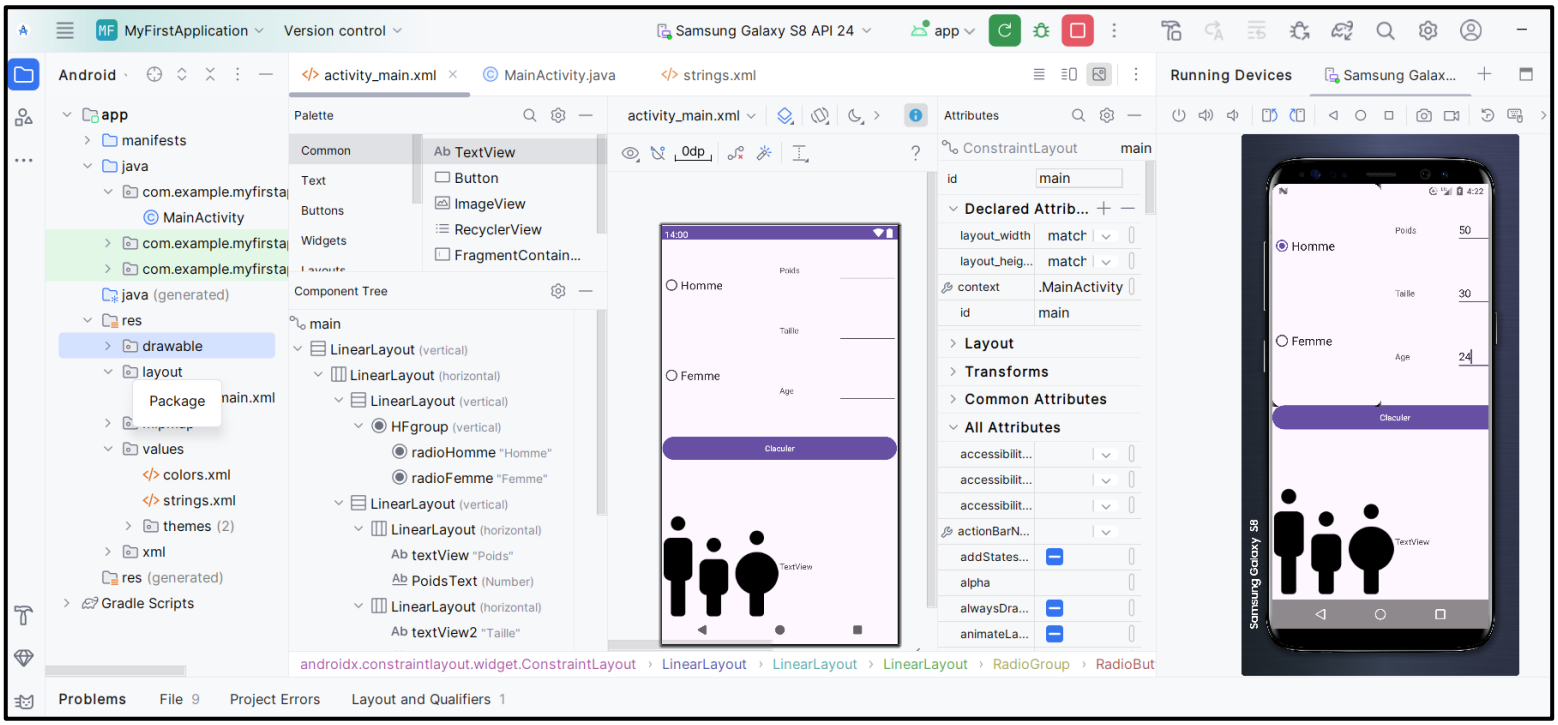
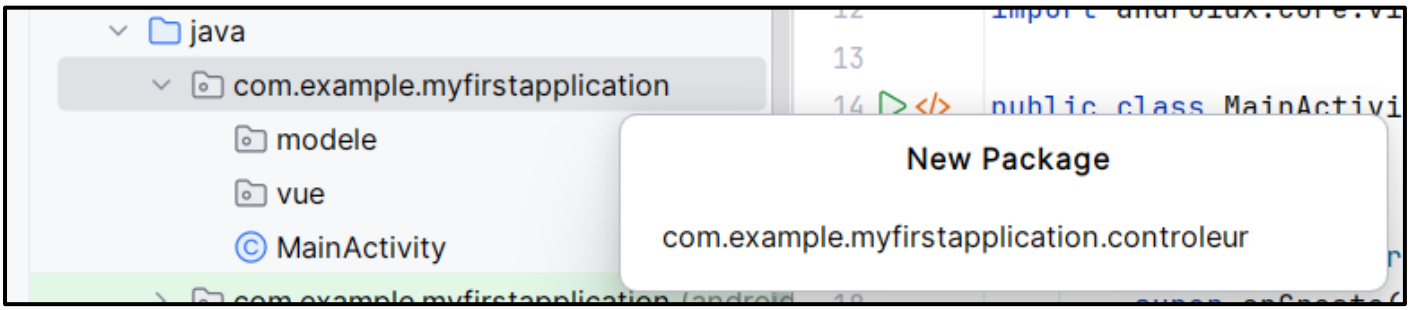
Création du modèle et mise en place des tests unitaires



Partie 2 : Création du Modèle

1. Contexte

Dans cette application Android, le modèle représente l'entité principale pour gérer les données et la logique métier. Le modèle, ici appelé **Profil**, est conçu pour calculer et analyser l'indice de masse grasse (IMG) en fonction des données fournies par l'utilisateur.



2. Structure du Modèle

Classe Profil

La classe **Profil** implémente l'interface Serializable afin de permettre la sérialisation des objets pour les sauvegarder ou les transmettre entre différentes activités Android.

Propriétés

- **Constantes statiques** : Définissent les seuils de l'IMG pour les hommes et les femmes.
    - minFemme, maxFemme, minHomme, maxHomme.
- **Attributs** :
    - dateMesure : Date de la mesure (type Date).
    - poids : Poids en kilogrammes (type Integer).
    - taille : Taille en centimètres (type Integer).
    - age : Âge de l'utilisateur (type Integer).
    - sexe : Sexe de l'utilisateur (0 pour femme, 1 pour homme).
    - img : Indice de masse grasse calculé (type float).
    - message : Résultat interprétatif de l'IMG (type String).

Constructeur

Le constructeur de la classe accepte les valeurs nécessaires pour initialiser un objet **Profil** :

- Appelle deux méthodes privées :
    - **calculIMG()** : Calcule l'IMG en fonction de la formule scientifique.
    - **resultIMG()** : Génère un message interprétatif basé sur les seuils définis.

```java
public class Profil {  6 usages

    // constantes statiques
    private static final Integer minFemme=15;  1 usage
    private static final Integer maxFemme=30;  1 usage
    private static final Integer minHomme=10;  1 usage
    private static final Integer maxHomme=25;  1 usage
    // proprietes
    private Integer poids;  3 usages
    private Integer taille;  3 usages
    private Integer age;  3 usages
    private Integer sexe;  4 usages
    private float img;  7 usages
    private String message;  8 usages

    public Profil(Integer poids, Integer taille, Integer age, Integer sexe) {  2 usages
        this.poids = poids;
        this.taille = taille;
        this.age = age;
        this.sexe = sexe;
        this.calculIMG();
        this.resultIMG();
    }

    public Integer getPoids() { return poids; }

    public Integer getTaille() { return taille; }

    public Integer getAge() { return age; }
```

Méthodes

1. **calculIMG()** :

- Calcule l'IMG selon la formule :

$$IMG = \left(\frac{1.2 \cdot poids}{tailleM^2} \right) + (0.23 \cdot age) - (10.83 \cdot sexe) - 5.4$$

  - tailleM représente la taille en mètres.

2. **resultIMG()** :
   - Interprète l'IMG en comparant avec les seuils prédéfinis.
   - Attribue une des valeurs suivantes à **message** :
     - "trop maigre"

- "trop de graisse"
- "normal"

```
    public void setMessage(String message) { this.message = message; }

    public void setImg(float img) { this.img = img; }

    private void calculIMG(){  1 usage
        float tailleM=(float)taille/100;
        this.img=(float)((1.2*poids/(tailleM*tailleM)) + (0.23*age) - (10.83*sexe) - 5.4);
    }

    private void resultIMG(){  1 usage
        if (sexe==0) {
                        if (img <minFemme) {message="trop maigre";}
                        else if (img>maxFemme) {message="trop de graisse";}
                        else {message="normal";}
                    }
        else        {
                        if (img <minHomme) {message="trop maigre";}
                        else if (img>maxHomme) {message="trop de graisse";}
                        else {message="normal";}

                     }
    }
}
```

Partie 3 : Test Unitaire

1. Contexte

Les tests unitaires servent à valider le comportement des méthodes de la classe **Profil**. Ici, nous utilisons **JUnit** pour vérifier :

- Le calcul de l'IMG.
- L'attribution correcte du message interprétatif.

2. Configuration du Projet

a. Ajouter JUnit à votre projet

Assurez-vous que JUnit est inclus comme dépendance dans votre fichier build.gradle (module/app). Ajoutez la dépendance suivante si elle n'est pas déjà présente :

Ensuite, synchronisez le projet pour télécharger les dépendances.

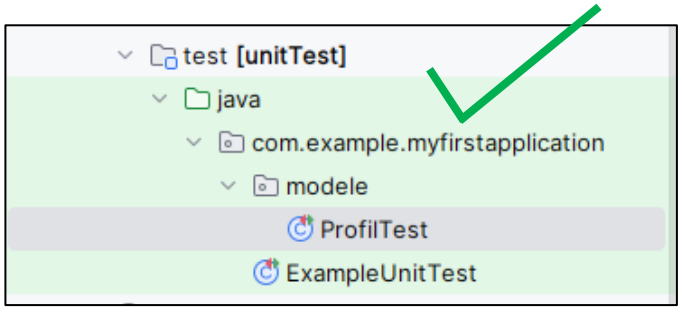
```
[libraries]
junit = { group = "junit", name = "junit", version.ref = "junit" }
ext-junit = { group = "androidx.test.ext", name = "junit", version.ref = "junitVersion" }
```

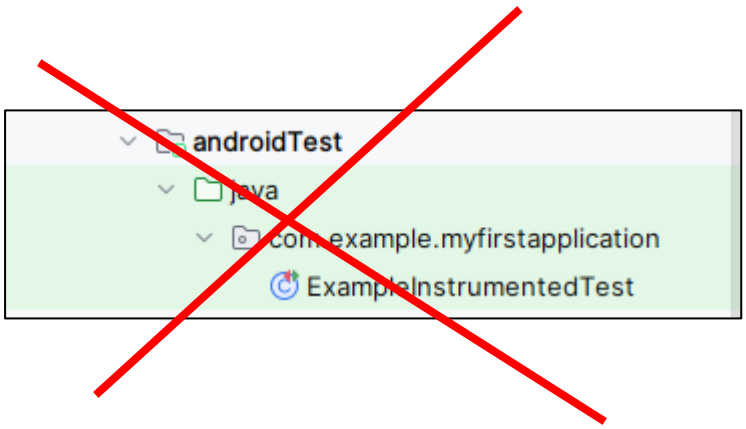
```
dependencies {

    implementation(libs.appcompat)
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
```

3. Créer la Classe de Test

a. Emplacement de la Classe de Test

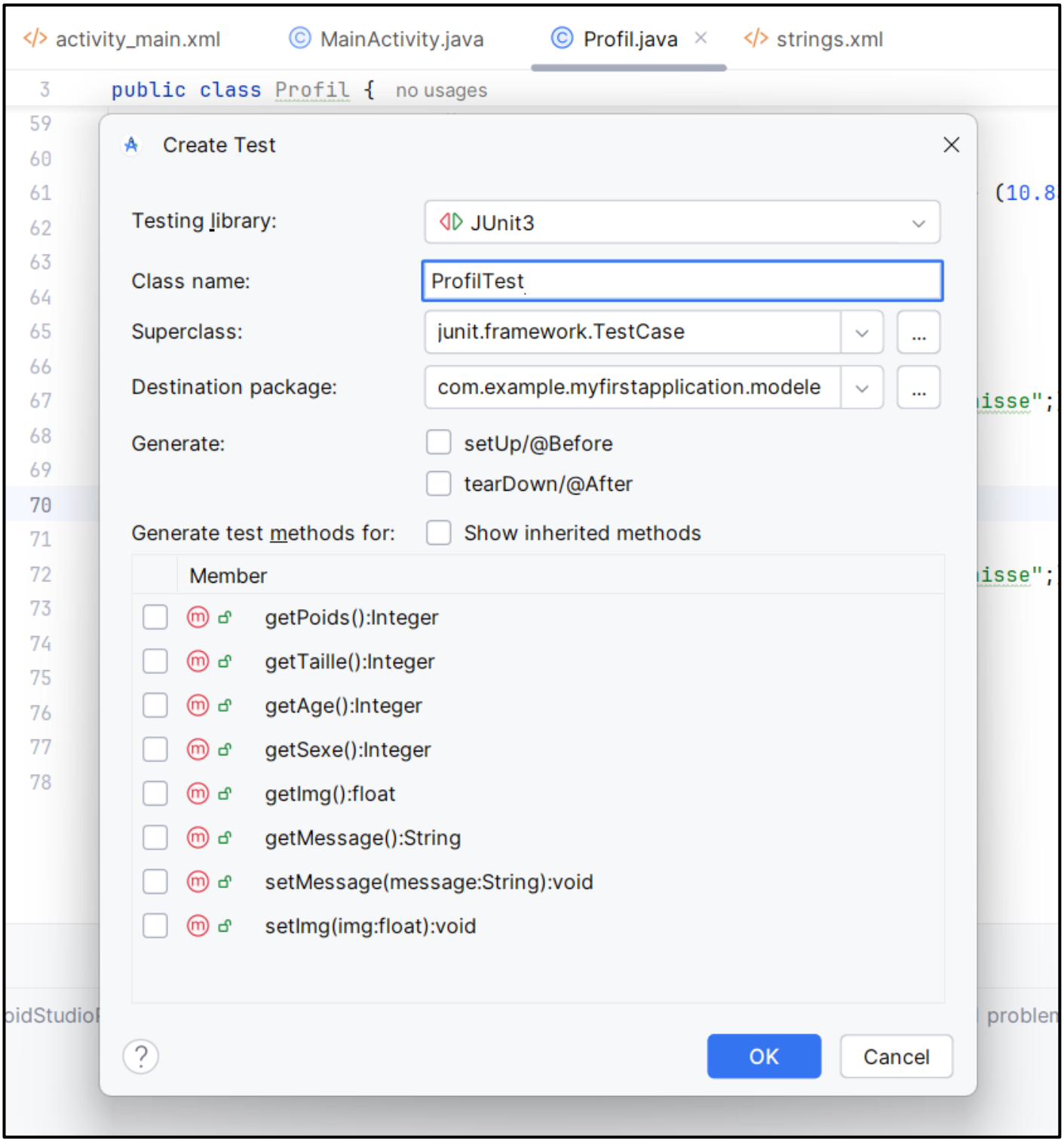
1. Dans votre projet Android, ouvrez le répertoire **src/test/java** (non pas **androidTest** qui est pour les tests instrumentés).





b. Créer la Classe

1. Cliquez avec le bouton droit sur le package, puis sélectionnez **New > Java Class**.
2. Donnez-lui le nom **ProfilTest**.



4. Implémenter les Tests

a. Importer les Bibliothèques

Dans la classe ProfilTest, ajoutez les imports nécessaires :

junit.framework.TestCase; et java.util.Date;

b. Étendre TestCase

La classe ProfilTest doit hériter de la classe junit.framework.TestCase pour utiliser les méthodes de test comme assertEquals.

c. Définir les Variables et Résultats Attendus

Créez une instance de la classe Profil avec des données fixes pour le test. Définissez les résultats attendus pour les assertions :

d. Implémenter les Méthodes de Test

Ajoutez des méthodes pour tester chaque fonctionnalité.

1. **Tester le Calcul de l'IMG :**

```
public void testGetImg() {
  assertEquals(img, profil.getImg(), (float) 0.1);  Vérifie l'égalité avec une marge d'erreur de 0.1}
```

2. **Tester le Message Interprétatif :**

```
public void testGetMessage() {
  assertEquals(message, profil.getMessage());  Vérifie si le message correspond à l'attendu}
```

---

5. Exécuter les Tests

a. Depuis Android Studio

1. Faites un clic droit sur la classe ProfilTest.
2. Sélectionnez **Run ProfilTest**.

b. Interpréter les Résultats

- **Vert** : Les tests sont réussis.
- **Rouge** : Les tests ont échoué. Cliquez sur l'erreur pour voir les détails et déboguer.

activity_main.xml | Profil.java | ProfilTest.java

```
5   public class ProfilTest extends TestCase {
6
7       private Profil profil=new Profil( poids: 67, taille: 165, age: 35, sexe: 0);  2 usages
8       // resultat
9       private float img =(float)32.4;  1 usage
10      private String message ="trop de graisse";  1 usage
11
12      public void testGetImg() {
13          assertEquals(img,profil.getImg(),(float)0.1);
14      }
15
16      public void testGetMessage() { assertEquals(message,profil.getMessage()); }
19  }
```

Tests failed: 1, passed: 1 of 2 tests – 5 ms

```
Expected :32.4
Actual   :32.181683
<Click to see difference>

junit.framework.AssertionFailedError Create breakpoint : expected:<32.4> but was:<32.181683> <5 internal lines>
    at com.example.myfirstapplication.modele.ProfilTest.testGetImg(ProfilTest.java:13) <26 internal lines>
    at jdk.proxy1/jdk.proxy1.$Proxy2.processTestClass(Unknown Source) <7 internal lines>
    at worker.org.gradle.process.internal.worker.GradleWorkerMain.run(GradleWorkerMain.java:69)
    at worker.org.gradle.process.internal.worker.GradleWorkerMain.main(GradleWorkerMain.java:74)
```