

# Programmation mobile

## Cours 1 : Généralités

Julien Grange <julien.grange@lacl.fr>

Mardi 10 septembre 2024

- CM : 6 séances de 1H30
- TP : 7 séances de 3H
- Enseignants : Ahmed Mimouni (TP), Julien Grange (CM et TP)
- Évaluation : 2 QCM + 1 atelier + 1 TP noté

$$\frac{\text{QCM} + \text{atelier} + \text{TP} - \min(\text{QCM}, \text{atelier}, \text{TP})}{2}$$

**Ajoutez une photo sur Eprel !**

**Numérique** : environ 4% des émissions de gaz à effet de serre. C'est

- autant que l'aviation (mais beaucoup plus démocratisé)
- ni négligeable, ni “dans les nuages”

**Numérique** : environ 4% des émissions de gaz à effet de serre. C'est

- autant que l'aviation (mais beaucoup plus démocratisé)
- ni négligeable, ni "dans les nuages"

**Conception d'un smartphone** : environ 70kg d'équivalent  $\text{CO}_2$ , ou

- l'équivalent de 3kg de bœuf
- l'équivalent de 350km en voiture thermique

**Usage** : pour 1h de vidéo par jour via 4G, 30kg d'équivalent  $\text{CO}_2$  par an

**Numérique** : environ 4% des émissions de gaz à effet de serre. C'est

- autant que l'aviation (mais beaucoup plus démocratisé)
- ni négligeable, ni "dans les nuages"

**Conception d'un smartphone** : environ 70kg d'équivalent  $\text{CO}_2$ , ou

- l'équivalent de 3kg de bœuf
- l'équivalent de 350km en voiture thermique

**Usage** : pour 1h de vidéo par jour via 4G, 30kg d'équivalent  $\text{CO}_2$  par an

Il importe donc

- de faire durer au maximum son matériel
- de limiter sa consommation de vidéos (a minima, de favoriser le wifi)

- IDE : Android Studio  
<https://developer.android.com/studio>
- Guide et documentation  
<https://developer.android.com/docs>
- Développement en **Java** ou Kotlin
- Exécution possible
  - sur vos smartphones et tablettes Android
  - sur le simulateur intégré

The screenshot displays the Android Studio interface. The top menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The main editor shows the MainActivity.java file with the following code:

```
17 // onCreate est appelée :
18 // - quand l'activité est créée
19 // - à chaque fois qu'elle est recrée, que ce soit
20 // - parce que l'OS l'a tuée pour libérer des ressources
21 // - parce que l'écran a été inversé (besoin de recalculer l'UI)
22
23
24 if
25
26 @Override
27 protected void onCreate(Bundle savedInstanceState) {
28     super.onCreate(savedInstanceState);
29     //affiche le layout tel que défini dans activity_main.xml
30     setContentView(R.layout.activity_main);
31
32     //et on ne vient pas juste d'être créé, on récupère la valeur du compteur sauvegardée
33     if(savedInstanceState!=null){
34         compteur=savedInstanceState.getInt(COMPTEUR, defaultValue 0);
35     }
36
37     //on récupère les views définies dans activity_main.xml
38     myTextView = findViewById(R.id.textView);
39     Button incrButton = findViewById(R.id.button);
40     Button displayButton = findViewById(R.id.button2);
41
42     //on affiche le compteur
43     myTextView.setText(Integer.toString(compteur));
44
45     //on définit l'action à effectuer (callback) quand l'utilisateur appuie sur incrButton
46     //on fait ici appel à une implémentation anonyme de l'interface View.OnClickListener, pour gagner du temps
47     incrButton.setOnClickListener(new View.OnClickListener(){
48         @Override
49         public void onClick(View view) {myTextView.setText(Integer.toString(++compteur)); }
50     });
51
52     displayButton.setOnClickListener(new View.OnClickListener() {
53         @Override
54         public void onClick(View view) {
55             //on crée un Intent dans le but de passer le main à l'activité HelloWorld
56             Intent intent = new Intent(getApplicationContext(), HelloWorld.class);
57             //on ajoute la valeur du compteur à l'intent, avec le clef COMPTEUR
58             intent.putExtra(COMPTEUR,compteur);
59             startActivity(intent);
60         }
61     });
62 }
```

On the right side, a virtual device emulator is shown with the following UI:

- Time: 17:40
- App Name: Cours1
- Text: 4
- Buttons: INCREMENT, DISPLAY

The bottom status bar shows: Launch succeeded (4 minutes ago), Event Log, Layout Inspector, 17:1 L F UTF-8 4 spaces.

- Les classes et interfaces seront encadrées : `Classe`
- Les méthodes (et champs) statiques d'une classe sont dénotés `Class.foo()`
- Les méthodes (et champs) non-statiques (i.e. propres aux objets) d'une classe sont dénotés `Class::foo()`
- Si la classe en question est claire, on se contentera de `foo()`



# Principes généraux

- Chaque application est composée d'une ou plusieurs `Activity`
- Chaque écran correspond (grosso modo) à une activité

- Chaque application est composée d'une ou plusieurs `Activity`
- Chaque écran correspond (grosso modo) à une activité
- Le layout d'une activité est constitué d'un arbre de `View`
  - Les feuilles sont des `View`, e.g.
    - `TextView`
    - `Button`
  - Les nœuds internes sont des `ViewGroup`, e.g.
    - `LinearLayout`
    - `ScrollView`

- Chaque application est composée d'une ou plusieurs `Activity`
- Chaque écran correspond (grosso modo) à une activité
- Le layout d'une activité est constitué d'un arbre de `View`
  - Les feuilles sont des `View`, e.g.
    - `TextView`
    - `Button`
  - Les nœuds internes sont des `ViewGroup`, e.g.
    - `LinearLayout`
    - `ScrollView`
- Une activité peut lancer une autre activité via un `Intent`
  - soit en la nommant (interne à une application)
  - soit à la crée (e.g. pour prendre une photo)

- Chaque application est composée d'une ou plusieurs `Activity`
- Chaque écran correspond (grosso modo) à une activité
- Le layout d'une activité est constitué d'un arbre de `View`
  - Les feuilles sont des `View`, e.g.
    - `TextView`
    - `Button`
  - Les nœuds internes sont des `ViewGroup`, e.g.
    - `LinearLayout`
    - `ScrollView`
- Une activité peut lancer une autre activité via un `Intent`
  - soit en la nommant (interne à une application)
  - soit à la crée (e.g. pour prendre une photo)
- Programmation événementielle, à base de callbacks

- Dès qu'une activité n'est plus visible, elle peut être tuée si l'OS a besoin de ressources
- Elle est alors relancée quand l'utilisateur la repasse en premier plan
- Idem à chaque rotation d'écran (nouveau calcul de l'UI)

- Dès qu'une activité n'est plus visible, elle peut être tuée si l'OS a besoin de ressources
- Elle est alors relancée quand l'utilisateur la repasse en premier plan
- Idem à chaque rotation d'écran (nouveau calcul de l'UI)

Il faut sauvegarder les données courantes dès qu'on quitte le premier plan.

Deux `Activity` :

- `MainActivity` : classe principale, accessible depuis le launcher
  - premier bouton → incrémente un compteur
  - deuxième bouton → lance `HelloWorld` via un `Intent`

Deux `Activity` :

- `MainActivity` : classe principale, accessible depuis le launcher
  - premier bouton → incrémente un compteur
  - deuxième bouton → lance `HelloWorld` via un `Intent`
- `HelloWorld`
  - affiche autant de fois "Hello World !" que la valeur du compteur



# MainActivity layout

The screenshot shows the Android Studio interface for editing the layout of MainActivity. The main workspace displays a visual representation of the ConstraintLayout with three widgets: a 'button' at the top left, a 'button2' at the top right, and a 'display' widget at the bottom center. The 'button' widget is selected, and the right-hand 'Attributes' panel shows its properties, including 'wrap\_content' for width and height, and various constraints. A 'Project update recommended' notification is visible at the bottom right.

**Declared Attributes**

Attribute	Value
layout_width	wrap_content
layout_height	wrap_content
layout_constraintBottom_toBottomOf	@+id/button2
layout_constraintHorizontal_toLeftOf	parent
layout_constraintHorizontal_toRightOf	parent
layout_constraintStart_toStartOf	@+id/textView
layout_constraintEnd_toEndOf	parent
layout_constraintTop_toTopOf	parent
id	button
text	@string/incr

**Layout**

Constraint Widget

Constraints (5)

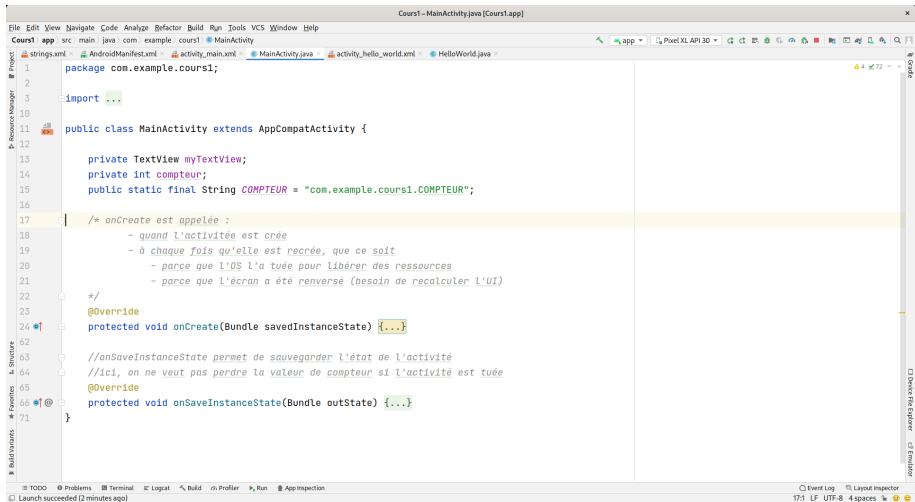
Attribute	Value
layout_width	wrap_content
layout_height	wrap_content
visibility	visible

**Transforms**

- Common Attributes
- style
- stateListDrawable

**Project update recommended**  
Android Gradle Plugin can be upgraded.

# MainActivity code



```
1 package com.example.cours1;
2
3 import ...
4
5
6
7
8
9
10
11 public class MainActivity extends AppCompatActivity {
12
13     private TextView myTextView;
14     private int compteur;
15     public static final String COMPTEUR = "com.example.cours1.COMPTEUR";
16
17     /* onCreate est appelée :
18      - quand l'activité est créée
19      - à chaque fois qu'elle est recrée, que ce soit
20      - parce que l'OS l'a tuée pour libérer des ressources
21      - parce que l'écran a été renversé (besoin de recalculer l'UI)
22     */
23
24     @Override
25     protected void onCreate(Bundle savedInstanceState) {...}
26
27
28
29
30
31     //onSaveInstanceState permet de sauvegarder l'état de l'activité
32     //ici, on ne veut pas perdre la valeur de compteur si l'activité est tuée
33
34     @Override
35     protected void onSaveInstanceState(Bundle outState) {...}
36
37
38
39
40
41 }
```

# MainActivity::onCreate()

The image shows an IDE window titled "Cours1 - MainActivity.java [Cours1.app]". The code editor displays the `onCreate` method of `MainActivity`. The code includes comments in French and implements the following logic:

- Checks if the activity is created or recreated.
- Retrieves the counter value from the saved instance state (defaulting to 0).
- Retrieves the `TextView` and `Button` views from `activity_main.xml`.
- Displays the counter value in the `TextView`.
- Defines a click listener for the `incrButton` that increments the counter and updates the `TextView`.
- Defines a click listener for the `displayButton` that creates an `Intent` to start `HelloWorld` activity, passing the current counter value as an extra.

On the right, a mobile emulator is shown with the following UI:

- Top bar: "Cours1"
- Center: A `TextView` displaying the number "4".
- Buttons: "INCREMENT" and "DISPLAY".

The IDE interface includes a menu bar (File, Edit, View, etc.), a toolbar, and a status bar at the bottom showing "Launch succeeded (4 minutes ago)" and system information like "17:1 L F UTF-8 4 spaces".

# Activity::onCreate()

```
public class MainActivity extends AppCompatActivity {  
  
    private TextView myTextView;  
  
    protected void onCreate(Bundle savedInstanceState) {  
  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        myTextView = findViewById(R.id.textView);  
        ...  
    }  
}
```

## View::setOnClickListener()

```
incrButton.setOnClickListener(new View.OnClickListener(){  
  
    @Override  
    public void onClick(View view){  
        myTextView.setText(Integer.toString(++compteur));  
    }  
  
});
```

# HelloWorld code

The screenshot shows an IDE window titled "Cours1 - HelloWorld.java [Cours1.app]". The code in the editor is as follows:

```
1 package com.example.cours1;
2
3 import ...
4
9 public class HelloWorld extends AppCompatActivity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_hello_world);
15
16         TextView textView = findViewById(R.id.textView2);
17
18         //on récupère l'intent qui a servi à nous démarrer, et on récupère la valeur du compteur
19         Intent intent = getIntent();
20         int compteur = intent.getIntExtra(MainActivity.COMPTEUR, defaultValue: 1);
21
22         StringBuilder helloworld= new StringBuilder();
23         for(int i = 0; i<compteur; i++){
24             helloworld.append("Hello World!\n");
25         }
26         textView.setText(helloworld.toString());
27
28     }
29 }
```

On the right, a mobile emulator displays the app's output. The status bar shows the time 13:36 and battery level. The app's title bar is purple and says "Cours1". The main content area displays "Hello World!" repeated four times on separate lines. A bottom navigation bar is visible at the bottom of the emulator.

At the bottom of the IDE, a green notification box says "Launch succeeded". Below that, a status bar shows "Launch succeeded (moments ago)".

At the bottom right of the IDE, there is a notification: "Project update recommended. Android Gradle Plugin can be upgraded." Below this, the status bar shows "Event Log", "Layout Inspector", "13:44", "LF", "UTF-8", "4 spaces", and a zoom icon.

# Activity::startActivity()

Dans `MainActivity` :

```
public static final String COMPTEUR = "cours1.COMPTEUR";

//dans le setOnClickListener() du second bouton
public void onClick(View view) {

    Intent intent = new Intent(getApplicationContext(), HelloWorld.
        class);
    Intent.putExtra(COMPTEUR, compteur);
    startActivity(intent);
}
```

Dans `HelloWorld` :

```
protected void onCreate(Bundle savedInstanceState) {

    Intent intent = getIntent();
    int compteur = intent.getIntExtra(MainActivity.COMPTEUR, 1);
    ...
}
```

## Activity::onSaveInstanceState()

Rappel : une `Activity` peut être tuée (malgré elle) en cas de

- passage au second plan (typiquement, en cas d'appel)
- rotation d'écran

On peut sauvegarder des clefs/valeurs dans un `Bundle` dans la méthode `onSaveInstanceState()`, qui permettront de retrouver les valeurs courantes lors du prochain appel à `onCreate()`.



# MainActivity::onSaveInstanceState()

```
1 package com.example.cours1;
2
3 import ...
4
5
6
7
8
9
10
11 public class MainActivity extends AppCompatActivity {
12
13     private TextView myTextView;
14     private int compteur;
15     public static final String COMPTEUR = "com.example.cours1.COMPTEUR";
16
17     /* onCreate est appelée :
18      - quand l'activité est créée
19      - à chaque fois qu'elle est recrée, que ce soit
20      - parce que l'OS l'a tuée pour libérer des ressources
21      - parce que l'écran a été renversé (besoin de recalculer l'UI)
22     */
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {...}
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57 //onSaveInstanceState permet de sauvegarder l'état de l'activité
58 //ici, on ne veut pas perdre la valeur de compteur si l'activité est tuée
59 @Override
60 protected void onSaveInstanceState(Bundle outState) {
61     //on se souvient de la valeur du compteur pour ne pas repartir de 0
62     outState.putInt(COMPTEUR, compteur);
63     super.onSaveInstanceState(outState);
64 }
65
66
67
68
69
70
71 }
```

The screenshot shows an IDE window titled "Cours1 - MainActivity.java [Cours1App]". The code editor displays the MainActivity.java file with the following content:

```
package com.example.cours1;

import ...

public class MainActivity extends AppCompatActivity {

    private TextView myTextView;
    private int compteur;
    public static final String COMPTEUR = "com.example.cours1.COMPTEUR";

    /* onCreate est appelée :
     - quand l'activité est créée
     - à chaque fois qu'elle est recrée, que ce soit
     - parce que l'OS l'a tuée pour libérer des ressources
     - parce que l'écran a été renversé (besoin de recalculer l'UI)
    */
    @Override
    protected void onCreate(Bundle savedInstanceState) {...}

    //onSaveInstanceState permet de sauvegarder l'état de l'activité
    //ici, on ne veut pas perdre la valeur de compteur si l'activité est tuée
    @Override
    protected void onSaveInstanceState(Bundle outState) {
        //on se souvient de la valeur du compteur pour ne pas repartir de 0
        outState.putInt(COMPTEUR, compteur);
        super.onSaveInstanceState(outState);
    }
}
```

On the right, a mobile emulator is shown. The app's title bar is purple and says "Cours1". The main screen has a white background with a purple "INCREMENT" button and a purple "DISPLAY" button. The number "4" is displayed in the center of the screen. The emulator's status bar shows the time "12:41" and various icons. The bottom navigation bar is black with three white icons.

# MainActivity::onSaveInstanceState()

```
private int compteur;
public static final String COMPTEUR = "cours1.COMPTEUR";

protected void onSaveInstanceState(Bundle outState) {

    outState.putInt(COMPTEUR, compteur);
    super.onSaveInstanceState(outState);

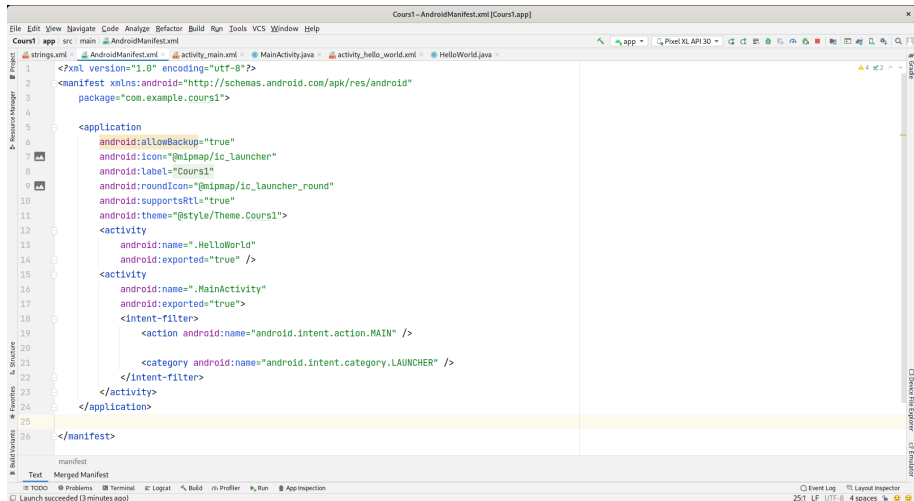
}

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    if(savedInstanceState != null){
        compteur = savedInstanceState.getInt(COMPTEUR, 0);
    }
    ...
}
```

# Android Manifest



The screenshot shows an IDE window titled "Cours1 - AndroidManifest.xml [Cours1.app]". The main editor displays the following XML code:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.cours1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/Theme.Cours1">
        <activity
            android:name=".HelloWorld"
            android:exported="true" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help), a toolbar with icons for app, Pixel XL API 30, and other actions, and a sidebar with Project, Resource Manager, Structure, Favorites, and Build Variants. The bottom status bar shows "Launch succeeded (3 minutes ago)", "25:1 LF UTF-8 4 spaces", and icons for Event Log, Layout Inspector, and other tools.