Avant de vous lancer dans ce TP, assurez-vous d'avoir suivi les instructions de *Prise en main d'Android Studio*.

On rappelle ici les conventions du cours :

— Les classes et interfaces sont encadrées : `Class`

— Les méthodes et champs statiques d'une classe sont dénotés `Class.foo()` ou `Class.bar`

— Les méthodes et champs non-statiques (i.e. propres aux objets) d'une classe sont dénotés `Class::foo()` ou `Class::bar`

— Si la classe en question est claire, on se contentera de `foo()` ou `bar`

1 Hello World !

Question 1 Créez un nouveau projet. Lorsque le type d'activité est demandé, choisir *Empty Views Activity*, en Java. Gardez les valeurs par défaut pour le reste.

Explorez l'arborescence du projet, qui peut se déplier/replier en cliquant sur le bouton *Project* de gauche, cf. Figure 1.

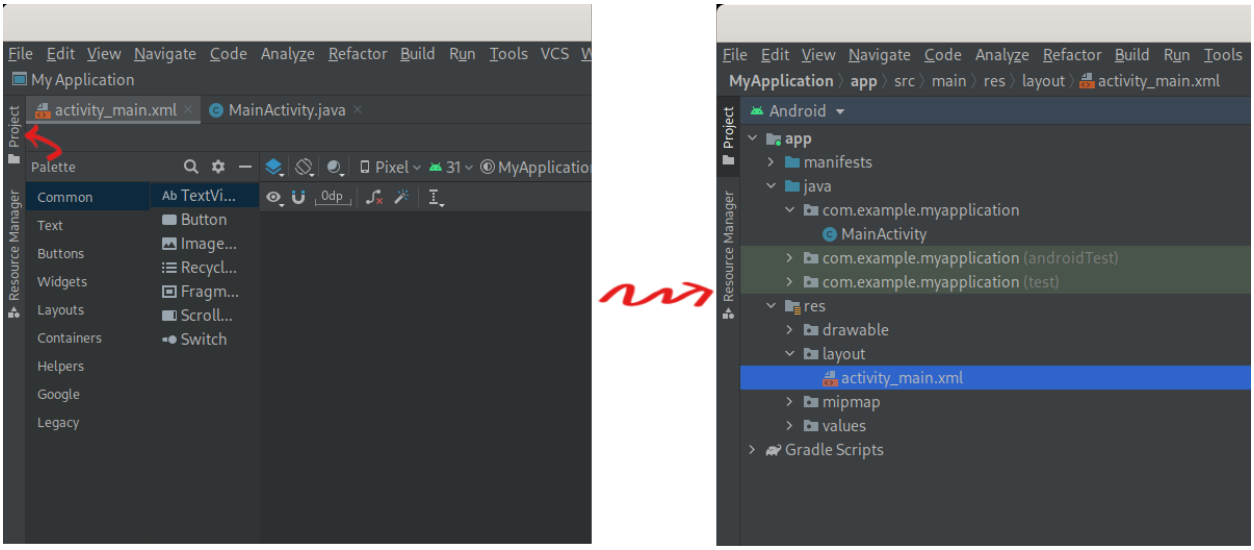

FIGURE 1 – Exploration de l'arborescence du projet.

En particulier, retrouvez le code de MainActivity, le fichier xml spécifiant son layout, et le Manifest de l'application.

Question 2 Compilez et exécutez l'application sur votre appareil (physique ou virtuel).

Question 3 Dans le fichier `activity_main.xml`, remplacez le "Hello world !" par votre prénom et augmentez la taille de la police du texte. Testez le résultat.

Question 4 Dans le fichier `activity_main.xml`, jetez un œil au code XML (onglet *Code* en haut à droite, cf. Figure 2). Puis cliquez sur l'onglet *Design*, cherchez dans la palette un *Button* (cf. Figure 2) et glissez-le dans l'interface.

Regardez les différences dans le code XML (vous pouvez ignorer le warning pour le moment). Pour observer d'un coup le code XML et le blueprint, cliquez sur l'onglet *Split*. Notez que l'id par défaut de ce bouton est *button* ; cela servira lors de la Question 5.

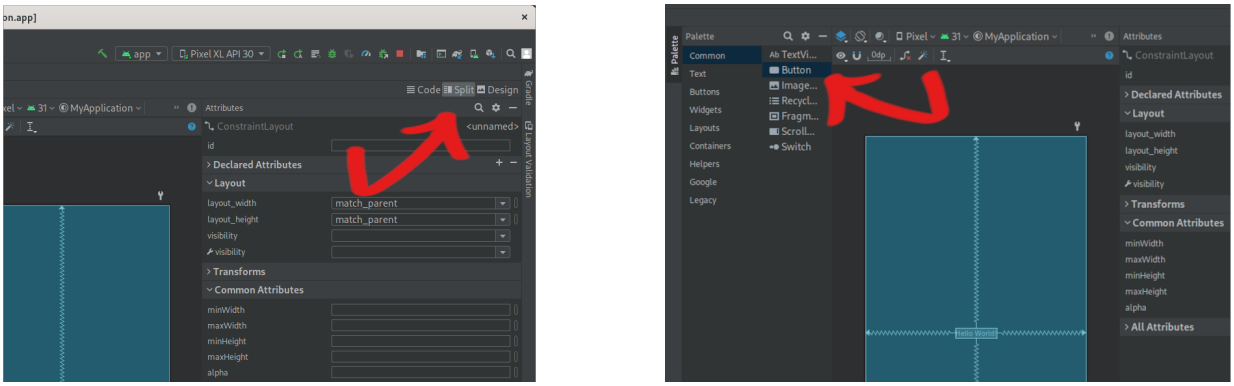

FIGURE 2 – Édition du layout en mode *Code*, *Split* ou *Design*. Utilisation de la Palette.

Question 5 On veut faire en sorte qu'un clic sur ce bouton affiche un texte dans une pop-up. Pour cela, on va écrire un écrire un callback et l'associer au bouton qu'on vient de créer.

On va utiliser un `Toast`, qui est une manière d'afficher un court message en bas de l'écran. À chaque clic sur le bouton, on va créer un `Toast` via `Toast.makeText()`, et le faire s'afficher via `Toast::show()`.

Dans le fichier `MainActivity.java`, ajoutez dans la méthode `onCreate()`, après les deux lignes intégrées par défaut, le code suivant :

```
Button button = findViewById(R.id.button);

button.setOnClickListener(new View.OnClickListener() {
  public void onClick(View view) {
    Toast.makeText(getApplicationContext(),"Clic␣!", Toast.LENGTH_SHORT)
      .show();
  }
});
```

Vérifiez que le code fonctionne bien comme prévu.

Question 6 Retournez dans le fichier `activity_main.xml`, et ajoutez des contraintes (clic droit → center, et en en cliquant sur les ronds situés autour du bouton) pour que le bouton soit centré horizontalement et soit situé juste au-dessous du champ de texte.

Vérifiez la correction de votre layout. Pour cela, on pourra explorer l'onglet *Layout Validation* sur la droite (cf. Figure 3), qui donne un aperçu de l'UI sur différents types d'écrans.

Question 7 Dans le fichier `MainActivity.java`, modifiez le code pour qu'un clic sur le bouton change la valeur du champ de texte.

On rappelle (cf. les slides du premier cours) que la méthode `TextView::setText()` permet de modifier le texte d'une `TextView`.

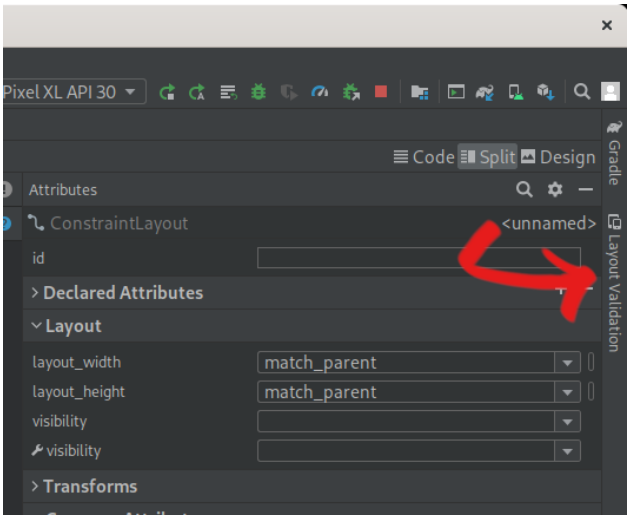

FIGURE 3 – Visualisation de l'UI sur différentes tailles d'écran.

2 Convertisseur

Question 8 Fabriquez un nouveau projet nommé *Convertisseur*.

Modifiez le fichier `activity_main.xml` pour que le layout soit composé de deux champs de texte, l'un éditable (`EditText`) et l'autre non (`TextView`), ainsi que deux `Button`, étiquetés par "€ → £" et "£ → €".

On fera en sorte que le `EditText` soit destiné à recevoir des caractères numériques.

Question 9 Modifiez le code de `MainActivity` pour implémenter un convertisseur Livre ↔ Euro.

Question 10 (☠) Généralisez l'application pour qu'elle supporte d'autres monnaies. On pourra par exemple utiliser deux `Spinner` : un pour la monnaie source, et le deuxième pour la monnaie cible.