

## Développement Mobile

# WorkShop 1 : Les Layouts


Année universitaire 2019/2020



# Ce que vous devriez déjà savoir



- Vous devriez être familier avec:
  - Comment installer et ouvrir Android Studio.
  - Comment créer l'application HelloWorld.
  - Comment exécuter l'application HelloWorld.



# Ce que vous allez apprendre

- ✓ Comment créer une application avec un comportement interactif.
- ✓ Comment utiliser l'éditeur de layout pour concevoir une layout.
- ✓ Comment éditer la layout en XML.



# Introduction

- L'interface utilisateur qui apparaît sur l'écran d'un appareil Android consiste en une hiérarchie d'objets appelée **View** (chaque élément de l'écran est un **View**).
- La classe **View** représente le bloc de construction de base pour tous les composants d'interface utilisateur tels que des boutons, des cases à cocher et des champs de saisie de texte.



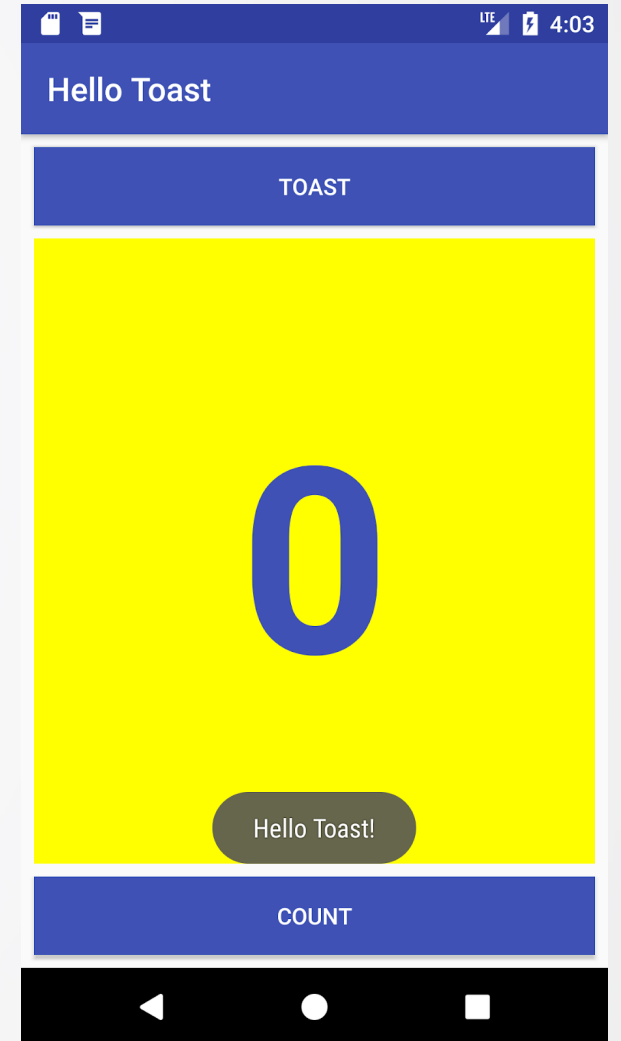
# Introduction



- Les sous-classes **View** couramment utilisées décrites dans plusieurs leçons comprennent:
  - **TextView** pour afficher du texte.
  - **EditText** pour permettre à l'utilisateur d'entrer et de modifier du texte.
  - **Button** et d'autres éléments cliquables (tels que **RadioButton**, **CheckBox**, et **Spinner**) pour fournir un comportement interactif.
  - **ScrollView** et **RecyclerView** pour afficher les éléments défilables.
  - **ImageView** pour afficher des images.
  - **ConstraintLayout** et **LinearLayout** pour contenir d'autres éléments **View** et les positionner.

# ▶ Aperçu de l'application à faire

- L'application **Hello Toast** est composée de deux **Button** et d'un **TextView**.
- Lorsque l'utilisateur appuie sur le premier **Bouton**, il affiche un court message (**Toast**) à l'écran.
- Appuyer sur le second **Bouton** augmente le compteur de "nombre de clique" affiché dans le **TextView**, qui commence à zéro.



# ► Tâche 1.1: Créer un nouveau projet



- Démarrez Android Studio et créez un nouveau projet avec les paramètres suivants:
  - Nom de l'application : **Hello Toast**
  - Company Name: **com.esprit.gl5x** (avec x le numéro de votre classe)
  - Minimum SDK: **API24: Android 7.0 Nougat**
  - Template: **Empty Activity**



## Tâche 1.2: Explorer l'éditeur de layout



- 1) Dans le dossier `app>res>layout` du volet `Projet>Android` , double-cliquez sur le fichier `activity_main.xml` pour l'ouvrir, s'il ne l'est pas déjà.
- 2) Cliquez sur l' onglet Conception s'il n'est pas déjà sélectionné.  
Vous utilisez l' onglet Conception pour manipuler les éléments et la présentation, et l' onglet Texte pour modifier le code XML de la présentation.
- 3) Le volet Palettes affiche les éléments d'interface utilisateur que vous pouvez utiliser dans la présentation de votre application.

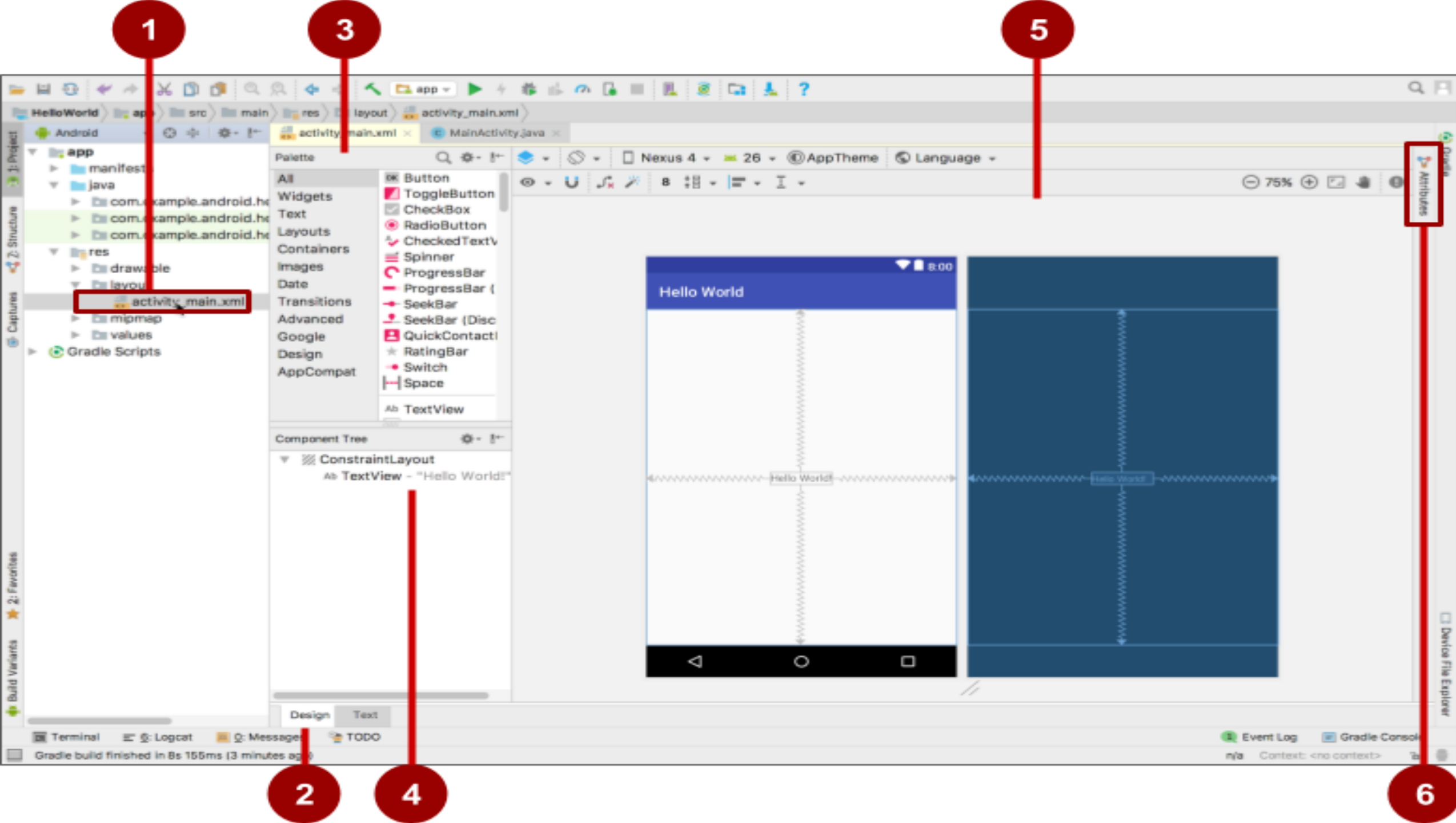




## Tâche 1.2: Explorer l'éditeur de layout



- 4) Le volet de l'arborescence des composants affiche la hiérarchie des éléments de UI. Les éléments sont organisés dans une arborescence de parents et d'enfants, dans laquelle un enfant hérite des attributs de son parent. Dans la figure ci-dessus, the [TextView](#) est un enfant de [ConstraintLayout](#).
- 5) Les panneaux de conception et de plan directeur de l'éditeur de disposition affichant les éléments d'interface utilisateur de la disposition. Dans la figure ci-dessus, la mise en page ne montre qu'un seul élément: un [TextView](#) qui affiche "Hello World".
- 6) L'onglet Attributs affiche le volet Attributs pour la définition des propriétés d'une [View](#).



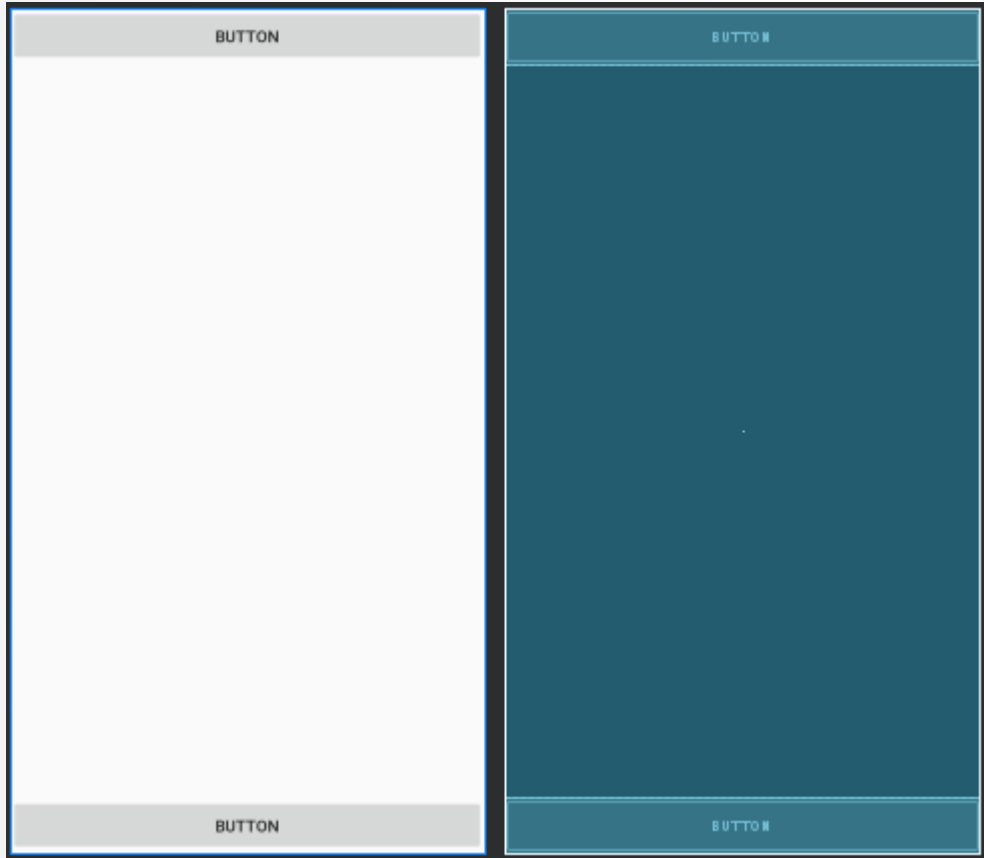
## Tâche 2.1: Créer l'interface



- 1) Commencez par supprimer le `TextView` pour avoir une interface vierge.
- 2) Utiliser la `ConstraintLayout` pour l'élément parent.

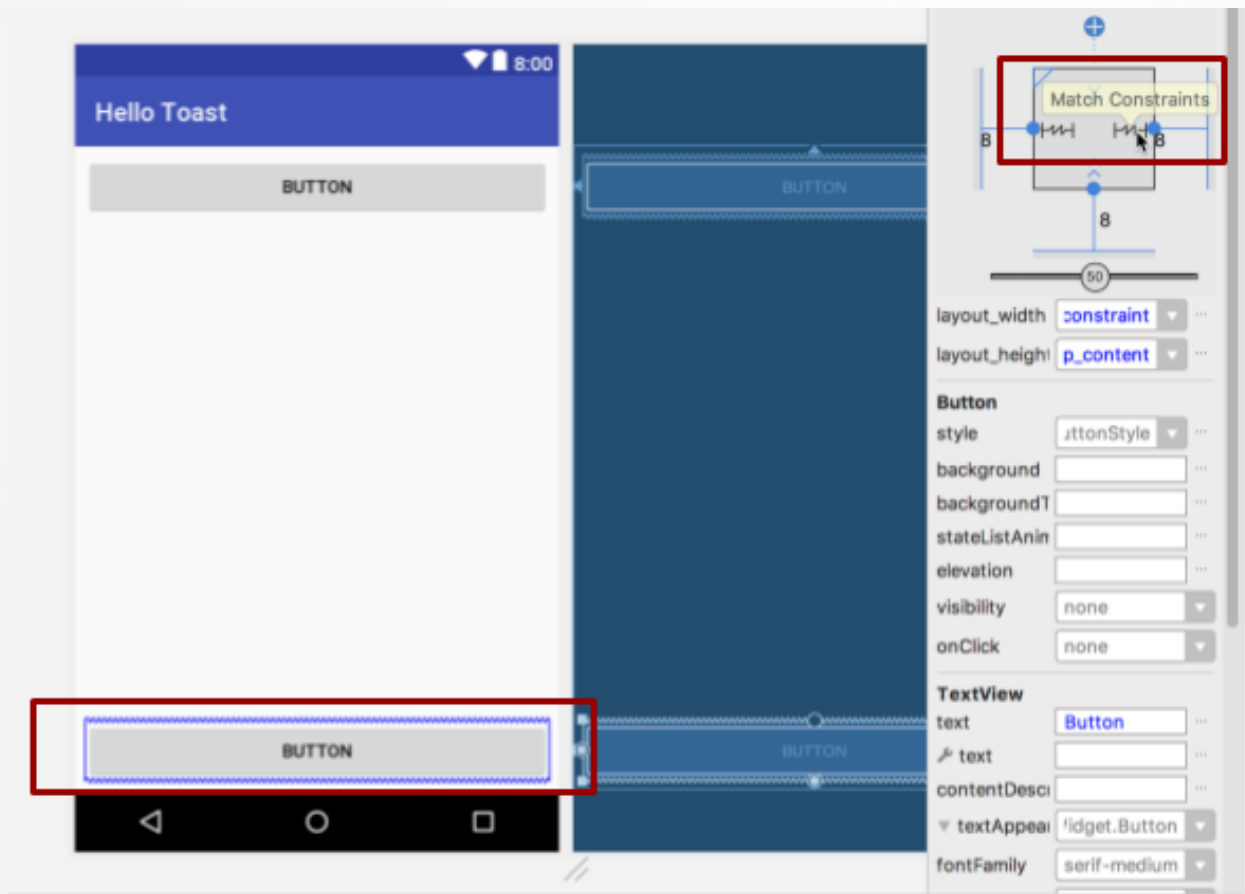
NB: Pour effacer toutes les contraintes d'une `ConstraintLayout`, cliquez sur le bouton **Effacer toutes les contraintes**  dans la barre d'outils.

## ► Tâche 2.1: Créer l'interface



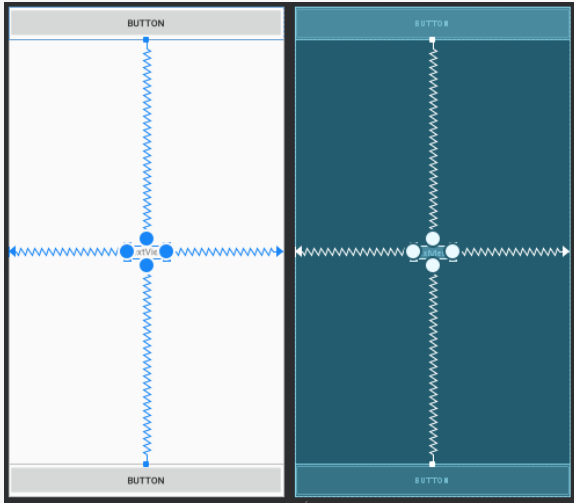
- 1) Ajouter un **bouton** et faites glisser les contraintes vers le haut, le côté gauche et le côté droit de l'interface
- 2) Ajouter un deuxième **bouton** et faites glisser les contraintes vers le bas, le côté gauche et le côté droit de l'interface
- 3) Ajuster les boutons de manière à avoir une largeur égale à la largeur du téléphone (voir figure)

# ► Tâche 2.1: Créer l'interface

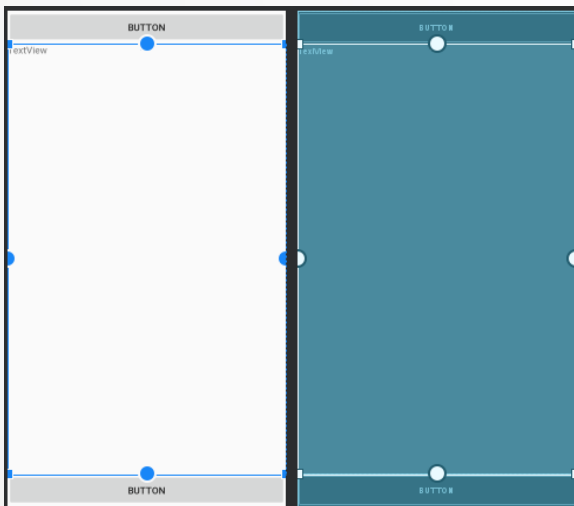


Pour ajuster il faut faire un double clique sur la position pour redimensionner l'élément

## ► Tâche 2.1: Créer l'interface



- 1) Ajouter un TextView et faites glisser les contraintes vers le haut, le bas, le côté gauche et le côté droit de l'interface (voir figure)



- 2) Ajuster le TextView de manière à avoir une largeur et une longueur égale à la largeur du téléphone (voir figure)



## ► Tâche 2.2: Modifier les propriétés

- Utilisez les propriétés suivantes pour le 1<sup>er</sup> Bouton:
  - id: button\_toast
  - background: colorPrimary
  - textColor: Blanc
  - text: Toast
  - layout\_marginLeft : 8dp
  - layout\_marginRight : 8dp
  - layout\_marginTop: 8dp



## ► Tâche 2.2: Modifier les propriétés

- Utilisez les propriétés suivantes pour le TextView:
  - id: show\_count
  - background: Jaune
  - textColor: colorPrimary
  - text: Count
  - gravity: center
  - textSize: 160sp
  - textStyle: bold
  - layout\_margin: 8dp





## ► Tâche 2.2: Modifier les propriétés

- Utilisez les propriétés suivantes pour le 2<sup>eme</sup> Bouton:
  - id: button\_count
  - background: colorPrimary
  - textColor: Blanc
  - text: Count
  - layout\_marginLeft : 8dp
  - layout\_marginRight : 8dp
  - layout\_marginBottom: 8dp



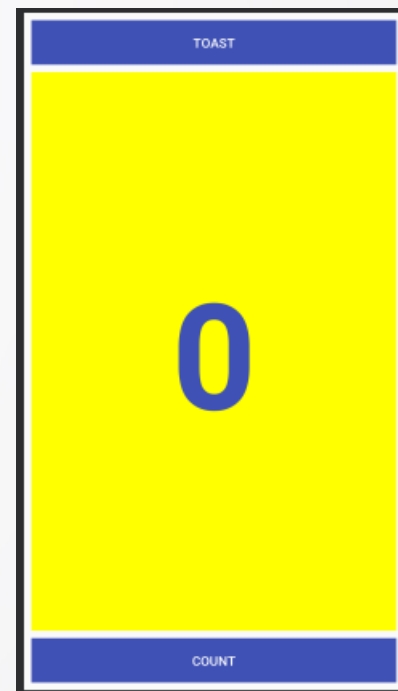
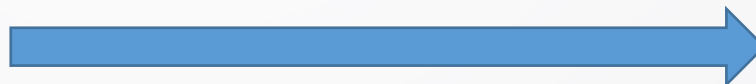
## Tâche 2.3: Modifier la mise en page en XML



- Extraire les valeurs de type texte écrit par des ressources dans le fichier strings.xml
- Extraire aussi chaque valeurs de couleur par des ressources dans le fichier colors.xml

```
17 android:text="Toast"
18 android:textColor="@android:color/white"
19 Hardcoded string "Toast", should use @string resource more... (⌘F1)
20 app:layout_constraintStart_toStartOf="parent"
21 app:layout_constraintTop_toTopOf="parent" />
```

A ce stade, vous devriez avoir une interface qui ressemble a cette figure





## Tâche 3.1: Ajouter les onClick handlers



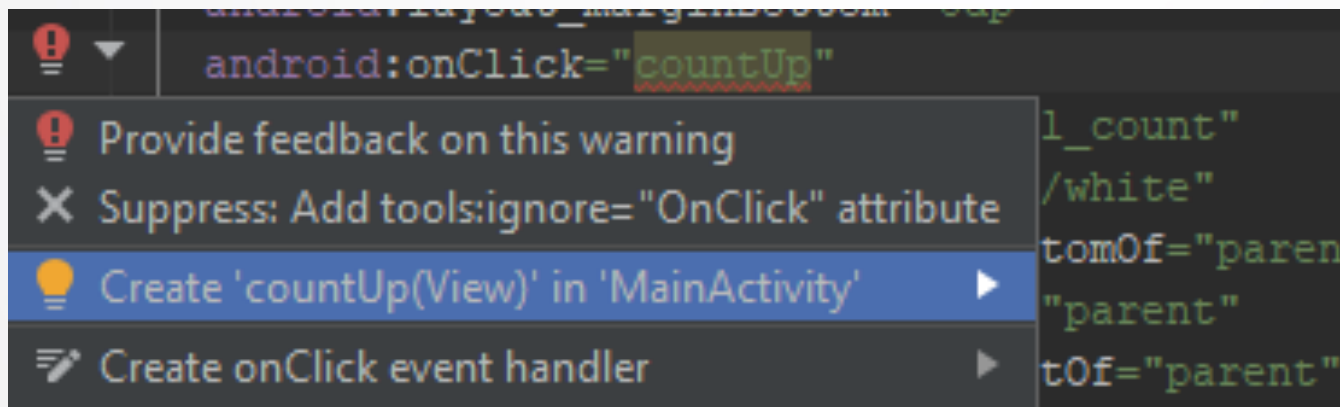
- Nous allons maintenant ajouter une méthode Java pour chaque bouton dans `mainActivity.java` qui s'exécute lorsque l'utilisateur tape sur le bouton.
  - 1) Ajoutez `android:onClick="showToast"` dans le code XML du 1<sup>er</sup> bouton
  - 2) Ajoutez `android:onClick="countUp"` dans le code XML du 2<sup>eme</sup> bouton



## Tâche 3.1: Ajouter les OnClick handlers



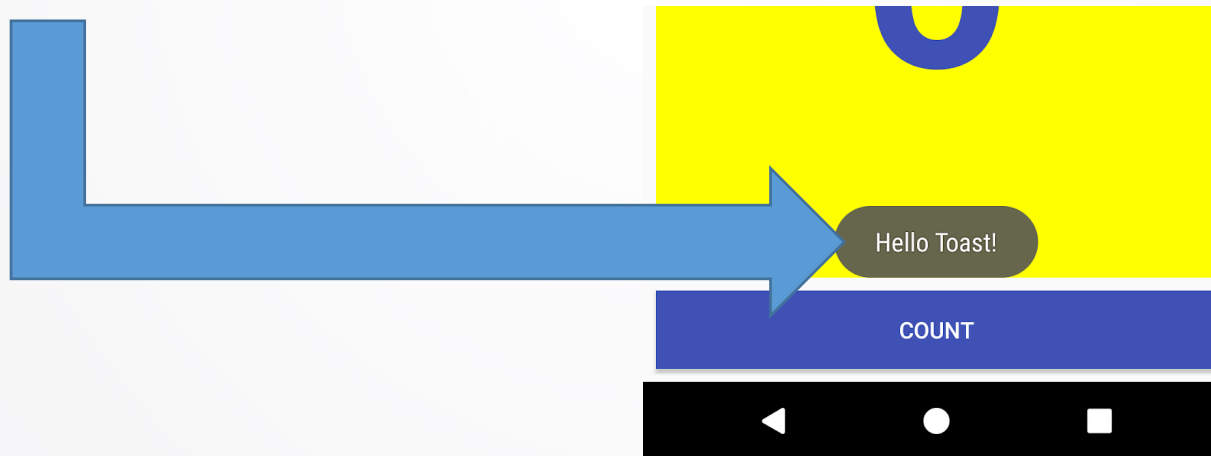
- Vous allez remarquer que c'est une erreur car les méthodes n'existent pas dans le code java, un click sur l'icone de l'ampoule peut résoudre le problème, ou vous pouvez créer les méthodes manuellement.



```
public void countUp(View view) {  
  
}
```

## ► Tâche 3.2: Modifier MainActivity.java

- Vous allez maintenant éditer la `showToast()` méthode
  - Un `Toast` fournit un moyen d'afficher un message simple dans une petite fenêtre contextuelle.



## ► Tâche 3.2: Modifier MainActivity.java

- 1) Pour créer une instance de Toast, appelez la `makeText()` méthode de la Toast classe.
- 2) Fournir le `contexte` de l'activité.
- 3) Fournissez le message à afficher.
- 4) Indiquez une durée pour l'affichage.  
`Toast.LENGTH_LONG` / `Toast.LENGTH_SHORT`
- 5) Montrer le `Toast` en appelant `show()`.

```
public void showToast(View view) {  
    Toast toast = Toast.makeText(this, R.string.toast_message,  
                                Toast.LENGTH_SHORT);  
    toast.show();  
}
```

## ► Tâche 3.2: Modifier MainActivity.java

Vous allez maintenant éditer la `countUp()` méthode mais d'abord:

- 1) Créer une variable pour le compteur.
- 2) Créer une variable de référence du `TextView` « `show_count` »

```
public class MainActivity extends AppCompatActivity {  
    private int mCount = 0;  
    private TextView mShowCount;  
}
```

## ► Tâche 3.2: Modifier MainActivity.java

- 3) Maintenant on peut obtenir une référence du `TextView` à l'aide de l'ID. Il faut obtenir cette référence une seule fois, spécifiez-la dans la méthode `onCreate()` avec `findViewById` qui prend l'identifiant d'une vue en tant que paramètre et le renvoie vers la `View`.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mShowCount = (TextView) findViewById(R.id.show_count);
}
```

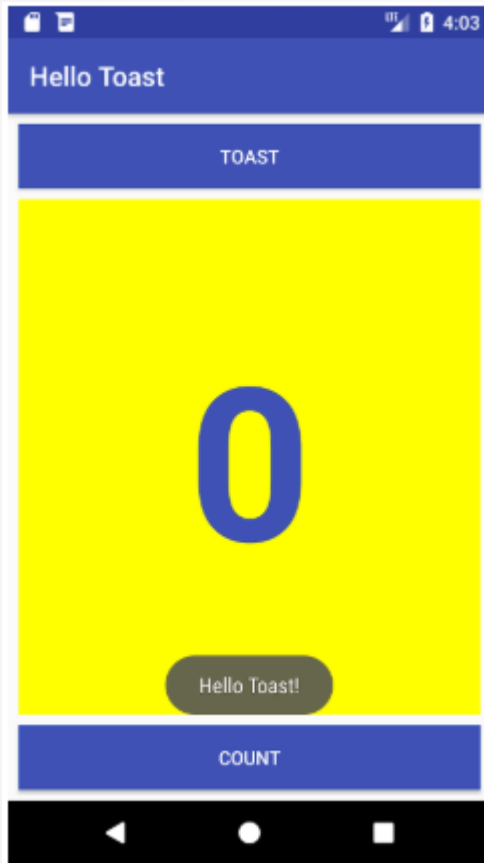


## ► Tâche 3.2: Modifier MainActivity.java

- 4) Maintenant que vous avez affecté le `TextView` à `mShowCount` vous pouvez utiliser la variable pour définir la valeur de `mCount` dans le `TextView`.
- 5) Ajoutez les éléments suivants à la `countUp()` méthode:

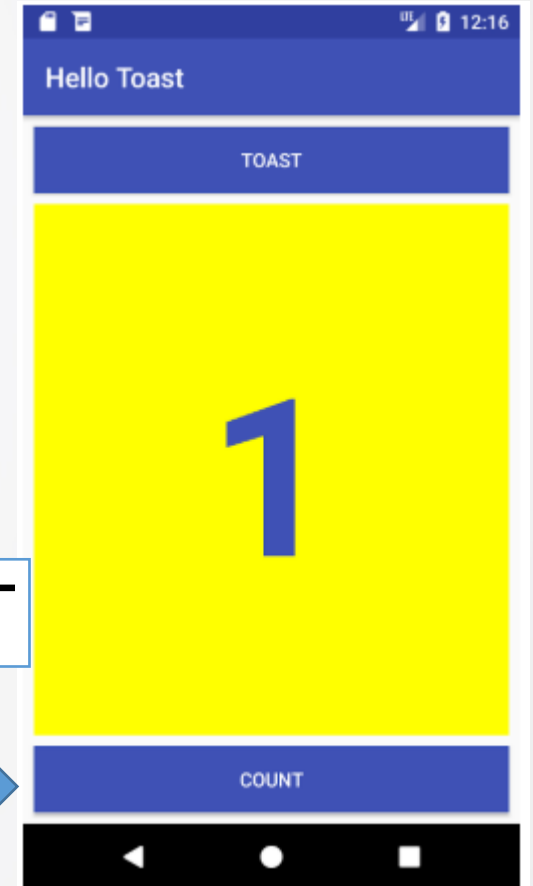
```
public void countUp(View view) {  
    ++mCount;  
    if (mShowCount != null)  
        mShowCount.setText(Integer.toString(mCount));  
}
```

## ► Tâche 4: Lancer l'application



Cliquer sur le bouton TOAST

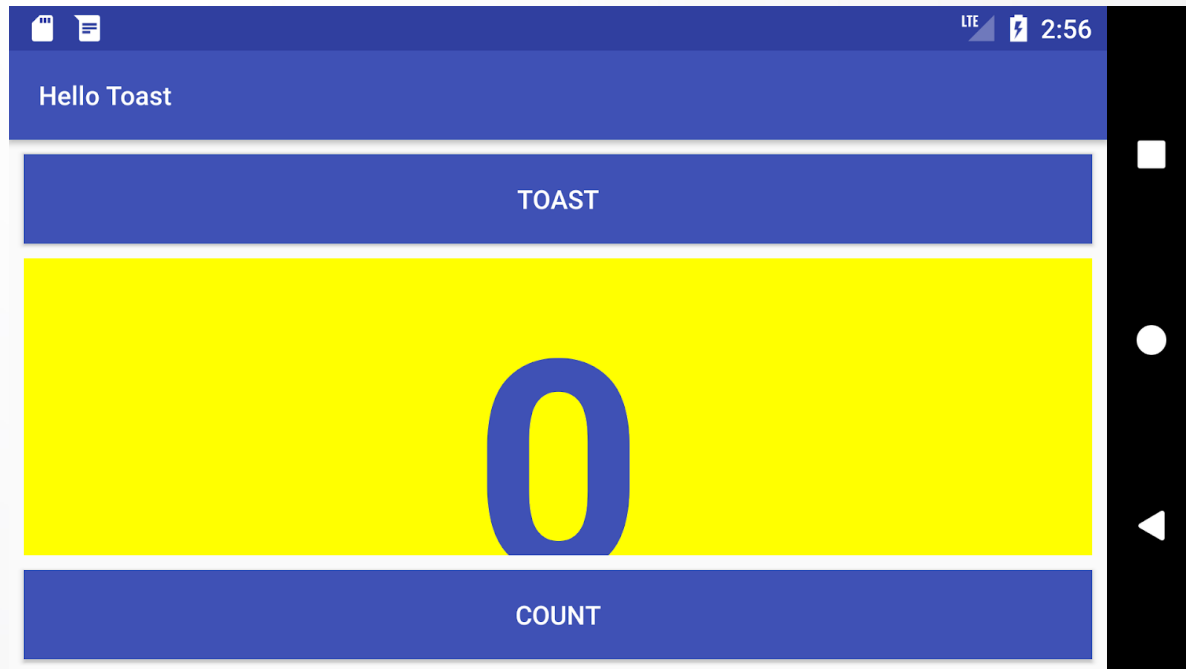
Cliquer sur le bouton COUNT




# ▶ Coding challenge



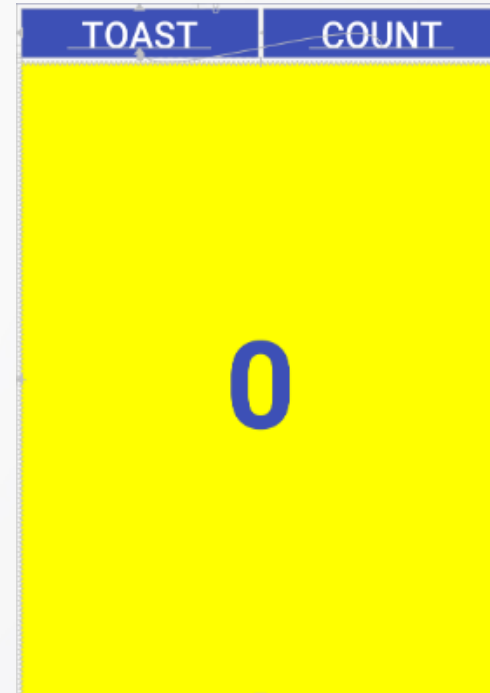
- Copiez le dossier du projet **HelloToast** et renommez-le en **HelloToastChallenge**
- Modifiez la mise en page de sorte qu'elle soit bien dans les orientations horizontale et verticale.





# ► Coding challenge

- Changer la mise en page afin que le **Toast Button** et le **Count Button** seront en haut, comme le montre la figure ci-dessous en utilisant `LinearLayout` ou `RelativeLayout`



# ► Layouts



- Vous avez maintenant manipuler les layouts et vous devez retenir:
  - Comment utiliser l'éditeur du layout
  - Réglage de la largeur et de la hauteur du Layout
  - Extraction des ressources de texte
  - Traitement des clics
  - Affichage des messages Toast

