



Atelier :développement mobile

TP2

Étudiantes: Farhaoui Eya

Guesmi Ikram

Enseignant:Maddouri Faouzi

Année universitaire:2024/2025

Table des matières

INTRODUCTION	1
I. Les fichiers de l'applicatin	1
1. Le fichier activity_main.xml	1
2. Modele/profil.....	2
3. Controleur/control.....	2
4. Vue/MainActivity.....	4
III. Les étapes de création d'un javadoc.....	7
IV. Les activités de l'applicatin	9

Liste des figures

- Figure 1 :partie du code du fichier activity_main.xml
- Figure 2 :le design
- Figure 3 :code de la classe controle
- Figure 4 :la partie des attributs
- Figure 5 : la methode onCreate.....
- Figure 6 :code de la methode init
- Figure 7 :code de la methode ecouteCalcul.....
- Figure 8 :code de la methode afficheResult
- Figure 9 :commenter le code
- Figure 10 :création du dossier
- Figure 11 :les étapes de création d'un javadoc
- Figure 12 :l'activité obtenue pour un gros homme
- Figure 13 :l'activité obtenue pour un homme ayant un faible IMG
- Figure 14 :l'activité obtenue pour une femme ayant un IMG normal

INTRODUCTION:

Dans cet atelier, on va créer une application qui va calculer l'Indice de Masse Grasse (IMG) de l'utilisateur en fonction de son poids, sa taille, son âge et son sexe. Elle va ensuite afficher les résultats sous forme de texte et d'images, indiquant si l'IMG est "normal", "trop faible" ou "trop élevé".

I. Les fichiers de l'application :

1. Le fichier activity_main.xml:

Pour créer l'interface qui va être destinée à saisir des informations physiques (sexe, poids, taille, âge) et à afficher un résultat visuel (smiley) et textuel après une action de calcul, on va utiliser un ConstraintLayout comme conteneur principal.

```
<Button
    android:id="@+id/btnCalc"
    android:layout_width="wrap_content"
    android:layout_height="90dp"
    android:layout_weight="1"
    android:text="calculer" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center|center_horizontal|center_vertical"
    android:layout_weight="1"
    android:gravity="center|center_horizontal|center_vertical"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/imgsmiley"
        android:layout_width="231dp"
        android:layout_height="283dp"
        android:layout_weight="1"
        app:srcCompat="@drawable/normale"
        tools:srcCompat="@drawable/normale" />
```

Figure 1: partie du code du fichier activity_main.xml

Voici une description de la structure :

LinearLayout (principal) : Ce layout vertical occupe tout l'écran et contient trois sections principales : une pour les informations utilisateur, une pour un bouton, et une pour une image et un label.

Section 1 : Informations utilisateur (contenue dans un autre LinearLayout) :

- RadioGroup avec deux boutons radio (Homme, Femme) pour choisir le sexe.
- Trois ensembles de vues pour saisir des informations de poids, taille, et âge via des EditText, avec des labels correspondants.

Section 2 : Bouton :

Un bouton "calculer" permettant probablement de déclencher une action (comme un calcul).

Section 3 : Résultat :

- Une image (ImageView) affichant un smiley.
- Un texte (TextView), probablement utilisé pour afficher un résultat après le calcul.

Après avoir ajouter ces dernier sections on obtient ce design;



Figure 2:le design

2. Modele/profil:

Ce fichier Java représente la classe Profil, qui stocke et gère les informations d'un utilisateur concernant son poids, sa taille, son âge, et son sexe, pour calculer son Indice de Masse Grasse (IMG).

3. Controleur/control:

ce fichier Java qui représente un contrôleur (Controle) dans une architecture MVC, gère la création du profil utilisateur et permet de récupérer des informations comme l'IMG et un message correspondant à ce profil. Voici une brève description de ses fonctionnalités :

Classe Controle :

Il s'agit d'un **singleton**, c'est-à-dire qu'il ne peut y avoir qu'une seule instance de cette classe dans l'application. Le constructeur privé empêche la création directe de nouvelles instances.

La méthode `getInstance()` garantit que seule une instance de la classe est créée et utilisée à travers l'application.

Attribut profil :

Un objet de la classe `Profil` est utilisé pour stocker les données de profil de l'utilisateur (comme le poids, la taille, l'âge et le sexe).

Méthode `creerProfil` :

Cette méthode crée un profil en utilisant les données saisies (poids, taille, âge et sexe).

Le sexe est représenté par un entier, avec 1 pour un homme et 0 pour une femme.

Méthode `getImg()` :

Cette méthode récupère et retourne l'indice de masse grasse (IMG) du profil, calculé dans la classe `Profil`.

Méthode `getMessage()` :

Elle retourne un message associé au profil, probablement un commentaire ou une interprétation de l'IMG (par exemple, si la personne est dans une fourchette normale, en surpoids, etc.).

```
5 public class Controle {
6     3 usages
7     private static Controle instance= null;
8     3 usages
9     private Profil profil;
10    1 usage
11    private Controle () { super () ; }
12
13    1 usage
14    public static final Controle getInstance () {
15        if (Controle.instance == null) {
16            Controle.instance = new Controle ();
17        }
18        return Controle.instance;
19    }
20    1 usage
21    public void creerProfil (Integer poids, Integer taille, Integer age, Integer sexe){
22        profil = new Profil(poids, taille, age, sexe);
23    }
24
25    1 usage
26    public float getImg() { return profil.getImg(); }
27
28    1 usage
29    public String getMessage() { return profil.getMessage(); }
```

Figure 3:code de la classe controle

4. Vue/MainActivity

Ce fichier Java représente l'activité principale (`MainActivity`) d'une application Android qui calcule et affiche l'Indice de Masse Grasse (IMG) de l'utilisateur en fonction de certaines données saisies. Voici une brève description :

- **Attributs :**

Des composants de l'interface utilisateur, tels que les champs de texte pour le poids, la taille, et l'âge (`EditText`), un bouton radio pour le sexe (`RadioButton`), une image (`ImageView`) et une étiquette pour afficher l'IMG (`TextView`).

Un objet `Controle` pour gérer la logique de l'application.

```
4 usages
private Controle controle;
2 usages
private EditText txtPoids;
2 usages
private EditText txtTaille;
2 usages
private EditText txtAge;
2 usages
private RadioButton rdHomme;
4 usages
private TextView lblIMG;
4 usages
private ImageView imgSmiley;
```

Figure 4:la partie des attributs

- **Méthode onCreate :**

Cette méthode initialise l'activité lorsque l'application est lancée. Elle charge le layout (`setContentView`) et appelle la méthode `init()` pour lier les composants de l'interface aux attributs Java.

Elle instancie également le contrôleur (`Controle`) en appelant `Controle.getInstance()`.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    init();
    controle = Controle.getInstance();
    ecouteCalcul(); |
}

```

Figure 5: la methode oncreate

- **Méthode init() :**

Cette méthode associe les composants de l'interface utilisateur (définis dans le fichier XML) à leurs équivalents Java à l'aide de findViewById().

```

↑ usage
private void init() {
    txtPoids = (EditText) findViewById(R.id.editTextNumber6);
    txtTaille = (EditText) findViewById(R.id.editTextNumber7);
    txtAge = (EditText) findViewById(R.id.editTextNumber8);
    rdHomme = (RadioButton) findViewById(R.id.rdHomme);
    lblIMG = (TextView) findViewById(R.id.lblIMG);
    imgSmiley = (ImageView) findViewById(R.id.imgsmiley);
}

```

Figure 6:code de la methode init

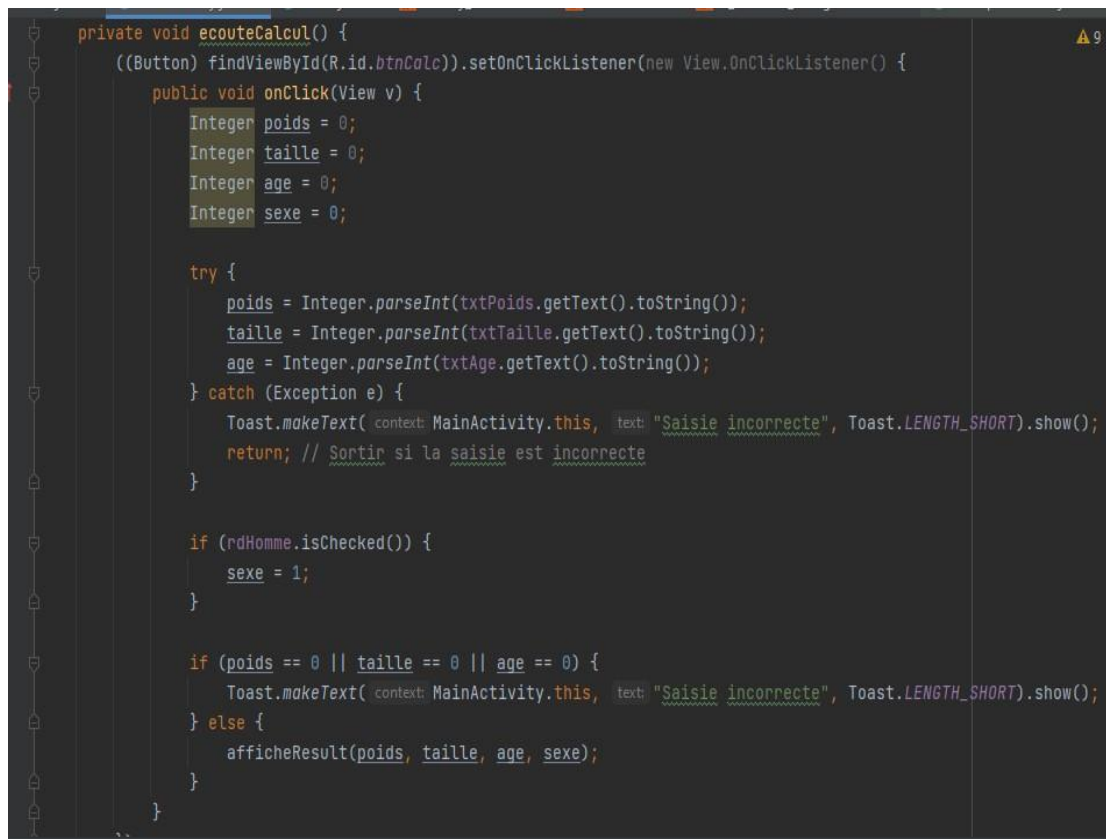
- **Méthode ecouteCalcul() :**

· Cette méthode écoute les clics sur le bouton de calcul (btnCalc). Lorsque l'utilisateur appuie sur le bouton :

Elle récupère les valeurs saisies (poids, taille, âge, sexe) en vérifiant si les champs sont remplis correctement.

Si la saisie est incorrecte, un message d'erreur est affiché via Toast.

Si la saisie est correcte, elle appelle afficheResult() pour afficher les résultats.



```
private void ecouteCalcul() {
    ((Button) findViewById(R.id.btnCalc)).setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Integer poids = 0;
            Integer taille = 0;
            Integer age = 0;
            Integer sexe = 0;

            try {
                poids = Integer.parseInt(txtPoids.getText().toString());
                taille = Integer.parseInt(txtTaille.getText().toString());
                age = Integer.parseInt(txtAge.getText().toString());
            } catch (Exception e) {
                Toast.makeText(context: MainActivity.this, text: "Saisie incorrecte", Toast.LENGTH_SHORT).show();
                return; // Sortir si la saisie est incorrecte
            }

            if (rdHomme.isChecked()) {
                sexe = 1;
            }

            if (poids == 0 || taille == 0 || age == 0) {
                Toast.makeText(context: MainActivity.this, text: "Saisie incorrecte", Toast.LENGTH_SHORT).show();
            } else {
                afficheResult(poids, taille, age, sexe);
            }
        }
    });
}
```

Figure 7:code de la methode ecouteCalcul

- **Méthode afficheResult() :**

Cette méthode crée un profil avec les données saisies en utilisant la méthode creerProfil() du contrôleur.

Elle récupère l'IMG et le message associés au profil.

En fonction du message ("normal", "trop faible", ou "trop élevé"), elle met à jour l'image (ImageView) et la couleur du texte (TextView).

Elle formate et affiche l'IMG avec le message correspondant.

```

1 usage
private void afficheResult(Integer poids, Integer taille, Integer age, Integer sexe) {
    controle.creerProfil(poids, taille, age, sexe);
    float img = controle.getImg();
    String message = controle.getMessage();

    if (message.equals("normal")) {
        imgSmiley.setImageResource(R.drawable.normale);
        lblIMG.setText(Color.GREEN);
    } else {
        lblIMG.setText(Color.RED);
        if (message.equals("trop faible")) {
            imgSmiley.setImageResource(R.drawable.maignre);
        } else {
            imgSmiley.setImageResource(R.drawable.graisse);
        }
    }

    lblIMG.setText(String.format("%.2f : IMG %s", img, message)); // Utilisation correcte du format
}

```

Figure 8:code de la methode afficheResult

III. Les étapes de création d'un javadoc:

Javadoc est un outil utilisé pour générer de la documentation en format HTML à partir des commentaires de documentation présents dans le code Java. Ces commentaires sont placés au-dessus des classes, méthodes, et variables et suivent une syntaxe spéciale. Ils permettent aux développeurs d'expliquer leur code et de fournir des informations utiles

1. Commenter la classe

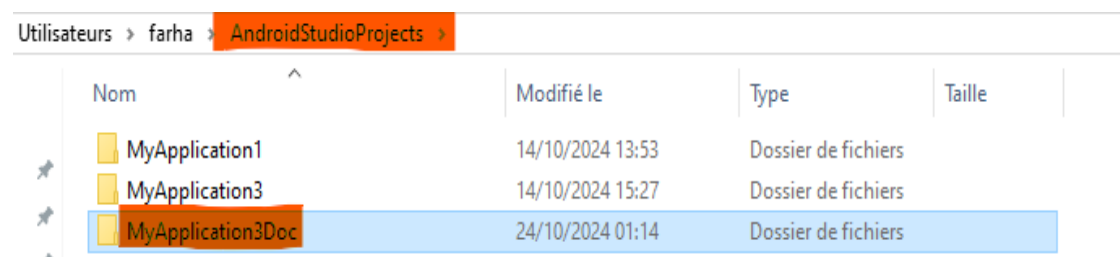
```

24  /**
25   * creation du profil
26   * @param poids
27   * @param taille en cm
28   * @param age
29   * @param sexe 1 pour homme 0 pour femme
30   */
31  no usages
32  public void creerProfil (Integer poids, Integer taille, Integer age, Integer sexe){
33      profil = new Profil(poids, taille, age, sexe);
34  }
35
36  /** recuperation img de profile
37   * @return
38   */
39  no usages
40  public float getImg(){
41      return profil.getImg();
42  }
43
44  /** recuperation message de profile
45   * @return
46   */
47  no usages

```

Figure 9:commenter le code

2. Créer un dossier dans le même dossier où se trouve le dossier de l'application



Utilisateurs > farha > AndroidStudioProjects >				
Nom	Modifié le	Type	Taille	
MyApplication1	14/10/2024 13:53	Dossier de fichiers		
MyApplication3	14/10/2024 15:27	Dossier de fichiers		
MyApplication3Doc	24/10/2024 01:14	Dossier de fichiers		

Figure 10:création du dossier

3. accédez à tools ,choisissez Generate JavaDoc ,allez à outputDirectory et ajoutez le path du dossier MyApplication3Doc

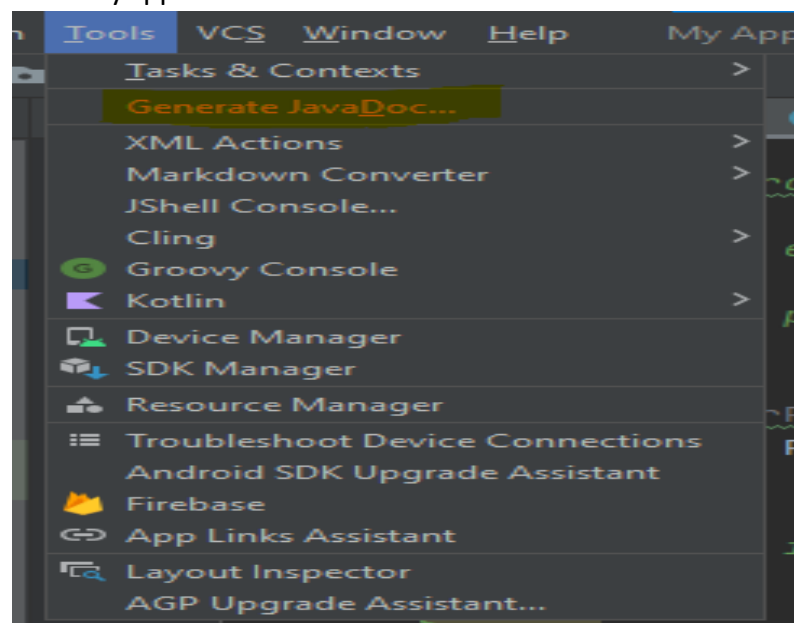


Figure 11:les étapes de création d'un javadoc

IV. Les activités de l'appli :

The screenshot shows a mobile application interface with a purple header bar. The status bar at the top displays the time 1:47 and various icons. The main form has a light pink background. It includes two radio buttons for gender: 'homme' (selected) and 'femme'. To the right, there are input fields for 'poids' (120), 'taille' (150), and 'age' (20). A large purple button labeled 'calculer' is positioned below the inputs. At the bottom, a large yellow cartoon character with a grumpy expression is shown. To its right, the text '52.37 : IMC trop élevé' is displayed in red.

Sexe	Poids (kg)	Taille (cm)	Age (ans)
homme	120	150	20

calculer

52.37 : IMC trop élevé

Figure 12:l'activité obtenue pour un gros homme

1:49

☒ homme poids 40

☐ femme taille 162

age 21

calculer

6.89 : IMG trop faible

Figure 13:l'activité obtenue pour un homme ayant un faible IMG

1.48 

☐ homme poids 48

☒ femme taille 162

age 21

calculer

 21.38 : IMG normal

Figure 14:l'activité obtenue pour une femme ayant un IMG normal