

Aula XIV e XV de Desenvolvimento de Aplicativo I

Revisão de Layout

Continuação do projeto Calculadora

Classe R - Android

Método findViewById();

Listener de Eventos

Métodos de Eventos

Fontes:

<https://developer.android.com/training/basics/firstapp?hl=pt-br>

<https://developer.android.com/training/basics/firstapp/building-ui?hl=pt-br>

<https://www.devmedia.com.br/tutorial-de-android-studio/34003>

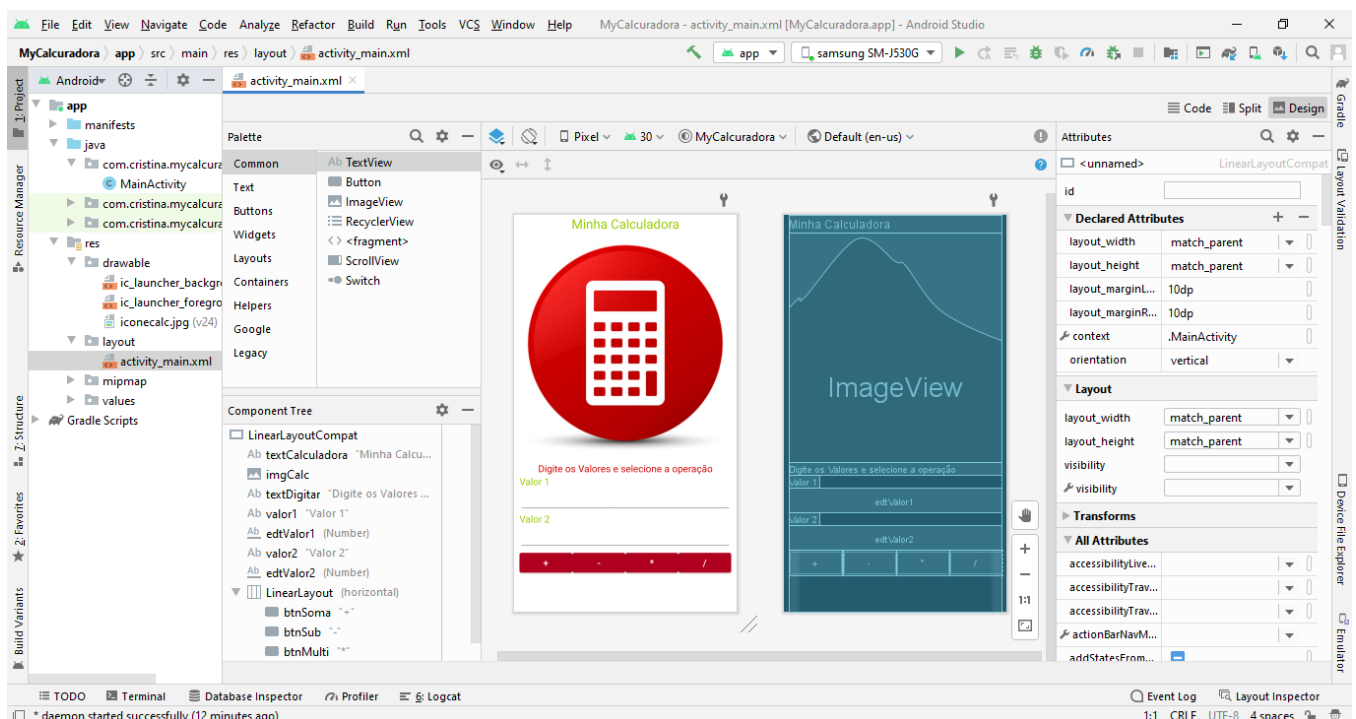
<https://developer.android.com/guide/topics/resources/providing-resources?hl=pt-br>

<https://tableless.com.br/manipulando-views-com-android/>

<https://developer.android.com/guide/topics/ui/ui-events?hl=pt-br#:~:text=Um%20listener%20de%20eventos%20C3%A9.com%20o%20item%20na%20IU.>

Continuação do projeto Calculadora

Vamos dar continuidade ao projeto da calculadora, na aula de ontem, finalizamos o nosso tutorial da seguinte forma:

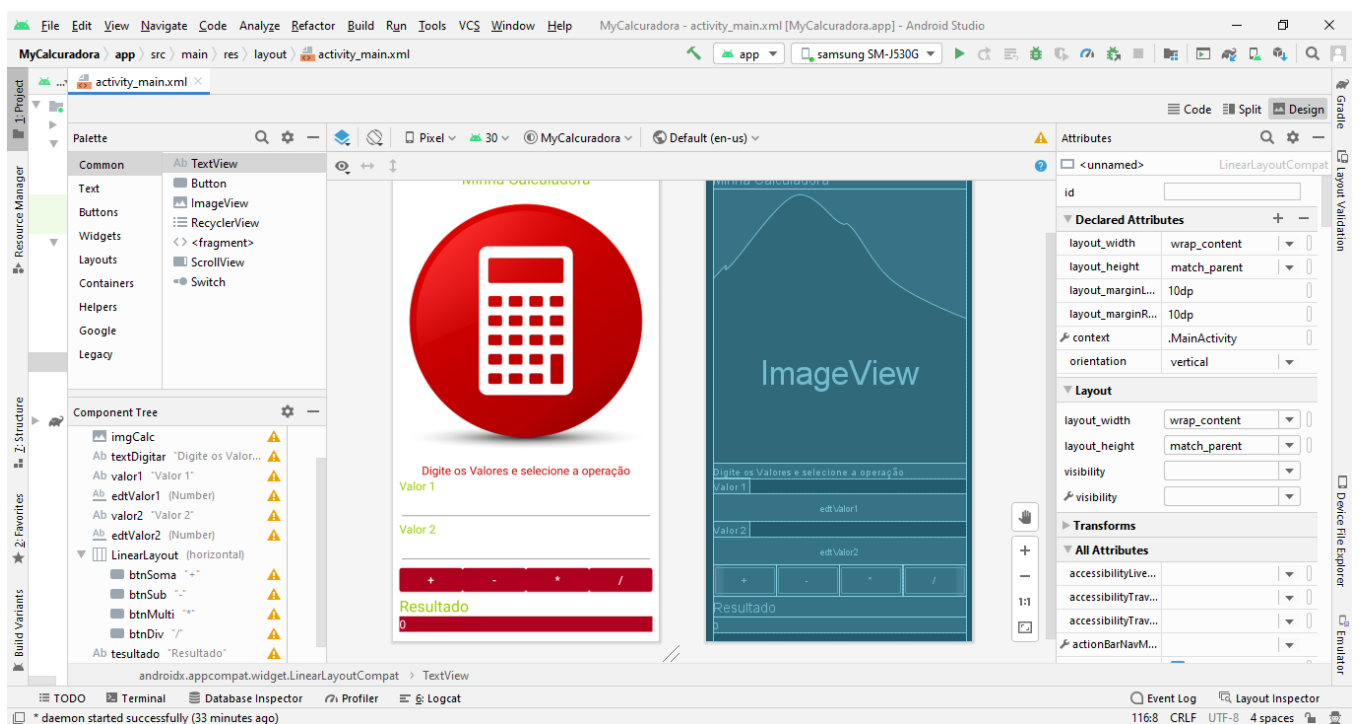


Observação: o tutorial está em cor diferente do documento anterior porque estes prints foram feitos em aula e este app ficou mais legal que o original ;)

Na próxima etapa iremos inserir mais um campo de texto para o resultado e para facilitar a nossa vida, iremos fazer via código de acordo com a imagem abaixo! Observem os nomes, eles serão muito importantes.

```
114     </LinearLayout>
115
116     <TextView
117         android:id="@+id/resultado"
118         android:layout_width="match_parent"
119         android:layout_height="wrap_content"
120         android:text="Resultado"
121         android:textColor="@android:color/holo_green_light"
122         android:textSize="24sp" />
123     <!--Para mostrar o resultado, iremos colocar outro texto-->
124     <TextView
125         android:id="@+id/txtResultado"
126         android:layout_width="match_parent"
127         android:layout_height="wrap_content"
128         android:background="@color/design_default_color_error"
129         android:text="0"
130         android:textColor="@color/white"
131         android:textSize="18sp" />
132
133 </androidx.appcompat.widget.LinearLayoutCompat>
```

Visualização do nosso App!

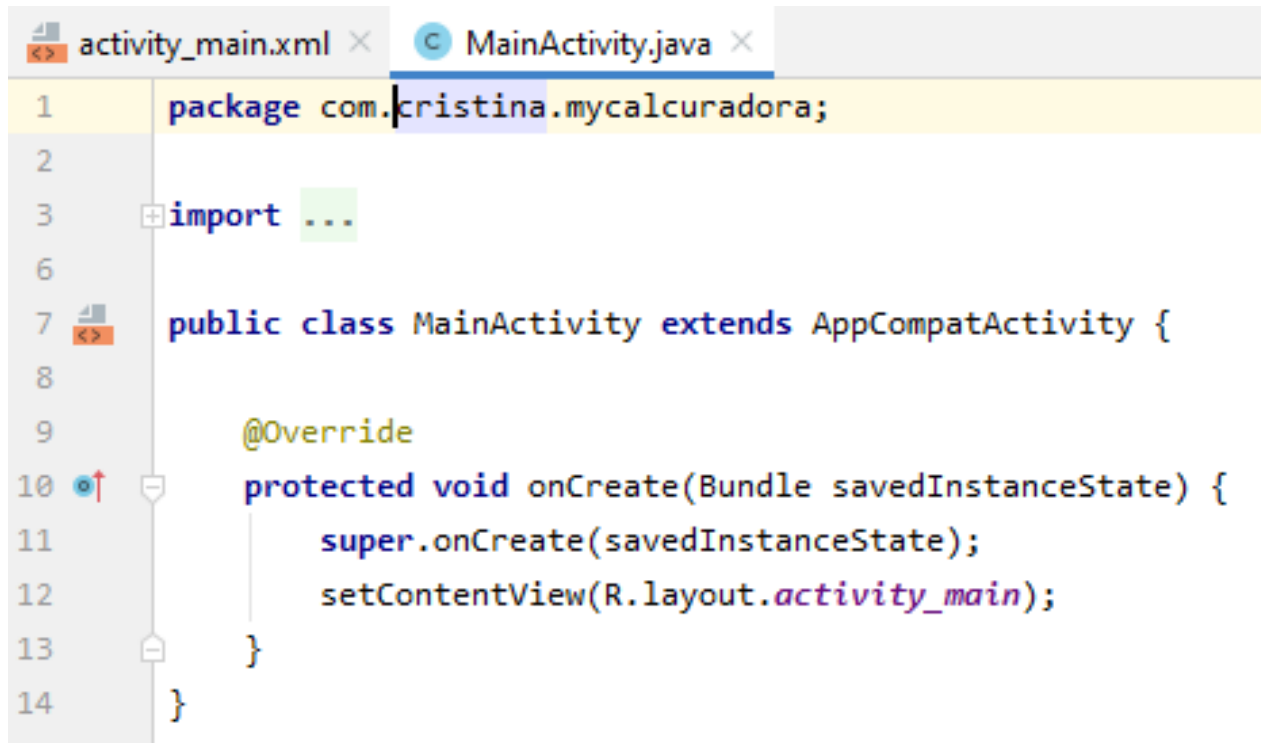


Ainda é possível colocar **margin_top** ou **bottom** para configurar o distanciamento dos componentes inseridos no layout do app, ou ainda: **zerar todos os atributos layout_height e configurar o layout_weight por valores, tipo 2.4, 3.5** desta forma estaremos configurando as alturas pela proporção, o que é aconselhável para controle de tamanho de imagens.

Classe R - Android

Afinal, qual é a classe R no Android?

No Android Studio, quando nós criamos um projeto usando uma empty activity, o AS gera o seguinte código!

A screenshot of the Android Studio IDE showing the MainActivity.java file. The code is as follows:

```
1 package com.cristina.mycalcuradora;
2
3 import ...
4
5
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14 }
```

A classe R é uma classe de mapeamento e/ou setamento que cria objetos de acesso para todos os componentes/objetos do nosso layout no app. Não é porque criamos os componentes no layout que não precisaremos criar no Java também! Nesta criação é construído um vínculo entre o elemento construído no design e mapeado em nós no xml e o Java, portanto, **a classe R é responsável por criar uma “interface de controle” ou comunicação entre as estruturas de um projeto Android - os arquivos xml e Java!**

Classe R contém as definições de todos os recursos de um pacote de aplicativo específico . Ele está no namespace do pacote do aplicativo.

Geralmente, há duas R classes com as quais trabalhamos:

1. Os recursos da estrutura em android.Re
2. Seu próprio namespace

Vamos para a classe R!

Alternamos para o modo Code e vamos acessar o arquivo **MainActivity.java**

1ª Parte!

Declarando os atributos Java e importando as classes destes dados.

```
activity_main.xml x MainActivity.java x
1 package com.cristina.mycalcuradora;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.widget.Button;
7 import android.widget.EditText;
8 import android.widget.TextView;
9
10 public class MainActivity extends AppCompatActivity {
11     EditText edtValor1, edtValor2;
12     Button btnSoma, btnSub, btnMulti, btnDiv;
13     TextView textResultado;
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_main);
19     }
20 }
```

Importação

Declaração dos atributos

Agora o mapeamento dos objetos criados no xml e no java!

2ª Parte!

Pois então! Todo o componente criado no Java precisa ser mapeado para o xml que é o nosso layout, por isso a importância de definirmos os “IDs” de forma correta e adequada ao que cada elemento significa na aplicação. Vamos usar o método **findViewById()**; Este método tem a função de achar a view do nosso layout e associar (mapear) ao objeto Java criado!

```
1 package com.cristina.mycalcuradora;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.widget.Button;
7 import android.widget.EditText;
8 import android.widget.TextView;
9
10 public class MainActivity extends AppCompatActivity {
11     EditText edtValor1, edtValor2;
12     Button btnSoma, btnSub, btnMulti, btnDiv;
13     TextView textResultado;
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_main);
19
20         edtValor1 = findViewById(R.id.edtValor1);
21     }
22 }
```

Nome	Valor	Id
btnDiv	(= 1000232)	int
btnMulti	(= 1000106)	int
btnSoma	(= 1000211)	int
btnSub	(= 1000286)	int
edtValor1	(= 1000281)	int
edtValor2	(= 1000282)	int
imgCalc	(= 1000099)	int
textResultado	(= 1000254)	int
textCalculadora	(= 1000324)	int
textDigitar	(= 1000161)	int
txtResultado	(= 1000214)	int

Método -> Classe R -> id

Vamos repetir para cada objeto criado no Java! Observe que as cores dos objetos criados em Java mudaram de cor após o mapeamento! Desta forma, após este mapeamento, a aplicação saberá que quando o componente de layout for “chamado” por uma ação do usuário, ele tem a referência do objeto Java especificado.

```
5      import android.os.Bundle;
6      import android.widget.Button;
7      import android.widget.EditText;
8      import android.widget.TextView;
9
10     public class MainActivity extends AppCompatActivity {
11         EditText edtValor1, edtValor2;
12         Button btnSoma, btnSub, btnMulti, btnDiv;
13         TextView textResultado;
14
15         @Override
16         protected void onCreate(Bundle savedInstanceState) {
17             super.onCreate(savedInstanceState);
18             setContentView(R.layout.activity_main);
19
20             edtValor1 = findViewById(R.id.edtValor1);
21             edtValor2 = findViewById(R.id.edtValor2);
22             btnSoma = findViewById(R.id.btnSoma);
23             btnSub = findViewById(R.id.btnSub);
24             btnMulti = findViewById(R.id.btnMulti);
25             btnDiv = findViewById(R.id.btnDiv);
26             textResultado = findViewById(R.id.txtResultado);
27         }
28     }
```

3ª Parte!

Configuração dos Listener de Eventos e os Métodos de Eventos.

Listener define sempre quando algum componente está "à espera" da ação que é o evento promovido pelo usuário que está manipulando o app que desenvolvemos.

Os métodos de eventos podem se resumir a:

onClick()

De **View.OnClickListener**. É chamado quando o usuário toca no item (no modo de toque) ou foca no item com as teclas de navegação ou o trackball e pressiona a tecla "Enter" adequada ou pressiona o trackball.

onLongClick()

De **View.OnLongClickListener**. É chamado quando o usuário mantém o item pressionado (no modo de toque) ou foca no item com as teclas de navegação ou o trackball e mantém pressionada a tecla "Enter" adequada ou mantém o trackball pressionado (por um segundo).

onFocusChange()

De **View.OnFocusChangeListener**. É chamado quando o usuário navega para ou do item usando as teclas de navegação ou o trackball.

onKey()

De **View.OnKeyListener**. É chamado quando o usuário está com foco no item e pressiona ou solta uma tecla de hardware no dispositivo.

onTouch()

De **View.OnTouchListener**. É chamado quando o usuário realiza uma ação qualificada como um evento de toque, incluindo o pressionamento, a liberação ou qualquer outro gesto de movimento na tela (dentro dos limites do item).

(fonte:

<https://developer.android.com/guide/topics/ui/ui-events?hl=pt-br#:~:text=Um%20listener%20de%20eventos%20%C3%A9,com%20o%20item%20na%20IU.>)

```
16      @Override
17      protected void onCreate(Bundle savedInstanceState) {
18          super.onCreate(savedInstanceState);
19          setContentView(R.layout.activity_main);
20          //mapeamento dos objetos
21          edtValor1 = findViewById(R.id.edtValor1);
22          edtValor2 = findViewById(R.id.edtValor2);
23          btnSoma = findViewById(R.id.btnSoma);
24          btnSub = findViewById(R.id.btnSub);
25          btnMulti = findViewById(R.id.btnMulti);
26          btnDiv = findViewById(R.id.btnDiv);
27          txtResultado = findViewById(R.id.txtResultado);
28
29          //configuração da lógica, dos listener e eventos
30          btnSoma.setOnClickListener(new View.OnClickListener() {
31              @Override
32              public void onClick(View v) {
33
34              }
35          });
36      }
37  }
```

Na imagem acima temos a configuração do `setOnClickListener()`. Vamos repetir a mesma codificação para os demais componentes, de acordo com a imagem abaixo.

