TP1 Programmation sous Android

Codage d'une calculatrice simple

IMT Lille Douai

1 L'interface graphique

L'objectif de ce TP est de réaliser une calculatrice basique (4 opérations, sans mémoire).
L'interface graphique se compose :

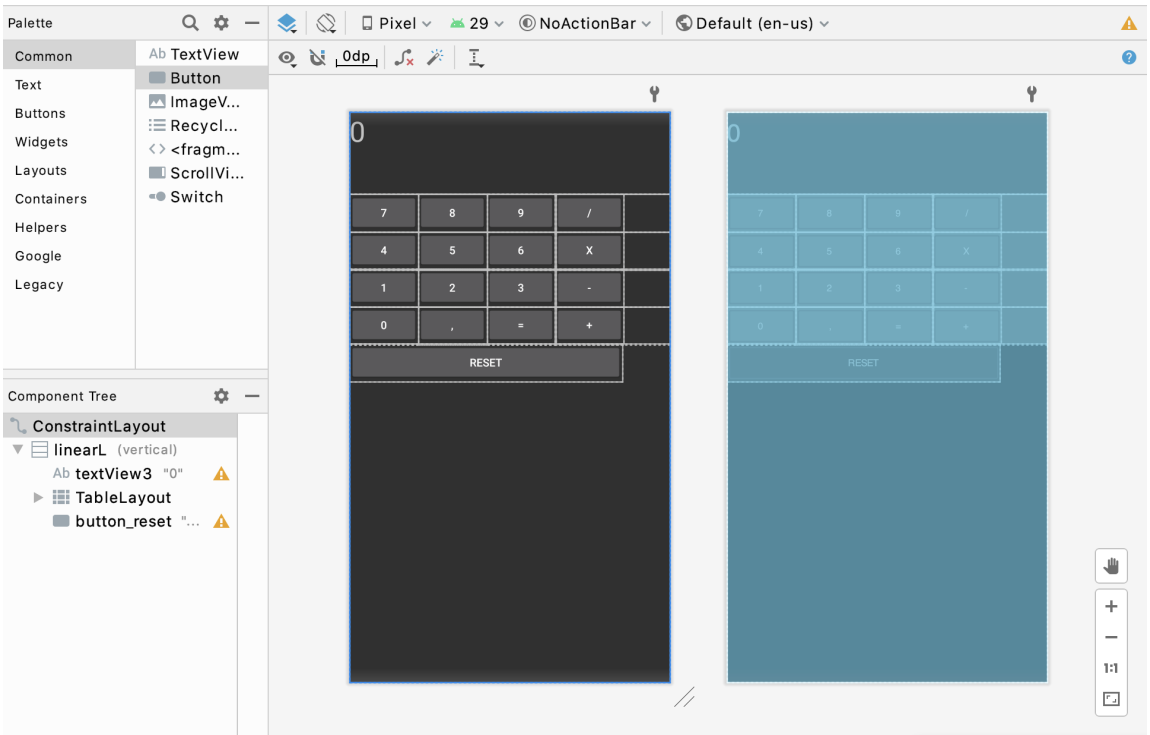
— d'un `TextView` permettant de visualiser les opérations demandées et le résultat ;
— et de 17 `Button` reprenant les chiffres de 0 à 9, la virgule, le signe égal, les 4 opérateurs arithmétiques, ainsi qu'un bouton RESET permettant d'effacer l'opération en cours.

Pour construire l'interface graphique, vous pouvez :

— utiliser une approche purement programmatique en déclarant chaque composant en syntaxe java (ne pas oublier le `new` puisqu'il s'agit de création d'objets à chaque fois)
— utiliser l'outil de composition d'interface : après avoir créé votre activité, cliquer sur le fichier xml layout de votre activité (`activity_main.xml` si votre activité s'appelle `MainActivity.java`), puis sélectionner la vue Design dans le coin supérieur droit.

Pour agencer les composants graphiques les uns par rapport aux autres, vous avez plusieurs possibilités :

— imbrication de `LinearLayout` verticaux dans un `LinearLayout` horizontal
— insertion d'un `TableLayout` dans un `LinearLayout` horizontal (voir figure ci-dessous).



2 La gestion des opérations et leur calcul

Vous pouvez déclarer dans votre activité (ou dans une classe dédiée), 3 attributs :

— un attribut `op1` représentant le premier opérande de l'opération à réaliser
— un attribut `op2` représentant le second
— un attribut operateur représentant l'une des 4 opérations arithmétiques
— un attribut res représentant le résultat de l'opération

Pour faciliter la mise à jour de l'affichage, vous pouvez considérer ces attributs sous forme de String et ne les convertir qu'au moment du calcul. Celui-ci sera à réaliser dans 2 cas : lorsque l'utilisateur appuie sur le bouton = ou lorsque l'utilisateur enchaine son calcul avec une autre opération (boutons +, x, -, /). La mise à jour de ces attributs est à faire à l'intérieur des méthodes onClick associées aux boutons. Avant de coder, réfléchissez au brouillon aux différents cas pouvant se présenter pour la mise à jour de ces attributs en fonction des différents boutons. Par exemple :

— si `op1` contient la chaine `''12''`, que operateur est vide et que l'utilisateur appuie sur un des boutons de 0 à 9, alors la valeur correspondante devra être concaténée à `op1`
— si `op1` contient la chaine `''12''` et que operateur est égal à `''+''` alors la valeur du prochain bouton touché par l'utilisateur sera affectée à `op2`
— etc

3 Pour aller plus loin

Vous pouvez ensuite gérer les "mauvaises" utilisations de l'application afin d'éviter les plantages et calculs erronés. Par exemple, vous pouvez empêcher l'utilisateur de toucher de fois d'affilée un bouton opérateur (le bouton peut être même désactivé dans l'interface via la méthode `setEnabled`).