

developement MVC & javadoc

Atelier 4 – Développement mobile



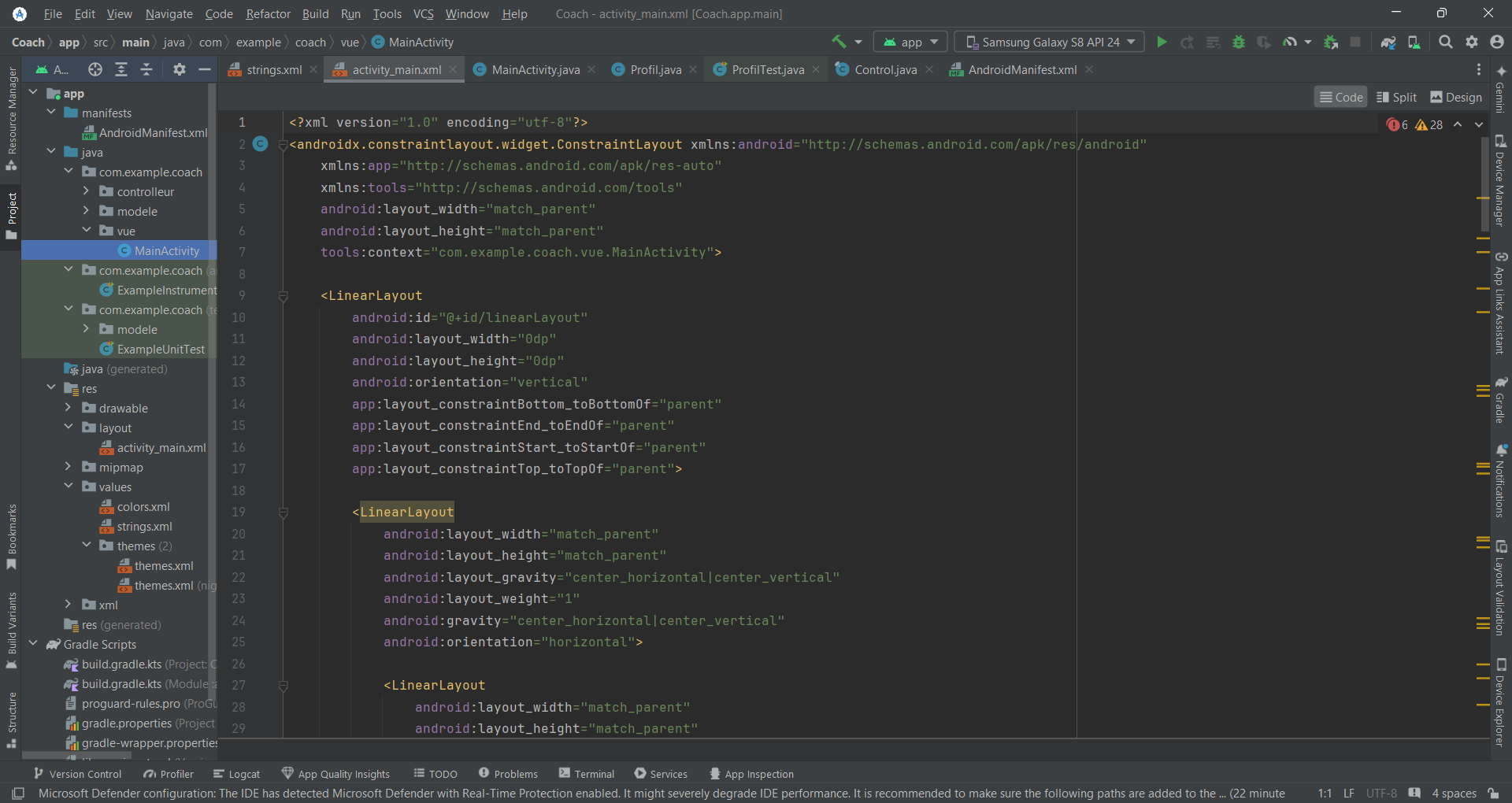
06 novembre 2024

Hafsi Mahdi

ISET RADES

Dans la première partie on a concentré sur la page Activity\_main.XML.

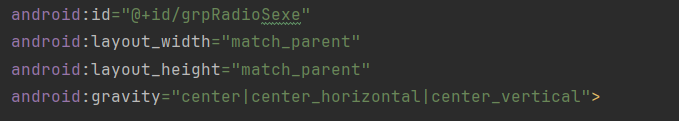
Alors quand on parle de activity\_main.xml qui est un fichier de Resource\layout\. On parle sur la partie graphique de l’application , chaque Activité de l’application a un fichier xml qui le représente comme dans l’exemple suivant :



Le fichier xml se compose des balises qui représente chaque composant ajouter dans l’activité comme dans notre application , je site quelques exemples :

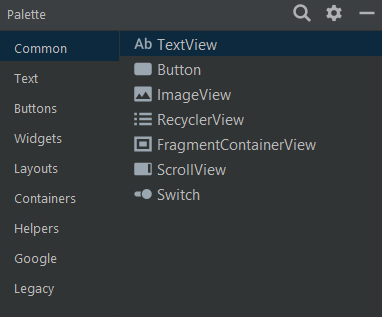
* LinearLayout
* ConstraintLayout
* RadioGroup
* RadioButton
* TextView
* EditText
* Button

Chaque composants à des attribut spécifier comme l’Id , la forme et la position …

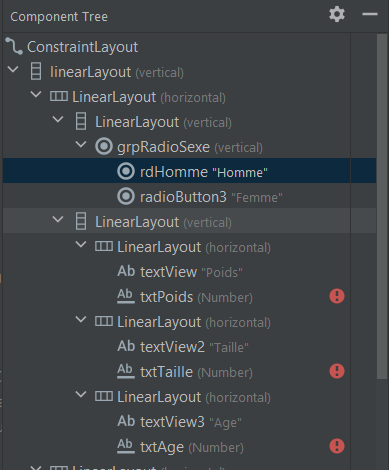


Dans la construction de l’interface de l’activité via l’onglet Design d’Android Studio (Générateur de composant ), on trouve une palette d’outils qui nous aides et nous facilite l’ajout de chaque view de l’activité si on parle des view ou des layout ou des graphique de géolocalisation.

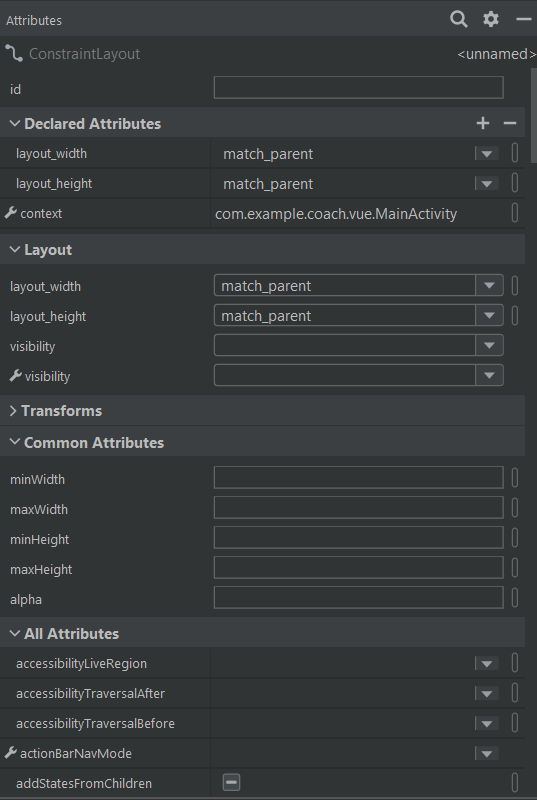
Voici la palette que j’ai parlé dans le paragraphe précédent :



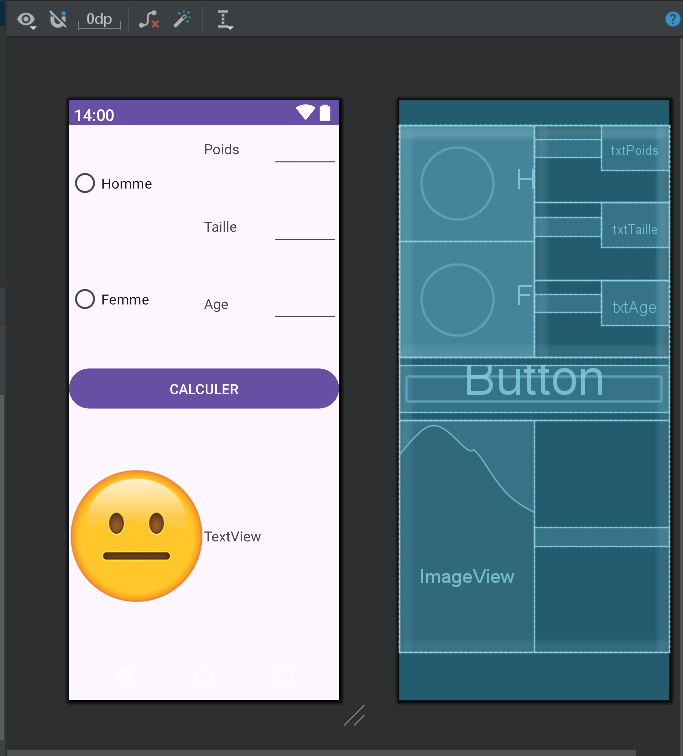
La construction se base sur un arbre de composant imbriqué entre eux ou séparer selon le bon choix de notre design.



Aussi Android studio nous facilite le travail avec une palette d’attribut pour modifier les attribut .



Alors composant par composant on va construire notre activité et on a obtenu ce résultat



Afin de construire l’activité et l’affichage il nous reste maintenant a la lié avec la partie logique de l’application. On fait ça à partir le fichier AndroidManifest.xml qui est le fichier qui comporte toutes les activités de notre application, donc il faut ajouter ce bloc d’activité

<activity  
 android:name=".vue.MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
</activity>

Ici l’élément <intent-filter> est essentiel pour permettre a une application Android de Reagir a des actions spécifique provenant d’autres application ou de système lui-même. Il indique :

-l’actions

-types de données

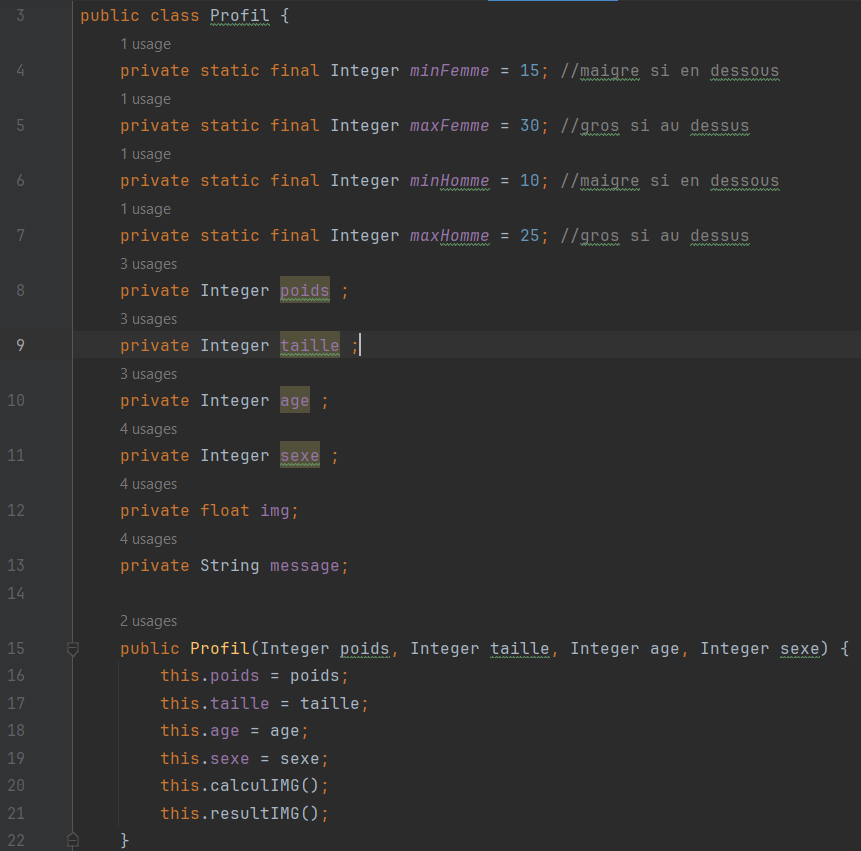
-catégorie

Ici l’action est MAIN c’est-à-dire la page de base de l’application et la catégorie LAUNCHER pour faire apparaitre l’icone de l’activité dans le menu principal de l’appareil.

Cette transition vers la partie logique en langage JAVA :

En appliquant le modèle MVC, On va débuter par créer notre model **Profil** par la création de ses attributs , son constructeur paramétré , les getters et les 2 méthodes qui concerne le calcul de l’IMG et son résultat .

Ci-dessous une capture d’écran du Modele :

En deuxieme lieu On va créer notre **Contrôleur** qui ce caractérise par:

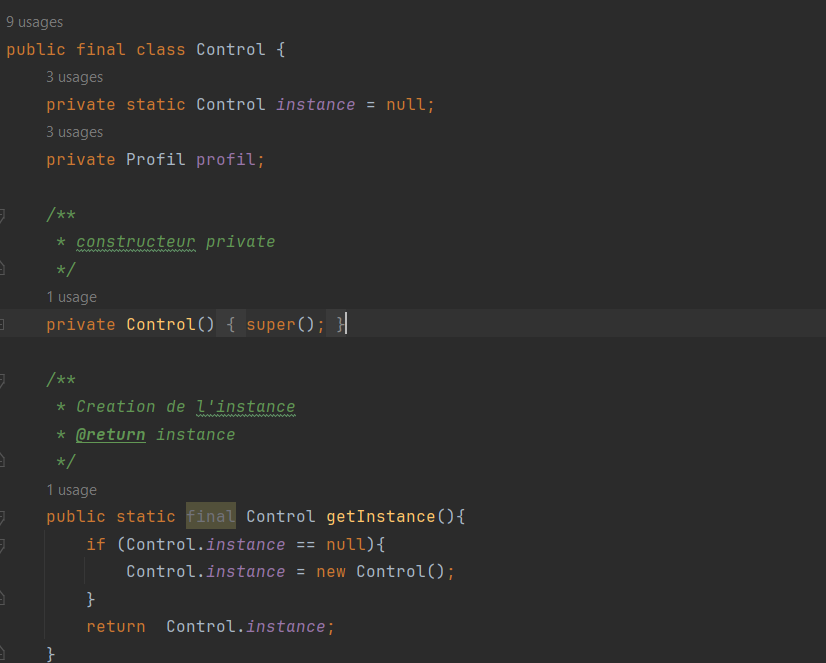
**Un class final publique .**

**Un attribut privé de type Control** qui est initiallement = null

**Un attribut privé de type Profil**

**Constructeur privé et la methode getInstance()** pour assurer le passage une seule fois vers la classe MainActivity .

Ci-dessous une capture de ces méthodes :



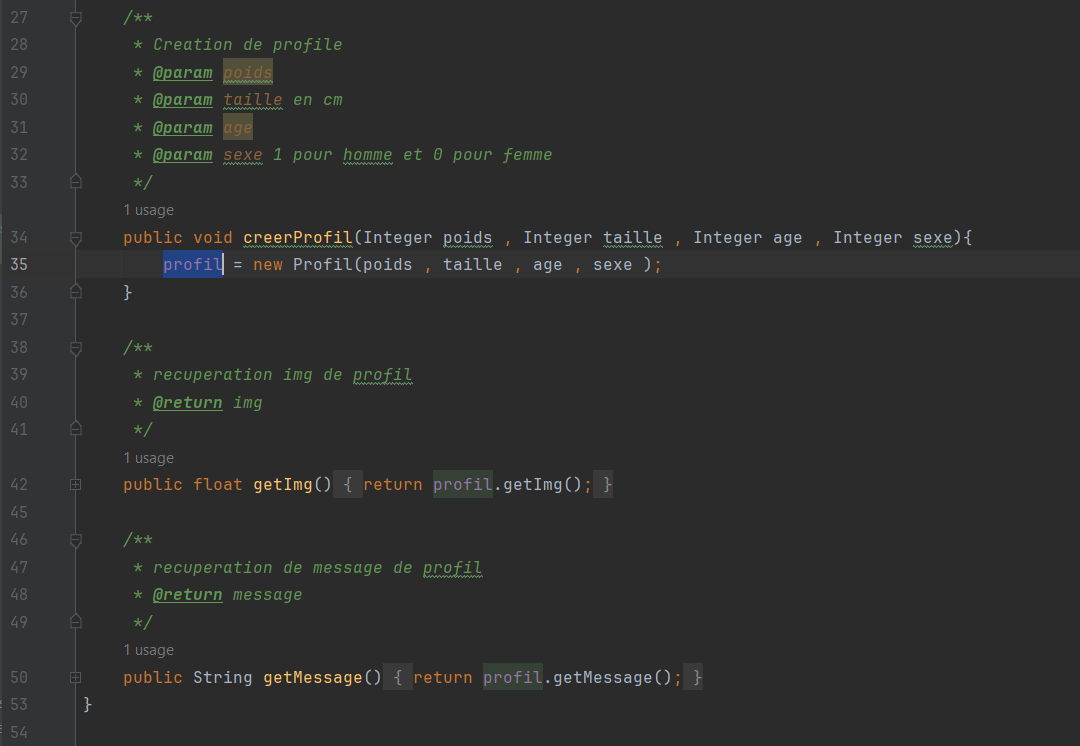
De plus on a les méthodes :

**créerProfil()** : qui permet de créer un Profil .

**getIMG()** : qui permet de récupérer l’IMG du profil

**getMessage()** qui assure de récupérer le message a afficher du profil .

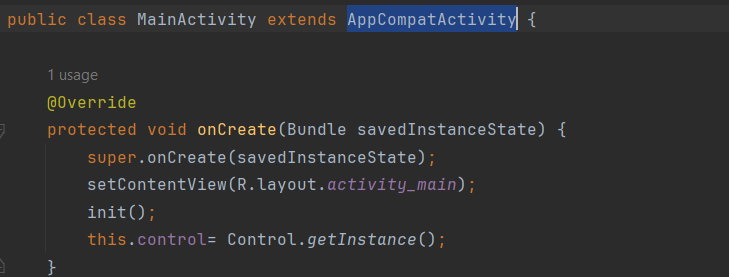
Ci-dessous une capture d’écran des méthodes :



Alors maintenant, on va détailler la Troisième phase c’est la View du model MVC :

On a créé un class MainActivity qui hérite de la classe AppCompatActivity qui represente l’activité principal de l’application et qui vas lié les autres deux parties du modèle MVC.

Premierement la methode **onCreate()**

****

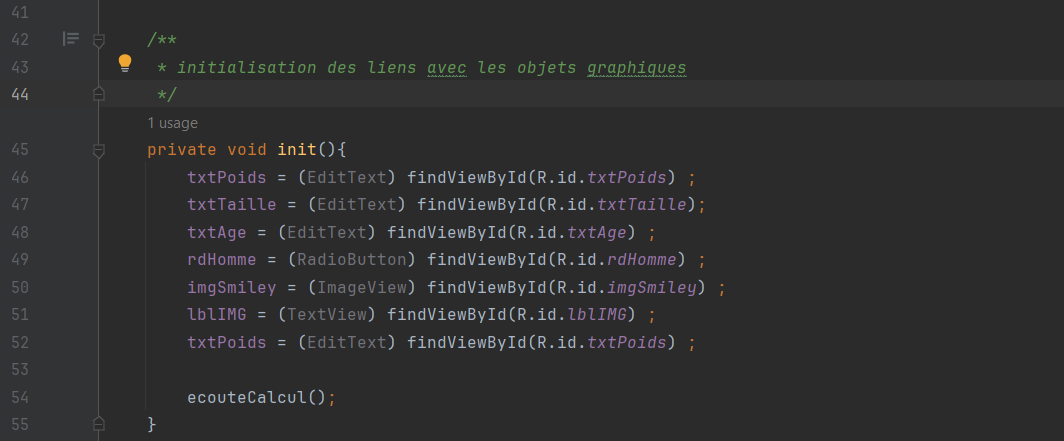
* Cette méthode prend en paramètre un attribut de type Bundle qui prend en charge de l’état de l’instance . L’appel de constructeur de class Activity : si on ferme et rouvrir l’application à partir de notre mobil elle garde son état.

onCreate est appelée au démarrage de l’activité. Elle configure la vue (setContentView) et initialise les composants graphiques avec init(). Ensuite, elle récupère l’instance de Control, qui gère les données et les opérations logiques.

Passant aux attributs qu’on va sauvegarder les données dedans .



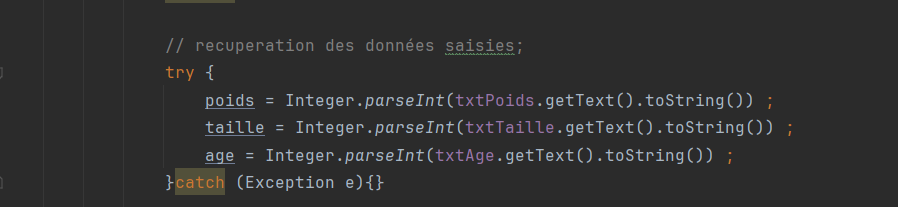
La methode init() qui lie chaque élément graphique à une variable. findViewById recherche les éléments définis dans le fichier XML activity\_main.



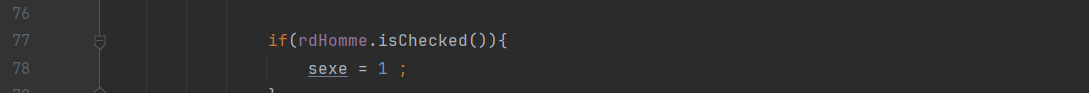
Passant à la methode ecouteCalcul() se debute par l’ecout du bouton Calculer dans l’application a traver un Listener . elle initialise les variables pour les données



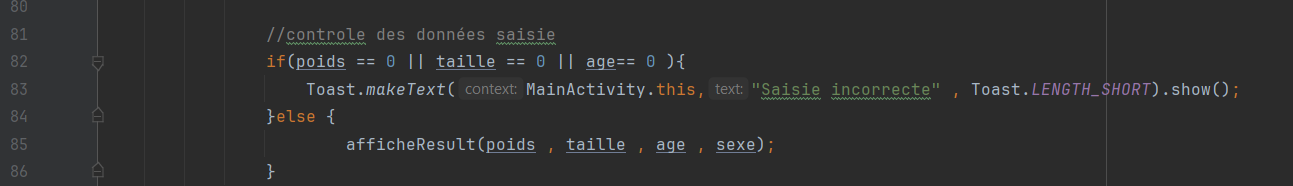
Le bloc try catch permet de convertir les valeurs saisie dans les champs en les affectants aux variables poids taille et age .



Si l’option Homme est sélectionnée (rdHomme est coché), la variable sexe est définie à 1 (représentant le sexe masculin).



La troisieme instruction du methode : Si le poids, la taille, ou l’âge sont égaux à zéro, un message toast "Saisie incorrecte" apparaît. Sinon, afficheResult est appelée pour afficher les résultats calculés



afficheResult crée un profil en appelant creerProfil dans le contrôleur. Elle récupère ensuite l’IMG et un message associé.

Cette methode comporte des methodes importantes de package widget comme :

**imgSmiley.setImageResource()**

**lblIMG.setTextColor(Color.GREEN)** ou **lblIMG.setTextColor(Color.RED)**

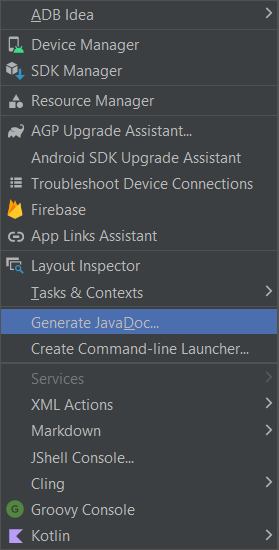
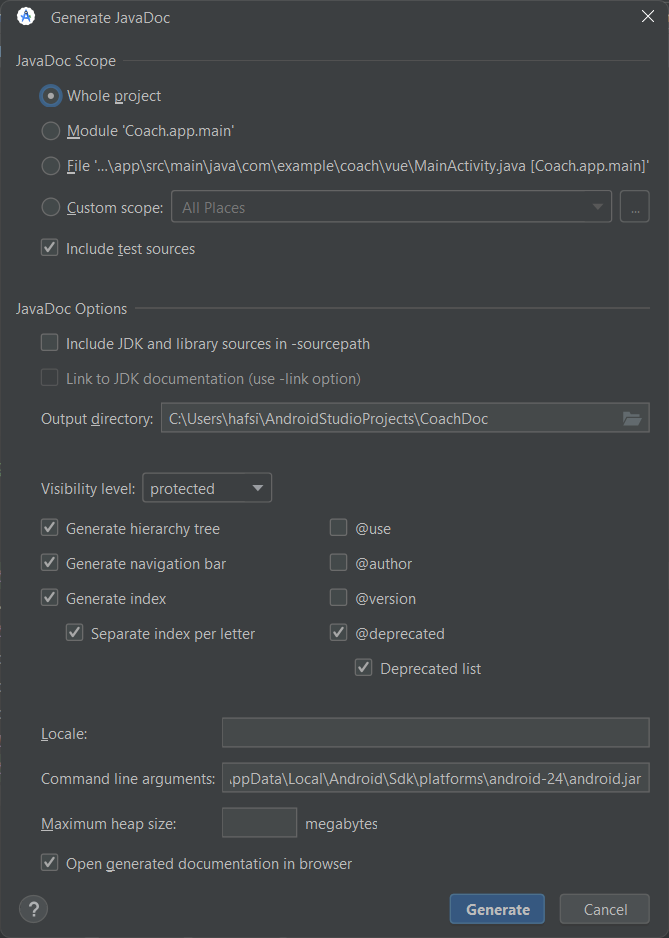
**lblIMG.setText(String)**

**JAVADOC**

JavaDoc est un outil permettant de générer automatiquement de la documentation à partir des commentaires dans le code source Java. Il permet aux développeurs de fournir des descriptions claires des classes, méthodes et variables, facilitant ainsi la compréhension du code. JavaDoc aide également à maintenir une documentation à jour et accessible pour les utilisateurs et les autres développeurs.

On appris dans cette leçon comment utiliser le javaDoc :

**etapes pour generer le javaDOC en important la libraire android.jar de dossier comportant le system image d’android :**



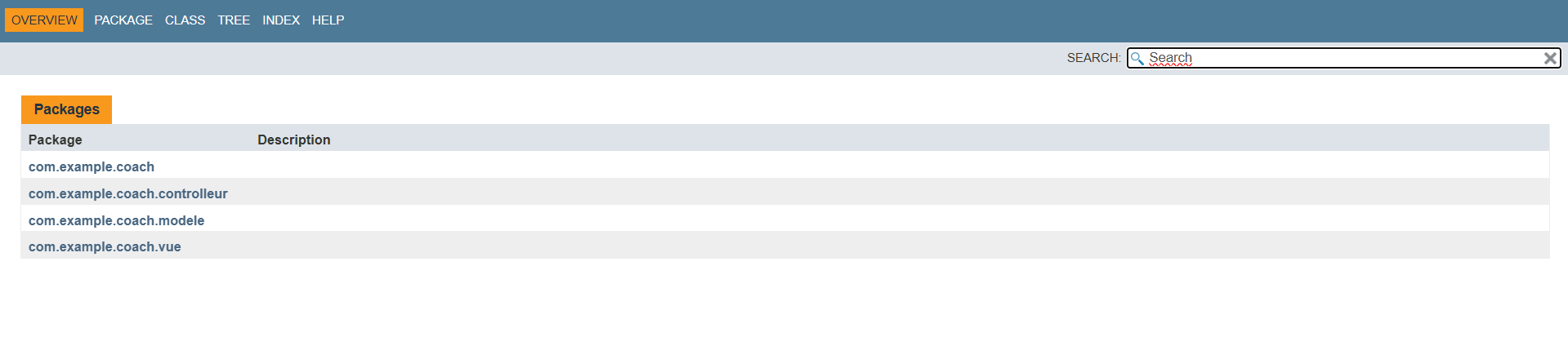
Ci-dessous un exemple de code pour la documentation javaDOC qui ne comporte ni une valeur de retour ni des paramètres :

*/\*\*  
 \* initialisation des liens avec les objets graphiques  
 \*/*private void init(){  
 txtPoids = (EditText) findViewById(R.id.*txtPoids*) ;  
 txtTaille = (EditText) findViewById(R.id.*txtTaille*);  
 txtAge = (EditText) findViewById(R.id.*txtAge*) ;  
 rdHomme = (RadioButton) findViewById(R.id.*rdHomme*) ;  
 rdFemme = (RadioButton) findViewById(R.id.*rdFemme*) ;  
 imgSmiley = (ImageView) findViewById(R.id.*imgSmiley*) ;  
 lblIMG = (TextView) findViewById(R.id.*lblIMG*) ;  
 txtPoids = (EditText) findViewById(R.id.*txtPoids*) ;  
 this.control= Control.*getInstance*(this);  
 ecouteCalcul();  
 recupProfil();  
}

Ci-dessous un exemple de code pour la documentation javaDOC qui comporte des parametres et pas de valeur de retour :

*/\*\*  
 \* Creation de profile  
 \** ***@param*** *poids  
 \** ***@param*** *taille en cm  
 \** ***@param*** *age  
 \** ***@param*** *sexe 1 pour homme et 0 pour femme  
 \*/* public void creerProfil(Integer poids , Integer taille , Integer age , Integer sexe , Context context){  
 *profil* = new Profil(new Date(), poids , taille , age , sexe );  
// accesLocal.ajout(profil);  
 *accesDistant*.envoi("enreg", *profil*.convertToJSONArray());  
// Serializer.serialize(nomFic, profil, context);  
 }

Ci-dessus la page index du javaDoc qui était générer :  
Là où on peut naviguer dans les packages pour voir les informations de chaque méthode



Par exemple : la classe contrôle du package com.example.coach.controlleur :

On trouve que les commentaires saisie sont ici pour bien informé le groupe de travail en cas de besoin.

