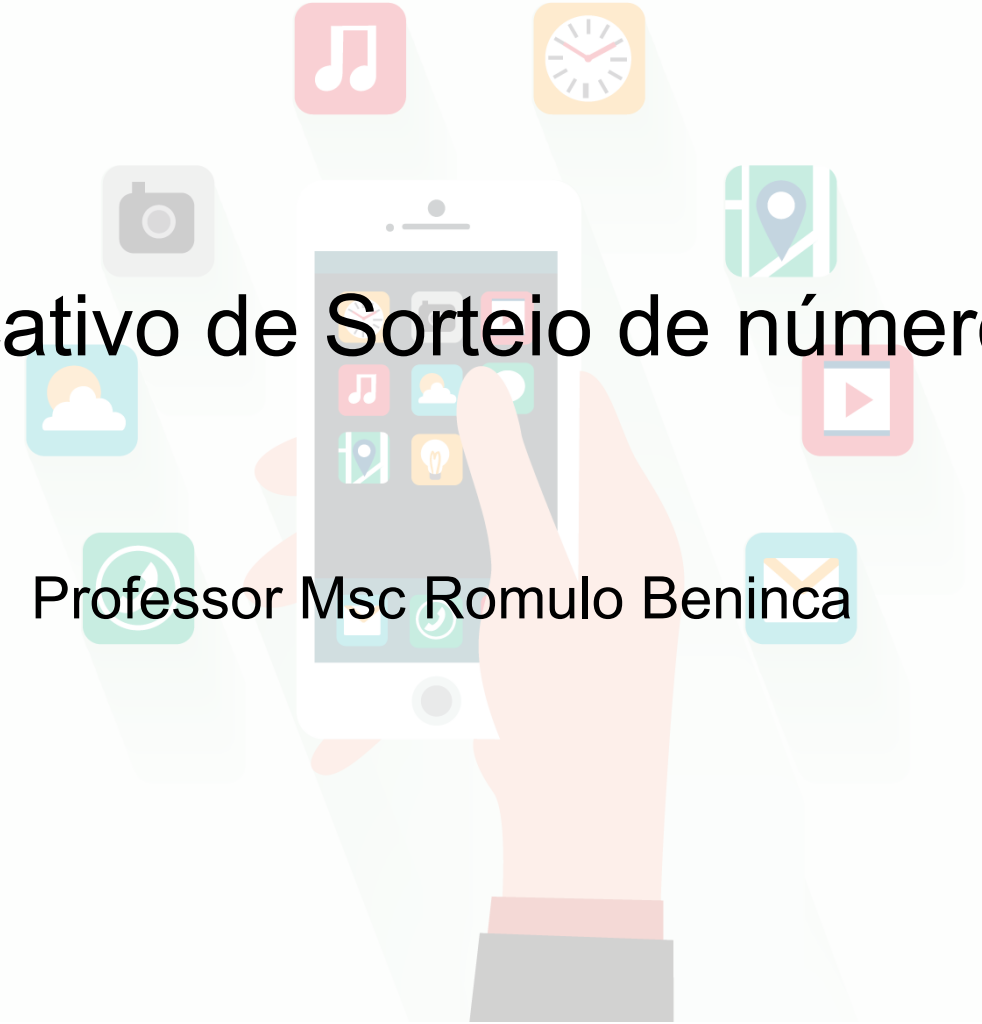


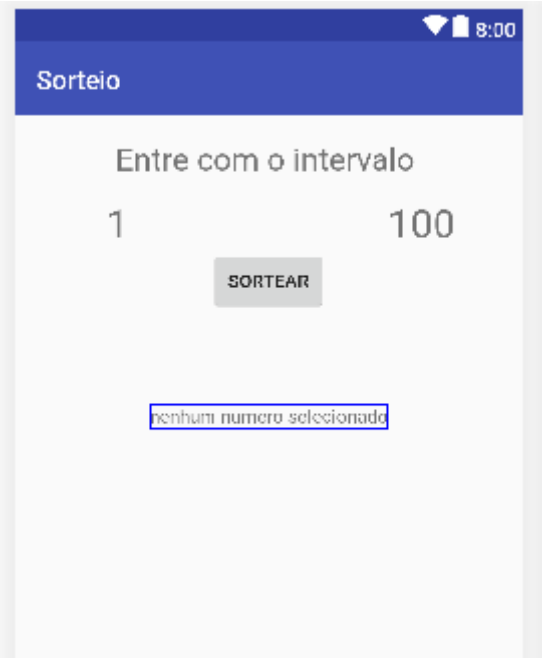
# Aplicativo de Sorteio de números

Professor Msc Romulo Beninca

# Objetivos da aula



Desenvolver uma aplicação de sorteio de números utilizando Android Studio e compreendendo os conceitos sobre a estrutura do projeto, chamada de métodos e iteração entre elementos da view com a Activity.



# Sumário

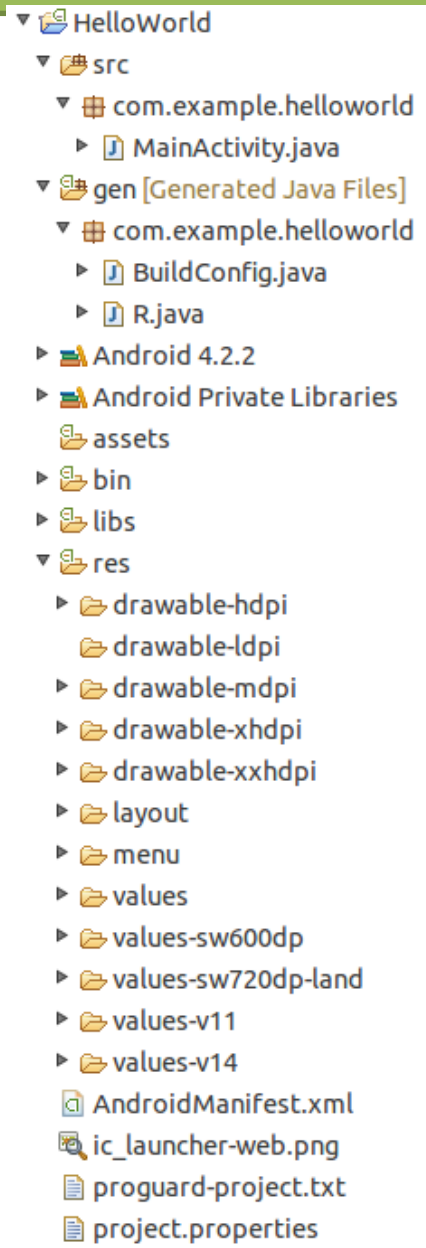


- Estrutura de um projeto Android.
- Definição de elemento na view.
- Chamada de método a partir de um evento
- Definição de identificadores.
- Como atribuir e recuperar valores de elementos da view.
- Aplicativo Muda Texto.
- Desenvolvimento do SorteioApp.

# Estrutura do Projeto de um App



- **src**: pasta que contém as classes java;
- **gen**: contém as classes geradas automaticamente, nunca devem ser alteradas manualmente.
  - Armazena a classe R.java;
- **assets**: arquivos opcionais do projeto, como fontes personalizadas;
- **libs**: pasta onde devem ser colocadas as bibliotecas .jar adicionais. Por padrão, traz a biblioteca android-support-v4.jar, que possui algumas classes de compatibilidade criadas no Android 3.x;
- **res**: recursos da aplicação: imagens, layouts, etc.
  - **drawable**: imagens da aplicação, divididas em subpastas de acordo com a resolução;
  - **layout**: arquivos XML utilizados para a contrução das telas;
  - **menu**: arquivos XML que criam os menus da aplicação;
  - **values**: arquivos XML para configurações em geral.



# Estrutura do Projeto de um App
## A pasta *res/layout*



A pasta */res/layout* armazena os arquivos de construção de telas e formulários;

- No android, cada **tela** é chama ***Activity***;
- Por padrão, cada *activity* é associada a um arquivo de layout, que define seus componentes.
- Na criação do projeto *HelloWorld*, foi gerada uma classe chamada ***MainActivity***, associada ao layout ***activity_main.xml***.
- Por que o nome do layout está ao contrário?
  - Nomenclatura da google, difere activities, menus, etc.

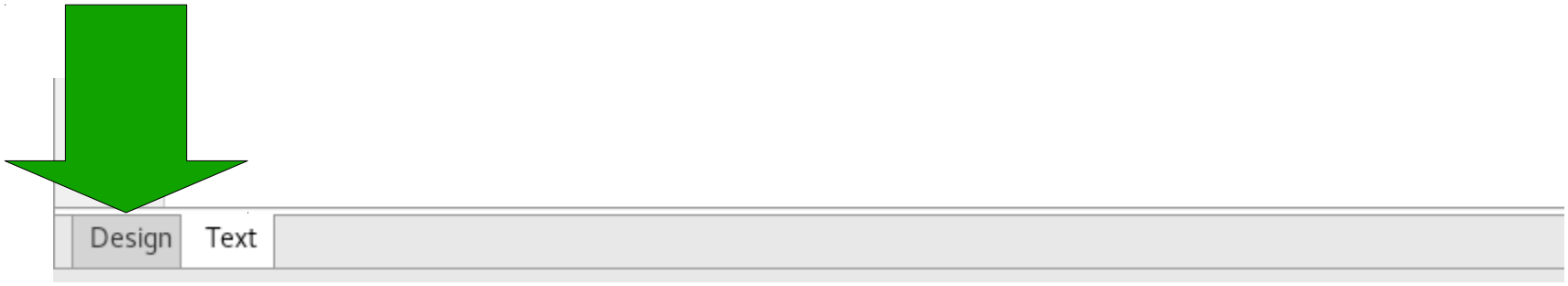
# Definindo um elemento View no layout: Adicionando um view ao layout *Text View*



Adicionando um *view* de texto por meio do código a baixo a ser inserido no arquivo res/layout/activity_main.xml

```
<TextView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:text="Texto 1"
/>
```

Você pode visualizar um *preview* do layout em construção na aba Design



# Definindo um elemento View no layout: Alterando uma propriedade do TextView



Alterando o texto contido.

```
<TextView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:text="Hoje eu acordei Feliz,
porquê tinha aula e eu adoro programar!
Não não dormir! Eu quero programar, Eu quero
programar!"
   />
```

Após alterar a propriedade text execute a aplicação e verifique o que ocorreu?

# Definindo um elemento View no layout: Alterando uma propriedade do TextView



Alterando o texto contido.

```
<TextView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:text="Hoje eu acordei Feliz,
porquê tinha aula e eu adoro programar!
Não não dormir! Eu quero programar, Eu quero
programar!"
   />
```

Como resolver o problema?

# Definindo um elemento View no layout:
## Como dimensionar o tamanho de um view no layout



Alterando o texto contido.

```
<TextView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:text="Hoje eu acordei Feliz, porquê tinha aula e eu
adoro programar!. Hoje eu acordei feliz eu do quero programar.
Não não dormir! Eu quero programar! Eu quero programar!"
   />
```

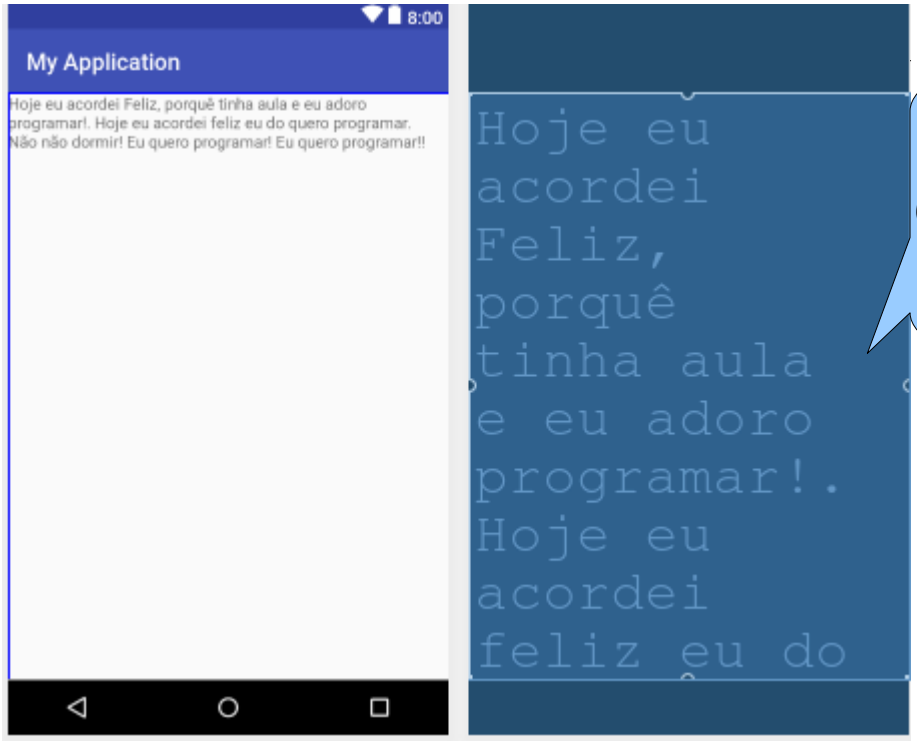
- **MATCH_PARENT**, o View assume o tamanho do elemento pai.
- **WRAP_CONTENT,** o View será suficientemente grande para incluir seu conteúdo (mais preenchimento), respeitando view pai

# Definindo um elemento View no layout:
## Parâmetro **wrap_content**



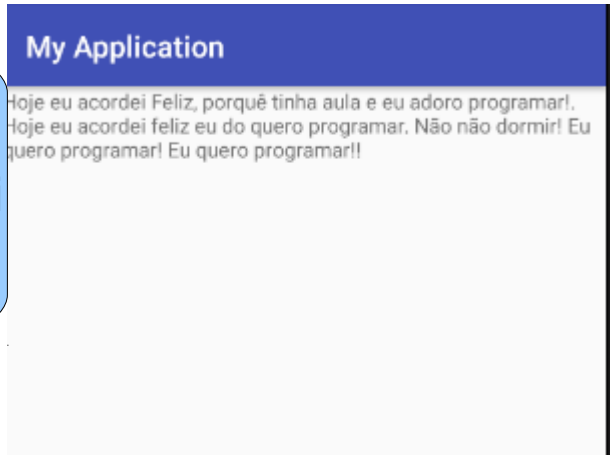
```
<TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="Hoje eu acordei Feliz, porquê tinha aula e eu
adoro programar!. Hoje eu acordei feliz eu do quero programar.
Não não dormir! Eu quero programar! Eu quero programar!"
   />
```

**Preview Design**




Observe que o view
Ocupa todo espaço do pai
Na largura e altura

**Aplicativo em execução**



# Definindo um elemento View no layout: Parâmetro **wrap_content**

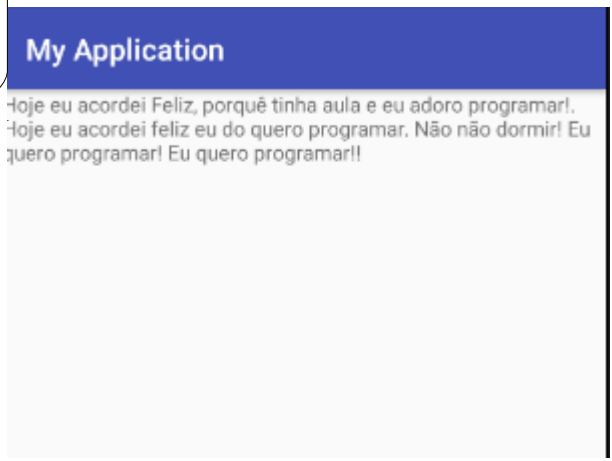


```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hoje eu acordei Feliz, porquê tinha aula e eu
adoro programar!. Hoje eu acordei feliz eu do quero programar.
Não não dormir! Eu quero programar! Eu quero programar!"
   />
```

**Preview Design**


Observe que o view ocupa todo espaço necessário para o texto, mesmo havendo mais espaço no objeto pai, não é utilizado

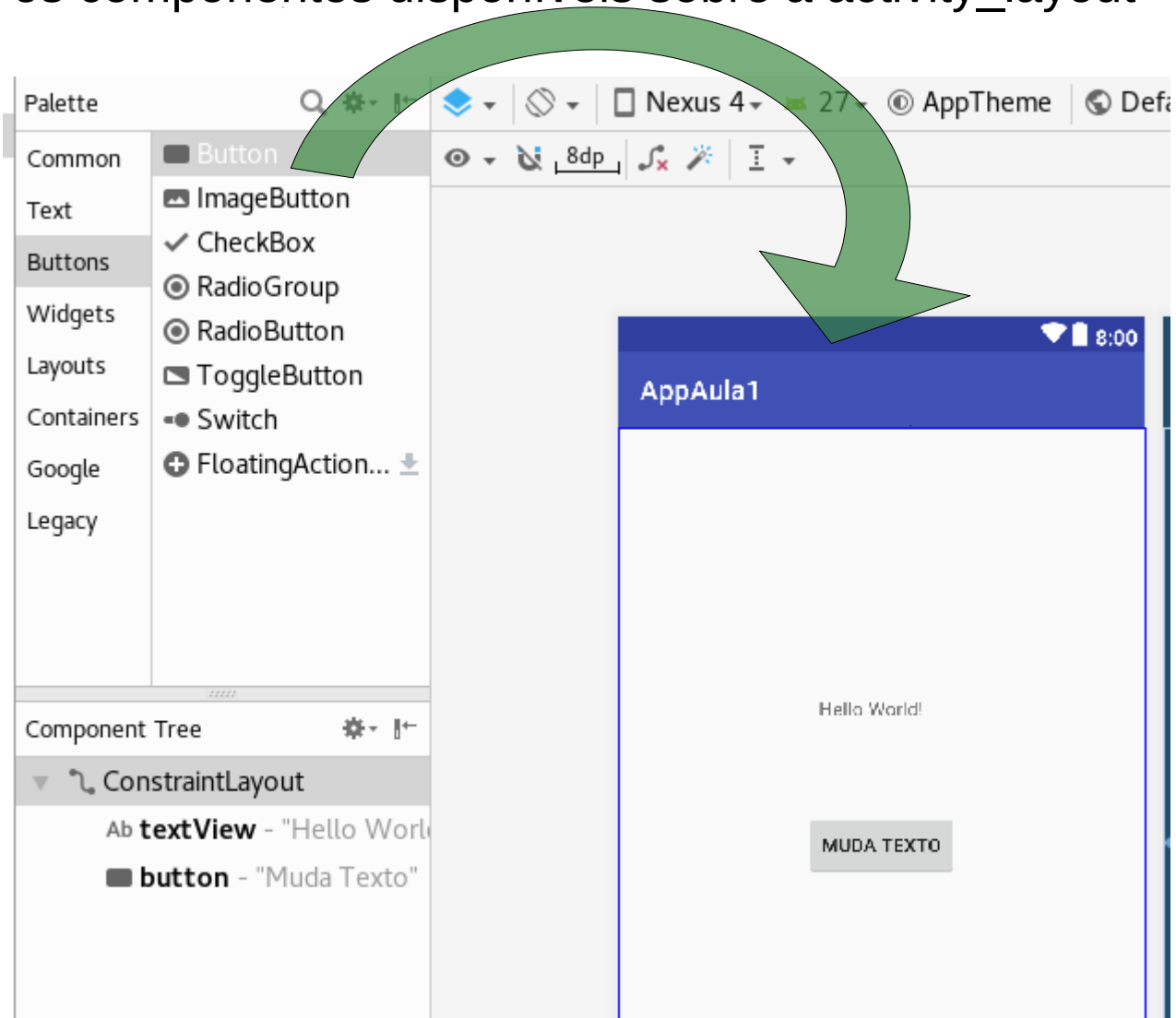
**Aplicativo em execução**



# Definindo Elemento da *view* pela ferramenta Design



Na ferramenta Design  podemos adicionar elementos predefinidos por arrastando os componentes disponíveis sobre a activity_layout



# Como ficou o código do *Layout* até agora

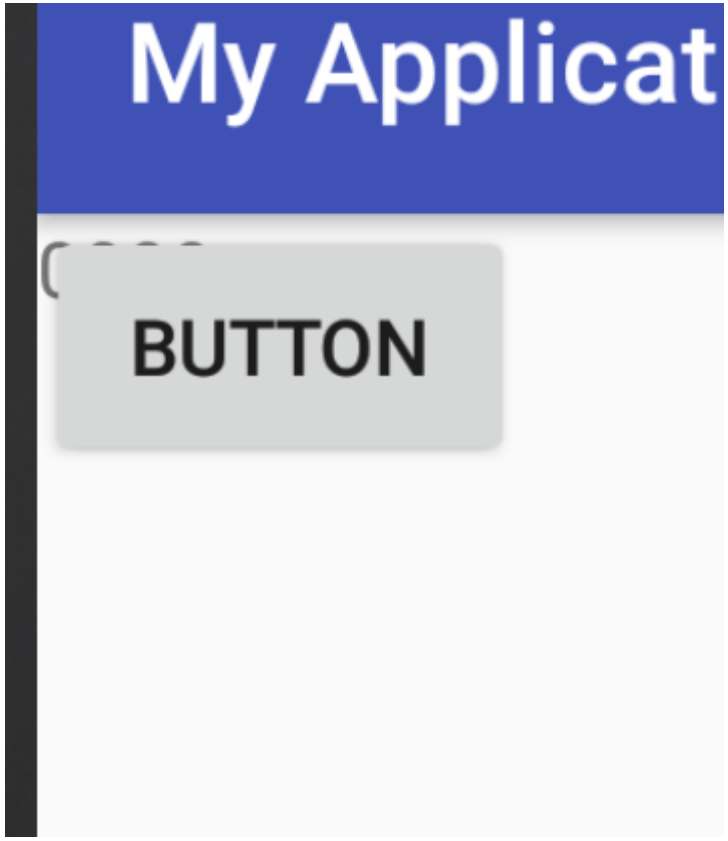


Altere seu código xml adicionando o botão como a seguir

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="0000"
/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button"
    />
```

# Executando a aplicação

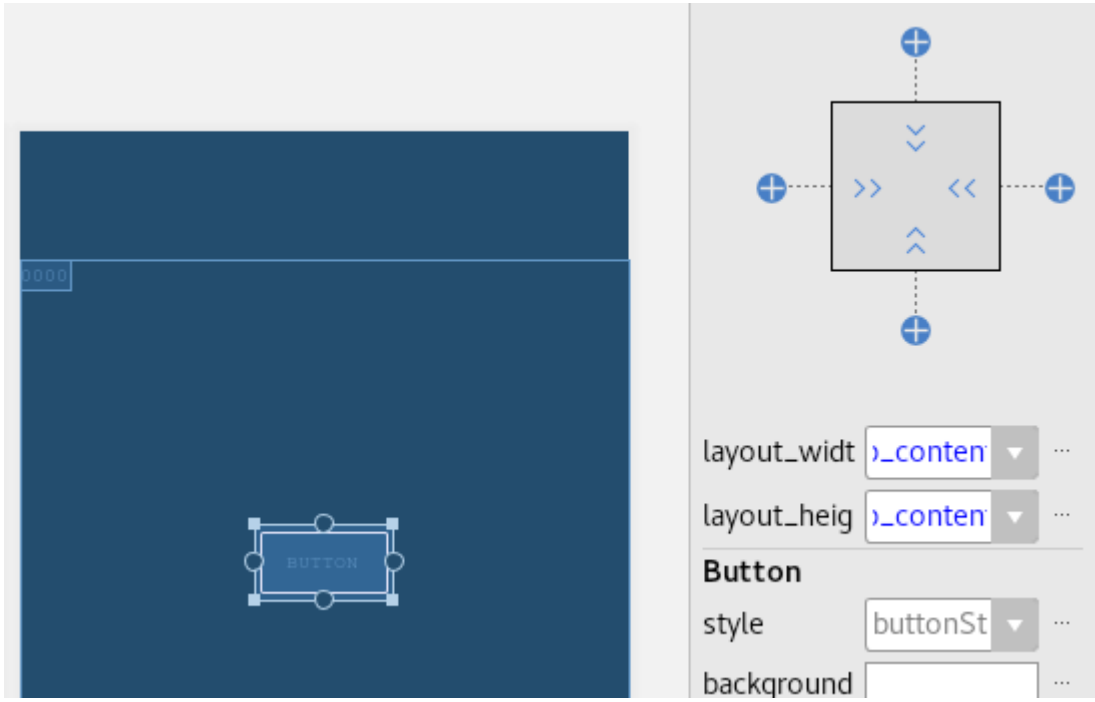




# Posicionando *views* pela ferramenta de design.



Na ferramenta desing são apresentado duas representações da tela, a do lado direito apresenta apenas restrições e conexões existentes entre os elementos.

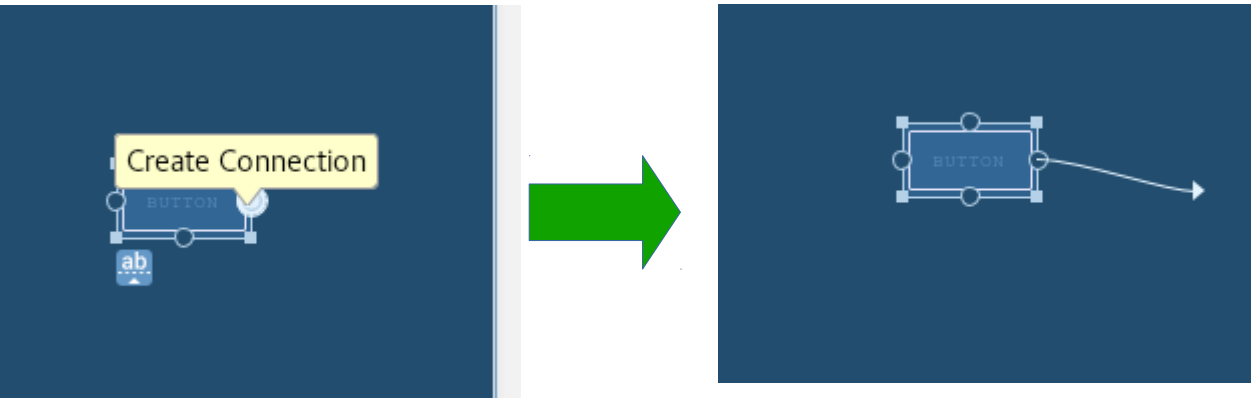


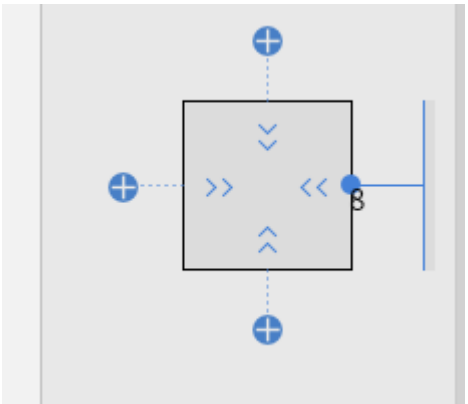
# Posicionando *views* pela ferramenta de design



Para associar a posição de um elemento em relação ao outro clique no elemento, na representação de restrições do layout e em seguida clique no outro elemento ao qual deseja-se associar o primeiro.

Exemplo:



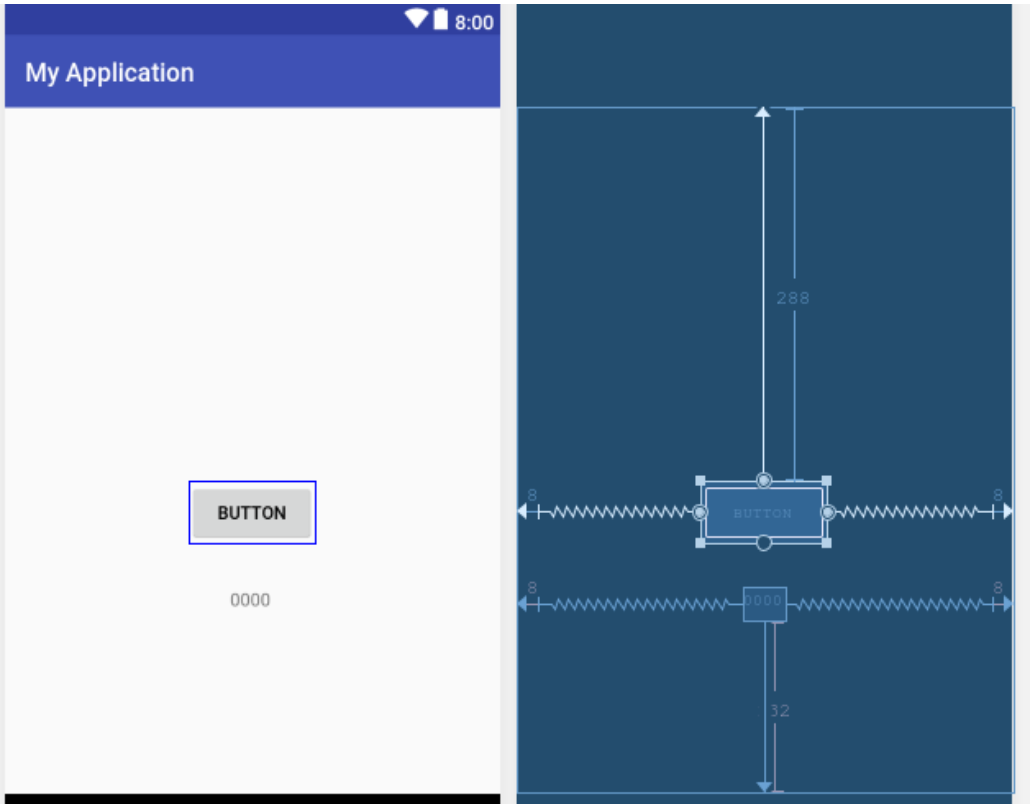



```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:text="Button"
    app:layout_constraintEnd_toEndOf="parent"
    />
```

# Posicionando views pela ferramenta de design



Utilizando a ferramenta Design do Android Studio faça o alinhamento do Botão e do TextView como na figura a seguir.



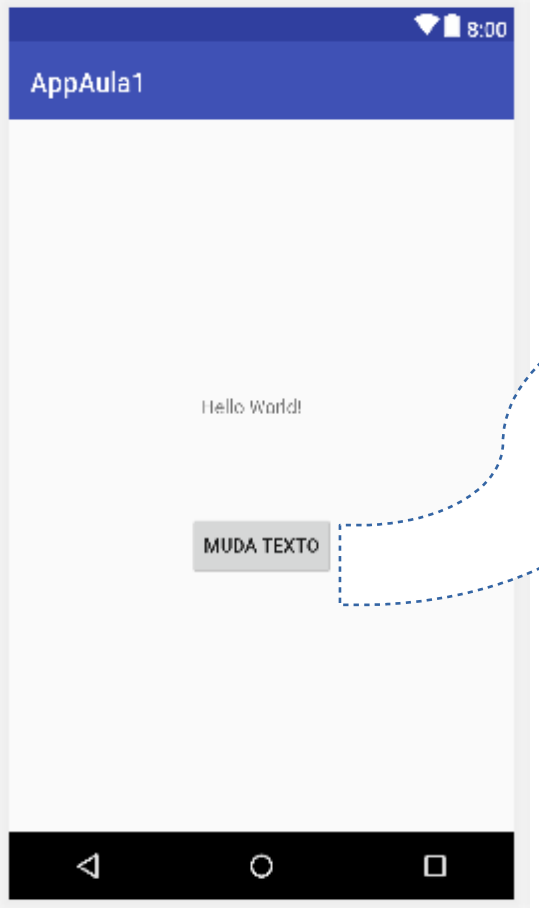

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="132dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:text="0000"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="320dp"
    android:text="Button"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

# Chamando método java a partir de um *view*


INSTITUTO FEDERAL
SANTA CATARINA



Para dar funcionalidade a qualquer elemento da view, que é definido no XML precisamos definir que ao detectar um evento sobre o view deve-se executar um metodo definido na classe Java.




?

```
public class MainActivity extends AppCompatActivity {

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

    public void mudatexto(View view){
        .....


    }
}
```

# Chamando método java a partir de um *view*


INSTITUTO FEDERAL
SANTA CATARINA



Para dar funcionalidade a qualquer elemento da *view*, que é definido no XML precisamos definir que ao detectar um evento sobre o *view* deve-se executar um método definido na classe Java. Esta propriedade também pode ser definida por meio da interface de design.

**Como exemplo a seguir:**

....
onclick=”metodoEmJavaAserExecutado”
....

**Exemplo:**

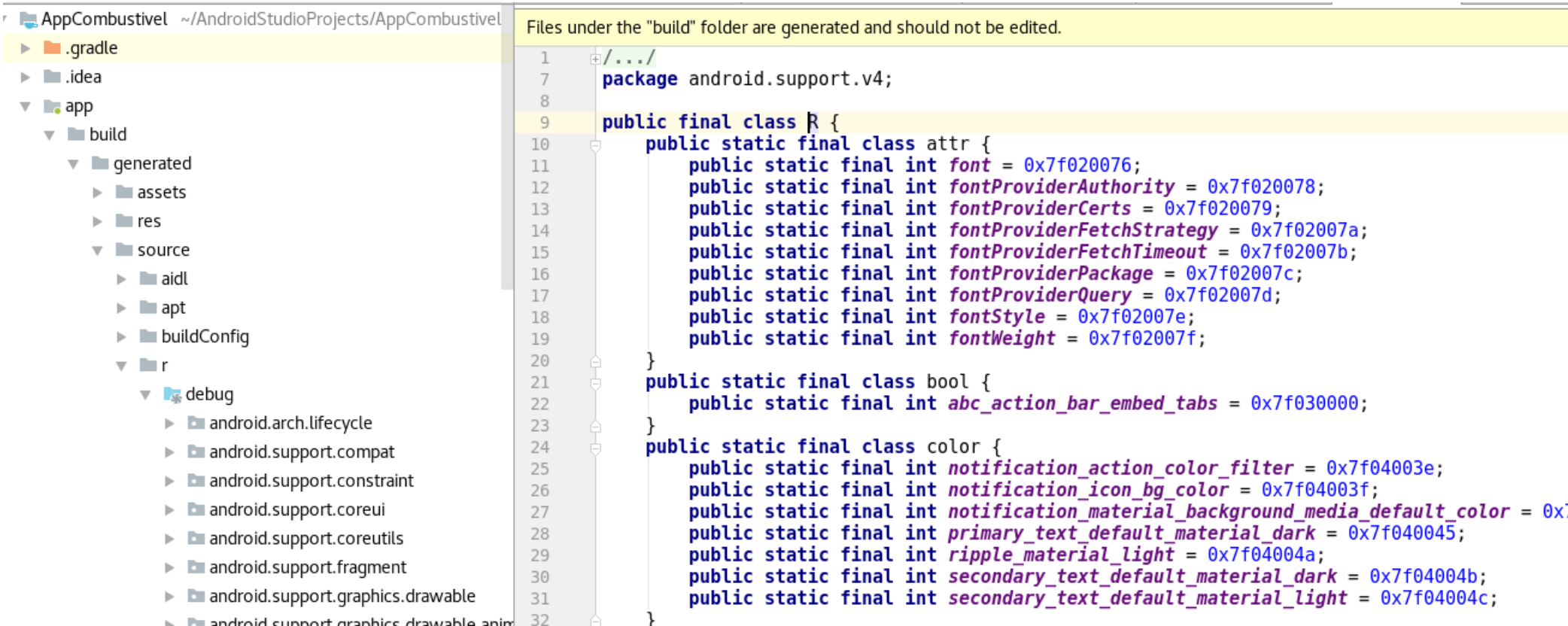
```
<TextView

  android:layout_width="wrap_content"
  android:layout_height="wrap_content"

  android:onClick="setandoTexto"

/>
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="320dp"
    android:text="Button"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:onClick="setandoTexto"/>
```

# Classe R



A classe R é criada em tempo de compilação e fornece acesso a endereço de todos elementos estáticos do projeto como  imagens, id de botões entre outros.



# Classe R



- A classe R é gerada e não deve ser alterada, ela é utilizada para identificar os componentes.
- Por ser uma classe *Static* podemos acessar o endereço dos elementos definidos no projeto por:

R.id.MeuComponente

# Exemplo de uso da classe R:
## Como referenciar *view* a partir do código java?

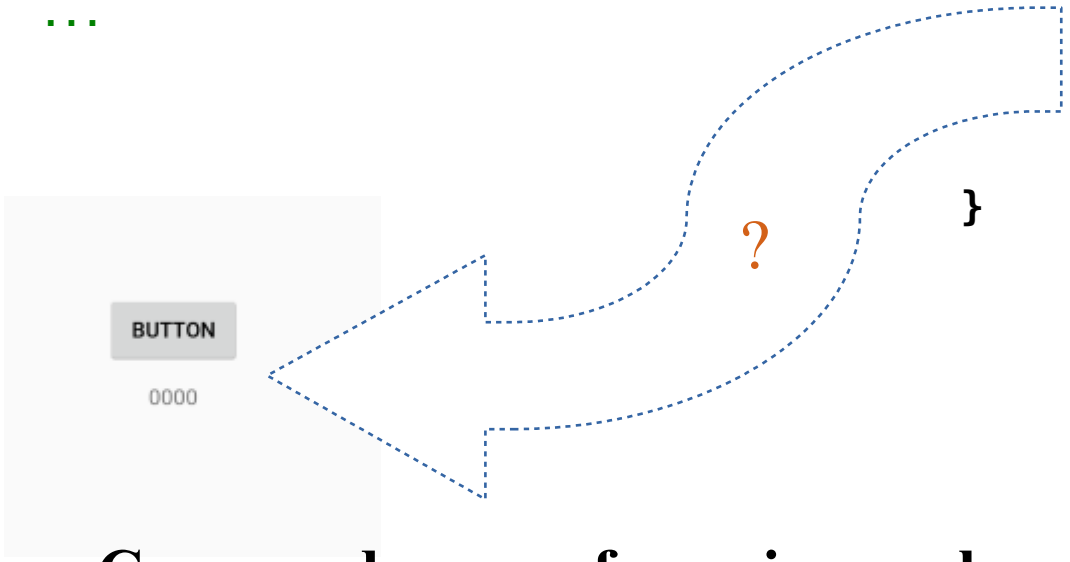


**XML Layout**

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="132dp"
            ...
            ...
/>
```

**Java**

```
public void setandoTexto(View view) {

    String texto=”IFSC é Legal!”

}
```




**Como podemos referenciar os elementos *views* a partir do código Java?**

# Exemplo de uso da classe R identificadores nas views



Para que  possam acessar os recursos do layout a partir do código Java precisamos identificar o cada view do layout por meio de identificadores únicos, os famosos id's.

**Definição do id de uma view**

```
android:id="@+id/idView"
```

Ao definir um identificador para cada view el pode ser acessado no activiy , por meio da classe R , que define recursos disponíveis no projeto.

**Como o id é referenciado no java**

```
R.id.idView;
```

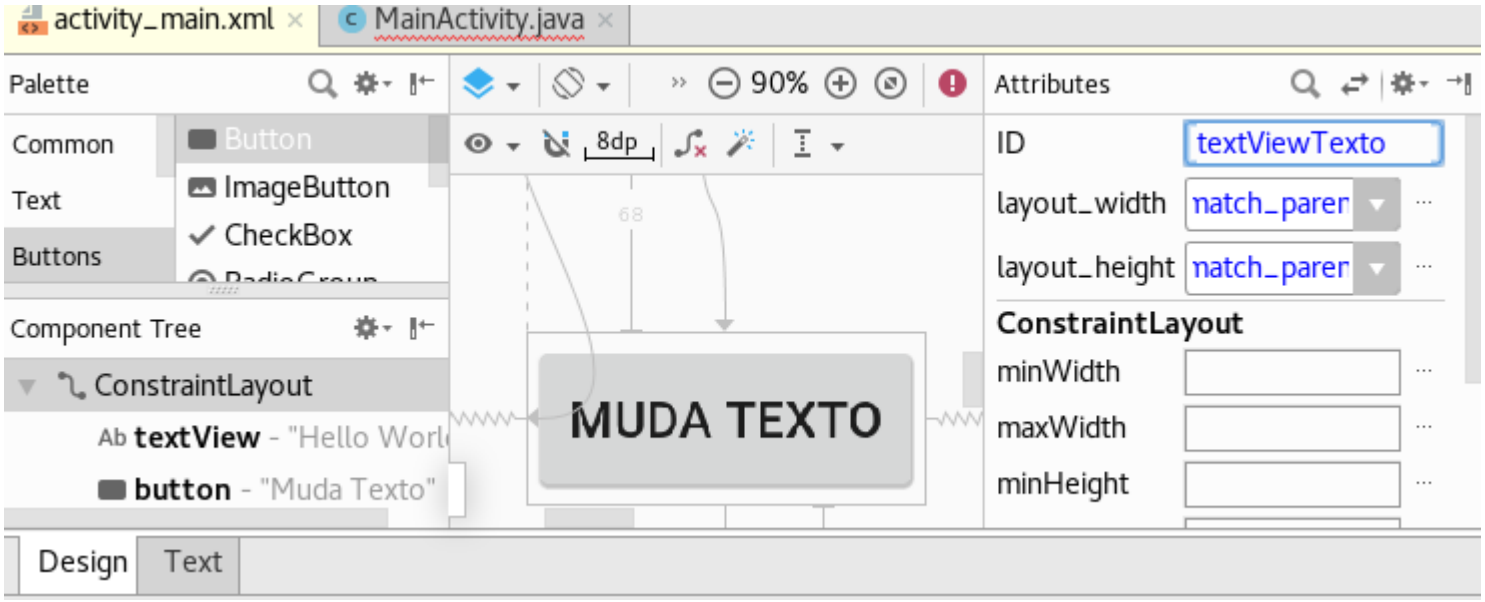
# Exemplo de uso da classe R: Identificadores pela ferramenta *Design*



Podemos definir o identificador de um view seguindo os passos:

1) Selecione o componente *TextView* na *Component Tree.*

2) Na seção *Attributes*(lado direito) encontre o campo id do componente selecionado  para *textViewTexto.*



# Como acessar o elemento *view* a partir do java?



A classe R da acesso a referencia do elemento, mas para acessar ele entre todos os *view* instanciados no layout é necessário utilizar a função *findviewbyid*

```
findViewById(R.id.idView)
```

# Como acessar o elemento *view* a partir do java?



A classe R da acesso a referencia do elemento, mas para acessar ele entre todos os *view* instanciados no layout é necessário utilizar a função *findviewbyid*

```
findViewById(R.id.idView)
```

**Exemplo:**

```
public void setandoTexto(View view) {
    TextView text=  findViewById(idView);
}
```

# Exemplo de uso da classe R
## Alterando a propriedade *text* a partir do código java



Faça a alteração no código para setar o texto ”Estudar no IFSC é legal” , quando o usuário clicar no botão.

# Aplicação Sorteio



## Agora que já sabemos :

- Definir elementos no layout de um *activity* e posicioná-los
- Realizar a chamada de um método para disparar o sorteio de números em java.
- Referenciar uma objeto da *view* e alterar suas propriedades.

**Atividade**

Crie o aplicativo SorteioApp, que deve receber como parâmetro dois números definido o intervalo do número a ser sorteado.

# Referências




- Lecheta, R. Google Android: Aprenda a criar aplicações para dispositivos móveis. 3a Ed. Novatec, 2013.
- http://developer.android.com/