# Lab 07: RecyclerView - Systéme Solaire

# RecyclerView - Systeme Solaire

Atelier Développement Mobile

Réaliser par : Mechergui Med Aziz Dhahbi Mohamed Amine

Encadré par:Maddouri Faouzi

**Problématique**:

Comment créer une application android contenant un recyclerView affichant la liste des planets du systeme solaire?

**Solution:**

1-on commence tout d’abord par la creation de ces classes et fichiers xml de cette arborescence du projet comme l’indique la figure 1:

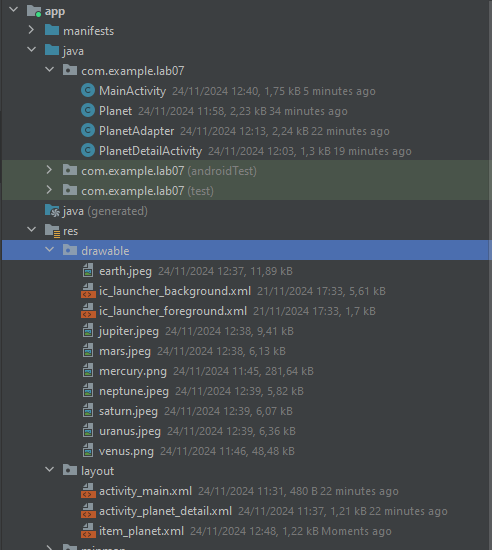


Figure : arborescence du projet

2-puis on ecrit le code de la classe **Planet** des figures 2,3 et 4 ci-dessous:

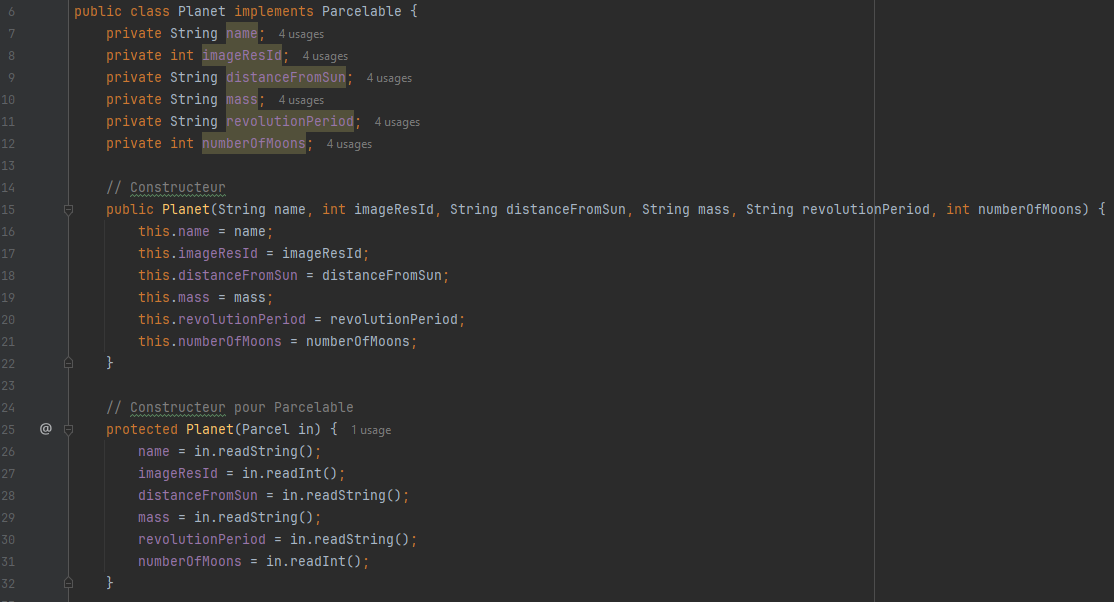


Figure :Code De La classe **Planet**

La classe **Planet** implémente l'interface **Parcelable** pour permettre de transférer des objets Planet d'une activité à une autre.

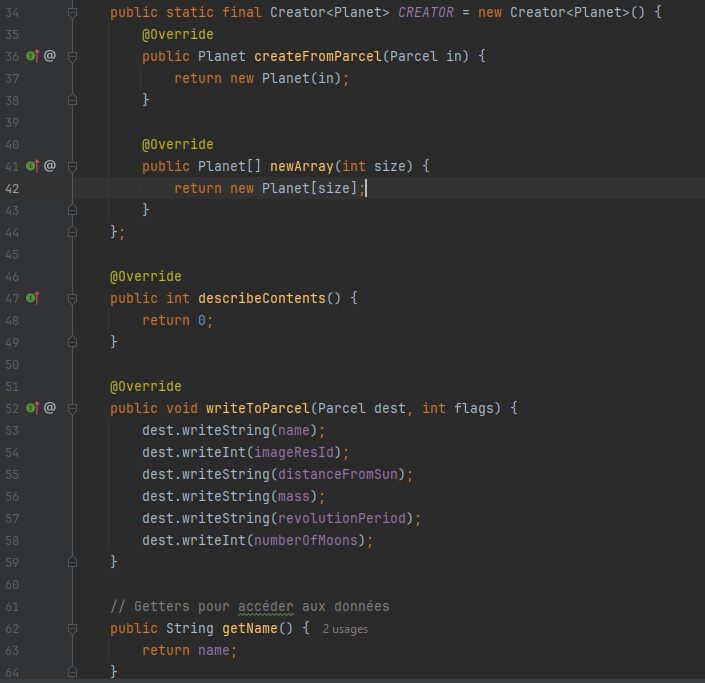


Figure :Code de la classe **Planet**

* **describeContents()** : Méthode requise par l'interface **Parcelable**. Elle renvoie une valeur

indiquant le type de contenu de l'objet. Ici, on retourne simplement 0, car nous n'avons

pas de données spéciales à décrire.

* **writeToParcel(Parcel dest, int flags)** : Cette méthode écrit les données de l'objet dans

un **Parcel**. Cela permet de "sérialiser" l'objet pour le transmettre entre activités

* La classe définit également un **CREATOR**, qui est responsable de la création d'un objet

**Planet** à partir d'un **Parcel**. Cela permet à Android de reconstituer l'objet lorsque vous le recevez dans une autre activité.



3-Maintenant on ecrit le code de la classe planetAdapter des figures 5 et 6 ci-dessous

Figure :Code de la classe planet



Figure :Code de la classe PlanetAdapter

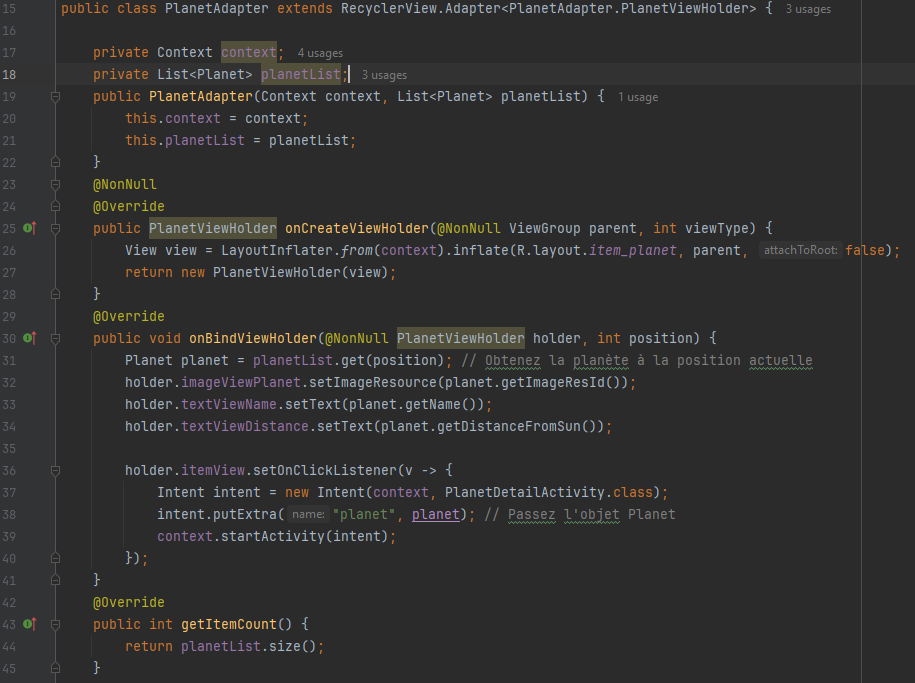


Figure : Code de la classe PlanetAdapter

**onCreateViewHolder(ViewGroup parent, int viewType)** :

* Crée une vue pour chaque élément de la liste en utilisant le fichier XMLitem\_planet.xml.

**onBindViewHolder(PlanetViewHolder holder, int position)** :

* Lie les données d'une planète (nom, distance, image) aux vues correspondantes dans le ViewHolder.

**ViewHolder** :

* Une classe interne qui contient les références des vues d'un seul élément de la liste (image, nom, distance).

4-Ensuite on ecrit le code de la classe **PlanetDetailActivity**

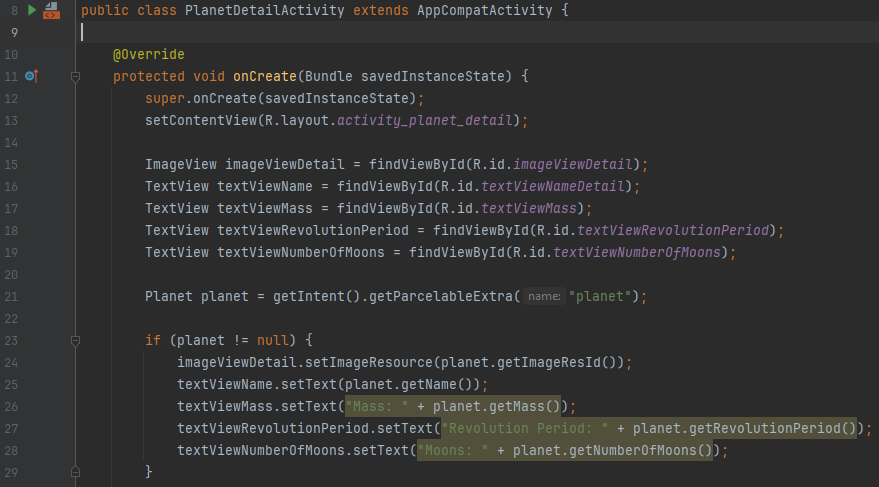


Figure :Code de la classe PlanetDetailActivity

**OnCreate**

* Charge le layout défini dans **activity\_planet\_detail** grâce à **setContentView**.

**getIntent().getParcelableExtra(“planet”)**

* Récupère l'objet **Planet** passé depuis l'activité principale (**MainActivity**) via un Intent.

5**-**On passe maintenant à l’implementation du XML et on commence avec le fichier **item\_planet.xml**

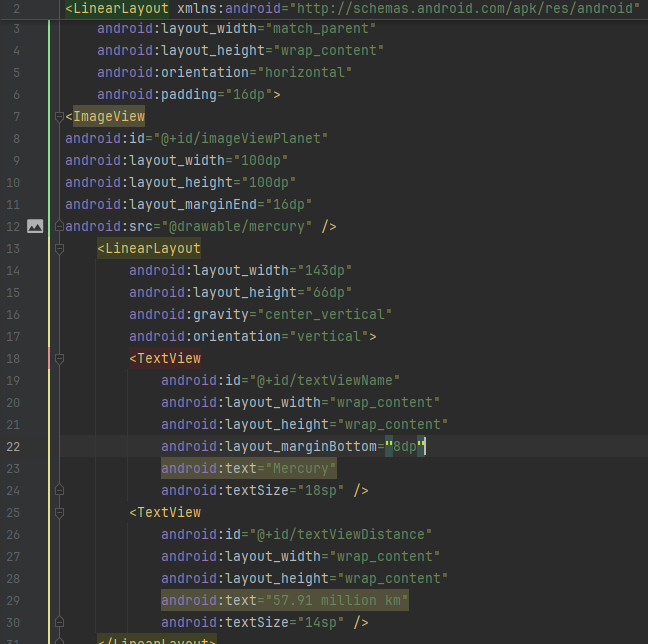


Figure :Code XML du fichier **item\_planet.xml**

6-Puis le fichier **activity\_planet\_detail.xml**



Figure :Code XML du fichier **activity\_planet\_detail.xml**

7-Ensuite le code Xml du fichier **activity\_main.xml**

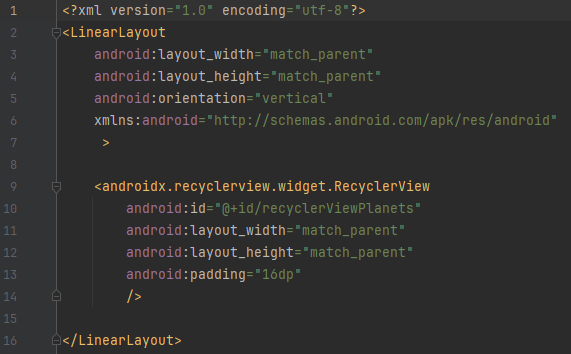


Figure :Code XML du fichier **activity\_main.xml**

8-Enfin on implemente le code de la classe MainActivity:

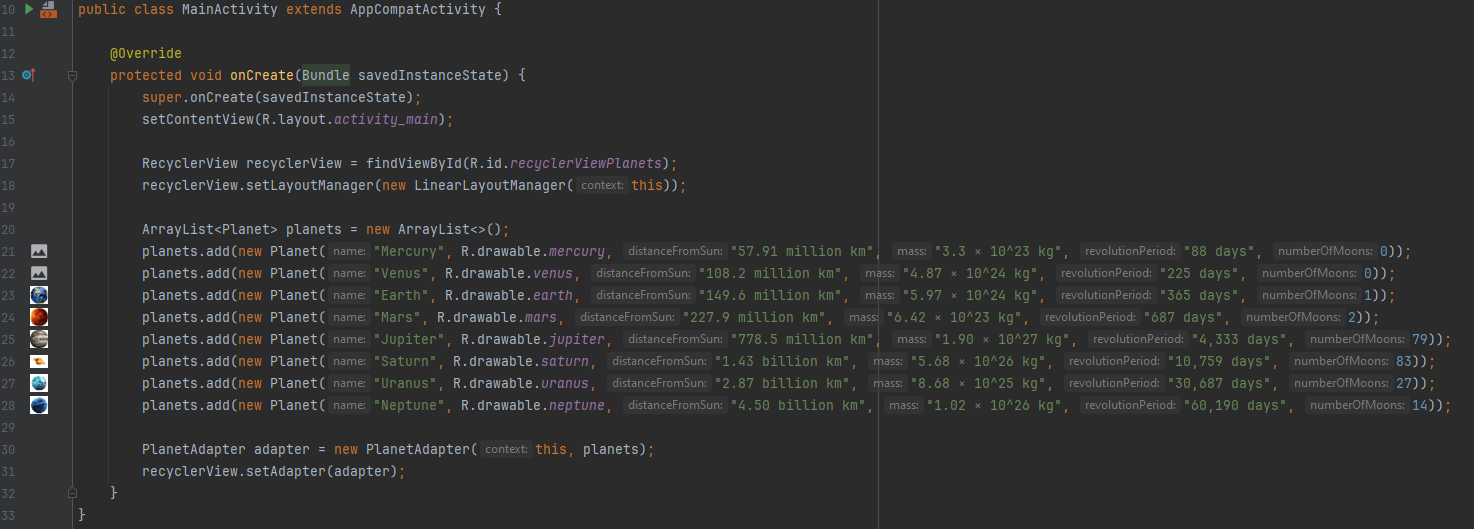


Figure :Code de la classe MainActivity

**onCreate(Bundle savedInstanceState)** :

* C'est la méthode appelée lors de la création de l'activité.
* Elle initialise l’interface graphique en chargeant le fichier xml principal.

**setContentView(int layoutResID)** :

* Spécifie quel fichier XML sera utilisé comme interface utilisateur pour cette activité.

**findViewById(int id)** :

* Permet de récupérer les vues définies dans le fichier XML (comme le **RecyclerView**).

**setLayoutManager(LayoutManager manager)** :

* Définit la disposition des éléments dans le **RecyclerView**. Ici, un LinearLayoutManager est utilisé pour afficher les éléments verticalement.

9-Maintenat on lance le projet et voila le resultat final :

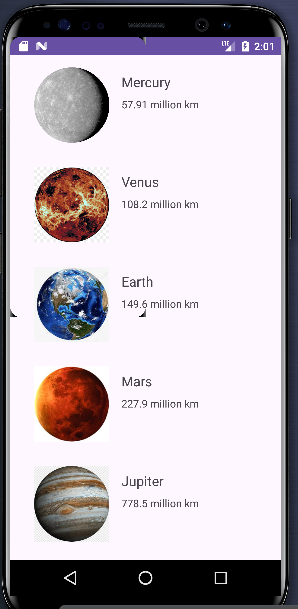
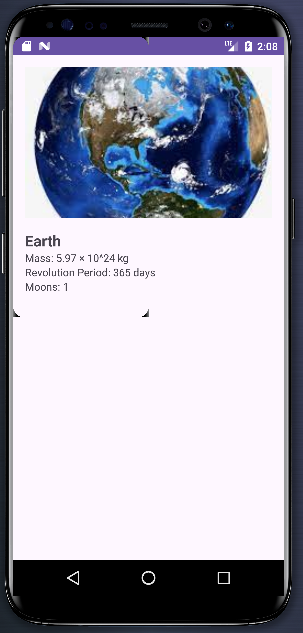


Figure 13:Les données d’une planéte

Figure 12 :La liste des planétes