Notre objectif calculer l'Indice de la Matière Corporelle : IMC = Poids / Taille²

1. Préliminaires : Activité et vue

Les applications android est un assemblage de fenêtres entre lesquelles il est possible de naviguer. Ces différentes fenêtres sont appelées des **activités**.

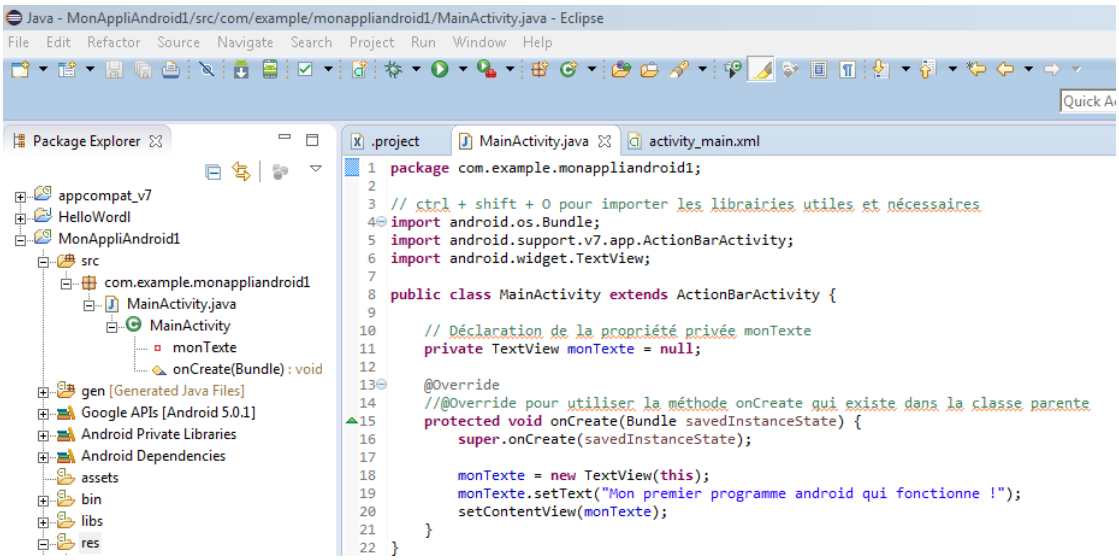
L'application android ne peut traiter qu'une seule activité à la fois qui « remplit » tout l'interface du mobile ==> il y a donc autant d'activités à produire que d'affichages différents sur l'interface du support mobile.

Une activité est liée à une vue spécifique. Chaque activité va donc imposer la description :

- de l'interface-vue affichée sur le support mobile : écrite en langage XML
- le code de gestion associé aux évènements qui peuvent se produire sur cette interface : en Java
- des informations sur l'état actuel de l'application : ces informations sont appelées « context »

Sur le support mobile une application doit pouvoir laisser la place à une autre de niveau de priorité plus élevé. Une activité passera donc par plusieurs états : active, en pause, stoppée.

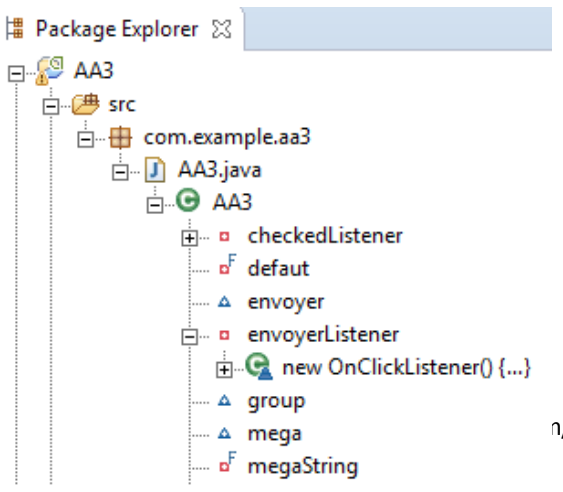
2. L'interface Eclipse



3. Les librairies : mot clé import *(équivalent using en C#)*

```
// ctrl + shift + O pour importer les librairies juste utiles et nécessaires
import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;
import android.widget.TextView;
```

Les ressources src : le dossier src contient les sources de l'application en langage java



Le nom de la **solution**

le **package** d'exécution

L'**activité** principale

une **propriété** privée

une **méthode**

un **objet de l'interface** ou **évènement**

4. Les ressources res

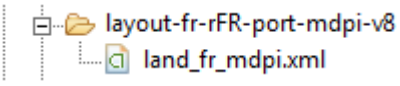
Un mot clé important **layout** : la **vue** produite en langage XML
layout ..... traduction : disposition ... on approfondira plus tard

Création d'une nouvelle ressource

Clic droit sur **res** **New / Other / Android XML File**

Donner un nom significatif au fichier .xml ; par exemple *land_fr_mdpi*

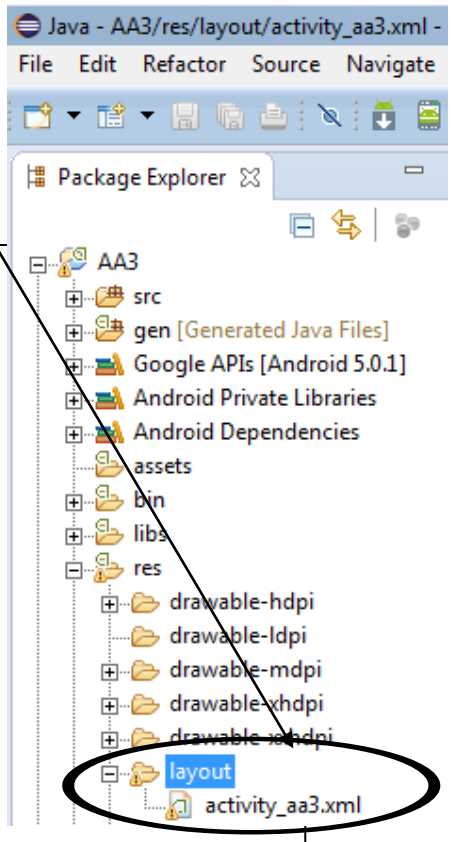
Cliquer sur Next et choisir les bonnes options pour créer un **layout-fr-rFR-port-mdpi-v8**



5. L'interface VUE de l'application IMC est gérée avec un fichier XML

La vue, fichier .xml, est rangé dans le sous-dossier ***layout*** du dossier ***res*** de l'application java.

Le fichier ***activity_aa3.xml*** contient la description des différents objets/widgets qui seront affichés.




```
<?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:textColor="#FF6600"
          android:textStyle="bold"
          android:text="Mon Indice de Masse Corporelle"
           android:textAppearance="?android:attr/textAppearanceLarge" />

      .....

       <EditText
          android:id="@+id/poids"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:hint="Quel est votre poids"
          android:inputType="numberDecimal" />
      .....
      <RadioGroup
          android:id="@+id/group"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:checkedButton="@+id/radio2"
          android:orientation="horizontal" >

          <RadioButton
                android:id="@+id/radio1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Mètre" />

       </RadioGroup>

      <CheckBox
          android:id="@+id/mega"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Mega fonction !" />

      <Button
          android:id="@+id/raz"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="RAZ" />

</LinearLayout>
```

6. Le fichier AndroidManifest.xml

Ce fichier déclare l'ensemble des éléments de l'application et définit le comportement de l' application au système Android. Le fichier manifest permet de décrire l'application. On y retrouve :

- le nom du package de l'application. Il servira d'identifiant unique.
- les composants de l'application, dont les classes qui implémentent les composants et leurs capacités.
- les permissions nécessaires pour le bon fonctionnement de l'application.
- les permissions nécessaires pour que les autres applications utilisent les composants de l'application.
- les informations contenant les versions de l'Android API requis pour exécuter l'application.
- les librairies utilisées par l'application.

7. Le contrôleur est écrit en langage Java

La gestion des évènements s'effectue par l'intermédiaire d'un fichier java. Chacun des objets peut réagir à différents types d'évènements. Ces évènements sont décrits et les actions générées sont décrites dans le fichier java associé à chaque vue. Pour réagir à un évènement, on utilise un objet qui détecte l'évènement. Cet objet s'appelle un **listener**.

Les listeners : gestionnaires d'évènements

Il existe plusieurs façons d'interagir avec une vue : cliquer sur un bouton, saisir un texte, sélectionner du texte, etc.

Pour intercepter un évènement clic sur un bouton, on applique ***View.OnClickListener(View vue)*** sur ce bouton, le paramètre de type View étant la « vue » sur laquelle le clic a été effectué
Pour gérer d'autres évènements, on utilisera d'autres méthodes (liste non exhaustive) :

- View.OnLongClickListener pour les clics qui durent longtemps
- View.OnKeyListener pour gérer l'appui sur une touche. On y associe la méthode boolean onKey(View vue, int code, KeyEvent event).

Pour associer un listener à une vue, on utilise une méthode du type
setOn[Evenement]Listener(On[Evenenement]Listener listener) avec Evenement = l'évènement concerné.

Exemple :

```
import android.view.View.OnTouchListener;                    Liste des bibliothèques à importer ctrl+shift+O
.....
// L'activité détectera les touchers et les clics sur les vues qui se sont inscrites

public class Main extends Activity implements View.OnTouchListener, View.OnClickListener
{
    private Button b = null;                                  /Propriété privée /
    @Override

    public void onCreate(Bundle savedInstanceState)
    {   super.onCreate(savedInstanceState);                   /super. permet d'en référer à une méthode mère
        setContentView(R.layout.main);
        b = (Button) findViewById(R.id.boutton);
        b.setOnTouchListener(this);
        b.setOnClickListener(this);
    }
    @Override
    public boolean onTouch(View v, MotionEvent event)         /* Réagir au toucher */
    {   return true;    }

    @Override
    public void onClick(View v) {          // On récupère l'identifiant de la vue, et en fonction de cet identifiant...
        switch(v.getId()) {
            case R.id.bouton1:             // Si l'identifiant de la vue est celui du premier bouton, actions bouton 1 */
                break;
            case R.id.bouton2:             // Si l'identifiant de la vue est celui du premier bouton, actions bouton 2 */
                break;
            /* etc. */
}   }
```

Notion de classe anonyme

L'inconvénient de la technique précédente est qu'elle allonge les méthodes des listeners s'il y a beaucoup d'éléments à gérer.
C'est pourquoi il est préférable de passer par **une classe anonyme**. Une classe anonyme est une classe qui dérive d'une superclasse ou implémente une interface dont on ne précise pas le nom.
Par exemple pour créer une classe anonyme qui implémente **View.OnClickListener()** on peut faire :

```
import android.app.Activity; .....
public class AnonymousExampleActivity extends Activity
{
    private Button touchAndClick = null;    // On cherchera à détecter les touchers et les clics sur ce bouton
    private Button clickOnly     = null;    // On voudra détecter uniquement les clics sur ce bouton
    @Override
    public void onCreate(Bundle savedInstanceState)
    {   super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        touchAndClick = (Button)findViewById(R.id.touchAndClick);
        clickOnly = (Button)findViewById(R.id.clickOnly);
        touchAndClick.setOnLongClickListener(new View.OnLongClickListener()
        {   @Override
            public boolean onLongClick(View v)
            {   // Réagir à un long clic
                return false;           }
        } );
        touchAndClick.setOnClickListener(new View.OnClickListener()
        {   @Override
            public void onClick(View v)
            {   // Réagir au clic       }
        } );
         clickOnly.setOnClickListener(new View.OnClickListener()
        {   @Override
            public void onClick(View v)
            {   // Réagir au clic       }
        } ) ;
    }
}
```

Synthèse sur les widgets

Il existe un grand nombre de widgets différents. Parmi les plus utilisés, nous avons :

- **TextView** destiné à afficher du texte sur l'écran.
- **EditText** qui hérite des propriétés de TextView et qui permet à l'utilisateur d'écrire du texte.
- **Button** qui hérite des propriétés de TextView et qui permet à l'utilisateur de cliquer sur du texte.
- **CheckBox** qui hérite des propriétés de Button et qui permet à l'utilisateur de cocher une case.
- **RadioButton** qui hérite des propriétés de Button et qui permet à l'utilisateur de choisir parmi plusieurs choix. De plus, **RadioGroup** est un layout spécifique aux RadioButton.

Les **toasts :** boîte de dialogue transitoire

A considérer comme un message d'information, d'avertissement. Ce message est dit transitoire car il ne nécessite aucune intervention de la part de l'utilisateur et ne prend même pas le focus: dans le cas où l'utilisateur serait en train d'effectuer une saisie dans un champ, la saisie continuera dans ce même champ durant tout le temps de l'affichage du message. Enfin, le message disparait de lui même. Exemple :

```
Toast.makeText(AA3.this, "Tu dois saisir ta taille !", Toast.LENGTH_SHORT).show();
```

Instanciation de listener

Dérivé de la méthode précédente : on implémente des classes anonymes en tant qu'objets de façon à pouvoir les utiliser dans plusieurs éléments graphiques différents qui auront la même réaction pour le même évènement.

Remarque : attention au ; qui termine une méthode qui instancie un nouvel évènement

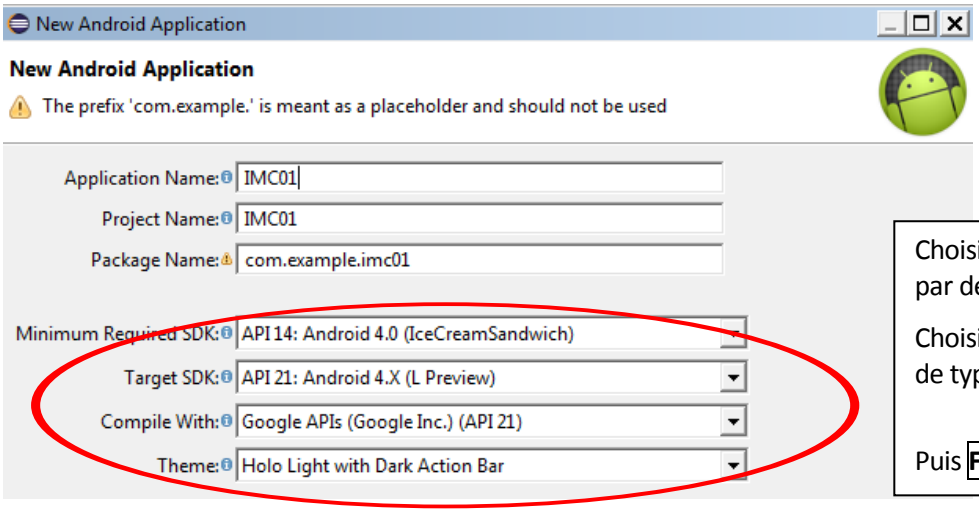
```
import android.app.Activity; .....
public class Main extends Activity {

    private OnClickListener clickListenerBoutons = new View.OnClickListener()
    {
        @Override
        public void onClick(View v) {    /* Réagir au clic pour les boutons 1 et 2*/    }
    };

     private OnTouchListener touchListenerBouton1 = new View.OnTouchListener()
    {
        @Override
        public boolean onTouch(View v, MotionEvent event) {    /* Réagir au toucher pour le bouton 1*/
            return onTouch(v, event);   }
    };

    private OnTouchListener touchListenerBouton3 = new View.OnTouchListener()
    {
        @Override    /* Réagir au toucher pour le bouton 3*/
        public boolean onTouch(View v, MotionEvent event) {
            return super.onTouch(v, event);         }
     };

    Button b1 = null;
    Button b2 = null;
    Button b3 = null;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        b1 = (Button) findViewById(R.id.bouton1);
        b2 = (Button) findViewById(R.id.bouton2);
        b3 = (Button) findViewById(R.id.bouton3);
        b1.setOnTouchListener(touchListenerBouton1);
        b1.setOnClickListener(clickListenerBoutons);
        b2.setOnClickListener(clickListenerBoutons);
        b3.setOnTouchListener(touchListenerBouton3);
  }
}
```

8. Créer un programme android pour calculer l'IMC sur un smartphone (simulé)

8.1 La création du projet

File/New/Android application projet ***IMC01***

Attention à bien choisir les paramètres SDK : Minimum Required SDK d'exécution : **API14: android 4.0**



Choisir tous les paramètres par défaut.

Choisir de créer une Activity de type « *blank activity* »

Puis **Finish**

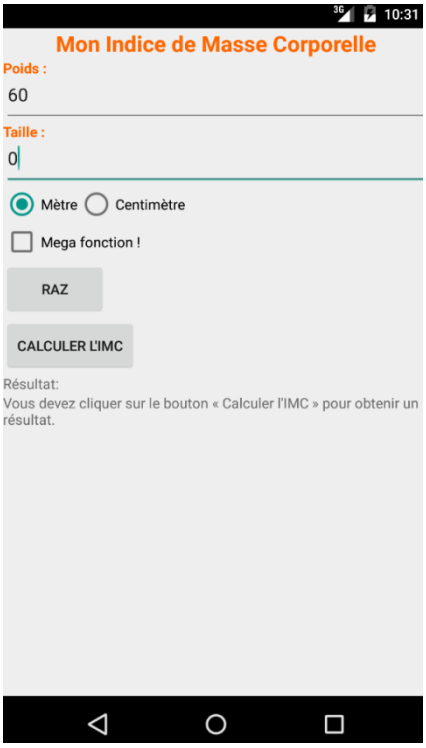
8.2 L'interface utilisateur

1ère étape générer le fichier xml de la vue dans le dossier res/layout.

Remplacer l'existant par le code xml qui permet de créer l'interface de calcul de l'IMC.

Extrait du programme XML qui gère les éléments de la vue :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
... <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
.... <EditText
        android:id="@+id/poids"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="Quel est votre poids"
        android:inputType="numberDecimal" />
    <RadioGroup
        android:id="@+id/group"
...     android:checkedButton="@+id/radio2"
        <RadioButton
        android:id="@+id/radio1"
        android:text="Mètre" />
... </RadioGroup>
    <CheckBox
        android:id="@+id/mega"
        android:text="Mega fonction !" />
    <Button
        android:id="@+id/raz"
        android:text="RAZ" />
</LinearLayout>
```



Le fichier complet activity_aa3.xml est ici : https://www.dropbox.com/home/2014-2015/SIO2/SLAM4/Android

8.3 Le programme java et les listeners de l'application

Essayer de comprendre le fonctionnement des différentes méthodes du code ci-dessous.
Importer ce code dans le programme MainActivity.java

```
package com.example.imc01;
// Bibliothèques utiles au projet
import android.app.Activity;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;   // à compléter : ctrl + shift + o pour déclarer les bibliothèques utiles et nécessaires

public class MainActivity extends Activity          {
    // Déclaration et initialisation des propriétés privées de la classe
    private final String defaut      = "Vous devez cliquer sur le bouton « Calculer l'IMC » pour obtenir un résultat.";
    private final String megaString      = "Vous faites un poids parfait ! ";   // La chaîne de caractères de la mégafonction
    Button        calculerIMC = null;     Button   raz        = null;
    EditText      poids       = null;     EditText taille     = null;
    TextView      result      = null;
    RadioGroup    group       = null;     CheckBox  mega      = null;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //R.id.calcul, R.id.raz, ... font référence au fichier xml : android:id="@+id/calcul"
        calculerIMC  = (Button)findViewById(R.id.calcul);
        raz                  = (Button)findViewById(R.id.raz);
        taille               = (EditText)findViewById(R.id.taille);
        poids                = (EditText)findViewById(R.id.poids);
        mega                 = (CheckBox)findViewById(R.id.mega);
        group                = (RadioGroup)findViewById(R.id.group);
        result               = (TextView)findViewById(R.id.result);
        // On affecte un listener adapté aux évènements qui peuvent se produire
        calculerIMC.setOnClickListener(envoyerListener);
        raz.setOnClickListener(razListener);
        taille.addTextChangedListener(textWatcher);
        poids.addTextChangedListener(textWatcher);
        mega.setOnClickListener(checkedListener);
    }
    //  Evènement onClick sur le bouton "calculerIMC" ==> on calcule l'IMC
    private OnClickListener envoyerListener = new OnClickListener()    {
        @Override
        public void onClick (View v)      {
            if(!mega.isChecked())      {                    // Si la megafonction n'est pas activée
                String t      = taille.getText().toString();     // on récupère la taille
                String p      = poids.getText().toString();      // on récupère le poids
                float tValue = Float.valueOf(t);                 // conversion de type de la variable t : taille
                float pValue  = Float.valueOf(p);                // conversion de type de la variable p : poids
                if(tValue == 0)                                  // Puis on vérifie que la taille est cohérente
                   Toast.makeText(MainActivity.this, "Saisis obligatoirement  ta taille !", Toast.LENGTH_SHORT).show();
                else          {
                   // Si l'utilisateur indique que la taille était en cm on vérifie que la Checkbox sélectionnée est la 2ème
                  if(group.getCheckedRadioButtonId() == R.id.radio2)          tValue = tValue / 100;  // on convertit en mètre
                  tValue      = (float)Math.pow(tValue, 2);      //Retourne tValue au carré,  à la puissance 2
                  float  imc = pValue / tValue;                  // Calcul de l'IMC
                  result.setText("Ton IMC est " + String.valueOf(imc));  // conversion de type pour affichage du résultat
                }
            }
            else    result.setText(megaString);
        }
    };
```

```
    / Listener du bouton de remise à zéro
    private OnClickListener razListener = new OnClickListener()        {
        @Override
        public void onClick(View v)                    {
            poids.getText().clear();
            taille.getText().clear();
            result.setText(defaut);
        }
    };
    // TextWatcher est un listener qui permet de surveiller les modifications dans la saisie d'un EditText
    private TextWatcher textWatcher = new TextWatcher()                {
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count)
        {              result.setText(defaut);          }
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count,  int after)
        {      }
        @Override
        public void afterTextChanged(Editable s)
        {      }
    };
    // Listener du bouton de la megafonction.
    private OnClickListener checkedListener = new OnClickListener()                {
        @Override
        public void onClick(View v)                            {
            // On remet le texte par défaut si c'était le texte de la megafonction qui était écrit
            if(!((CheckBox)v).isChecked() && result.getText().equals(megaString))
                result.setText(defaut);
        }
    };
}
```

9. Exercice

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

La taille doit être exprimée en mètre.

Exemple de calcul

- 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.

Interprétation de l'IMC d'un adulte

- 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

Travail à faire

En vous aidant des ressources à disposition dans ce cours, créez l'application android IMC01 qui calcule l'IMC d'un utilisateur saisissant son poids et sa taille.