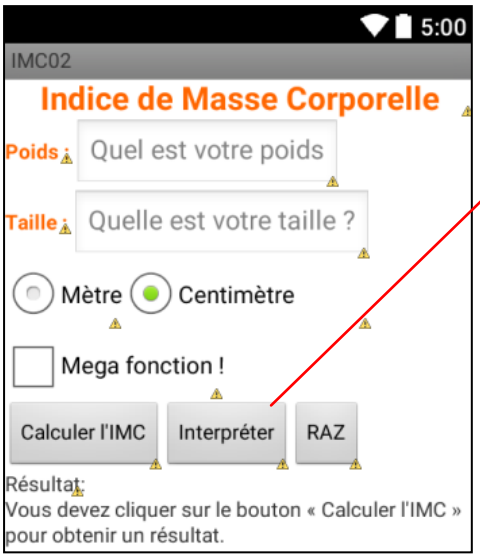
**Notre objectif : améliorer l'application IMC en proposant une interprétation**

## 1. Que voulons-nous obtenir cette fois-ci ?

On va créer cette fois deux activités :

- la première activité s'appuie sur celle déjà développée mais nous l'améliorons,
- une seconde activité s'exécute sur clic de l'utilisateur sur le bouton "Interprétation" et propose d'interpréter l'IMC calculé par l'activité N°1




*Vue = activity_main.xml*
*Contrôleur = MainActivity.java*


*Vue = activity_interpretation.xml*
*Contrôleur = Interpretation.java*

## 2. Retour et approfondissement sur le layout

Il faut toujours éviter d'utiliser le RelativeLayout, préférer un **LinearLayout** vertical ou horizontal.

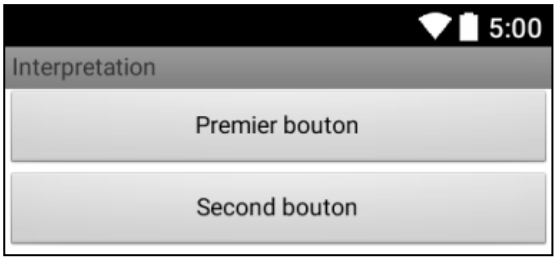
Le LinearLayout se charge de mettre les vues selon une certaine orientation. L'attribut pour préciser cette orientation est ***android:orientation*** qui peut prendre deux valeurs :

- **vertical** pour que les composants soient placés de haut en bas, **les uns au-dessous des autres**,
- **horizontal** pour que les composants soient placés de gauche à droite, les **uns à côté des autres**.

**Exemple 1 : les boutons sont placés les uns en dessous des autres**

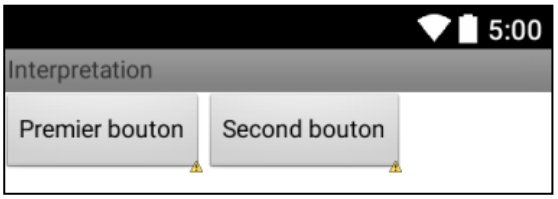
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <Button
        android:id="@+id/premier"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Premier bouton" />
    <Button
        android:id="@+id/second"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Second bouton" />
</LinearLayout>
```

***width*** : largeur
***height*** : hauteur
***fill_parent*** : s'adapte à la dimension du parent
***wrap_content*** : s'adapte au composant lui-même



**Exemple 2 : les boutons sont placés les uns à côté des autres**

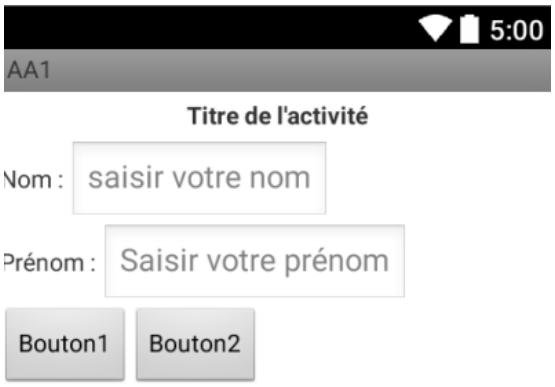
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <Button
        android:id="@+id/premier"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Premier bouton" />
    <Button
        android:id="@+id/second"    .....
        android:text="Second bouton" />
</LinearLayout>
```




**Exemple 3 : des placements horizontaux et verticaux**

On peut sur une même vue avoir des objets placés verticalement et d'autres horizontalement. Dans ce cas on décrit autant que LinearLayout que d'orientations différentes.

```
<<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical" >
 <TextView
    android:id="@+id/textView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
     android:text="Titre de l'activité"
     android:textStyle="bold"
    android:padding="5dip"  />
    <LinearLayout
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:orientation="horizontal">
          <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="left"
                android:text="Nom : " />
          <EditText
                android:id="@+id/nom"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:hint="saisir votre nom" />
    </LinearLayout>
    <LinearLayout
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:orientation="horizontal">
           <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="left"
                android:text="Prénom : " />
           <EditText
           android:id="@+id/prenom"
```



```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Saisir votre prénom" />
    </LinearLayout>
    <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >
            <Button
                android:id="@+id/bouton1"
                 android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Bouton1" />
             <Button
                android:id="@+id/Bouton2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Bouton2" />
      </LinearLayout>
</LinearLayout>
```

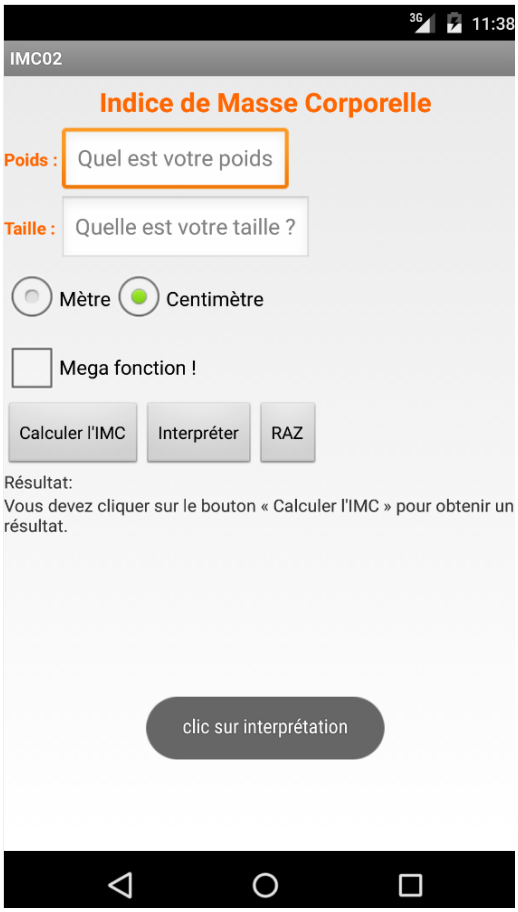
**Travail à faire**

**Créer trois nouvelles activités de test que vous pouvez raccrocher à votre projet en cours . Pour chacune d'elle tester les différentes fenêtres en XML qui sont proposées ci-dessus.**

Après avoir compris le fonctionnement des layout vous devez modifier la fenêtre créée au projet IMC01.

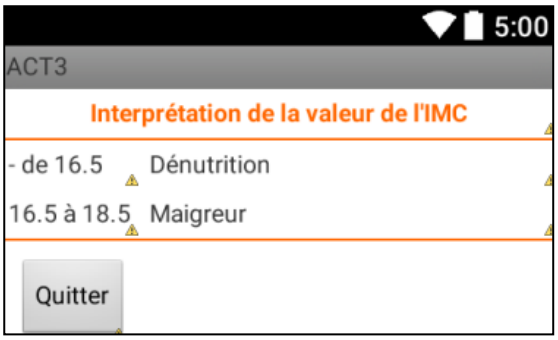
**Travail à faire : projet IMC01**

**<u>1ère étape</u> de ce travail modifier la vue existante *activity_main.xml* pour qu'elle se présente sous un format similaire à celui proposé ci-contre.**



### 3. La présentation sous forme de tableau

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="2">
    <TextView
        android:id="@+id/textView2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Interprétation de la valeur de l'IMC"
        android:padding="5dip"  ..... />
    <View
        android:layout_height="1dip"
        android:background="#FF6600" />
    <TableRow>
        <TextView
            android:text="- de 16.5"
            android:layout_column="1"
            android:gravity="left"
            android:padding="5dip"     />
        <TextView
            android:text="Dénutrition"
            android:layout_column="2"
            android:gravity="left"
            android:padding="5dip"     />
    </TableRow>
    <TableRow>
        <TextView
            android:text="16.5 à 18.5"
            android:layout_column="1"
            android:gravity="left"
            android:padding="5dip"     />
        ....
    </TableRow> ....
    <View
        android:layout_height="1dip"
        android:background="#FF6600"
    />
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="left"
        android:padding="10dip" >
        <Button
            android:id="@+id/quitter"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Quitter" />
    </LinearLayout>
</TableLayout>
```



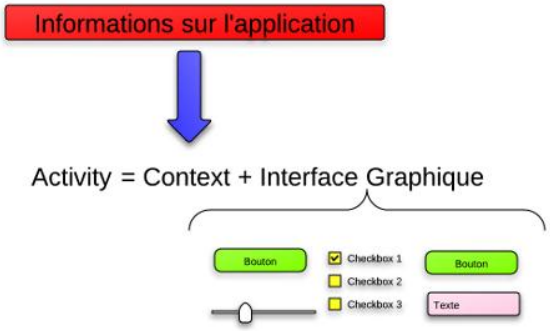

**Travail à faire**

**<u>2ème étape</u>, créer une nouvelle activité nommée "*Interpretation*" et associée à la vue *activity_interpretation.xml, dont l'affichage permet à l'utilisateur de visualiser l'interprétation du calcul de son IMC.* Format similaire à celui proposé en page 1. *S'aider du code présenté ci-dessus.***

## 4. L'enchainement des activités

### 4.1 Qu'est-ce qu'une activité ?

Chaque fenêtre/vue d'une application android est une **activité**.
Une activité remplit tout l'écran du support mobile, l'application ne peut en afficher qu'une à la fois.
Une activité contient des informations sur l'état actuel de l'application : ces informations s'appellent le «**context**».
Ce contexte constitue un lien avec le système Android ainsi qu'avec les autres activités de l'application.
Une activité est constituée du contexte de l'application et d'une seule et unique interface graphique.




### 4.2 Les états d'une activité ?

Si un utilisateur reçoit un appel, il est important qu'il puisse y répondre et que soit stoppée la chanson qui était diffusée par l'application musicale. Pour pouvoir répondre à ce besoin d'arrêt immédiat d'une activité, les développeurs d'android ont eu recours à un système particulier : à tout moment l' application peut laisser place à une autre application, qui a une priorité plus élevée. Si l'application utilise trop de ressources système, alors elle empêchera le système de fonctionner correctement et android l'arrêtera immédiatement.

Rappelons qu'au cours de son existence, une activité passe par plusieurs états : active, en pause, stoppée.

### 4.3 La création de nouvelles activités

Si un projet android qui doit afficher deux interfaces, il faut créer deux activités.

- L'activité principale *MainActivity* et la vue associée *mainActivity.xml* seront exécutées au lancement de l'application,
- Un évènement particulier, clic sur un bouton d'action par exemple, mettra en pause l'activité en cours et exécutera la nouvelle activité,
- L'évènement sera pris en charge dans le code java par un **listener.**

## 5. Le code du listener

Dans le code de l'activité principale *MainActivity.java*, associé à la méthode onCreate, on doit ajouter le code qui permet de lancer la nouvelle activité :

- On déclare le bouton d'action lié à l'évènement
- On associe le clic sur ce bouton à l'appel d'une méthode *setOnClickListener*
- On écrit le code de l'action associé à cet évènement

### 5.1 Testons le click sur le bouton " Interpreter " : comment faire ? 1er essai ... pour comprendre

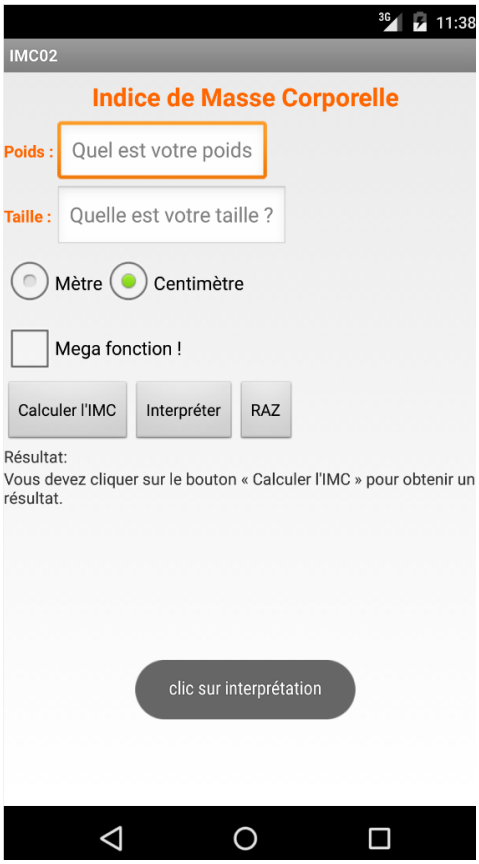
**Travail à faire**

**Il faut modifier la méthode onCreate du *MainActivity.java* par ajout de code :**

- **déclaration du bouton nommé "*interpreter*", associé à *R.id.interprétation* du fichier xml**
- **association de ce bouton à un évènement *setOnClickListener***
- ***ajout d'une nouvelle méthode OnClickListener permettant la gestion de l'évènement***

```
@Override
public void onCreate(Bundle savedInstanceState)   {
......
    interpreter = (Button) findViewById(R.id.interpretation);
    interpreter.setOnClickListener(btnclick);     }
```

```
// méthode qui gère les actions liées à l'évènement onClick
private OnClickListener btnclick = new OnClickListener()
{
    public void onClick(View v)
    {
        switch (v.getId())
        {    case R.id.interpretation:
             Toast.makeText(getApplicationContext(),"Clic détecté sur interprétation",
                    Toast.LENGTH_LONG).show();
            break;
        }
    }
};  // attention au ; obligatoire en java dès
qu'une méthode procède d'une instanciation
```




**On teste pour voir !**

Le clic sur le bouton Interpréter doit afficher un message à la volée (Toast.makeText)
C'est pas cela que l'on veut obtenir mais cela nous a permis de comprendre comment ça marche ...

### 5.2 Le click doit provoquer l'exécution de la 2ème activité

On fait c'est assez simple !
Au lieu d'afficher un message à la volée avec la méthode Toast.makeText, on va utiliser un composant de la plateforme android. Ce composant est un ***Intent***

Le composant ***intent*** d'android doit se comprendre comme un message que la plate-forme android envoie quand elle veut demander le service d'une autre ressource, comme le lancement d'une autre activity ou d' un autre service .

Comment utiliser ce composant ?

A la place du message à afficher, on écrit :

```
case R.id.interpretation:
//Toast.makeText(getApplicationContext(),"clic sur interprétation",
Toast.LENGTH_LONG).show();
    Intent i ; // déclaration d'une variable i, de type Intent
    i = new Intent(getApplicationContext(), Interpretation.class);
       startActivity(i);  //  Interpretation est le nom de la classe associée à
                              l'activité d'affichage de l'interprétation de l'IMC.
```

**On teste pour voir !**

Super l'activité n° 1, l'affichage principal a été mis en attente et remplacée par une nouvelle activité qui cette fois affiche le tableau de l'interprétation de l'IMC.

On est satisfait on sait lancer une activité à partir d'une autre.
Maintenant il faut pouvoir lancer l'activité N° 1 depuis le clic que le bouton d'action de l'activité N° 2.

### 5.3 Programmer le bouton Quitter pour revenir à l'activité principale

Extrait du fichier de la vue activity ***interpretation.xml***

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="left"
    android:padding="10dip" >
    <Button
        android:id="@+id/quitter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Quitter" />
</LinearLayout>
```

*Le code de* ***Interpretation.java***

```
package com.example.imc02;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class Interpretation extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_interpretation);
        // déclaration du bouton d'action quitter et association avec un listener

        Button quitter ;
        Button) findViewById(R.id.quitter);
        quitter.setOnClickListener(btnclick);
    }
quitter = (

    // private obligatoire
    private OnClickListener btnclick = new OnClickListener()
    {
        public void onClick(View v)
        {
            Intent i = new Intent ();
            setResult (Activity.RESULT_OK, i);
            finish();  // arrêt de l'activité ==> on revient à l'activité précédente
        }
    };
}
```

## 6. Exercice

La formule de l'Indice de la Matière Corporelle est : **IMC = Poids / Taille$^2$**

La taille doit être exprimée en mètre.

### 6.1 Le calcul de l'IMC fonctionne très bien sur l'activité 1

- Poids = 70 kg
- Taille = 1.60 m

IMC = 70 / $1.60^2$ = 27.3 : surpoids

Le corps médical a défini des tranches en fonction de la valeur de l'IMC, voir ci-dessous.

### 6.2 Une nouvelle activité affiche sous forme de tableau l'interprétation de l'IMC

| | |
|---|---|
| - de 16.5 | dénutrition |
| 16.5 à 18.5 | maigreur |
| 18.5 à 25 | corpulence normale |
| 25 à 30 | surpoids |
| 30 à 35 | obésité modérée |
| 35 à 40 | obésité sévère |
| + de 40 | obésité massive |

### 6.3 Conseils diététiques

Vous avez réussi à lier deux activités et si on en faisait une troisième ?

Cette fois on veut afficher sur une fenêtre vierge des conseils diététiques et d'hygiène alimentaire, ou pas, .... c'est en fonction de l'indice de masse corporelle calculé.

A partir de la deuxième activité on veut pouvoir en ouvrir une troisième qui afficherait le résultat de l'IMC calculée et quelques conseils alimentaires pour améliorer la condition physique du testeur.

**Travail à faire**

**A partir de la deuxième activité on ouvre donc une troisième fenêtre qui affiche le résultat de l'IMC calculée et quelques conseils alimentaires pour améliorer la condition physique du testeur.**

**A vous d'inventer et de produire un résultat qui soit satisfaisant !**

**Bien entendu, l'activité N° 3 doit se terminer ....**

A vous d'inventer et de produire un résultat qui soit satisfaisant !

Quand tout ça est opérationnel, il est temps d'importer le fichier .apk sur votre propre smartphone !