

Programmation mobile

Cours 1 : Généralités

Julien Grange <julien.grange@lacl.fr>

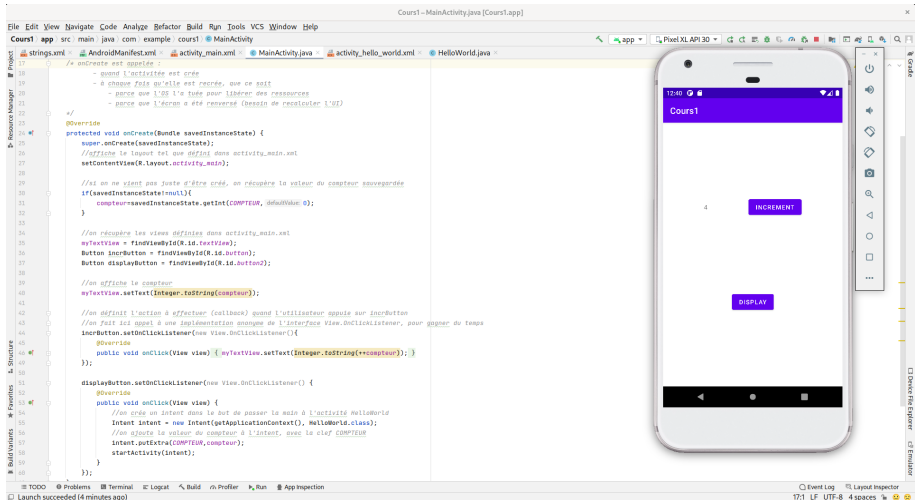
Mardi 9 septembre 2025

- CM : 6 séances de 1H30
- TP : 7 séances de 3H
- Enseignants : Quentin Le Houérou (TP), Julien Grange (CM et TP)
- Évaluation : 3 CC, moyenne des 2 meilleurs

Ajoutez une photo sur Eprel !

- 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



- 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

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`

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 créée (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)

Gestion agressive des ressources par l'OS

- 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` :

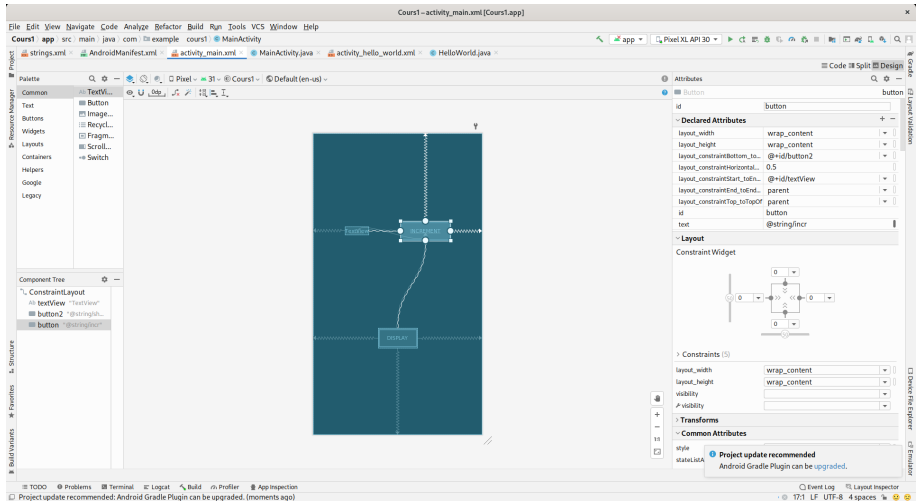
- `MainActivity` : classe principale, accessible depuis le launcher
 - premier bouton → incrémente un compteur
 - deuxième bouton → lance `HelloWorld` via un `Intent`

Première application : Hello World !

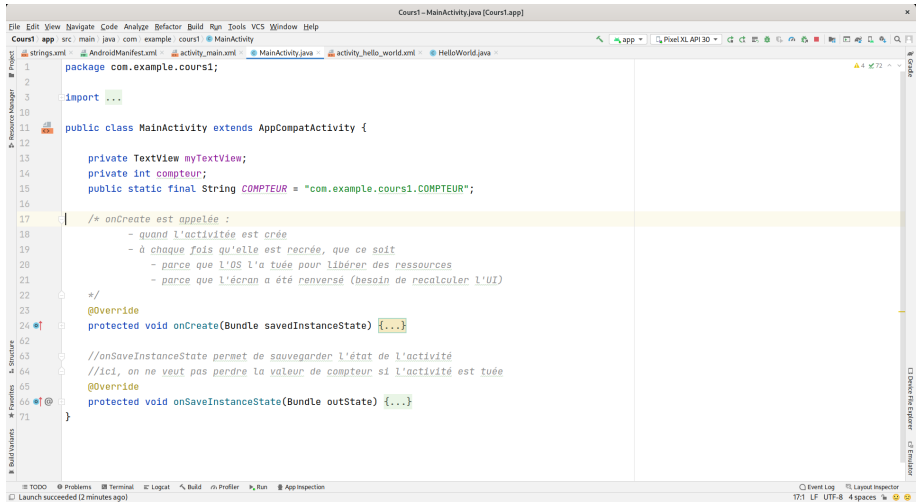
Deux `Activity` :

- `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



MainActivity code



```
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 }
```

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 handles button clicks. A virtual device emulator is shown on the right, displaying the app's interface.

```
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);
33     }
34
35     //on récupère les vues 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
51     displayButton.setOnClickListener(new View.OnClickListener() {
52         @Override
53         public void onClick(View view) {
54             //on crée un Intent dans le but de passer la main à l'activité HelloWorld
55             Intent intent = new Intent(getApplicationContext(), HelloWorld.class);
56             //on ajoute la valeur du compteur à l'Intent, avec la clef COMPTEUR
57             intent.putExtra(COMPTEUR, compteur);
58             startActivity(intent);
59         }
60     });
61 }
```

The virtual device emulator shows the app's UI with a purple header labeled "Cours1", a text view displaying "4", and two buttons: "INCREMENT" and "DISPLAY".

Launch succeeded (4 minutes ago)

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 displays the Android Studio IDE with the following components:

- Code Editor:** Shows the `HelloWorld.java` file with the following code:

```
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 }
```
- Virtual Device Emulator:** A smartphone screen showing the app's UI. The status bar at the top displays the time 12:36 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. The bottom navigation bar is black.
- Bottom Bar:** Contains a green "Launch succeeded" button and a status bar with the text "Launch succeeded (moments ago)".
- Right Panel:** Includes a "Project update recommended" notification stating "Android Gradle Plugin can be upgraded." and a "Device file explorer" button.
- Bottom Status Bar:** Shows the time 13:44, layout (LF), UTF-8 encoding, and 4 spaces.

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 `MainActivity.java` file open. The code implements a counter application. The `onSaveInstanceState()` method is highlighted, showing how it saves the counter value to a bundle before the activity is destroyed. To the right, a mobile emulator shows the app's UI with a purple header, a counter at 4, and 'INCREMENT' and 'DISPLAY' buttons.

```
1 package com.example.cours1;
2
3 import ...
4
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
62
63     //onSaveInstanceState permet de sauvegarder l'état de l'activité
64     //ici, on ne veut pas perdre la valeur de compteur si l'activité est tuée
65     @Override
66     protected void onSaveInstanceState(Bundle outState) {
67         //on se souvient de la valeur du compteur pour ne pas repartir de 0
68         outState.putInt(COMPTEUR, compteur);
69         super.onSaveInstanceState(outState);
70     }
71 }
```

Launch succeeded (5 minutes ago)

17:1 LF UTF-8 4 spaces

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

