Programmation mobile

Cours 1 : Généralités

Julien Grange <julien.grange@lacl.fr>

Mardi 10 septembre 2024

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL

Informations pratiques

- 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 et environnement

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 et environnement

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 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 CO^2 par an

Numérique et environnement

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 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 CO^2 par an

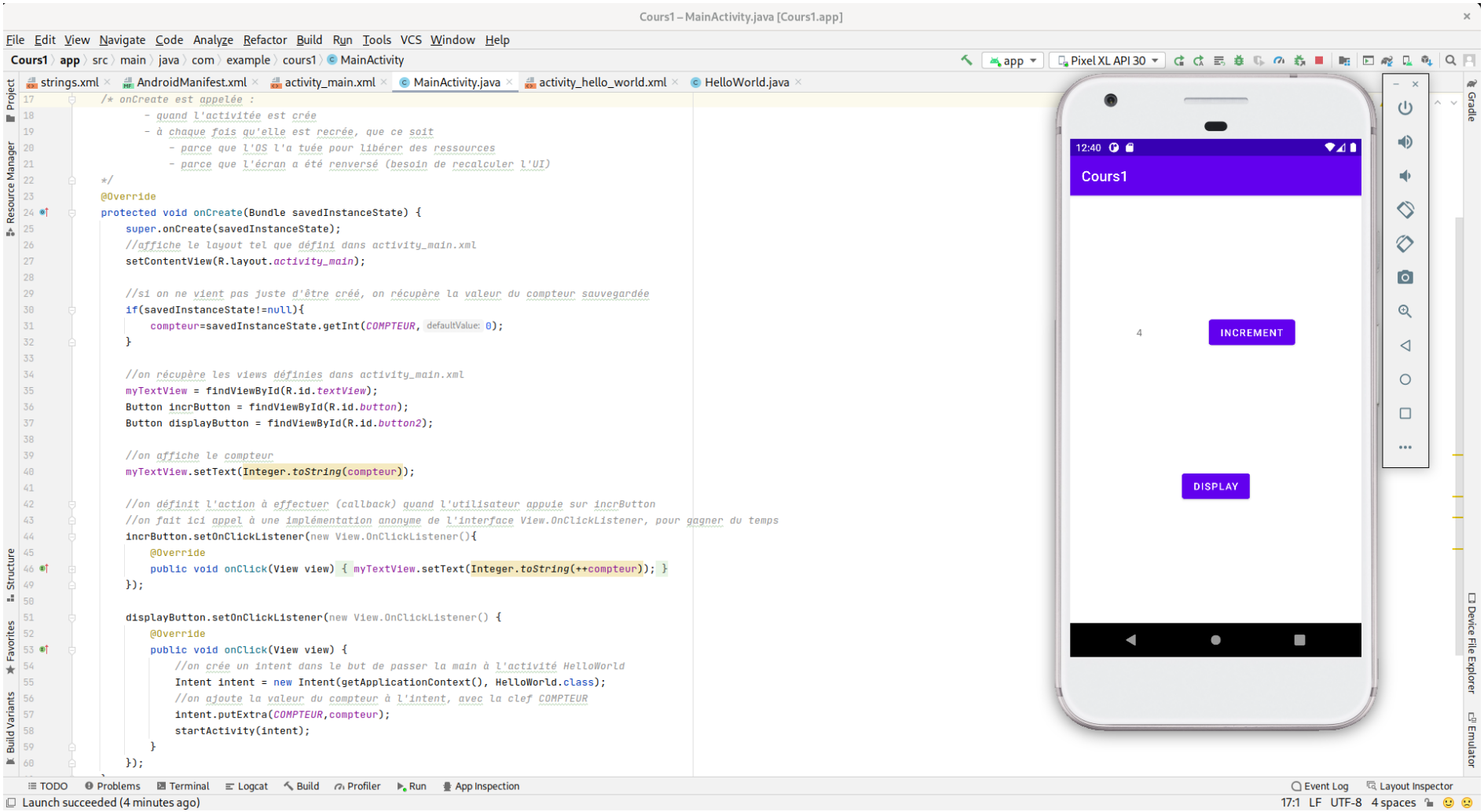
Il importe donc

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

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` :

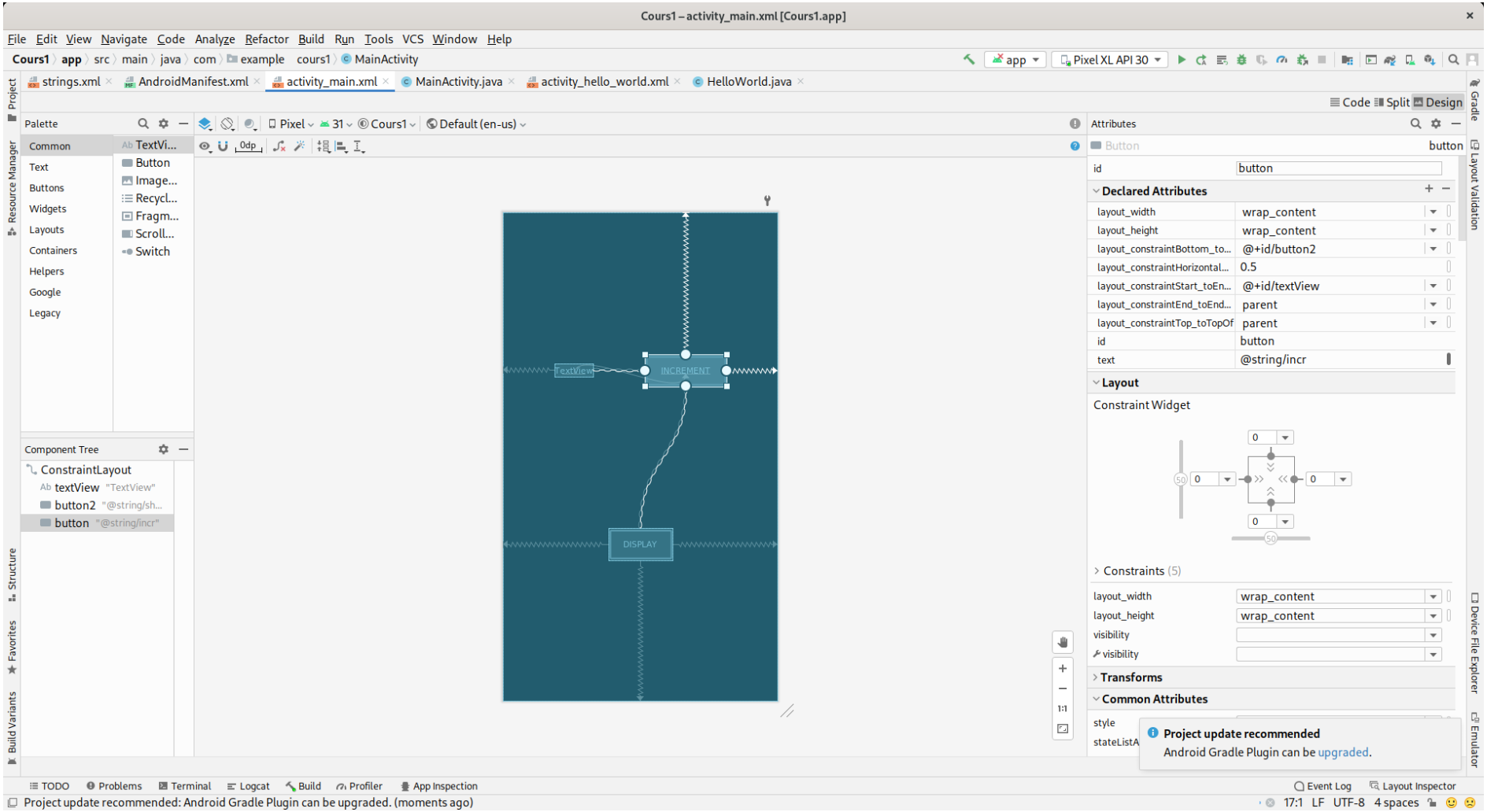
- `MainActivity` : classe principale, accessible depuis le launcher
  - premier bouton → incrémente un compteur
  - deuxième bouton → lance `HelloWorld` via un `Intent`

Première application : Hello World !

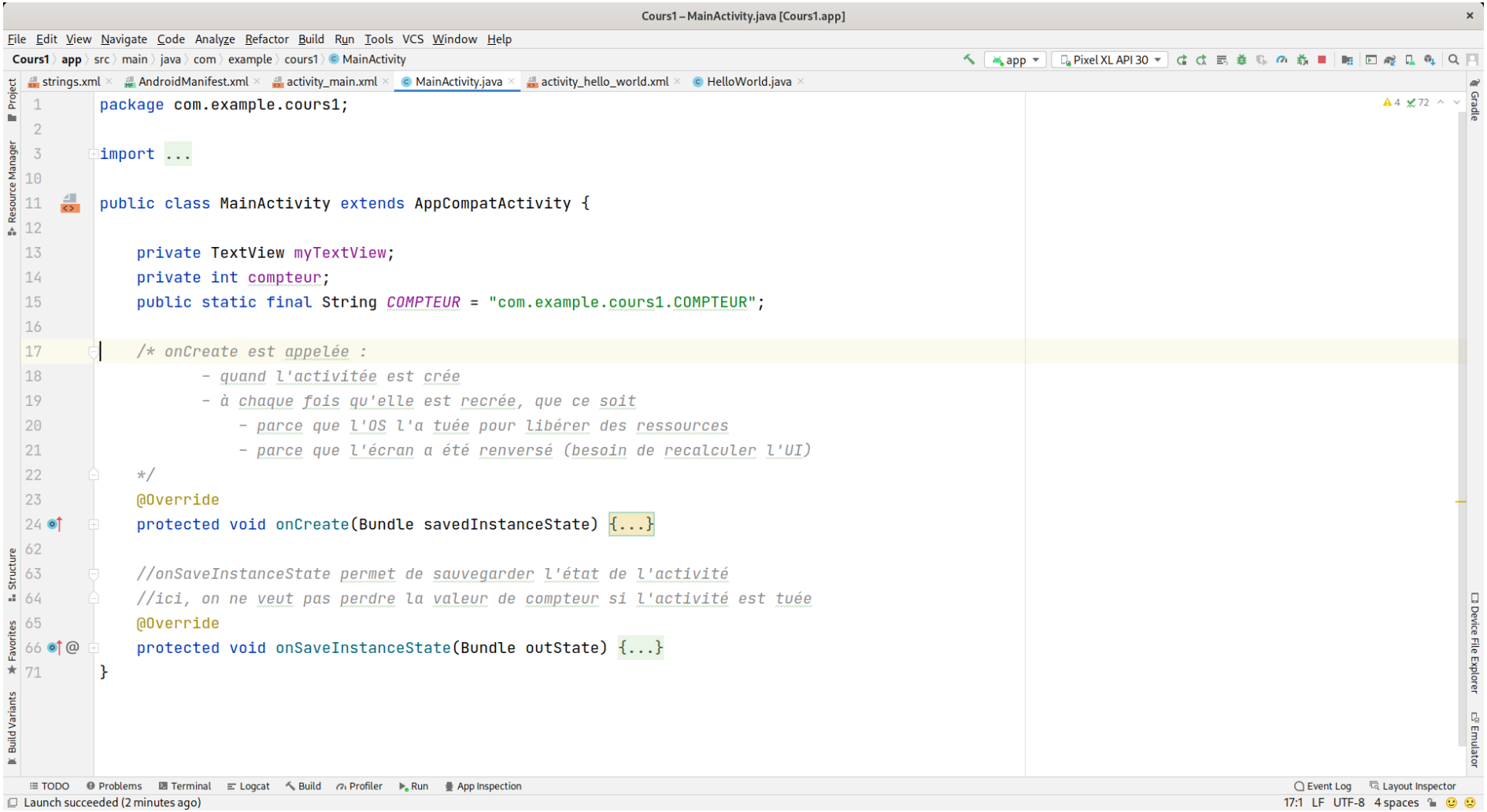
Deux `Activity` :

- `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

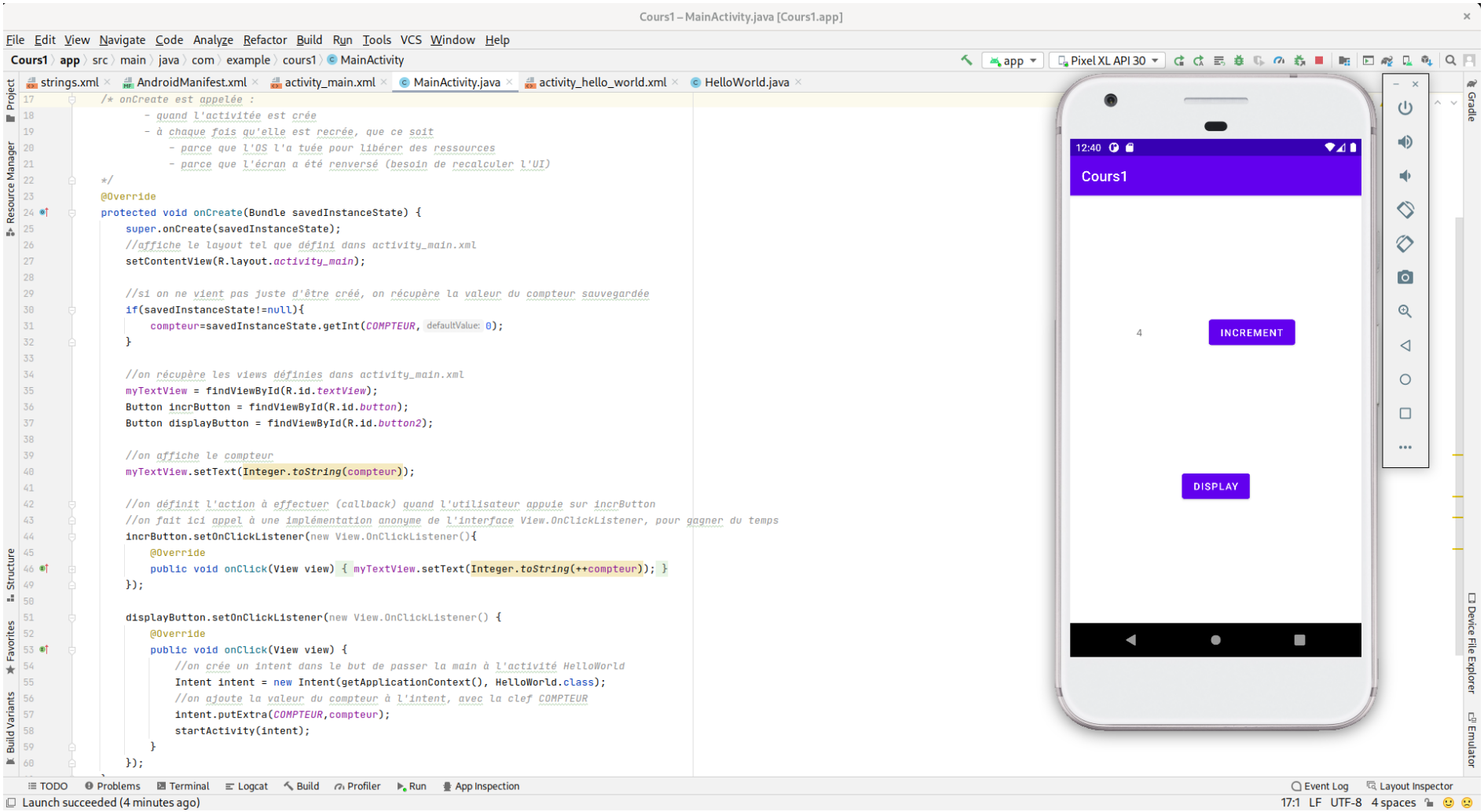
MainActivity layout



MainActivity code



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);
    ...
  }
}
```

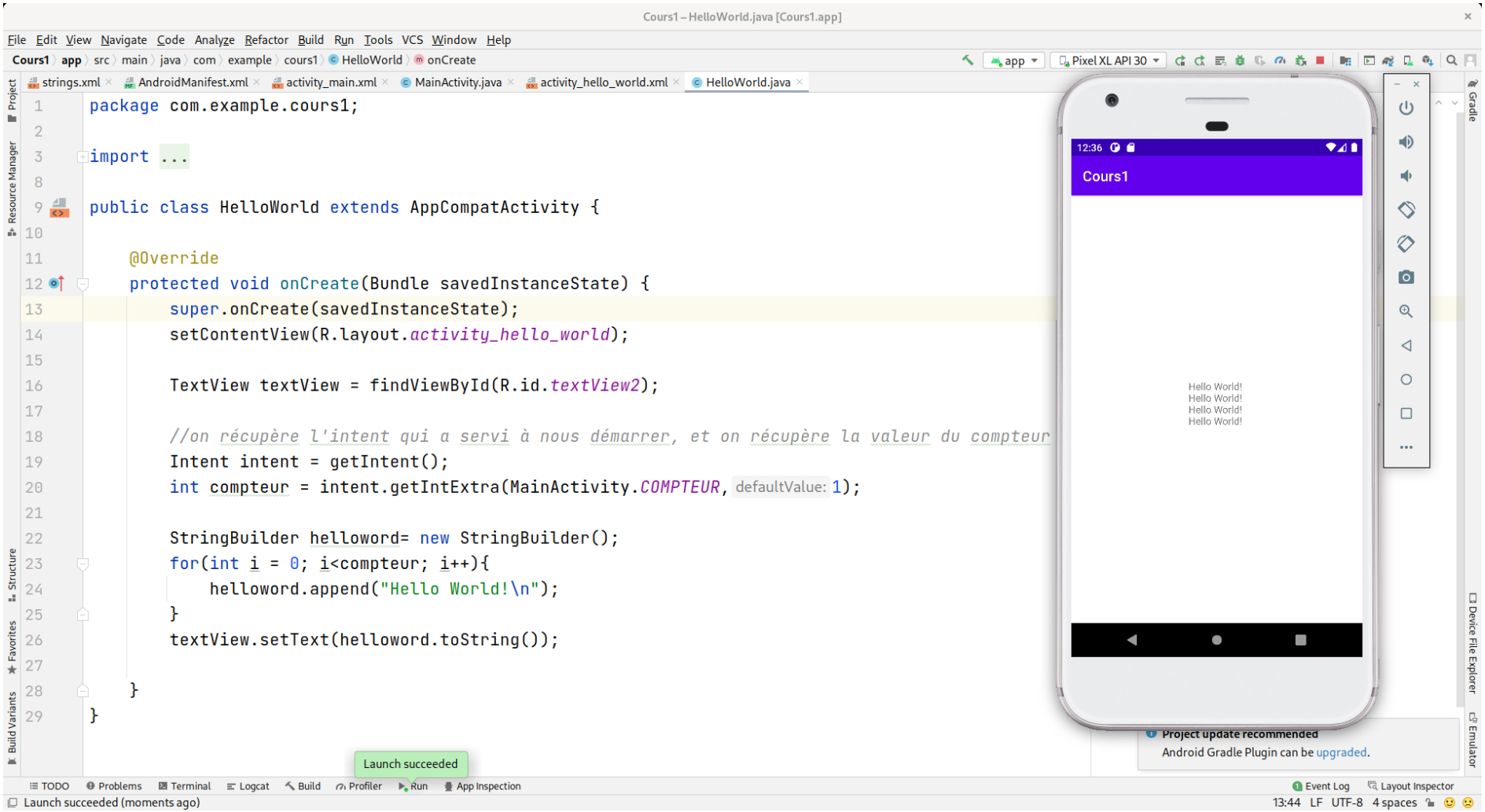
View::setOnClickListener()

```
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);
  ...
}
```

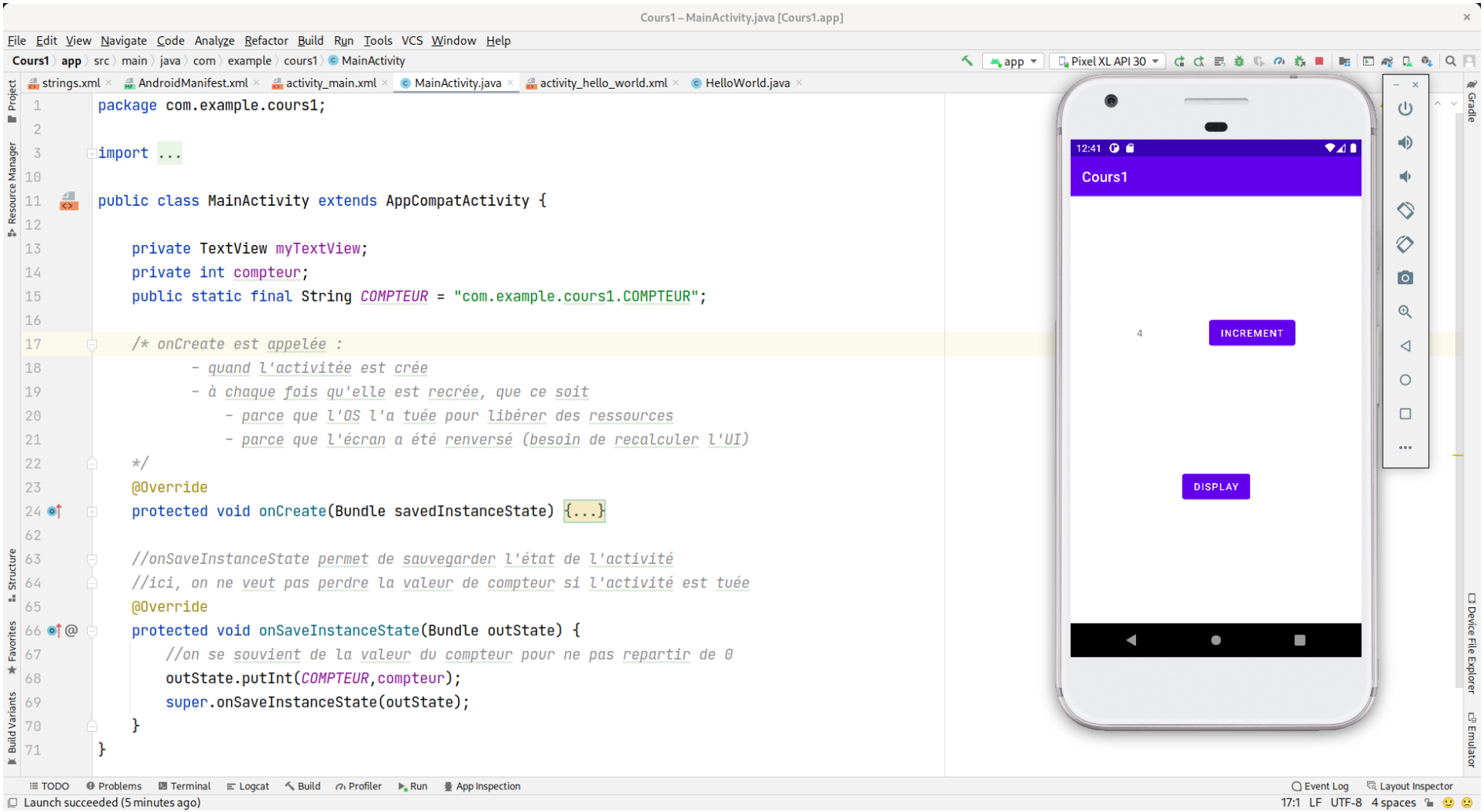
`Activity::onSaveInstanceState()`

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