

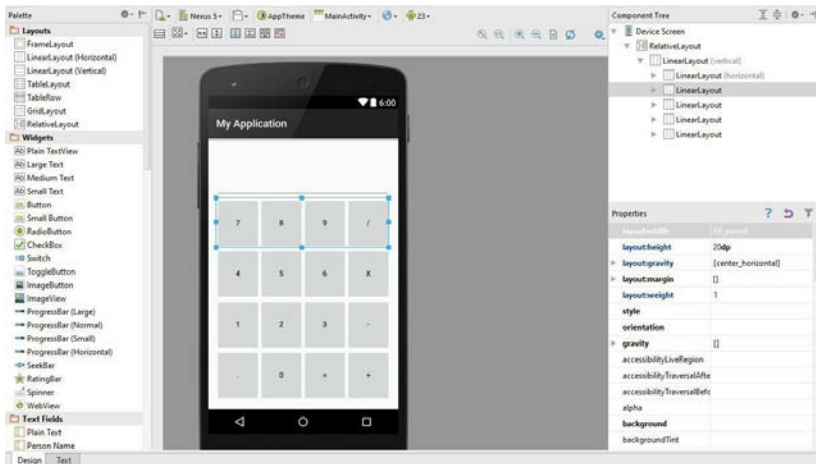
Developper des Applications Android

SEG2505 - Introduction au génie logiciel – Automne 2023

Exercice de laboratoire - Simple Calculator



Travail avec ton interface d'utilisateur

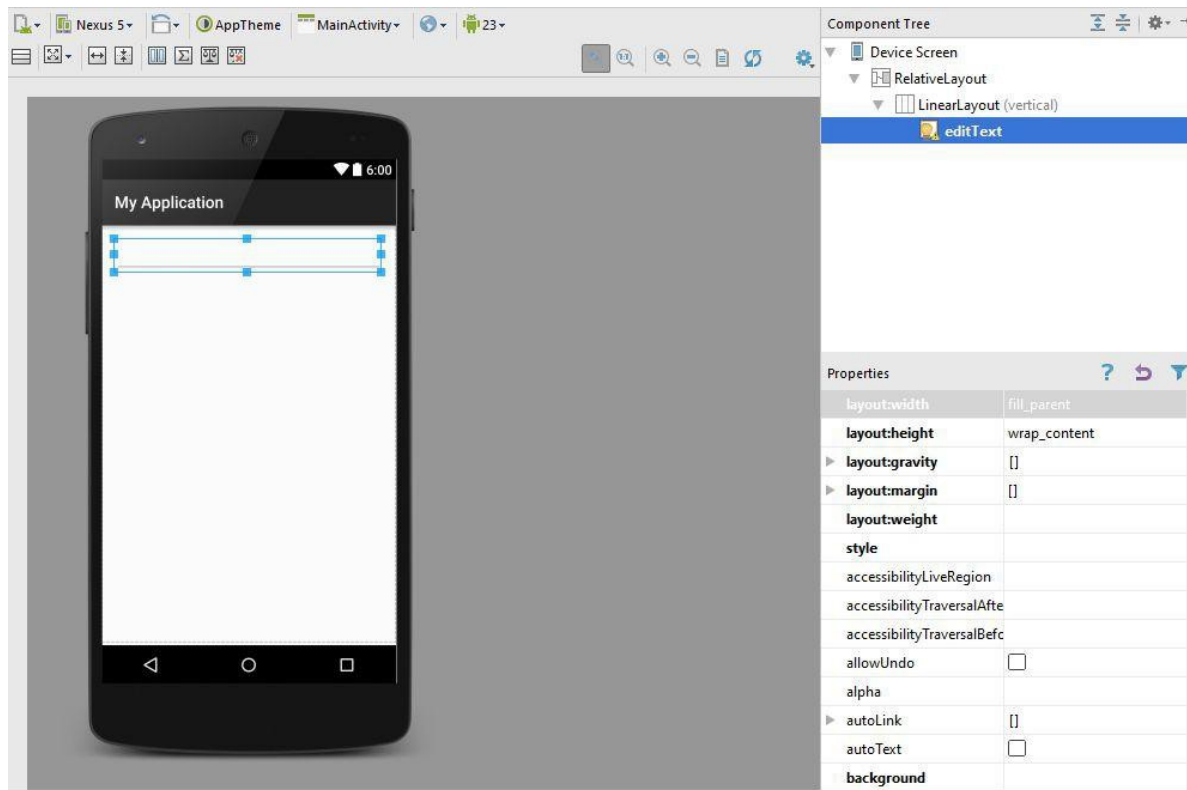


- L'objectif de cette slide est de vous apprendre à utiliser les outils disponible
- Votre application android n'a pas besoin de ressembler exactement à celle de l'exemple, mais elle a besoin de fonctionner comme une calculatrice
- Pour ajouter un nouvel objet à votre interface, prenez un composant à partir de la "Palette" à la "Component Tree"
 - Ne prenez jamais un composant du preview, TOUJOURS du "Component Tree"
 - Si vous prenez un composant du android preview screen, cela va désorganiser tout l'interface et rendra plus dur pour vous de compléter cette slide manuellement.

Simple Calculator : Layout

1. Suivez les étapes dans l'exemple (du tutoriel 2) pour créer une application. Appelez votre application "Simple Calculator".
2. Enlevez le *TextView* "**Hello World**" (si votre layout n'est pas vide)
3. Ajoutez un "**EditText**" a votre "**MainLayout**" et changer son id en "**resultEdit**"
 - *Pour modifier une propriété dans un objet vous pouvez double cliquez sur ce dernier dans le "preview panel" ou dans le "properties table". Comme ça, vous serez capable de changer les propriétés de l'ID de vos composants*
4. Ajoutez un "Layout" (Verticale) a votre hierarchie et déplacez "**resultEdit**" afin qu'il soit en dessous dans la hiérarchie
 - Assignez l'ID "**mainlayout**" a 'VerticalLayout'.
5. Changez la propriete **ResultEdit's** "layout_width" a **fill_parent**.
 - *Cela permettra à votre zone de texte d'occuper toute la largeur de l'écran.*

Simple Calculator : Résultat de l'étape 1



Simple Calculator : Boutons

1. Ajoutez un **HorizontalLayout** au **VerticalLayout** et appelez le "ButtonRow01"

1. Ajoutez 4 boutons à l'HorizontalLayout que vous venez de créer

2. Renommez chaque boutons de l'ID avec un nom "**btn0X**" appropriée

1. *(e.x.: : Le bouton 1 sera appelé "btn01" et le bouton = "buttonEQ")*

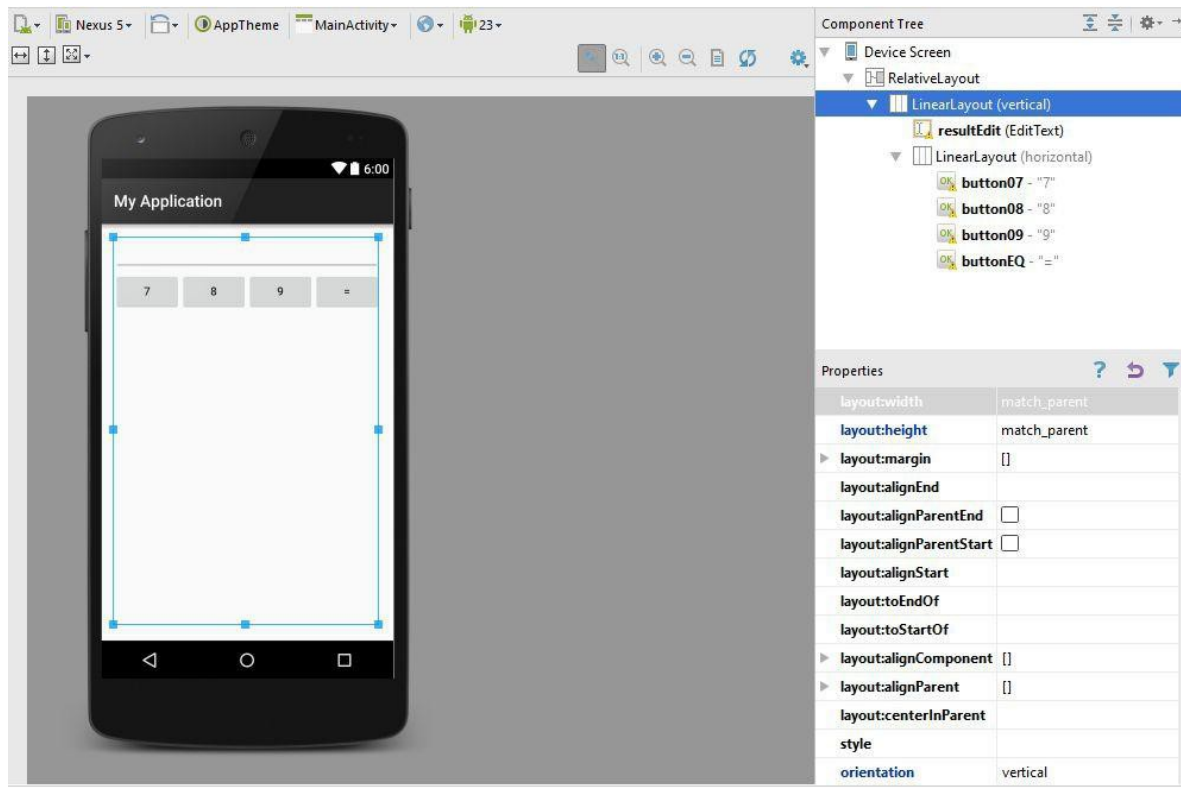
A ce niveau, vos boutons devraient sembler déséquilibré, avec des items compressés. C'est parce que certains boutons prennent tout l'espace disponible sur l'écran.

Pour remédier à ça, faites ce qui suit:

1. Changez la propriété 'layout_weight' de l'*HorizontalLayout* à 1. Faites la même chose pour tous les boutons.

2. Maintenant, tous les boutons devraient occuper le même espace sur l'écran peu importe le texte qu'ils contiennent.

Simple Calculator : Résultat de l'étape 2



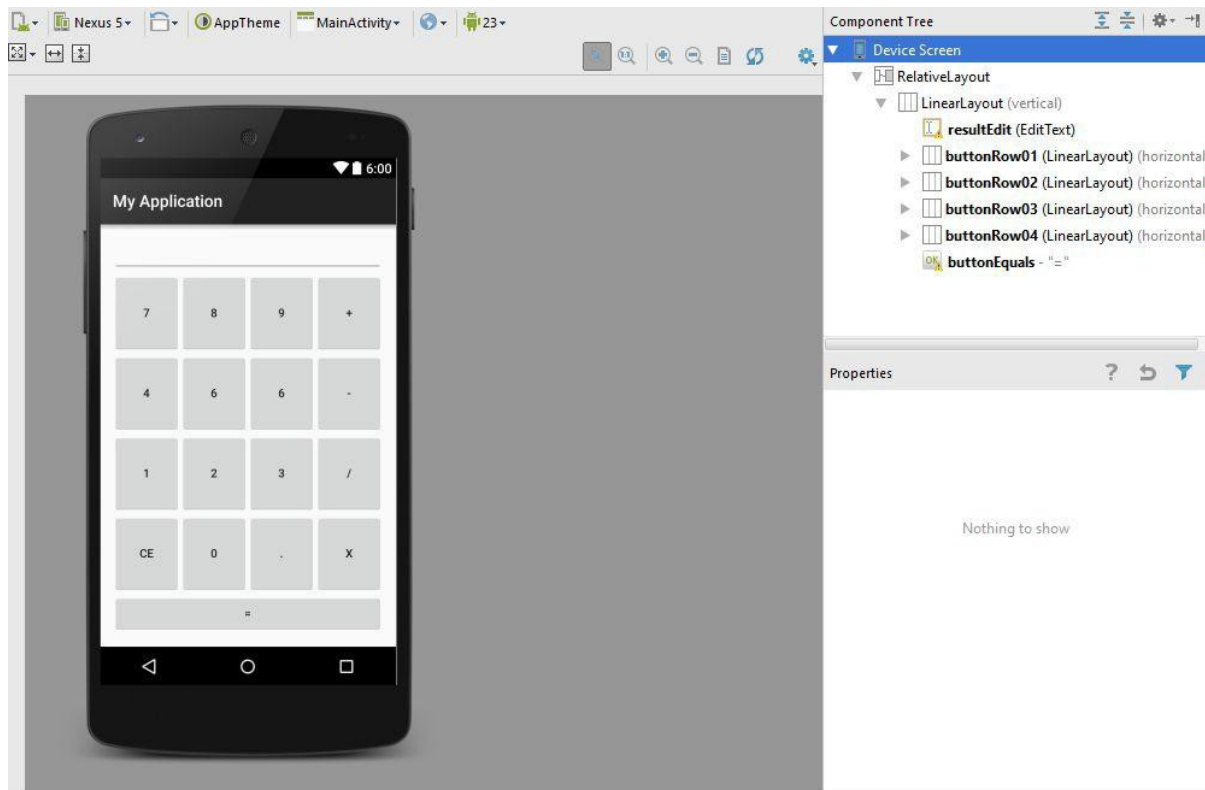
Simple Calculator : Plus de buttons

1. Refaites les étapes de la précédente slide 3 fois pour créer une matrice de 4x4 boutons

Vous noterez que dans la prochaine rangée de boutons sera également aplati, mais cette fois ci vers le bas. Pour résoudre cela, donner à l'HorizontalLinearLayout le même weight (layout:weight = 1) dans le "properties panel".

2. Ajoutez un bouton en extra en bas du plus bas HorizontalLinearLayout (ca sera le bouton "=")
 1. Définissez le **layout:width** a "**fill_parent**".
 2. Pour remplir n'importe quel espace verticale vide entre les boutons définissez leur propriété **layout:height** a "**fill_parent**"

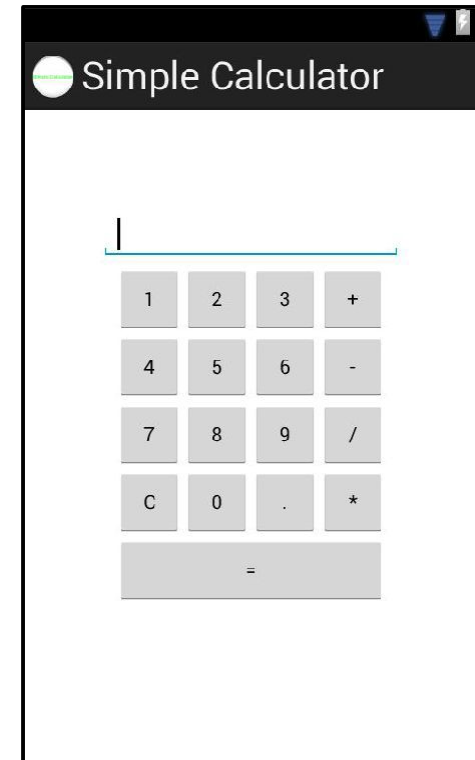
Simple Calculator : Résultat de l'étape 3



Votre design final ne doit pas exactement ressembler à celui de l'exemple du moment ou vous avez inclus toutes les fonctions basiques.

Simple Calculator : Encore des Buttons

- Changez le texte de tous les boutons comme vous pouvez le voir dans la photo
- Changez l'ID de tous les boutons comme suit:
 - Le numéro de l'ID devrait commencer par "btn0" + son numéro. Par exemple l'ID du bouton 1 devrait être "btn01", pour le bouton 0 "btn00" ainsi de suite.
 - Pour les autres, regardez le tableau sur le côté de l'écran

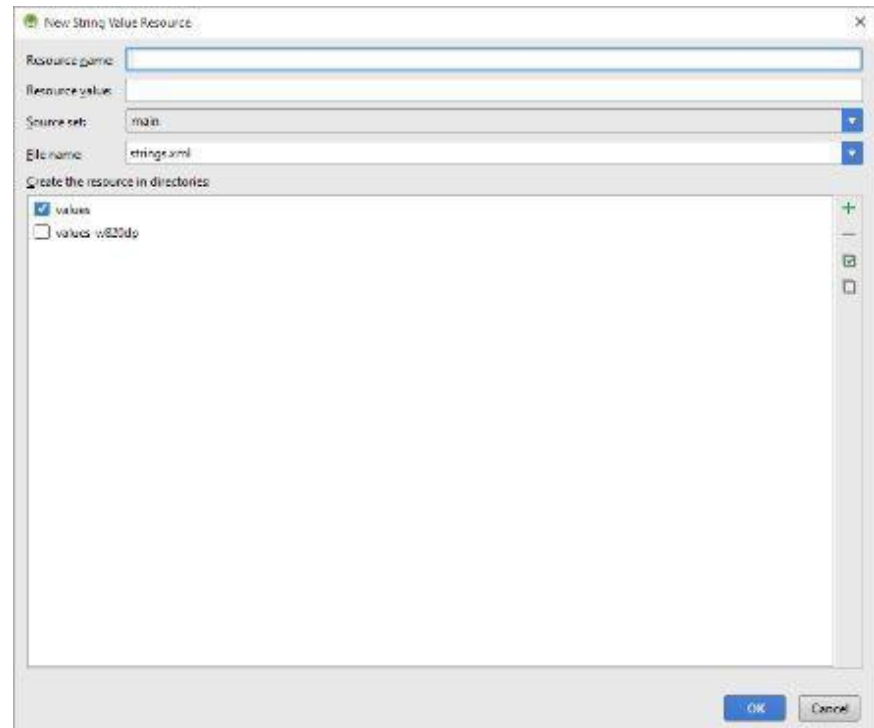
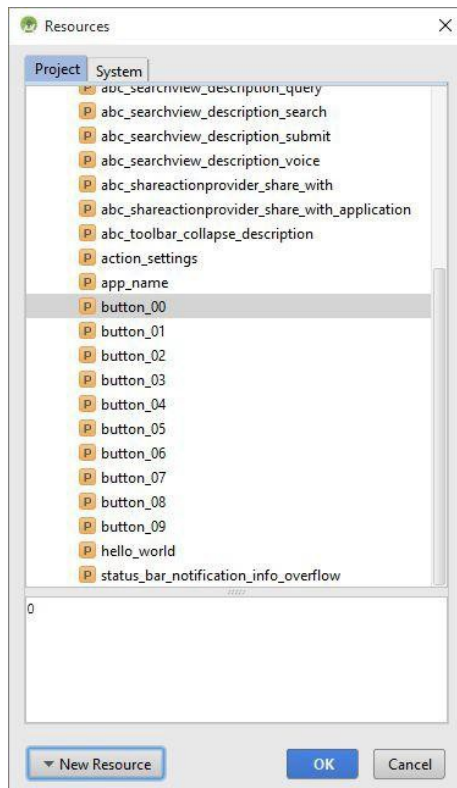


Symbol	ID
"C"	btnClear
"."	btnDot
"+"	btnAdd
"-"	btnMinus
"*"	btnMultiply
"/"	btnDivide
"="	btnResult

Simple Calculator : Strings

Pour separer les strings du layout code, suivez ces étapes

1. Sélectionnez un bouton dans la section "**component tree**" et cliquez dessus.
2. Allez jusqu'à la propriété "**Text**" dans le "properties panel"
3. Sélectionnez la propriété "**Text**" et cliquez sur le bouton a trois points
4. Sélectionnez "**New Resource**" et ensuite "**New String Resource**" dans la "Resources Window".
5. Définissez un nom pour le nouveau bouton (e.x.: bouton_07), ajoutez la valeur actuelle du bouton (e.x.: 7), sauvegardez la nouvelle ressource.
6. Répétez ce processus pour tous les nombres et opérateurs.



Simple Calculator (cont.)

- Ajouter toutes les "onClick" méthodes au fichier MainActivity.java

- La signature de la méthode devra être la suivante

```
Public void <<methodName>> (View view) {  
    <YOUR CODE GOES HERE>>  
}
```

- E.x. onClick methode pour btnResult est:

```
Public void btnResultClick (View view) {  
    <YOUR CODE GOES HERE>  
}
```

- N'oubliez pas d'importer "**android.view.View**" et "**android.widget.EditText**" aussi bien que les autres classes nécessaire.
- Maintenant écrivez votre code pour chaque bouton dans votre **ACTIVITY JAVA FILE !**
 - Les prochaines slides sont des échantillons de ce à quoi votre code devrait ressembler.

Simple Calculator (code section 1)

```

activity_main.xml | MainActivity.java
package vahdat.android.simple.calculator;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends Activity {

    private enum Operator {none, add, minus, multiply, divide}
    private double data1 = 0, data2 = 0;
    private Operator optr = Operator.none;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    public void btn00Click(View view) {
        EditText eText = (EditText) findViewById(R.id.resultEdit);
        eText.setText(eText.getText() + "0");
    }
}

```

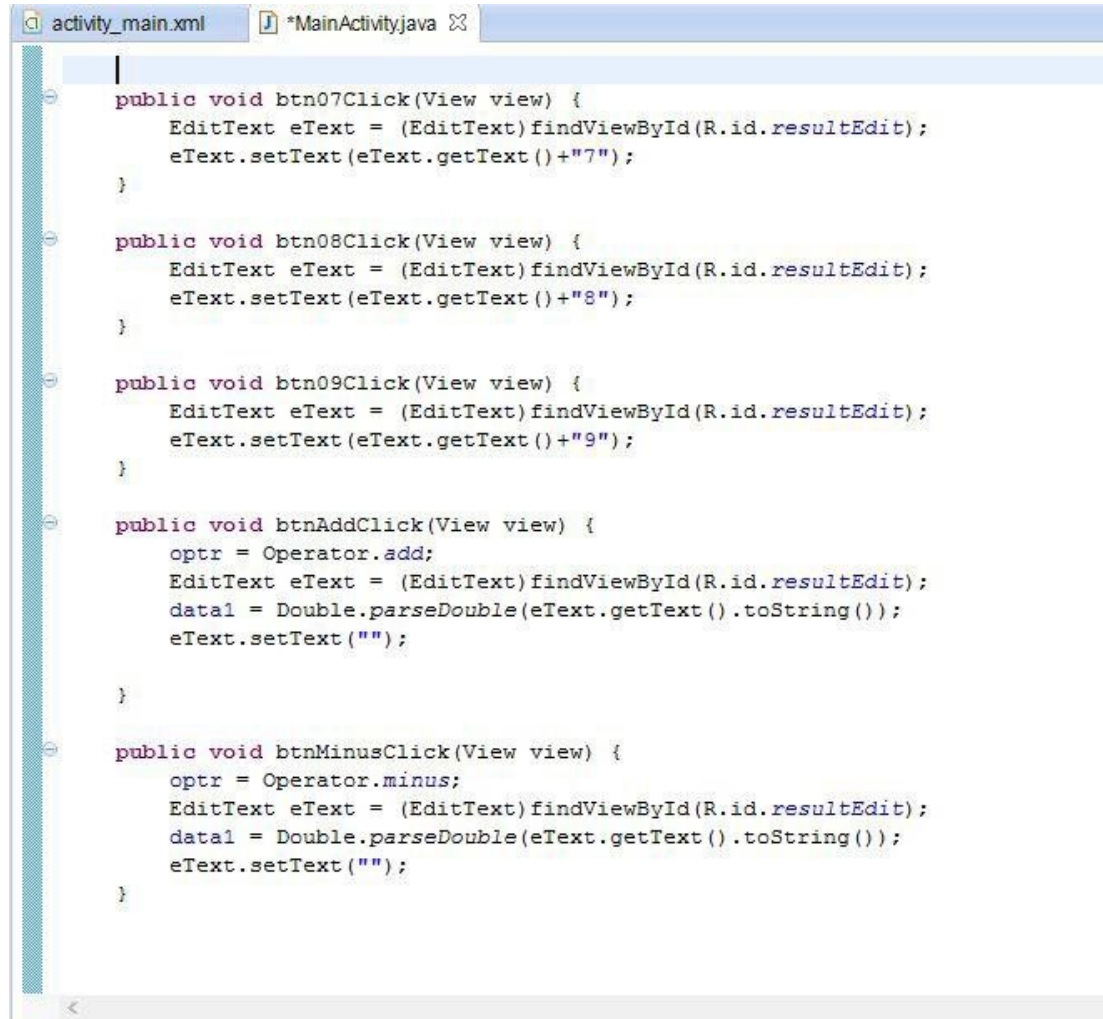
These variables were added by us.

This portion of the code is generated automatically for all activities and contains the basic constructor code.

Simple Calculator (code section 2)

```
activity_main.xml  *MainActivity.java  ⌕  
  
public void btn01Click(View view) {  
    EditText eText = (EditText)findViewById(R.id.resultEdit);  
    eText.setText(eText.getText()+"1");  
}  
  
public void btn02Click(View view) {  
    EditText eText = (EditText)findViewById(R.id.resultEdit);  
    eText.setText(eText.getText()+"2");  
}  
  
public void btn03Click(View view) {  
    EditText eText = (EditText)findViewById(R.id.resultEdit);  
    eText.setText(eText.getText()+"3");  
}  
  
public void btn04Click(View view) {  
    EditText eText = (EditText)findViewById(R.id.resultEdit);  
    eText.setText(eText.getText()+"4");  
}  
  
public void btn05Click(View view) {  
    EditText eText = (EditText)findViewById(R.id.resultEdit);  
    eText.setText(eText.getText()+"5");  
}  
  
public void btn06Click(View view) {  
    EditText eText = (EditText)findViewById(R.id.resultEdit);  
    eText.setText(eText.getText()+"6");  
}
```

Simple Calculator (code section 3)



```
activity_main.xml  MainActivity.java

public void btn07Click(View view) {
    EditText eText = (EditText)findViewById(R.id.resultEdit);
    eText.setText(eText.getText()+"7");
}

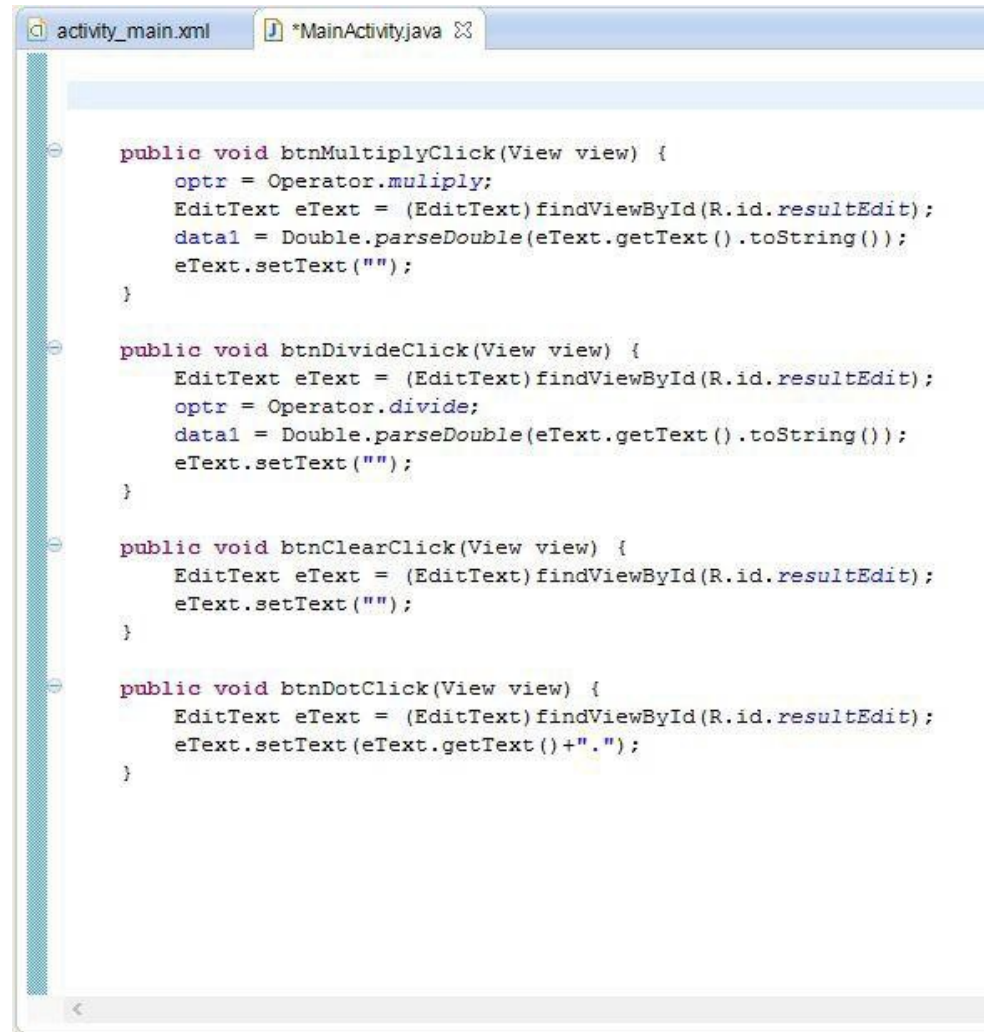
public void btn08Click(View view) {
    EditText eText = (EditText)findViewById(R.id.resultEdit);
    eText.setText(eText.getText()+"8");
}

public void btn09Click(View view) {
    EditText eText = (EditText)findViewById(R.id.resultEdit);
    eText.setText(eText.getText()+"9");
}

public void btnAddClick(View view) {
    opr = Operator.add;
    EditText eText = (EditText)findViewById(R.id.resultEdit);
    data1 = Double.parseDouble(eText.getText().toString());
    eText.setText("");
}

public void btnMinusClick(View view) {
    opr = Operator.minus;
    EditText eText = (EditText)findViewById(R.id.resultEdit);
    data1 = Double.parseDouble(eText.getText().toString());
    eText.setText("");
}
```


Simple Calculator (code section 4)



```
activity_main.xml  *MainActivity.java

public void btnMultiplyClick(View view) {
    optr = Operator.multiply;
    EditText eText = (EditText)findViewById(R.id.resultEdit);
    data1 = Double.parseDouble(eText.getText().toString());
    eText.setText("");
}

public void btnDivideClick(View view) {
    EditText eText = (EditText)findViewById(R.id.resultEdit);
    optr = Operator.divide;
    data1 = Double.parseDouble(eText.getText().toString());
    eText.setText("");
}

public void btnClearClick(View view) {
    EditText eText = (EditText)findViewById(R.id.resultEdit);
    eText.setText("");
}

public void btnDotClick(View view) {
    EditText eText = (EditText)findViewById(R.id.resultEdit);
    eText.setText(eText.getText()+".");
}
```


Simple Calculator (code section 5)



```
activity_main.xml  *MainActivity.java X

public void btnResultClick(View view) {
    if (optr != Operator.none){
        EditText eText = (EditText)findViewById(R.id.resultEdit);
        data2 = Double.parseDouble(eText.getText().toString());
        double result = 0;
        if (optr == Operator.add) {
            result = data1+data2;
        } else if (optr == Operator.minus) {
            result = data1-data2;
        } else if (optr == Operator.muliply) {
            result = data1*data2;
        } else if (optr == Operator.divide) {
            result = data1/data2;
        }
        optr = Operator.none;
        data1 = result;
        if((result-(int)result)!=0)
            eText.setText( String.valueOf(result));
        else
            eText.setText( String.valueOf((int)result));
    }
}
```

Simple Calculator : Input Events

- Chaque view a son propre Events (e.x. onTouchEvenet()) et devra être réécrite pour avoir votre propre implémentation.
- Au lieu de d'étendre une view pour avoir le control events, utilisez chaque listeners.
- Un event listener est une interface dans la classe View qui contient une callback méthode unique. Ces methodes peuvent etre appeles par l'Android framework.
 - onClick()
 - onLongClick()
 - onFocusChange()
 - onKey()
 - onTouch()
 - onCreateContextMenu()

Simple Calculator : Les fonctions boutons

Pour que vos boutons puissent exécuter le code en java, ils ont besoin de savoir quelle fonction ils doivent appeler.

Pour le faire :

- Ajoutez la propriété **onClick** a tous les boutons.
 - Dans “Design view” définissez la propriété onClick dans le “properties panel” au nom de la fonction correspondante dans votre code.
 - E.x.: la fonction **btn01click()** correspond au **button01** donc la propriété **onClick** devrait lire “**btn01click**”
 - Si vous regardez la version texte de l’interface, vous verrez quelque chose comme : *android:onClick="btn01Click"*
 - *Cela signifie que le button01 a la bonne fonction onClick.*
- Répétez ce processus pour tous les boutons numériques et les opérateurs !

AU SECOURS ! Mon code ne fonctionne pas.

- Les lignes ondulées signifie que votre expression ne peut être compiler
- 9 fois sur 10, il y'a un **TYP**O dans le code
 - **onClick()** est different de **onclick()**
 - **button01** est different de **Button01** et de **button_01**
- N'oubliez pas d'inclure les librairies que vous appelez!
 - Les textes en **ROUGE** signifie que Android Studio ne sait pas quelle classe vous voulez, probablement à cause d'un manquement
 - Alt+Enter sur les manquements importe automatiquement la classe dont il pense que vous avez besoin.
- Les lampes **jaunes** et **rouges** vous donnent des astuces sur quoi faire.

Améliore ton code !

- Précédemment, chaque bouton avait sa propre fonction mais il pourrait être intéressant de réduire le nombre de fonctions redondantes. Cela peut être fait en assignant la même fonction a plusieurs boutons
- Pour lutter contre ce changement, tout ce dont vous avez à faire est de vérifier quel bouton était appuyé à l'intérieur de la fonction onClick
- `View.getID()` retourne le nombre entier qui représente l'item, qui peut être associée à une ID dans l'ID resource list

```
//Function called every time a number button is pressed
public void onClickNumericalButton(View view) {
    //Getting ID of pressed Button
    int pressID = view.getId();

    //Getting Text object where we display the current number value
    TextView curText = (TextView)findViewById(R.id.resultText);

    //Figuring out which button was pressed and updating the represented text field object
    switch (pressID) {
        case R.id.button00:
            curText.setText(curText.getText() + "0");
            break;
        case R.id.button01:
            curText.setText(curText.getText() + "1");
            break;
        case R.id.button02:
            curText.setText(curText.getText() + "2");
            break;
        case R.id.button03:
            curText.setText(curText.getText() + "3");
            break;
        case R.id.button04:
            curText.setText(curText.getText() + "4");
            break;
        case R.id.button05:
            curText.setText(curText.getText() + "5");
            break;
        case R.id.button06:
            curText.setText(curText.getText() + "6");
            break;
        case R.id.button07:
            curText.setText(curText.getText() + "7");
            break;
        case R.id.button08:
            curText.setText(curText.getText() + "8");
            break;
        case R.id.button09:
            curText.setText(curText.getText() + "9");
            break;
        case R.id.buttonDot:
            curText.setText(curText.getText() + ".");
            break;
        default:
            curText.setText("ERROR");
            Log.d("Error", "Error: Unknown Button pressed!");
            break;
    }
}
```