Université Hassan II de Casablanca
Ecole Nationale Supérieur de l'Enseignement
Technique Mohammedia

Ingénierie Multimédia

Développement Mobile Natif - Android

Compte rendu

TP 1 : Codez votre propre calculatrice

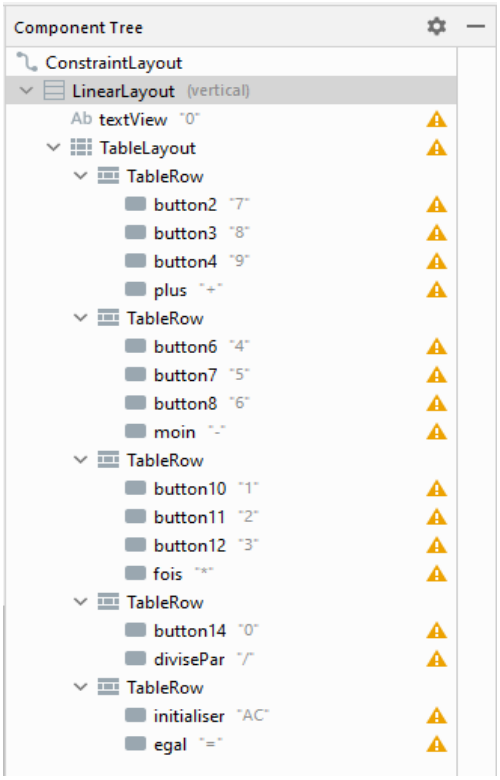
Réalisé par: **AHADJANE Mohcine** **GLSID2**

But

Le but de ce TP est la réalisation d'une application qui permet de mettre en pratique la gestion des événements avec une interface graphique "une calculaculatrice bi-operandes".

Partie I : L'interface graphique de l'application

L'organisation générale demandée peut se décomposer en un texte en haut de l'écran et un tableau de boutons en bas. En utilisant le `LinearLayout` avec une orientation verticale, on peut donc placer un `TextView` en haut de l'écran sur l'intégralitéde la largeur avec un texte aligné à droite ainsi qu'un `TableLayout` en bas. Il faut mentionner que les cases du tableau sont redimensionnables pour occuper un certain nombre de cases si nécessaire.



Partie II : Code JAVA

Au niveau du comportement, on se rend compte que pour faire des opérations binaires (avec deux opérandes), il faudra mémoriser deux opérandes et l'opération effectuée. L'action du bouton égal (=) sera celle qui fera le calcul. Il faut aussi mémoriser si on est en train de saisir le premier ou le second opérande. Ainsi, lancer le calcul ne correspondra qu'à faire l'opération demandée entre les 2 opérandes en mémoire, stocker le résultat en premier opérande et mettre à jour l'affichage

1. Déclarer les attributs nécessaires au traitement dans la classe *MainActivity* : *Int valeur1, Int valeur2, String operation* et *Boolean isOp1* (qui teste si on est sur le premier ou le deuxième opérande).

```
public class MainActivity extends AppCompatActivity {
    private int valeur1=0;
    private int valeur2=0;
    private String operation="";
    private Boolean isOp1=true;
    private TextView textView;
```

2. Ecrire une méthode **void afficher()** qui affiche l'opérande en cours de saisie. Utiliser la (si nécessaire) dans les méthodes qui suivent.

```
public void afficher() {
    if (!isOp1) {
        textView.setText(String.valueOf(valeur1) + " " + operation +
String.valueOf(valeur2));
    } else textView.setText(String.valueOf(valeur1));
}
```

1. Ecrire une méthode **void setOperator(View view)** pour gérer les 4 boutons d'opérateurs (+, -, * et /). Cette méthode devra prendre une *View* en paramètre et retrouver le bouton d'opération qui a été pressé pour agir en conséquence. Pour ce faire, vous pouvez utiliser la méthode *getId()* présente dans toutes les vues et qui permet de récupérer l'identifiant de la vue qui a généré l'événement.

```
public void setOperator(View view){
    switch (view.getId()){
        case R.id.plus: operation="+";      break;
        case R.id.moin: operation="-";      break;
        case R.id.fois: operation="*";      break;
        case R.id.divisePar: operation="/"; break;
        default: return;
    }
    isOp1=false;
    afficher();
}
```

2. Ecrire une méthode **void ajouterNbr(View view)** qui gère les boutons associés aux chiffres par conversion numérique de la valeur présente dans le texte du bouton (ajouter un digit à un opérande revient ensuite à multiplier la valeur actuelle de l'opérande par 10 et ajouter à y la valeur du nouveau digit).

```
public void ajouterNbr(View view){
    int val = Integer.parseInt(((Button)view).getText().toString());
    if(isOp1){
        valeur1 = valeur1*10+val;
    }
    else {
        valeur2=valeur2*10+val;
    }
    afficher();
}
```

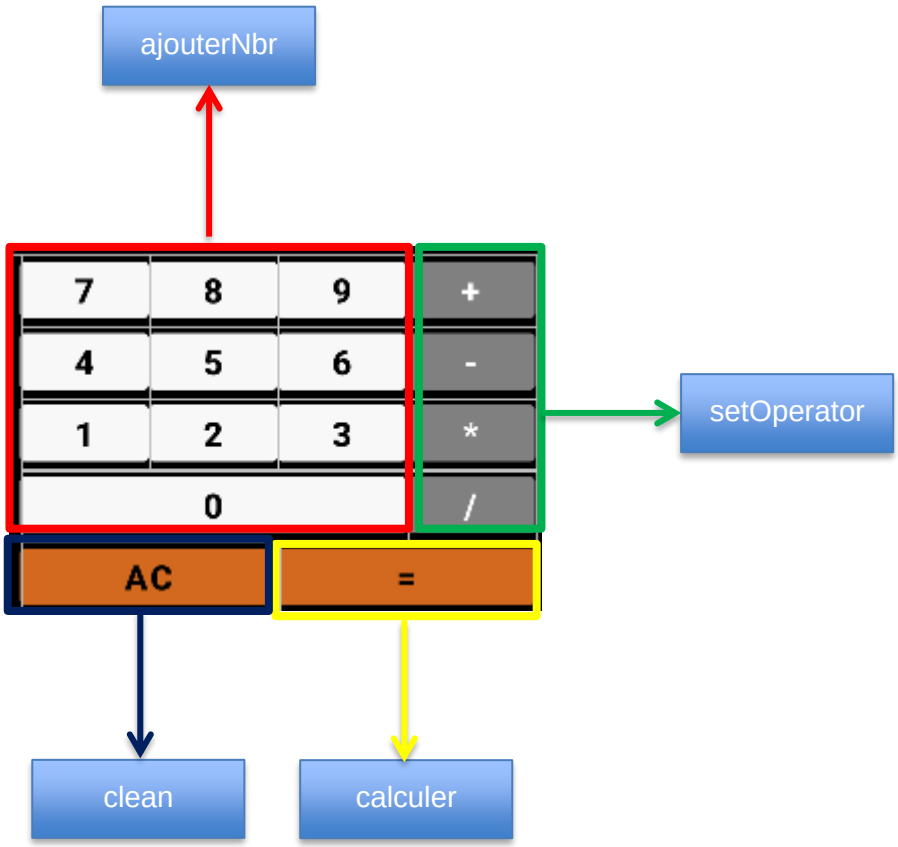
3. Ecrire une méthode **void calculer(View view)** qui effectue l'opération demandée entre les deux opérandes, stocker le résultat en premier opérande et mettre à jour l'affichage.

```
public void calculer(View view){
    if (!isOp1){
        switch (operation){
            case "+": valeur1=valeur1+valeur2;  break;
            case "-": valeur1=valeur1-valeur2;  break;
            case "*": valeur1=valeur1*valeur2;  break;
            case "/": valeur1=valeur1/valeur2;  break;
            default: return;
        }
        valeur2=0;
        isOp1=true;
        afficher();
    }
}
```

4. Ecrire une méthode **void clean()** qui initialise l'affichage.

```
public void clean(View view){
    valeur1=0;
    valeur2=0;
    isOp1=true;
    operation="";
    afficher();
}
```

5. Associer ces méthodes aux boutons concernés dans le fichier XML.



Partie III: Simulation