Programmation mobile

Cours 1 : Généralités

Julien Grange <julien.grange@lacl.fr>

Mardi 9 septembre 2025

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL

Informations pratiques

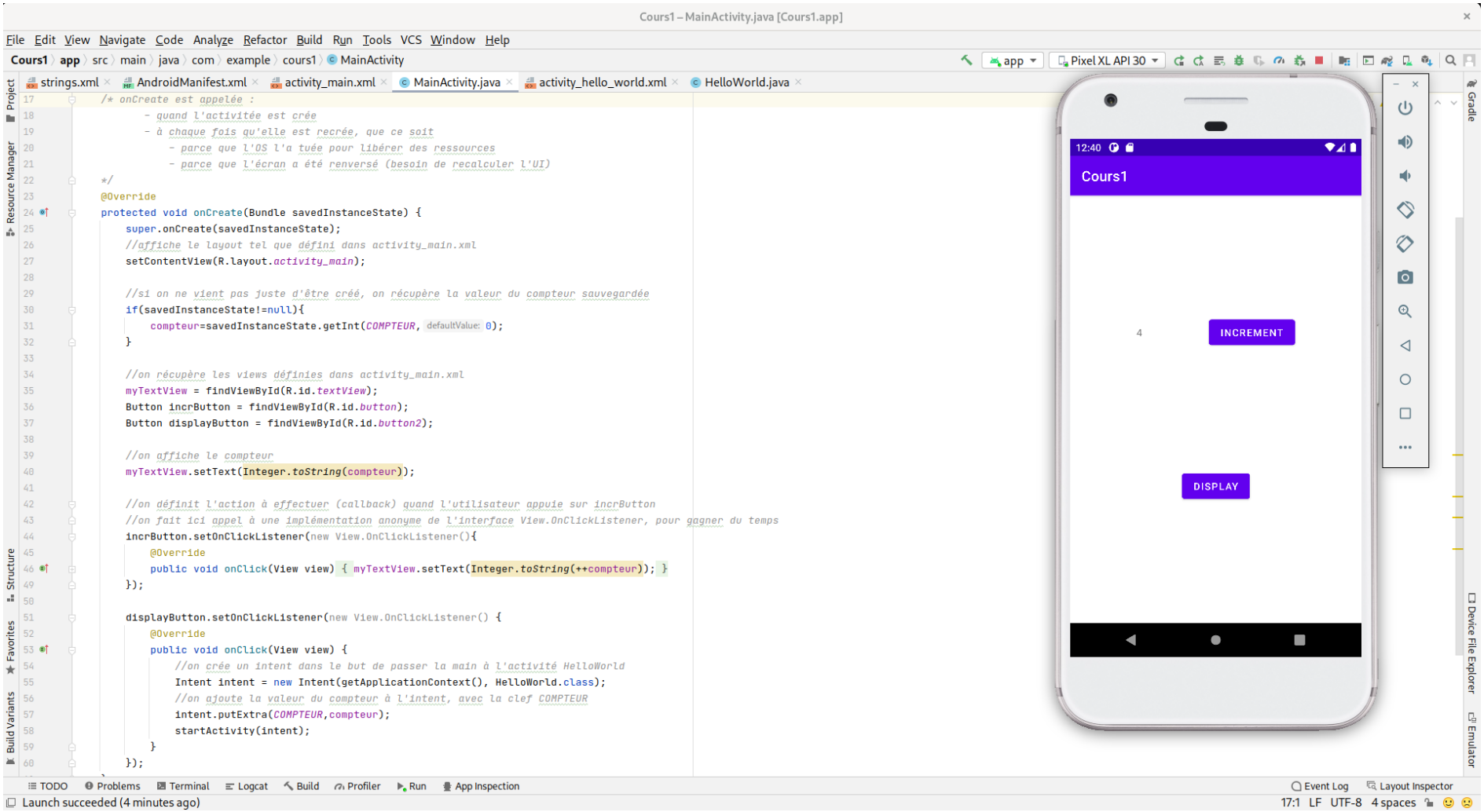
- CM : 6 séances de 1H30
- TP : 7 séances de 3H
- Enseignants : Quentin Le Houérou (TP), Julien Grange (CM et TP)
- Évaluation : 3 CC, moyenne des 2 meilleurs

Ajoutez une photo sur Eprel !

Outils

- 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



Conventions du cours

- 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 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 criée (e.g. pour prendre une photo)
- Programmation événementielle, à base de callbacks

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)

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` :

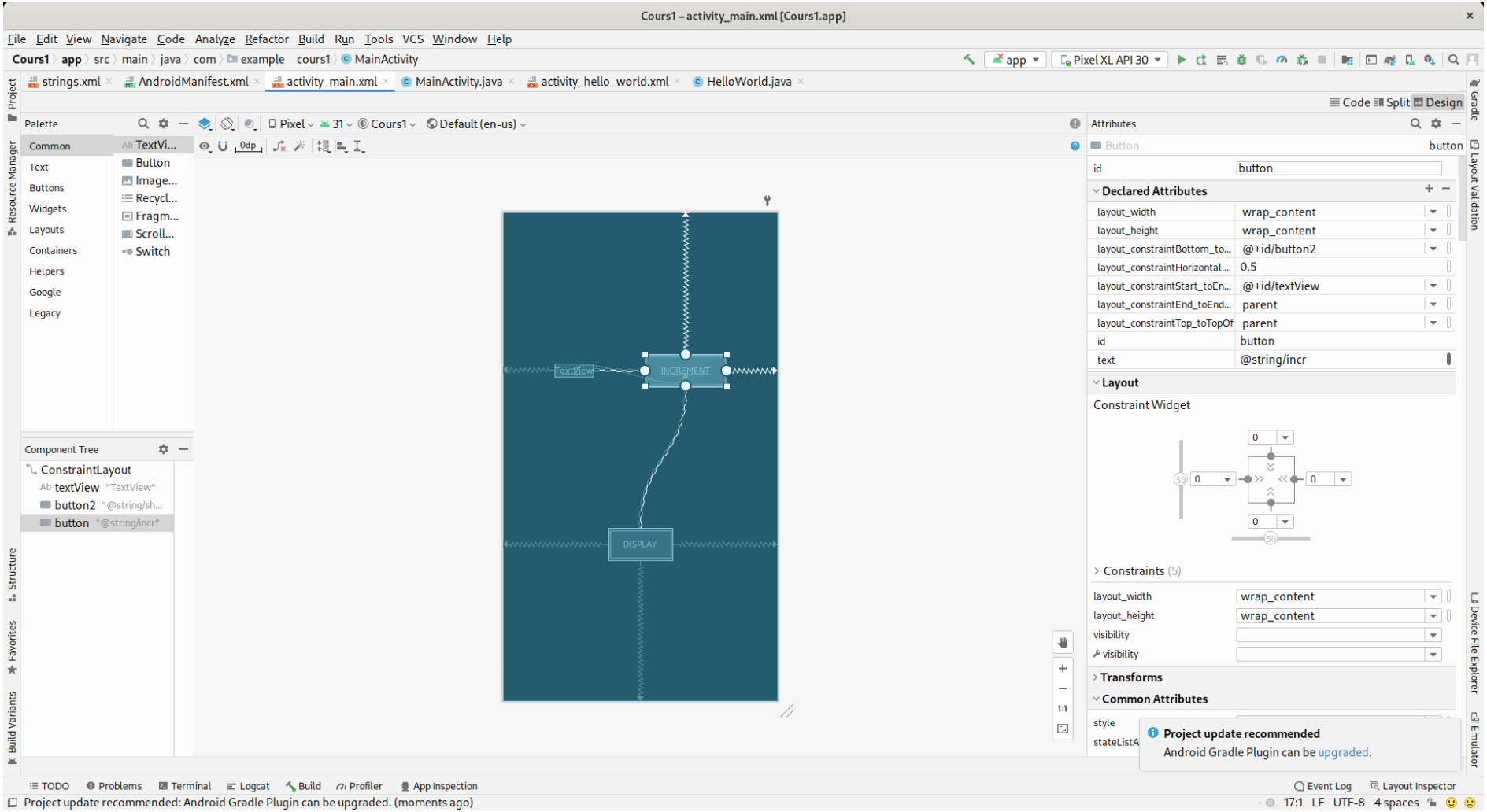
- `MainActivity` : classe principale, accessible depuis le launcher
  - premier bouton $\rightarrow$ incrémente un compteur
  - deuxième bouton $\rightarrow$ lance `HelloWorld` via un `Intent`

Première application : Hello World !

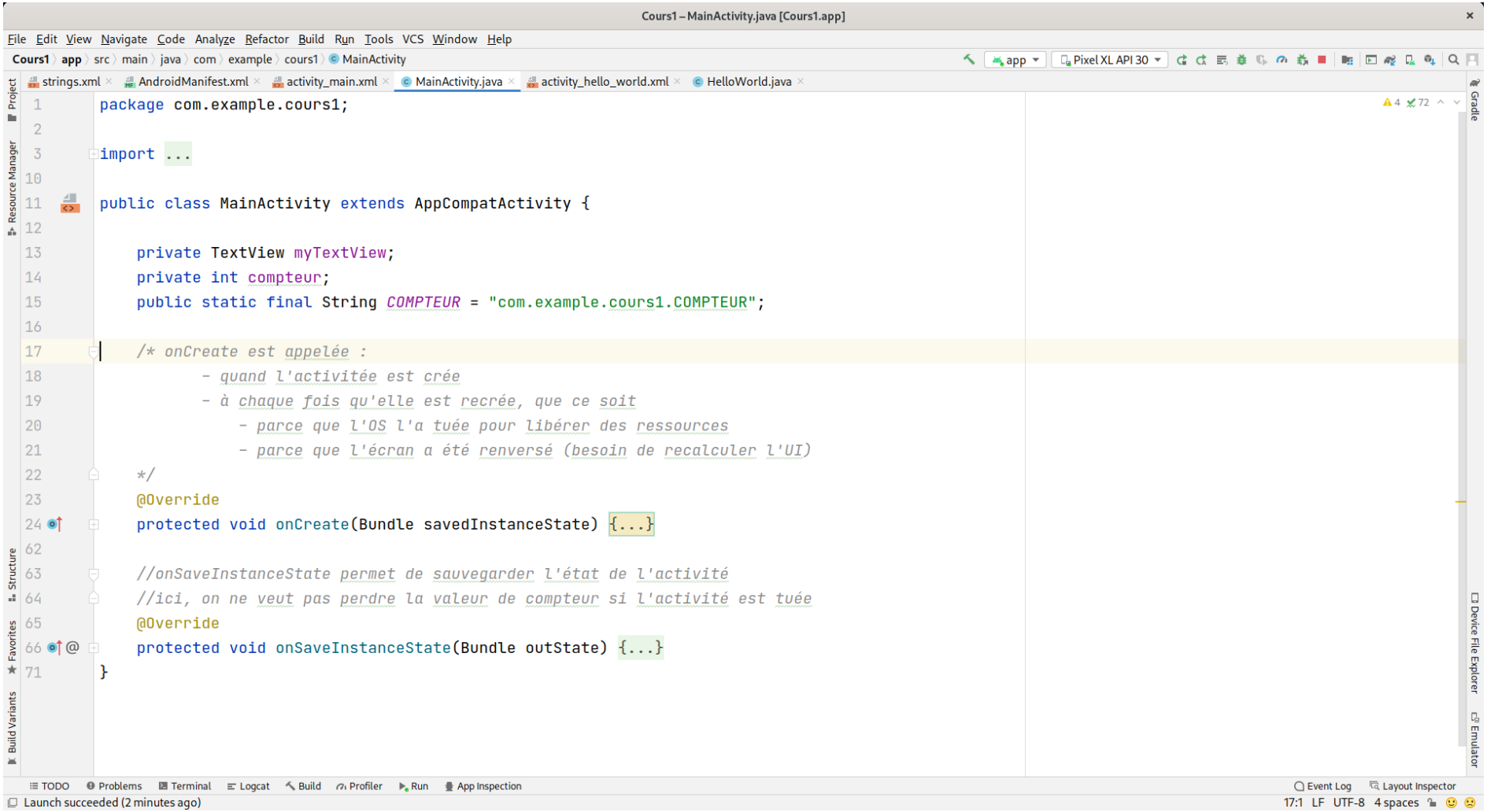
Deux `Activity` :

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

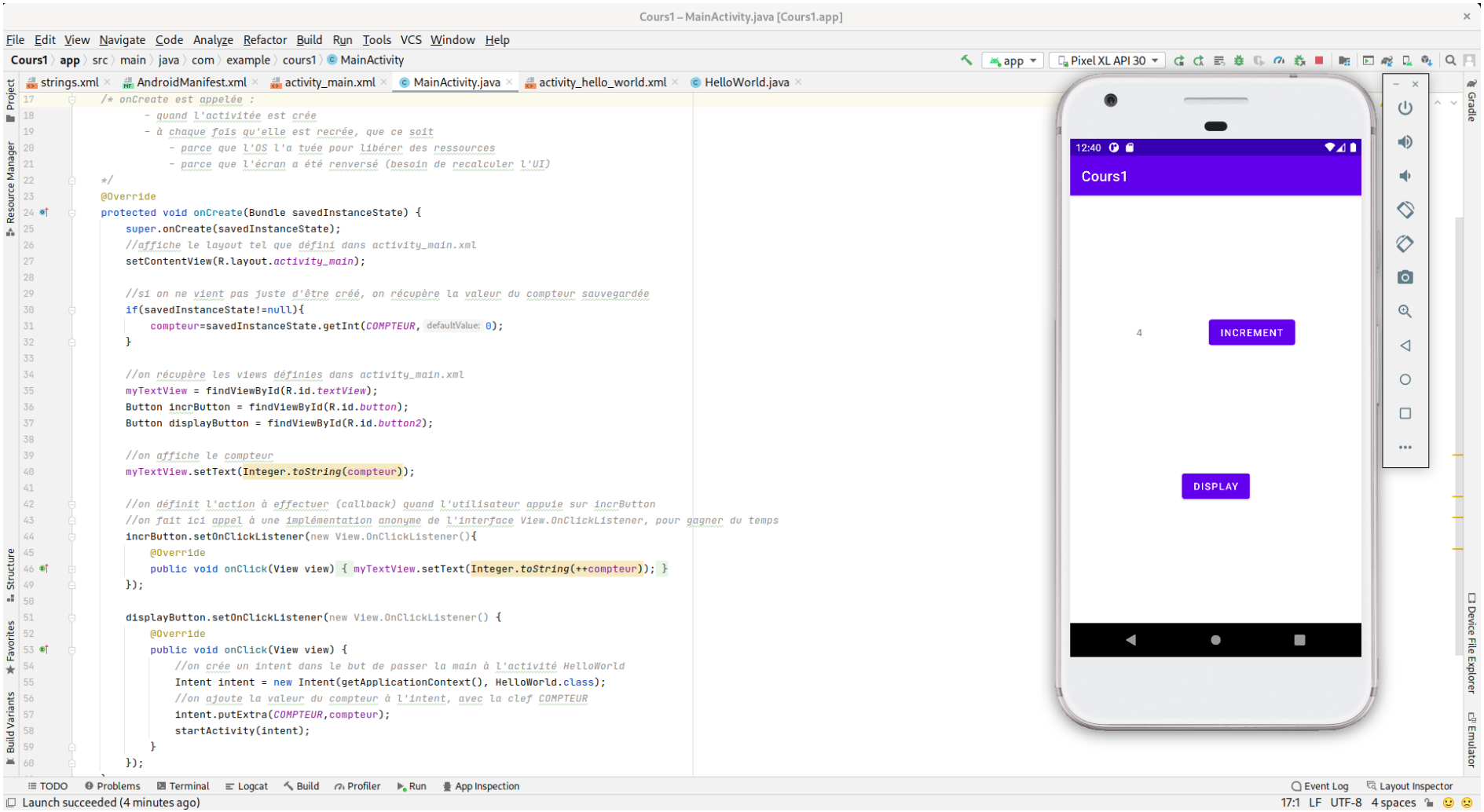
MainActivity layout



MainActivity code



MainActivity::onCreate()



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);
    ...
  }
}
```

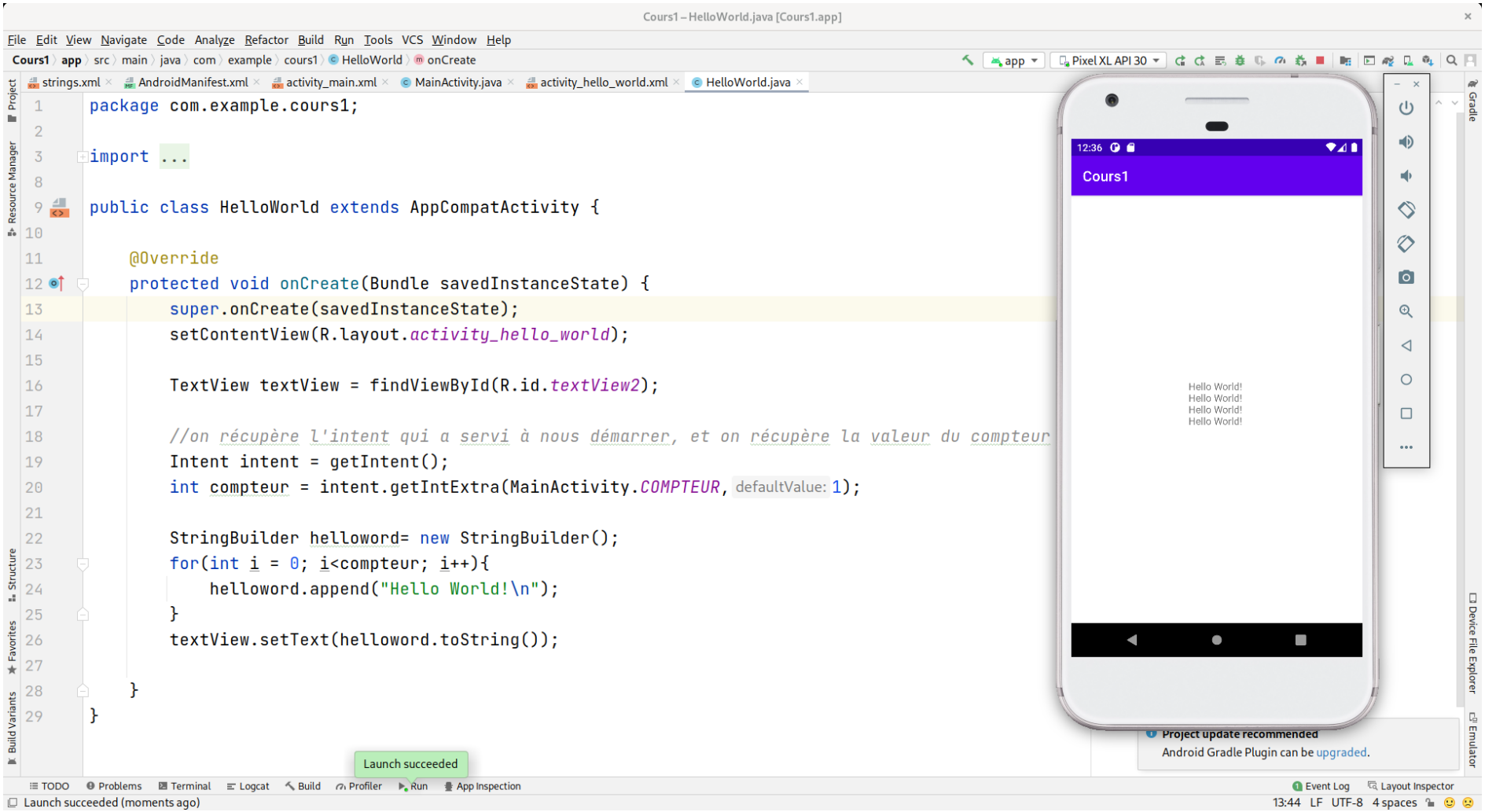
View::setOnClickListener()

```
incrButton.setOnClickListener(new View.OnClickListener(){

  @Override
  public void onClick(View view){
    myTextView.setText(Integer.toString(++compteur));
  }

});
```

HelloWorld code



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);
  ...
}
```

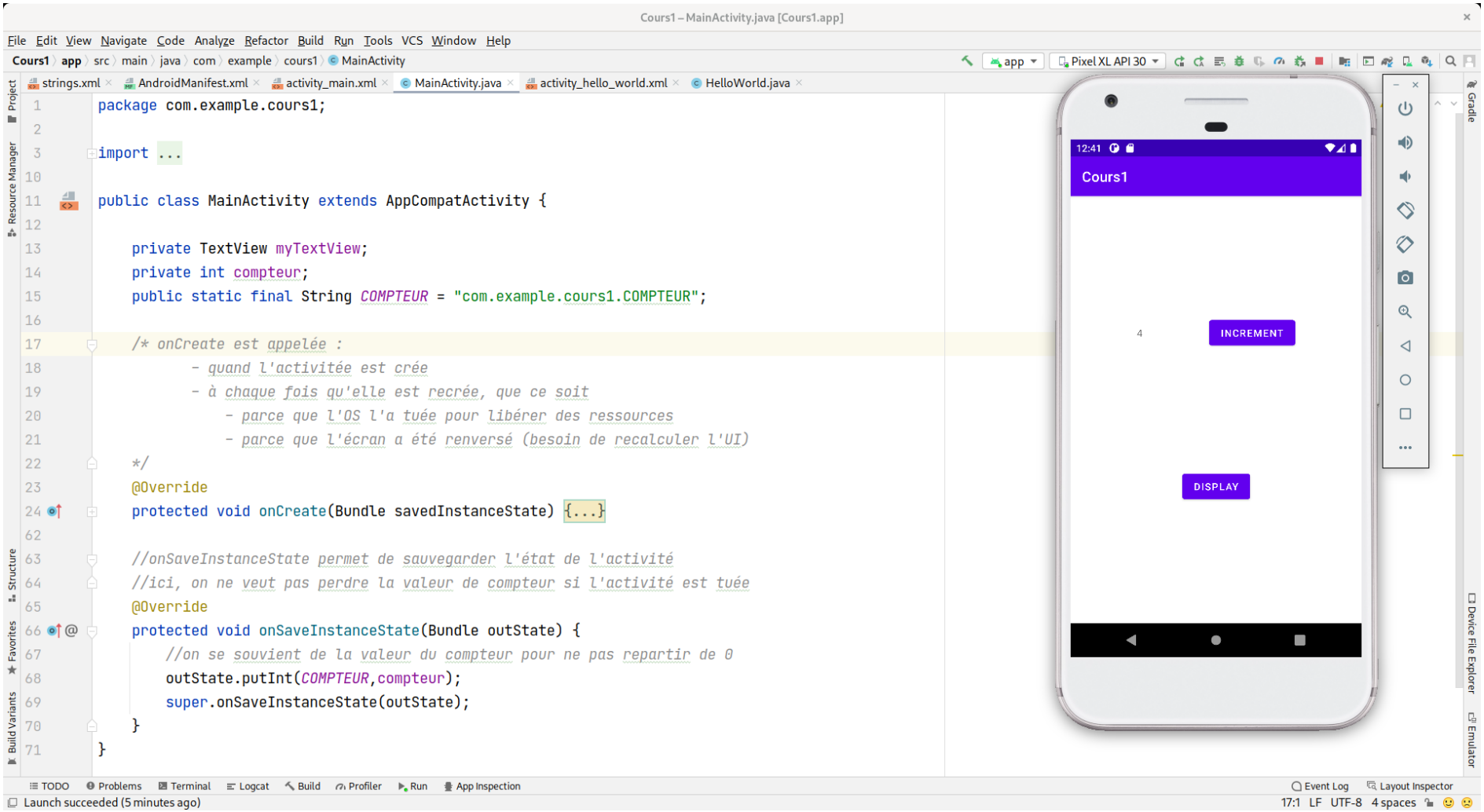
`Activity::onSaveInstanceState()`

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()



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

Cours1 – AndroidManifest.xml [Cours1.app]

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

Cours1 › app › src › main › AndroidManifest.xml

app | Pixel XL API 30

strings.xml | AndroidManifest.xml | activity_main.xml | MainActivity.java | activity_hello_world.xml | HelloWorld.java

```xml
<?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="Cours1"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="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>
```

manifest

Text | Merged Manifest

TODO | Problems | Terminal | Logcat | Build | Profiler | Run | App Inspection | Event Log | Layout Inspector

Launch succeeded (3 minutes ago) 25:1 LF UTF-8 4 spaces