
style_dune_ligne.xml, puis on réutilise convertView
        val uneLigneView = convertView ?:
        LayoutInflater.from(parent!!.getContext()).inflate(ma_ligne,
parent, false)

        //Ces variables représentent les TextView de la ligne à créer.
        val tvNom =
uneLigneView.findViewById<TextView>(R.id.tv_nom)
        val tvPrenom =
uneLigneView.findViewById<TextView>(R.id.tv_prenom)

        //Récupère les données à afficher.
        val personne = getItem(position)

        //Met les données dans les TextView
        tvNom.text = personne.nom
        tvPrenom.text = personne.prenom

        //La ligne est créée, on la retourne.
        return uneLigneView
    }
}
```

Cet exemple présente comment définir un adapter qui permettra de remplir la liste. En gras, les éléments intéressants et relatifs à l'adapter.

1.2.5 Lier l'adapter à la ListView

La classe `MainActivity` se situe dans le fichier `MainActivity.kt`. La fonction `onCreate` de la classe `MainActivity` est appelée lors de la création de l'activité. Un bouchon (données de test) constitué d'une liste d'objets `Personne` est créé. Un objet `PersonneAdapter` est créé en lui passant en paramètre la liste des objets `Personne` et le style des éléments de la liste. L'adapter est ensuite lié à la `ListView`.

MainActivity.kt

```
/**
 * Activité qui affiche la ListView.
 */
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //Bouchon
        val p1 = Personne(1,"Adnet","Romain")
        val p2 = Personne(2,"Roussel","Guenole")
        val p3 = Personne(3,"Baudouin","Vincent")
        val personnes = mutableListOf(p1,p2,p3)

        //Création de l'adapter
        val adapter =
PersonneAdapter(this,R.layout.style_dune_ligne,personnes)

        //On lie l'adapter à la ListView
        la_liste_view.adapter = adapter
    }
}
```

Cet exemple présente comment combiner tous les éléments dans le but de remplir la liste. En gras, les éléments intéressants.

1.3 RecyclerView

1.3.1 Présentation

Pour créer une liste à l'aide des RecyclerView il est nécessaire de passer par cinq étapes :

- Étape 1 : ajouter la dépendance pour utiliser les RecyclerView dans Gradle.
- Étape 2 : définir un élément de type RecyclerView dans l'IHM d'une activité ou d'un fragment.