# Adaptateur de Liste d'Éléments


Macha DA COSTA

@MachaDaCosta


# Adaptateur de Liste d'Éléments

A. Adaptateur 

B. Adaptateur et Liste 

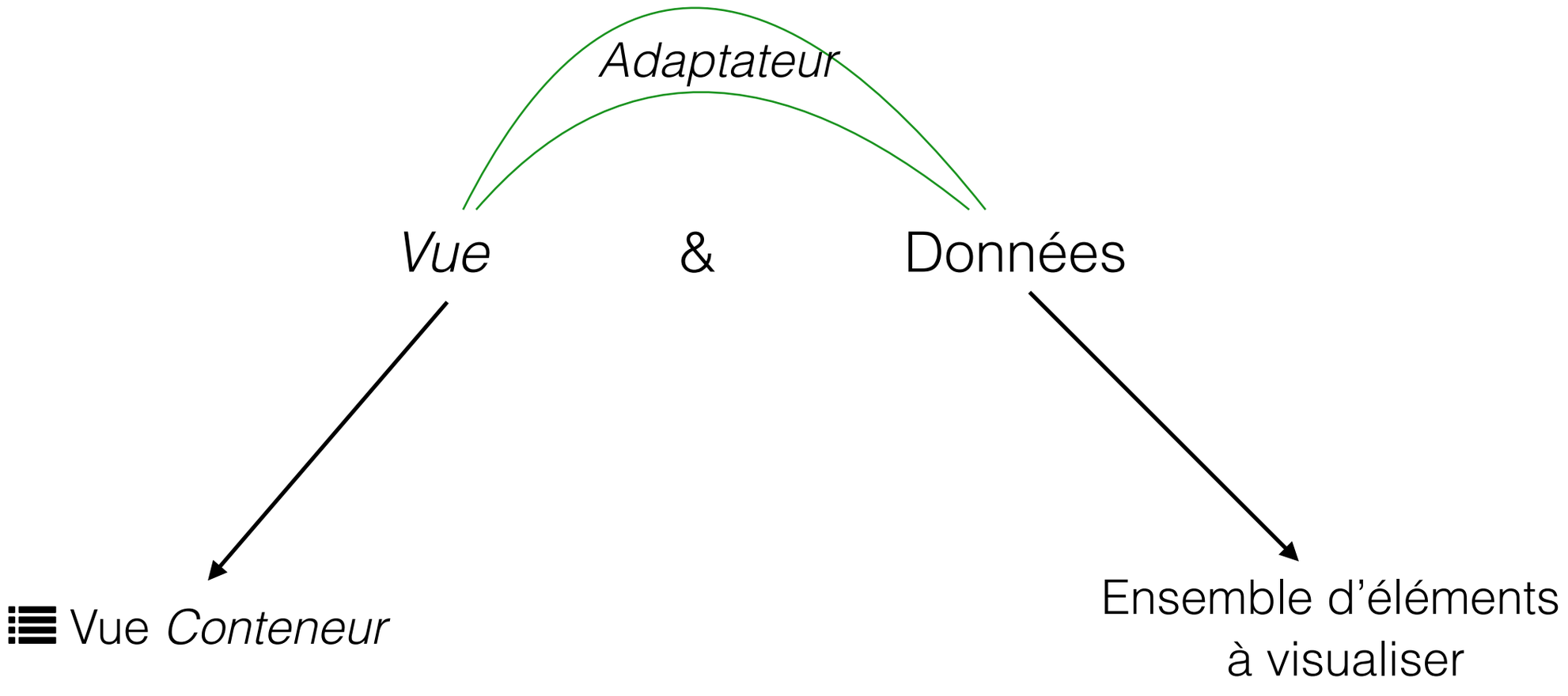
# A. Adaptateur

⟷ Pont entre Vue et Données

Exemples de Vue

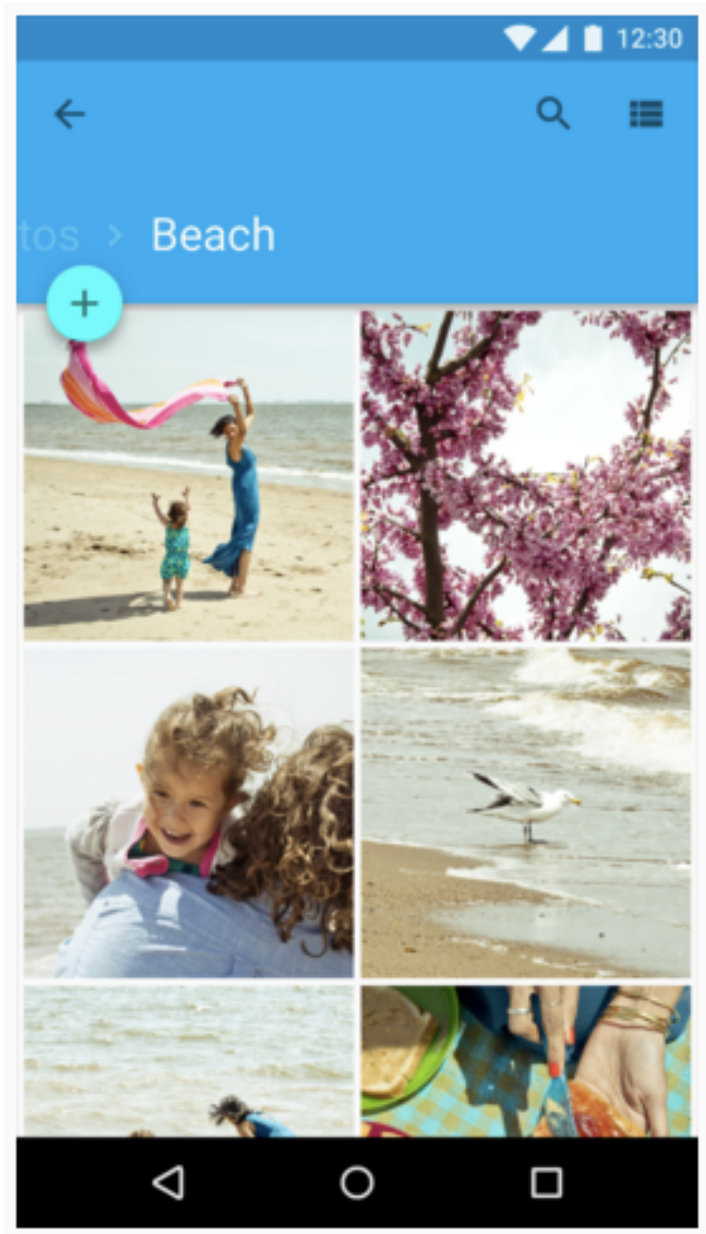
# A. Adaptateur

## ⟷ Pont entre Vue et Données



# A. Adaptateur

## Exemples de Vue

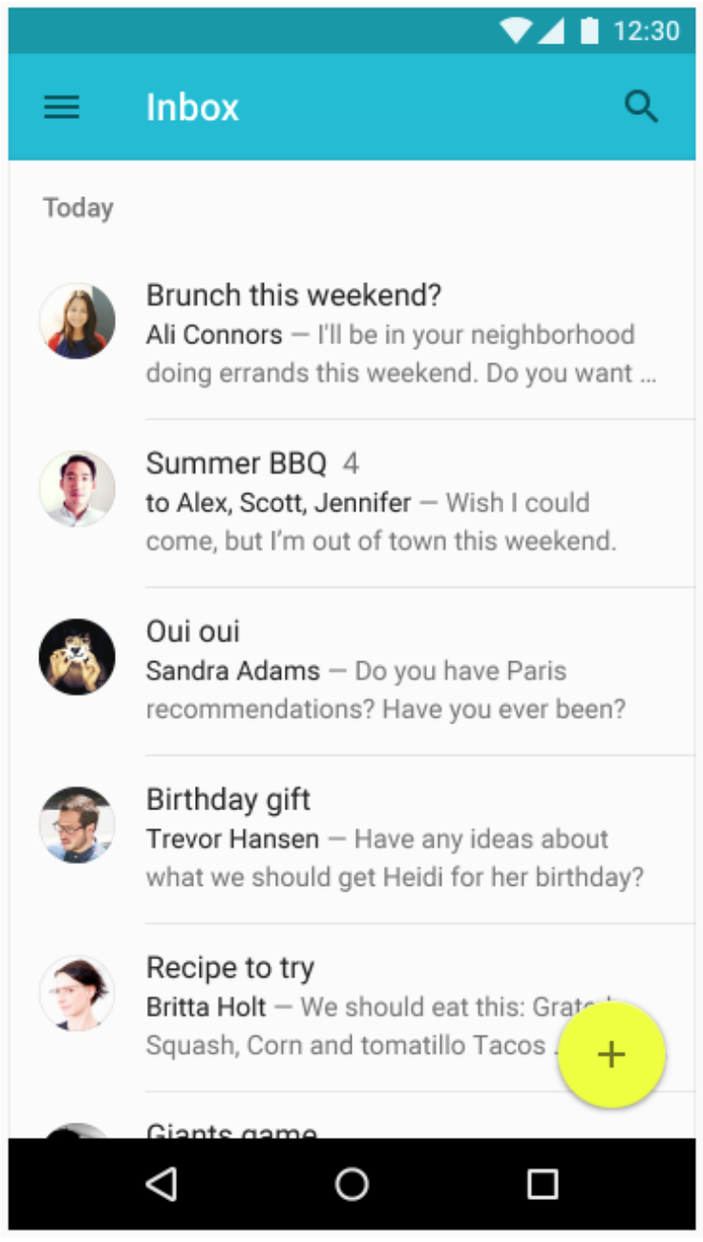



GridView

RecyclerView.Adapter

# A. Adaptateur

## Exemples de Vue

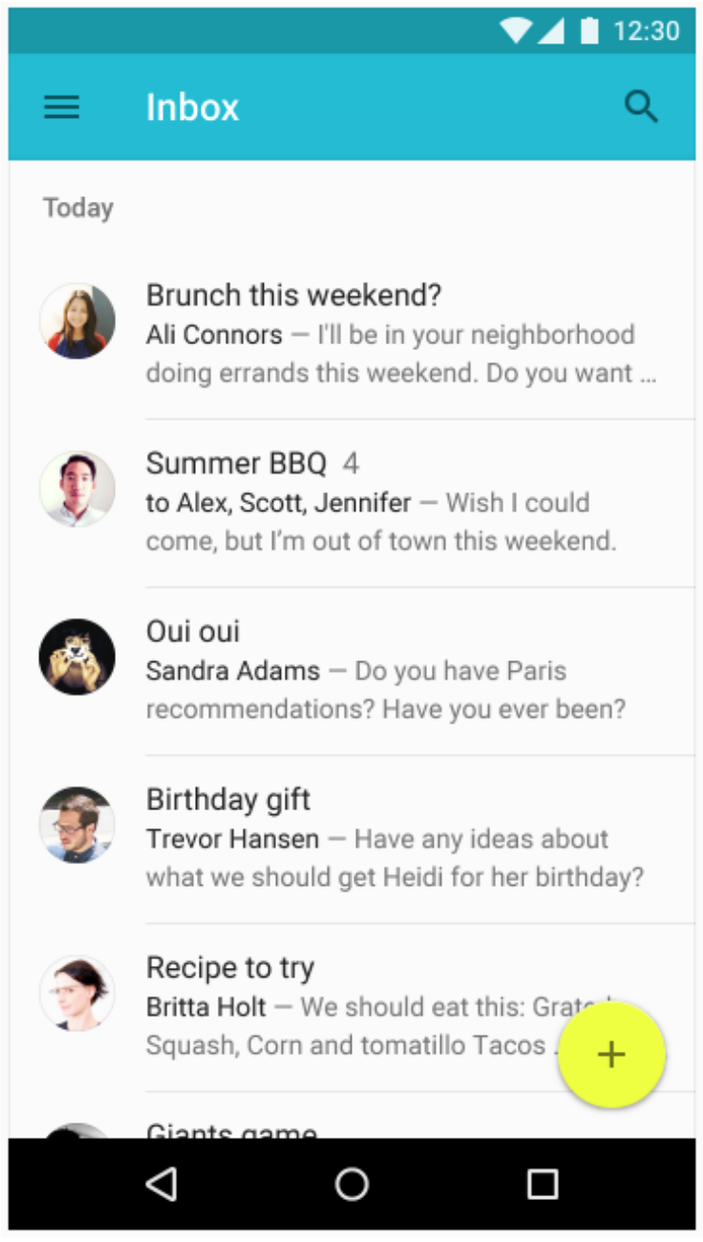



`ListView`

`ArrayAdapter`

# A. Adaptateur

## Exemples de Vue

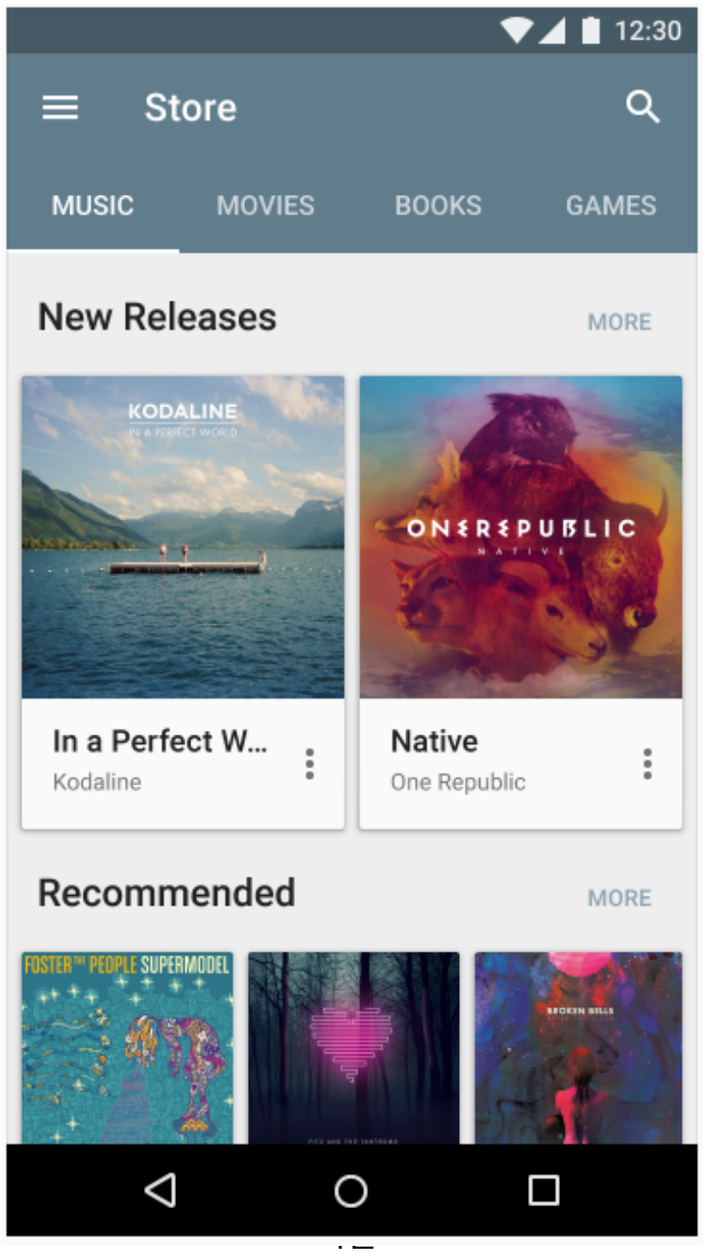



`RecyclerView`

`RecyclerView.Adapter`

# A. Adaptateur

## Exemples de Vue

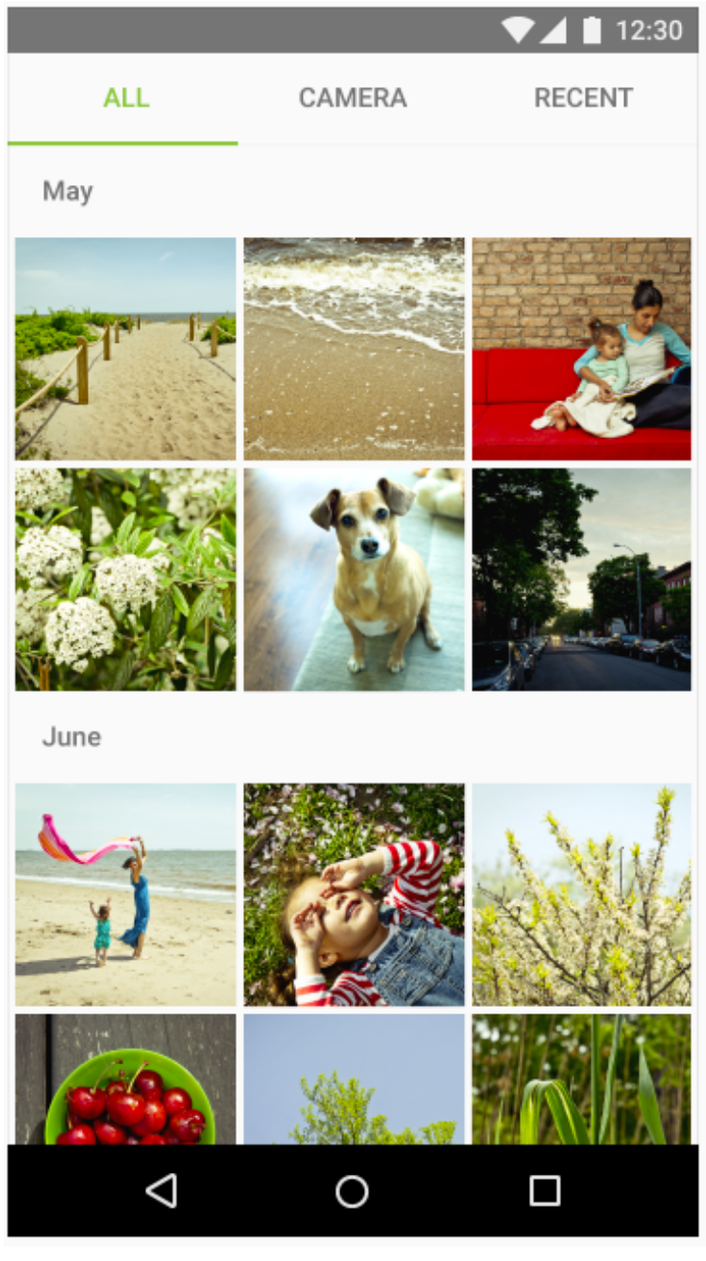



↔ Tabs

🔗 `FragmentPagerAdapter`
`FragmentStatePagerAdapter`

# A. Adaptateur

## Exemples de Vue

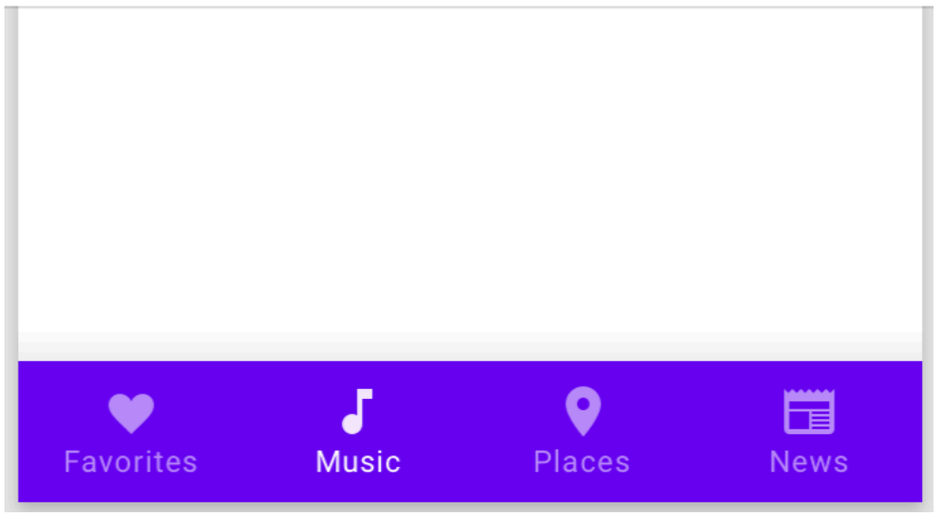



↔ ViewPager

🔗 FragmentPagerAdapter
FragmentStatePagerAdapter

# A. Adaptateur

## Exemples de Vue

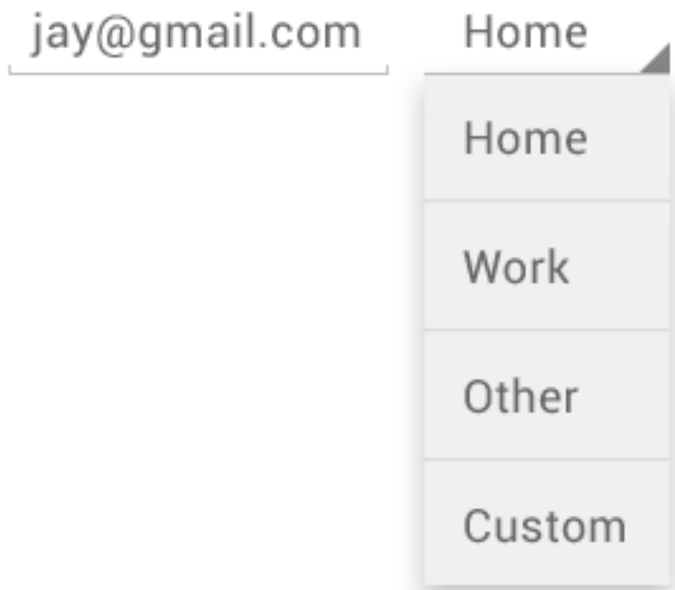



↔ `BottomNavigation`

🔗 `FragmentPagerAdapter`
`FragmentStatePagerAdapter`

# A. Adaptateur

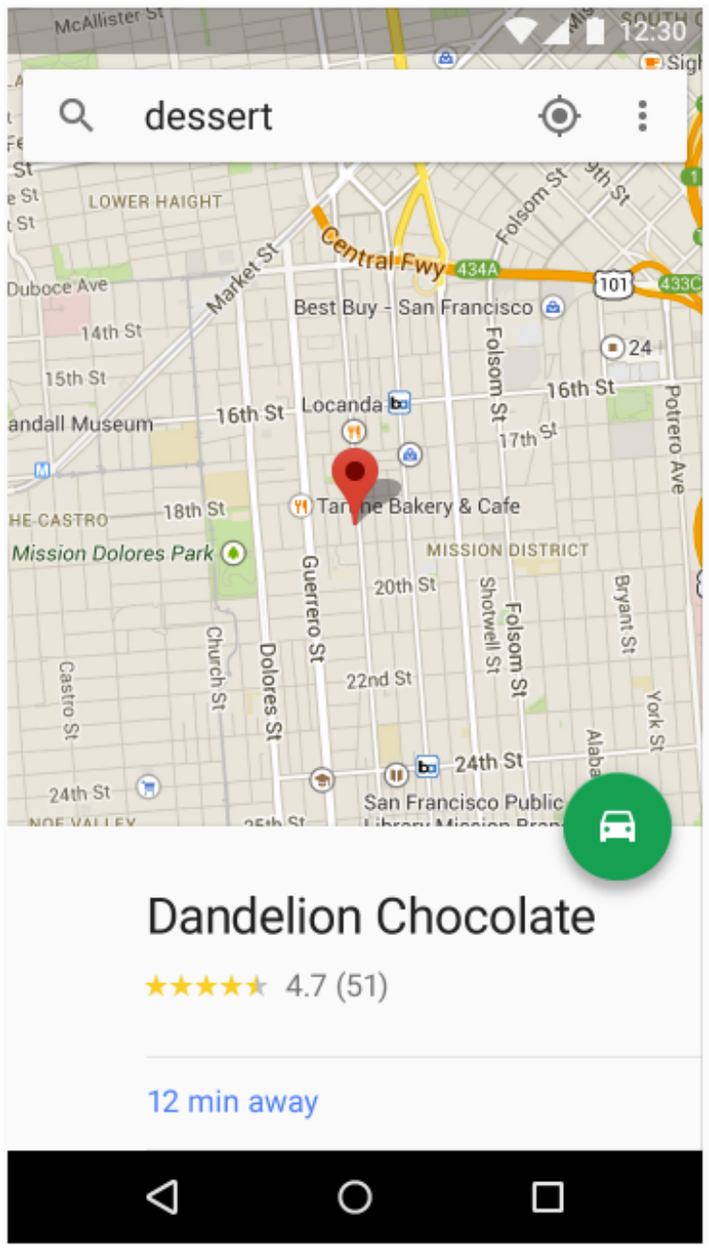
## Exemples de Vue




- Spinner
- `SpinnerAdapter`

# A. Adaptateur

## Exemples de Vue

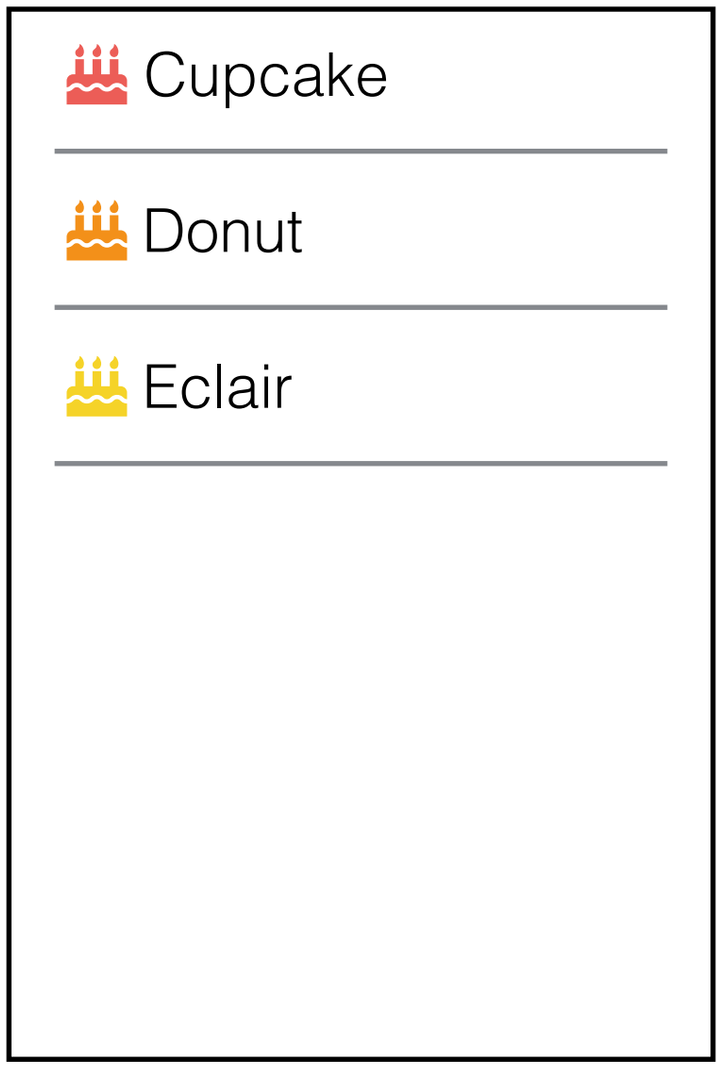



Map

`InfoWindowAdapter`

# B. Adaptateur et Liste

## </> Implémentation



# B. Adaptateur et Liste 

</> Implémentations liées aux données

</> Implémentation de l'adaptateur

</> Implémentation de l’activité principale

# B. Adaptateur et Liste 

</> Implémentation liées aux données

1. Créer une classe représentant l'objet à afficher
2. Initialiser un tableau d'objet dans l'`Activity` principale

</> Étapes d'implémentation

# B. Adaptateur et Liste

## </> Implémentation : ✚ class `extends Activity`

```
class MainActivity : AppCompatActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
  }
}
```

<0/> Créer un nouveau projet

# B. Adaptateur et Liste

</> Implémentation : `AndVersion`

```
data class AndVersion (var name: String=“Toto”)
```

<1/> Créer une classe représentant l’objet à afficher

# B. Adaptateur et Liste

</> Implémentation : `strings.xml`

```xml
<string-array name="name">
    <item>Cupcake</item>
    <item>Donut</item>
    <item>Eclair</item>
</string-array>
```

<2 Bis/> Créer un tableau de chaînes de caractères

# B. Adaptateur et Liste

## </> Implémentation : ✚ tableau de `MyObject`

```
class MainActivity : AppCompatActivity() {

    var items = Array<AndVersion>(10, { AndVersion() })

    override fun onCreate(savedInstanceState: Bundle?) {
            ...
     seedItems()
    }

    fun seedItems(){
      val nameArray = resources.getStringArray(R.array.name)
      for (k in 0..2) {
       items[k] = AndVersion(nameArray[k])
      }
    }
}
```

## <2/> Initialiser un tableau d’objet

# B. Adaptateur et Liste

## </> Implémentation de l'adaptateur

3. Créer une classe héritant de `RecyclerView.Adapter`
4. Créer un layout XML affichant une ligne de la liste
5. Créer une composition de 4., de type `RecyclerView.ViewHolder`
6. Implémenter les méthodes liées à l'`Adapter`

### </> Étapes d'implémentation

# B. Adaptateur et Liste

## </> Implémentation

ℹ `getView()` : Appel à `findViewById()`

→ Récupèrer elts de la ligne

⚠ Coûteux

💡 `ViewHolder` : Système de cache pour les vues

Vue de la ligne → Cache

Éviter de recharger à chaque affichage

Diminue appel à `findViewById()`

# B. Adaptateur et Liste

</> Implémentation : ✚ class `extends RecyclerView.Adapter`

```
class AndVersionAdapter(val items: Array<AndVersion>) :
RecyclerView.Adapter<AndVersionAdapter.ViewHolder>() {
```

<3/> Créer une classe héritant de `RecyclerView.Adapter`

# B. Adaptateur et Liste

## </> Implémentation : ✚ item_and_version.xml

⚠ Height ⟶

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeight">

    <TextView
        android:id="@+id/andVersionName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</LinearLayout>
```

<4/> Créer un layout XML affichant une ligne de la liste

# B. Adaptateur et Liste

## </> Implémentation : ✚ item_and_version.xml

### Bonus

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeight">

    <TextView
        android:id="@+id/andVersionName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</LinearLayout>
```

Utilisation Style Android

<4/> Créer un layout XML affichant une ligne de la liste

# B. Adaptateur et Liste

## </> Implémentation : ✚ `RecyclerView.ViewHolder`

ℹ Pâtisserie à Afficher

```
class AndVersionAdapter(val items: Array<AndVersion>) :
RecyclerView.Adapter<AndVersionAdapter.ViewHolder>() {

    class ViewHolder(val view: View) : RecyclerView.ViewHolder(view) {
        fun bindAndVersion(andVersion: AndVersion) {
            with(andVersion) {
                itemView.andVersionName.text = "$name"
      }
    }
```

ℹ Un elt de la Vue

ℹ Initialisation d'un des elts de la vue

<5/> Créer un objet de type `RecyclerView.ViewHolder`

# B. Adaptateur et Liste

</> Implémentation : la méthode *getItemCount()* liées à l'`Adapter`

ℹ Retourne nombre d'elts à afficher

```
class AndVersionAdapter(val items: Array<AndVersion>) :
RecyclerView.Adapter<AndVersionAdapter.ViewHolder>() {
    override fun getItemCount(): Int = items.size

}
```

<6.1/> Retourner le nombre d'éléments de la vue

# B. Adaptateur et Liste

## </> Implémentation : Bonus

```
class AndVersionAdapter(val items: Array<AndVersion>) :
RecyclerView.Adapter<AndVersionAdapter.ViewHolder>() {

    fun ViewGroup.inflate(@LayoutRes layoutRes: Int, attachToRoot:
    Boolean = false): View {
        return LayoutInflater.from(context).inflate(layoutRes, this,
    attachToRoot)
    }


}
```

ℹ Fonction d'extension Kotlin

## <6.2 Bis/> Implémenter une fonction outils en Kotlin

# B. Adaptateur et Liste

</> Implémentation : la méthode *onCreateViewHolder()* liées à l'`Adapter`

```
class AndVersionAdapter(val items: Array<AndVersion>) :
RecyclerView.Adapter<AndVersionAdapter.ViewHolder>() {

  override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
  AndVersionAdapter.ViewHolder {
      return ViewHolder(parent.inflate(R.layout.item_and_version))
  }

}
```

ℹ Charge
et initialise vue

layout Android

<6.2/> Retourner la vue d'une ligne via le `ViewHolder`

# B. Adaptateur et Liste

</> Implémentation : la méthode *onBindViewHolder()* liées à l'`Adapter`

ℹ Lie holder et données

```
class AndVersionAdapter(val items: Array<AndVersion>) :
RecyclerView.Adapter<AndVersionAdapter.ViewHolder>() {

  override fun onBindViewHolder(holder: ViewHolder, position: Int) {
      holder.bindAndVersion(items[position])
  }

}
```

<6.3/> Fournir l'objet à afficher à la vue via le `ViewHolder`

# B. Adaptateur et Liste 

## </> Implémentation de l'activité principale

7. Ajouter un `RecyclerView dans` le *layout* principal
8. Initialiser le `RecyclerView` et `LinearLayoutManager`
9. Initialiser et lier l'objet `Adapter` au `RecyclerView`

</> Étapes de l'Implémentation

# B. Adaptateur et Liste

</> Implémentation

`ListView` → `RecyclerView`

# B. Adaptateur et Liste 

</> Implémentation : ✚ dépendances ‘support:recyclerview-v7’

```
dependencies {

    implementation “com.android.support:recyclerview-v7:$support_version”
}
```

<7 Bis/> Inclure la bibliothèque pour le **RecyclerView**

# B. Adaptateur et Liste

</> Implémentation : `activity_main.xml`

✚ XML avec `<RecyclerView`

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.RecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:id="@+id/recyclerView"
     android:layout_width="match_parent"
     android:layout_height="match_parent" />
```

<7/> Ajouter un `RecyclerView dans` le *layout* principal

# B. Adaptateur et Liste

</> Implémentation : ✚ `LinearLayoutManager`

```
class MainActivity : AppCompatActivity() {

 override fun onCreate(savedInstanceState: Bundle?) {
        …
      recyclerView.layoutManager = LinearLayoutManager(this)
 }
}
```

✚ `LinearLayoutManager`
| `GridLayoutManager`

<8/> Initialiser le `RecyclerView` avec un `LinearLayoutMana`

# B. Adaptateur et Liste

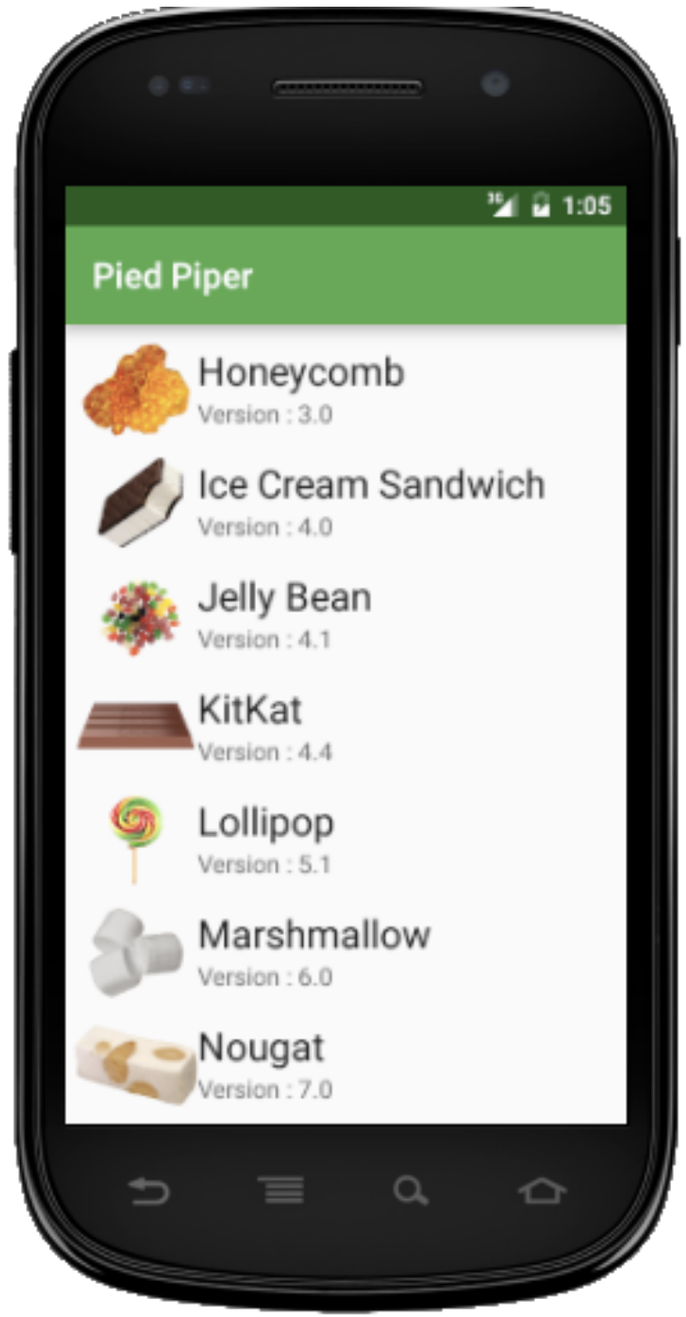
</> Implémentation : ✚ `LinearLayoutManager`

```
class MainActivity : AppCompatActivity() {

 override fun onCreate(savedInstanceState: Bundle?) {
        …
       recyclerView.adapter = AndVersionAdapter(items)

   }
}
```

<9/> Initialiser puis lier l'objet `Adapter` au `RecyclerView`

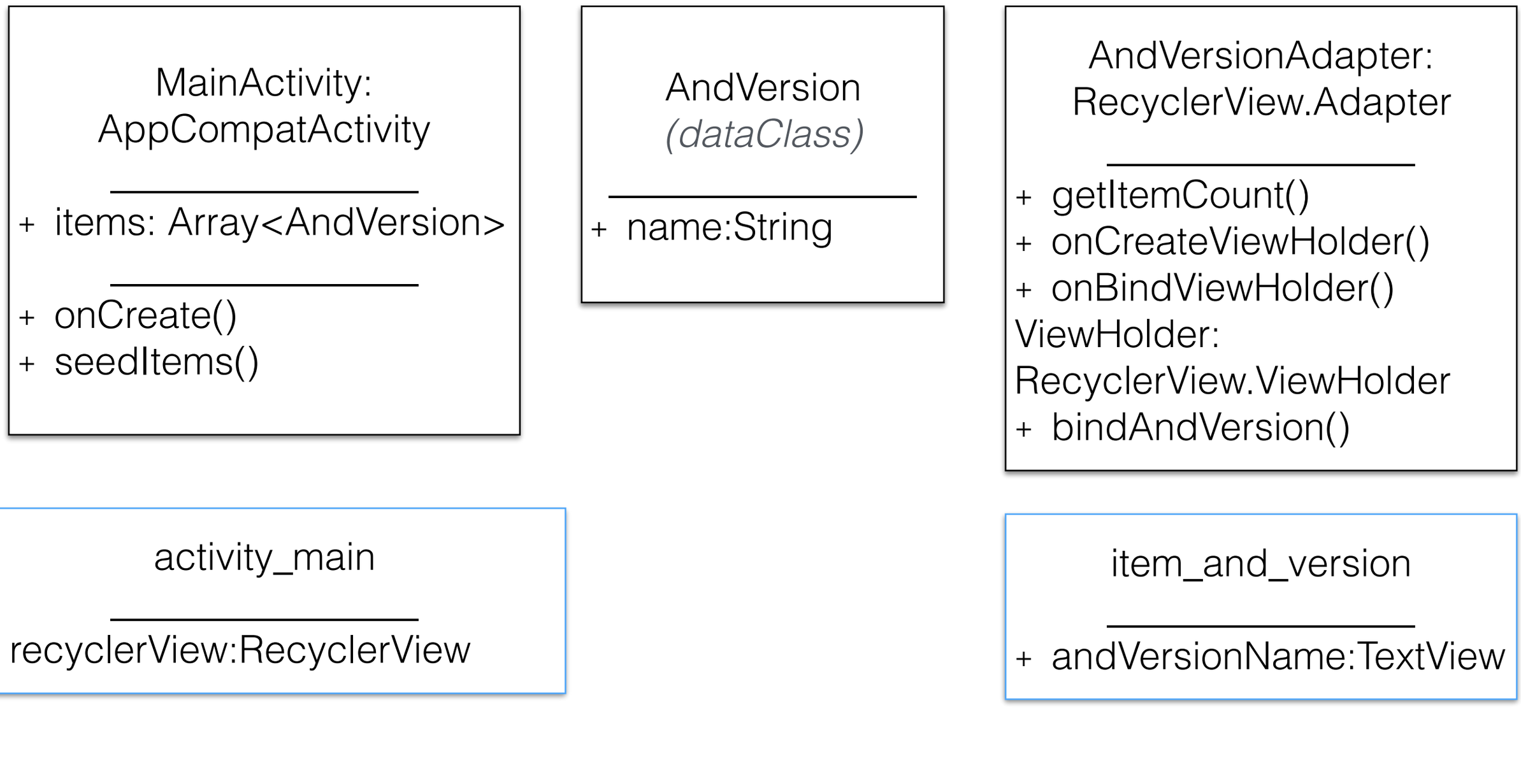
# B. Adaptateur et Liste

## </> Implémentation : Résultat



# B. Adaptateur et Liste

## </> Diagramme de classes UML



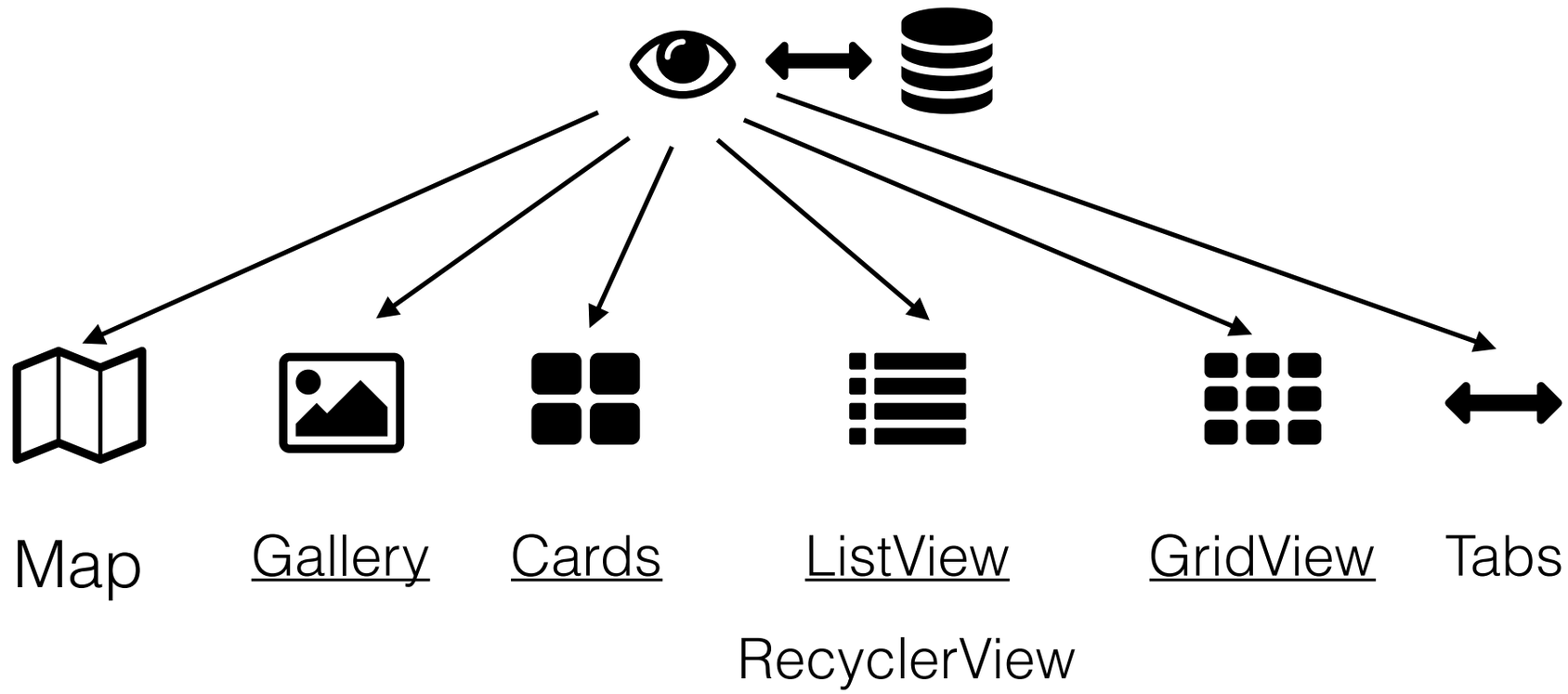
# B. Adaptateur et Liste

## </> Implémentation : Mise à jour

```
public class MainActivity extends AppCompatActivity {

    private void updateObjectList(MyAdapter adapter){
        adapter.notifyDataSetChanged();
    }

}
```

MainActivity.java

# Conclusion




</> Lien vers le code

DIFFICULT ROADS
OFTEN
LEAD TO
BEAUTIFUL
DESTINATIONS.

# Sources


- Quizz Contrôleur d'Interface
- https://antonioleiva.com/recyclerview-adapter-kotlin/
- Edition Eni : Android 5 Les fondamentaux du développement d'applications Java (Livre)
- developer.android : Support Library Features
- developer.android : Creating Lists and Cards
- Adapter
- developer.android : Creating swipe views with tabs
- developer.android : Fragments