

# Programmation mobile

## Cours 1 : Généralités

Julien Grange <julien.grange@lacl.fr>

Mardi 5 septembre 2023

- CM : 6 séances de 1H30
- TP : 7 séances de 3H
- Enseignants : Julien Cervelle (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

# AndroidStudio

The screenshot displays the Android Studio environment. The top menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The main editor window shows the code for MainActivity.java, which implements the onCreate method. The code includes comments in French and logic for displaying a counter and handling button clicks.

```
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, a virtual smartphone emulator is shown. The screen displays the text "Cours1" at the top, a counter "4" in the center, and two buttons: "INCREMENT" and "DISPLAY". The status bar at the top of the emulator shows the time 17:40 and battery level. The bottom of the emulator shows the standard Android navigation bar.

- 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 criée (e.g. pour prendre une photo)

# Principes généraux

- 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.

# Première application : Hello World !

Deux `Activity` :

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

# Première application : Hello World !

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

Cours1 - activity\_main.xml [Cours1.app]

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Cours1 app | src | main | java | com | example | cours1 | MainActivity

strings.xml | AndroidManifest.xml | activity\_main.xml | MainActivity.java | activity\_hello\_world.xml | HelloWorld.java

Palette

- Common
  - TextView
- Text
  - Button
- Buttons
  - Image
- Widgets
  - Recycl
- Layouts
  - Fragm
- Containers
  - Scroll
- Helpers
  - Switch
- Google
- Legacy

Component Tree

- ConstraintLayout
  - textView "textView"
  - button2 "@string/hello\_world"
  - button "@string/increment"

Attributes

button

Declared Attributes

id	button
layout_width	wrap_content
layout_height	wrap_content
layout_constraintBottom_toBottomOf	@+id/button2
layout_constraintHorizontal_toHorizontal	0.5
layout_constraintStart_toStartOf	@+id/textView
layout_constraintEnd_toEndOf	parent
layout_constraintTop_toTopOf	parent
id	button
text	@string/incr

Layout

Constraint Widget

Constraints (5)

layout_width	wrap_content
layout_height	wrap_content
visibility	visible

Transforms

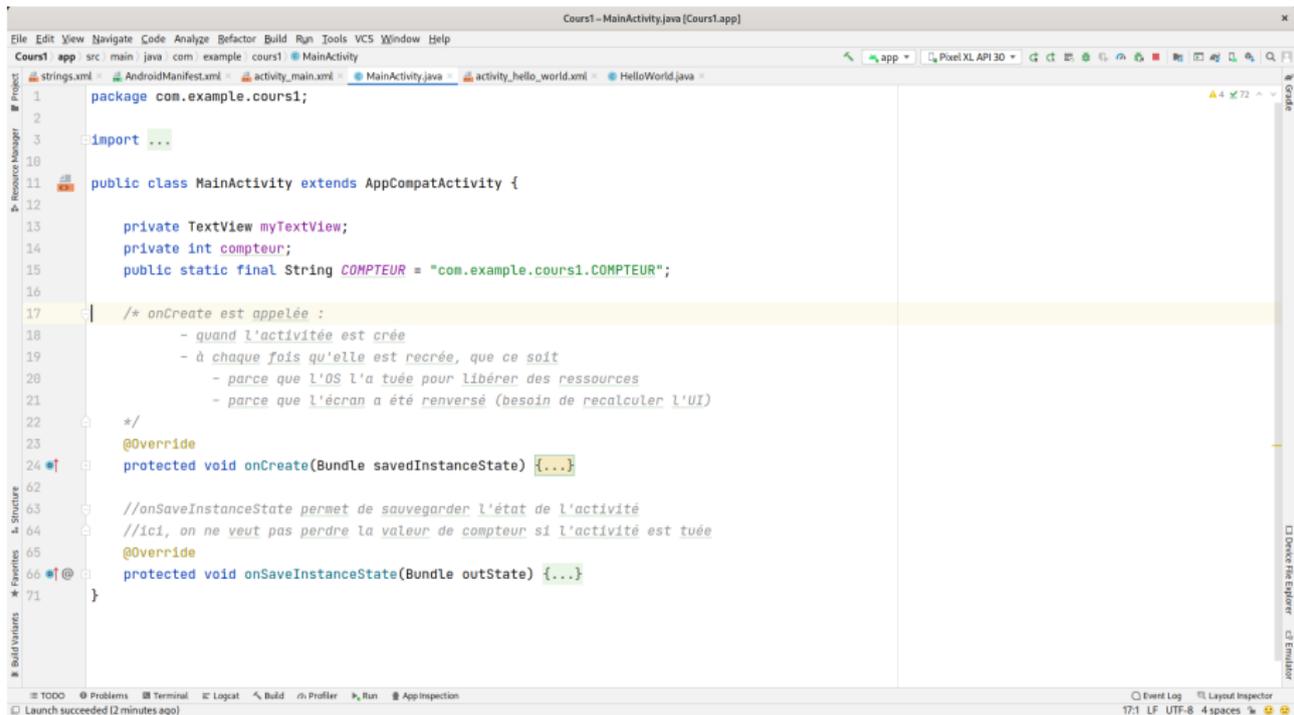
- Common Attributes

Project update recommended  
Android Gradle Plugin can be upgraded.

Event Log | Layout Inspector

17:1 LF UTF-8 4 spaces

# 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
32
33
34     //onSaveInstanceState permet de sauvegarder l'état de l'activité
35     //ici, on ne veut pas perdre la valeur de compteur si l'activité est tuée
36
37     @Override
38     protected void onSaveInstanceState(Bundle outState) {...}
39
40 }
41
```

Launch succeeded [2 minutes ago] | 17:1 LF UTF-8 4 spaces

# MainActivity::onCreate()

The screenshot displays the Android Studio IDE with the `MainActivity.java` file open. The code implements the `onCreate()` method, which initializes the UI and sets up click listeners for two buttons: `INCREMENT` and `DISPLAY`. The `INCREMENT` button is currently showing the value `4`.

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

The right side of the IDE shows a preview of the app's UI on a smartphone. The screen displays the text "Cours1" at the top, a counter showing "4", and two buttons: "INCREMENT" and "DISPLAY". The status bar at the top of the phone shows the time 17:40 and battery level.

# 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 the Android Studio IDE with the following code in the main editor:

```
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 }
```

The virtual device on the right displays the app's output, which consists of the text "Hello World!" repeated four times on separate lines. A notification at the bottom of the IDE states: "Project update recommended. Android Gradle Plugin can be upgraded."

# 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()

The screenshot displays the Android Studio IDE with the following code in MainActivity.java:

```
1 package com.example.cours1;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     private TextView myTextView;
8     private int compteur;
9     public static final String COMPTEUR = "com.example.cours1.COMPTEUR";
10
11     /* onCreate est appelée :
12      * - quand l'activité est créée
13      * - à chaque fois qu'elle est recrée, que ce soit
14      *   - parce que l'OS l'a tuée pour libérer des ressources
15      *   - parce que l'écran a été renversé (besoin de recalculer l'UI)
16      */
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {...}
19
20     //onSaveInstanceState permet de sauvegarder l'état de l'activité
21     //ici, on ne veut pas perdre la valeur de compteur si l'activité est tuée
22     @Override
23     protected void onSaveInstanceState(Bundle outState) {
24         //on se souvient de la valeur du compteur pour ne pas repartir de 0
25         outState.putInt(COMPTEUR, compteur);
26         super.onSaveInstanceState(outState);
27     }
28 }
```

The mobile emulator on the right shows a purple app interface with the text "Cours1" at the top, a counter value "4" in the center, and two buttons: "INCREMENT" and "DISPLAY".

## MainActivity::onSaveInstanceState()

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

protected void onSaveInstanceState(Bundle outState) {

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

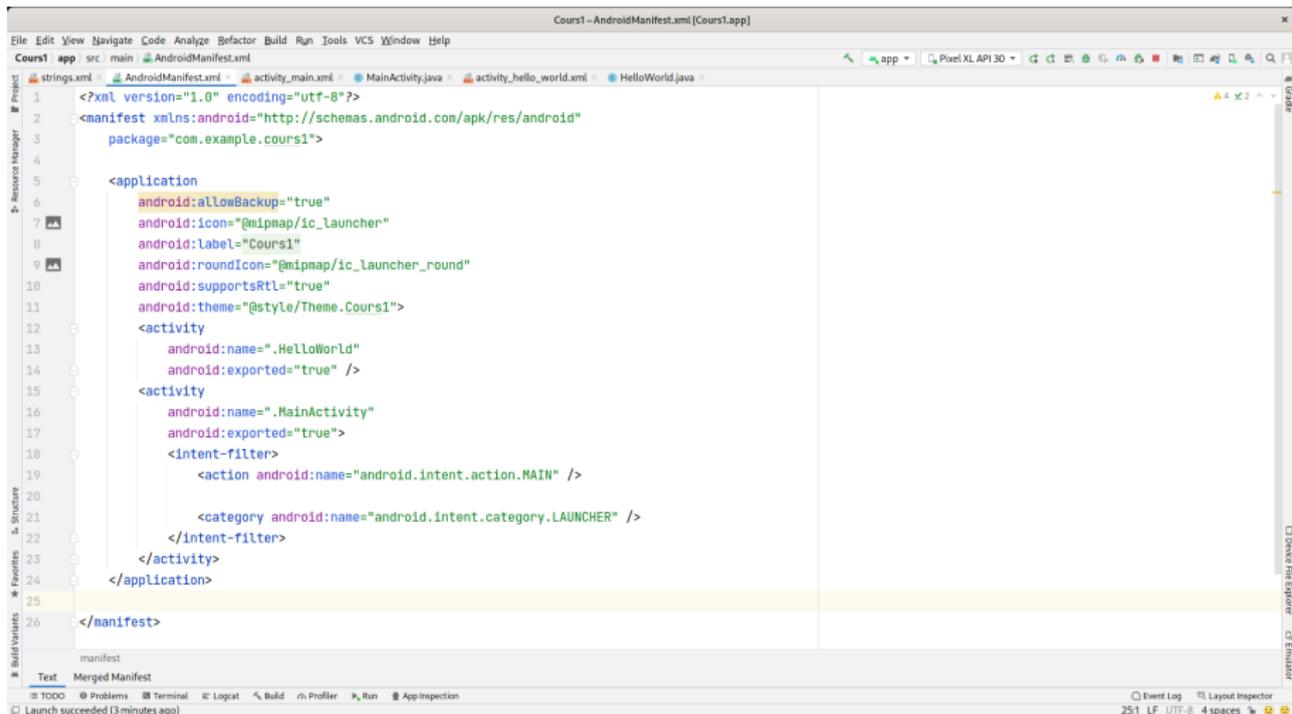
}

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 menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The toolbar shows various icons for file operations and development tools. 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 Project view on the left showing the file structure, a Resource Manager, and a Build view at the bottom. The status bar at the bottom indicates "Launch succeeded (3 minutes ago)", "25:1 LF UTF-8 4 spaces", and icons for Event Log, Layout Inspector, and other tools.