Desenvolvimento com Android Studio

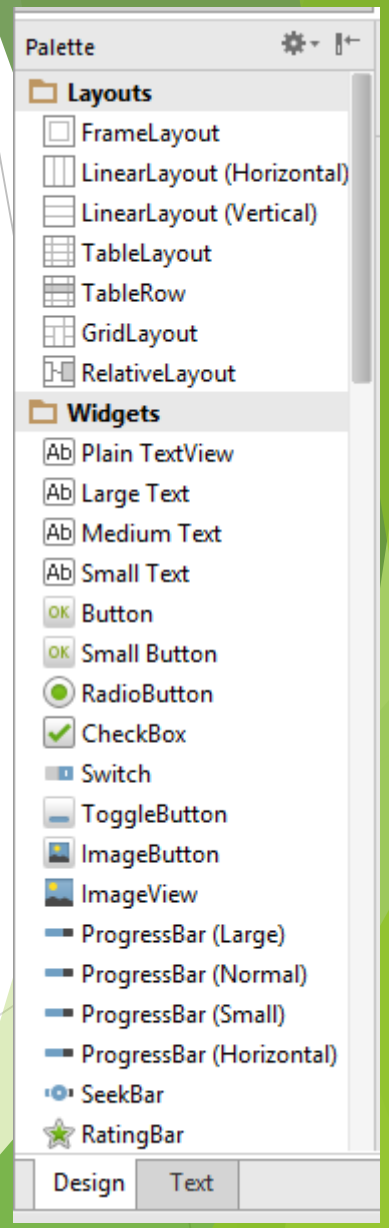
Aula 02 – Widgets, Manipulação de Dados e Programação de Eventos

Widgets

TextView

- O Widget TextView é utilizado para apresentar um texto não editável na tela.
- Qualquer componente gráfico pode ser adicionado arrastando da paleta até a tela gráfica ou criando o código XML deste elemento.
- O código XML que representa um TextView pode ser representado:

```
<TextView
        android:id="@+id/label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TEXTO 1:"
/>
```



TextView

- Propriedades:
  - **android:text=“Texto 1"**: Este parâmetro define o texto que é exibido na TextView
  - **android:id="@+id/tvNome** “: Este parâmetro define um identificado textView. Caso seja necessário manipular este elemento via código Java, usamos este identificador para acessá-lo
- OBS: De acordo com a documentação do android é considerado uma boa prática “exteriorizar” strings, arrays de string, imagens, cores, e outros recursos que você ou outra pessoa possa gerenciá-los separadamente do código de seu aplicativo.
- Para isto, adicione uma nova String: r**es -> values -> strings**

  <string name="nome">Nome</string>
- Em seguida configure o parâmentro android:text conforme abaixo:

  android:text="@string/nome"

TextView

- Propriedades adicionais:
  - **android:textColor="#A5B6C7“**: Este parâmetro define uma cor ao texto exibido. A cor definida deve está em formato hexadecimal.
  - **android:textSize="20dp“**: Este parâmetro define o tamanho do texto.
  - **android:textStyle="bold“**: Define o estilo do texto(negrito, itálico ou normal)
  - **android:textAllCaps="true“**: Define se o texto exibido aparecerá em caixa alta (true) ou em caixa baixa(false)
  - **android:layout_gravity="center_horizontal“**: Define o alinhamento do texto
  - **android:typeface="serif“**: Define os padrões de fonte, ou famílias, que no caso do Android são 3 famílias: Droid Sans, Droid Sans Mono e Droid Serif

TextView

- Propriedades:
  - **android:shadowColor**: cor da sombra
  - **android:shadowRadius**: o raio da sombra
  - **android:shadowDx**: o distanciamento horizontal da sombra em relação ao texto
  - **android:shadowDy**: o distanciamento vertical da sombra em relação ao texto

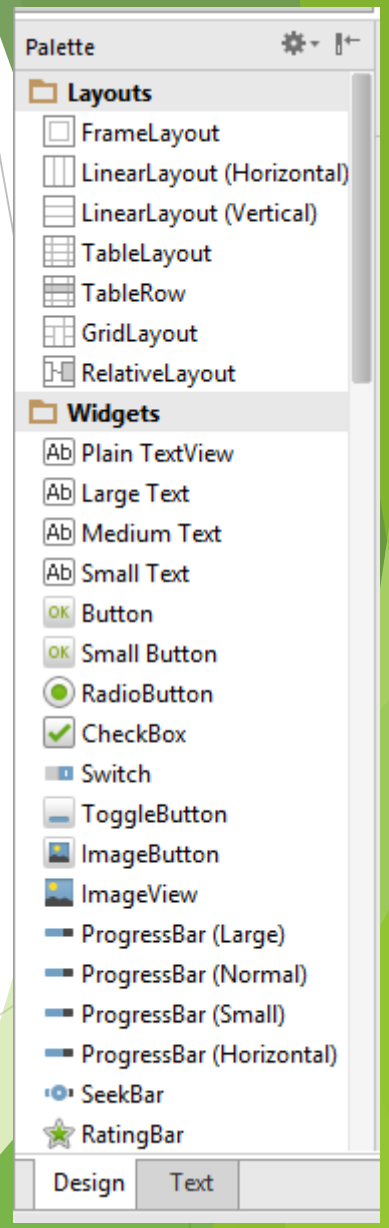
ImageView

- O Widget ImageView é usado para adicionar uma imagem em uma activity(tela)
- Os parâmetros id, gravity, e outras propriedades comuns a todos os widgets são configurados da mesma forma aos já apresentados
- Definindo a imagem (propriedade src):

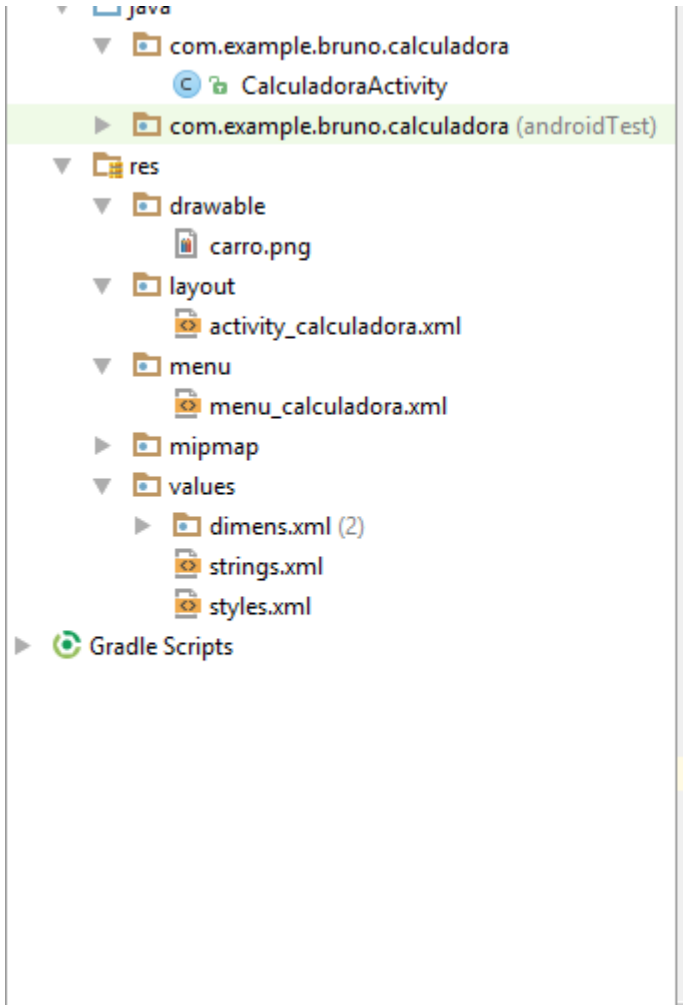
  android:src="@drawable/figura“

- OBS: Antes de utilizar uma imagem, é necessário coloca-la na pasta de imagem(@drawable). Para isto copie e cole a imagem na pasta específica.

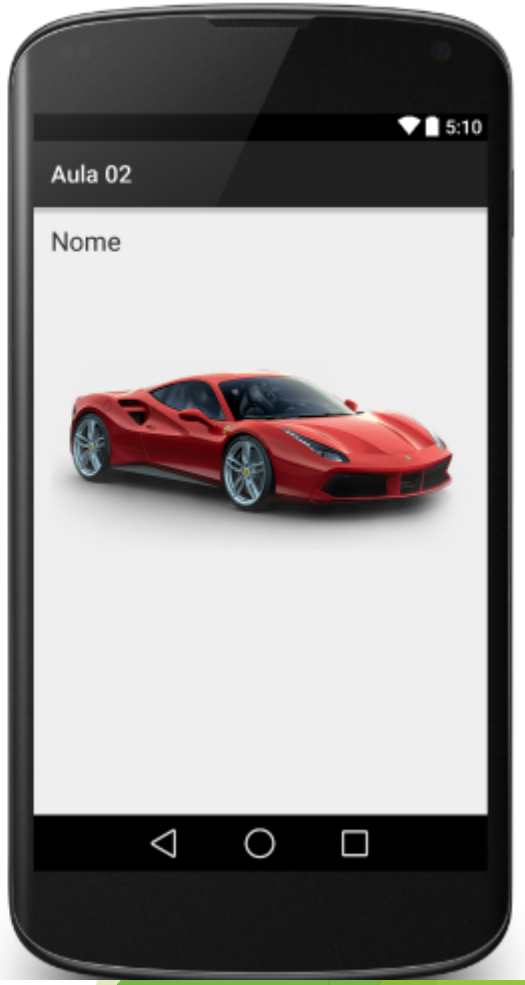
```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageView"
    android:layout_below="@+id/textView"
    android:layout_centerHorizontal="true"
    android:src="@drawable/carro" />
```



ImageView

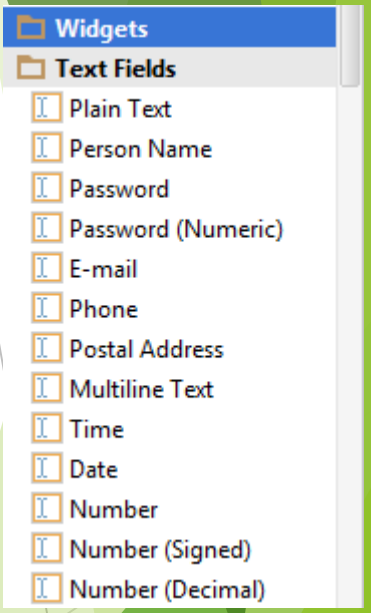



```
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:cor
    android:id="@+id/app">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/nome"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true"
        android:src="@drawable/carro" />
</RelativeLayout>
```



EditText

- Um EditText é um componente gráfico que permite ao usuário interagir com o aplicativo através da inserção de textos.
- Quando o usuário tocar em um EditText , automaticamente será exibido o teclado virtual para que uma informação seja passada.
- Na paleta de Widgets é possível incluir EditText com entradas pré-configuradas para permitir apenas números, campos no formato senha(password), etc.
- Uma EditText também poderá ser adicionada via código XML, conforme abaixo:

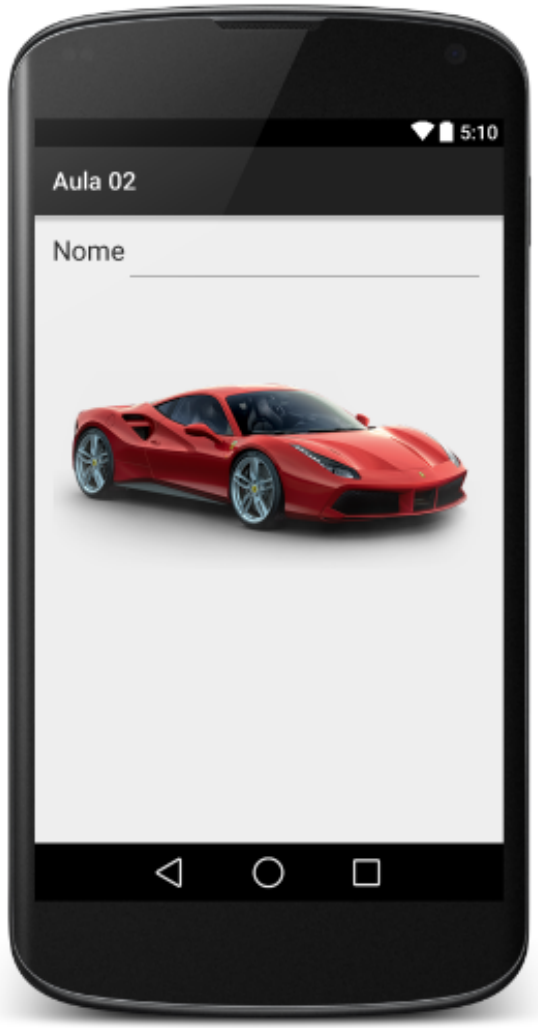



```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editText"
    android:layout_alignParentTop="true"
    android:layout_toRightOf="@+id/textView"
    android:layout_alignRight="@+id/imageView"
    android:layout_alignEnd="@+id/imageView" />
```

EditText

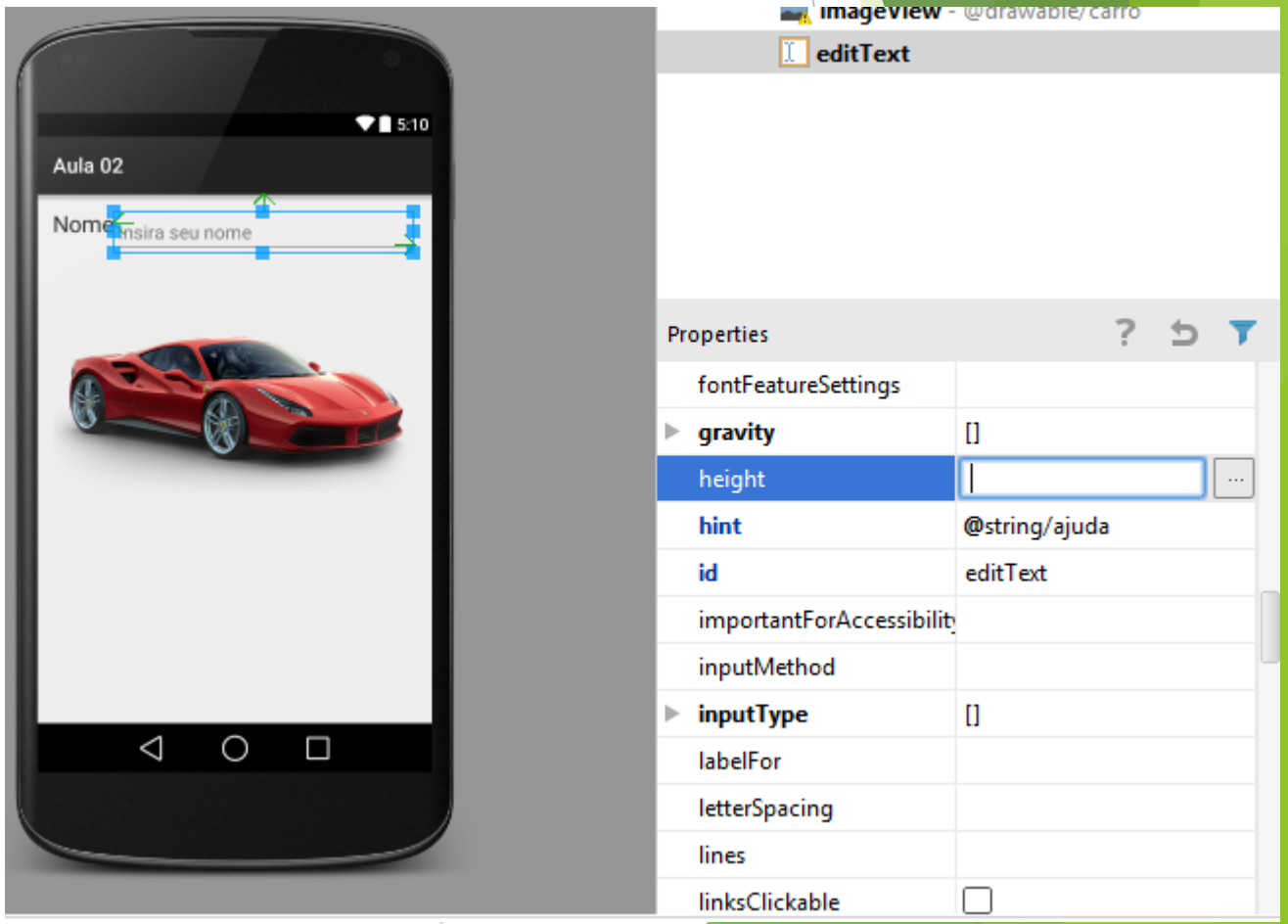
```
        android:text="@string/nome"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true"
        android:src="@drawable/carro" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@+id/textView"
        android:layout_alignRight="@+id/imageView"
        android:layout_alignEnd="@+id/imageView" />
</RelativeLayout>
```

Design | Text



EditText

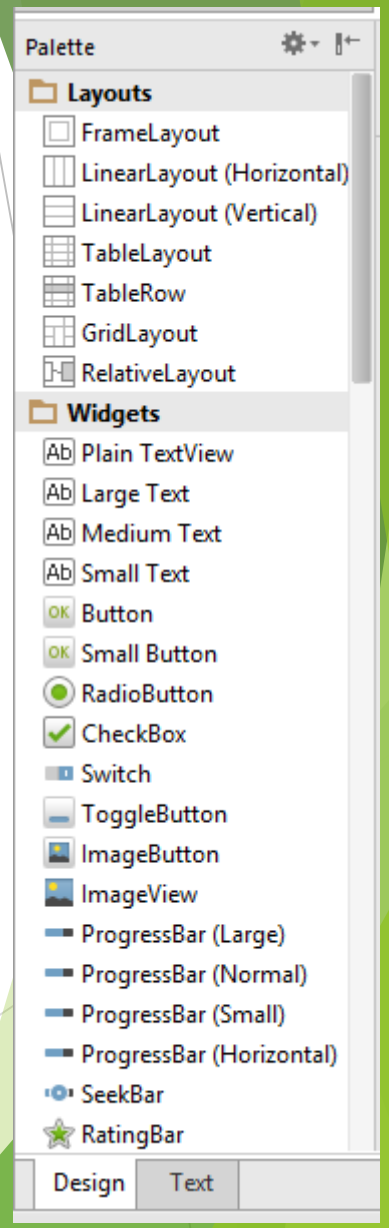
- Exibindo ajuda:
  - **android:hint**: este atributo exibe uma dica dentro de um componente EditText, a qual ajuda o usuário a entender o objetivo do componente. Quando o usuário iniciar a digitação neste componente, a dica de ajuda desaparece.



Button

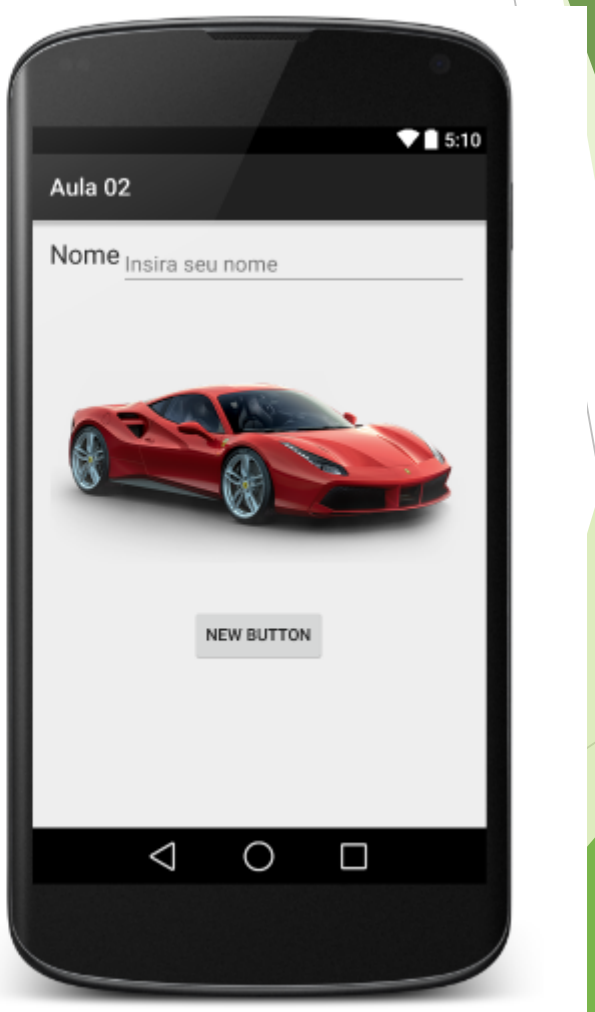
- Um Button é um componente gráfico que permite ao usuário interagir com o aplicativo através de cliques(toques) no botão.
- Em geral os botões acompanham código JAVA que é acionado para realizar uma determinada função assim que o usuário do aplicativo toca-lo. Usamos para isto a propriedade onClick para chamar uma função no código JAVA a qual o formulário está relacionado.
- As propriedades id, text, background, margin e outras propriedades comuns a todos os widgets são configuradas da mesma forma que os controles apresentados

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button"
    android:layout_alignBottom="@+id/imageView"
    android:layout_centerHorizontal="true" />
```



Button

```
        android:id="@+id/imageView"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true"
        android:src="@drawable/carro" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@+id/textView"
        android:layout_alignRight="@+id/imageView"
        android:layout_alignEnd="@+id/imageView"
        android:hint="@string/ajuda" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button"
        android:layout_alignBottom="@+id/imageView"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```



Relacionando Widgets no código Java

Manipulando Widgets através de código JAVA

- Para manipular os componentes que adicionamos via XML é necessário um código JAVA que esteja relacionado com a interface gráfica do aplicativo.
- Na programação em Android, temos em geral um arquivo .XML que contém o layout e interface gráfica do aplicativo e uma classe JAVA associada a uma Activity.
- Ao criar um novo projeto para Android, foi criado automaticamente pela ferramenta um Layout .XML (res -> layout -> nome do layout.xml) e uma classe Java (src -> nome do pacote -> nome da classe.java
- Observe que na classe JAVA, existe um método **OnCreate**:

```
public class Aula02Activity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_aula02);
    }
```

Manipulando Widgets através de código JAVA

- O método **onCreate(Bundle)** é onde você inicia sua atividade e define a interface gráfica a qual está relacionada através do método **setContentView**.
- Assim podemos obter os *widgets* (elementos de tela) através de um método chamado **findViewById().**
- Para que um arquivo .JAVA consiga se comunicar com o layout XML e outros recursos, existe uma classe intermediária, criada e atualizada automaticamente pela ferramenta, responsável por realizar esta ligação. Esta classe é chamada de **R.Java**.
- Observe que o método setContentView relaciona a interface gráfica usando a classe R: (`setContentView(R.layout.activity_main);`

Manipulando Widgets através de código JAVA

- O método ***onCreate()*** - É a primeira função a ser executada em uma Activity (Tela). Geralmente é a responsável por carregar os layouts XML e outras operações de inicialização. É executada apenas uma vez.
- O método **onCreate()** é importante pois é nele que iniciaremos nossos objetivos visuais e relacionamos os eventos(onClick,onTouch, etc)
- Neste método, **programamos todas as funções** que gostaríamos que fossem inicializadas quando o aplicativo for executado.
- **OBS: Este método só é executado uma única vez, no início da aplicação.**

A classe R.Java

- Esta classe é o "coração" do sistema Android. Ela representa, em forma de atributos Java, todos os recursos da sua aplicação que estão dentro dos diretórios já explicados de um aplicativo Android.
- **Ela é gerada e atualizada automaticamente e não deve ser editada manualmente; o Eclipse fará isto automaticamente.**
- Por exemplo, temos dentro do diretório "res/drawable" a imagem "icon.png"; podemos acessá-la de dentro da nossa aplicação Android com a seguinte expressão: '**R.drawable.icon**', onde "R " é a classe, "drawable" é o diretório e "icon" é o nome do recurso.
- Isto serve para quaisquer recursos presentes dentro dos diretórios de recursos.

Manipulando Widgets através de código JAVA

- Os widgets podem ser usados no nosso código Java. Se no código XML tivermos um widget do tipo EditText, para acessar esse componente pelo Java, é preciso fazer uso da classe EditText.
- Cada widget no XML possui o seu respectivo em classe Java, logo, se possui um widget TextView, para acessa-lo devemos fazer uso da classe TextView e assim vai.
- Podemos relacionar um widget a qualquer momento(onCreate, no clique de um botão, etc.)
- No exemplo abaixo, associamos os widget no método onCreate da activity:

```
TextView tvBoasVindas;
tvBoasVindas = (TextView)findViewById(R.id.tvBoasVindas);

EditText txtNome;
txtNome = (EditText)findViewById(R.id.txtNome);
```

Manipulando Widgets através de código JAVA

- O código abaixo cria um objeto do tipo EditText no código Java

```
EditText txtNome;
```

- Para relacionar o objeto criado com o seu respectivo correspondente na interface gráfica usamos o método **findViewById** e especificamos qual elemento está relacionado através da classe **R.id.**
- Como JAVA é uma linguagem fortemente tipada, faz-se obrigatório converter o elemento para o tipo correto de dado, e isto é feito fazendo o casting (EditText), (TextView), etc.

```
txtNome = (EditText) findViewById(R.id.txtNome);
```

- Este processo é feito semelhantemente para quaisquer recursos presentes dentro da interface gráfica XML.

Programando Eventos

Botões e Ações

Programando eventos

- A programação para Android, semelhantemente a outros ambientes, linguagens e ferramentas gráficas, é orientada a eventos, neste caso, aos cliques e toques na tela.
- Cada vez que um usuário clica em um botão, seleciona um item em uma lista, ou pressiona uma tecla, o sistema operacional gera um evento
- Se uma aplicação está interessada em um evento específico (por exemplo, clique em um botão), deve solicitar ao sistema para “escutar” o evento. Se a aplicação não está interessada, seu processamento continua de forma normal.
- É importante observar que a aplicação não espera pela ocorrência de eventos isso é controlado pelo sistema.

Programando eventos

- Para que um componente ou container possa “escutar” eventos, é preciso instalar um **listener**.
- Listeners são classes criadas especificamente para o tratamento de eventos.
- Um event listener é uma interface da classe View que contém um método simples de chamada. Esse método pode ser chamado pela framework Android quando a View a qual o listener está registrado é chamado por uma interação de usuário com um item da interface, ou seja, quando a ação correspondente ocorre no objeto.
- Por exemplo, quando um botão é clicado, o método onClick() é chamado no objeto.

Programando eventos

- Os principais eventos presentes na programação para Android são:
  - ***onClick()***
    Vem de View.OnClickListener. É chamado quando o usuário toca o item (quando estiver em modo de toque) ou foca o item através de teclas de navegação ou trackball e pressiona o botão de enter correspondente ou pressiona o trackbak (que também serve como enter).
  - ***onLongClick()***
    Vem de View.OnLongClickListener. É chamado quando o usuário toca um item e o segura (quando estiver em modo de toque) ou foca o item através de teclas de navegação ou trackball e pressiona o botão de enter correspondente e o segura ou pressiona o trackbak por pelo menos um segundo.

Programando eventos

- Os principais eventos presentes na programação para Android são:
  - ***onFocusChange()***
    Vem de View.OnFocusChangeListener. É chamado quando o usuário navega para dentro ou para fora de um item, usando as teclas de navegação ou trackball.
  - ***onKey()***
    Vem de View.OnKeyListener. É chamado quando o usuário foca em um item e pressiona ou solta uma tecla no dispositivo. Exemplo: quando se está escrevendo uma mensagem, você pode tocar no botão virtual (ou físico) da letra A e, ao soltá-lo, a letra A é mostrada na tela.

Programando eventos

- Os principais eventos presentes na programação para Android são:
  - ***onTouch()***
    Vem de View.OnTouchListener. É chamado quando o usuário performa uma ação qualificada como um evento de toque, incluindo pressionar, soltar ou qualquer movimento de gesto na tela (dentro dos limites do item).
  - ***onCreateContextMenu()***
    Vem de View.OnCreateContextMenuListener. É chamado quando um menu de contexto está sendo criado (como resultado de um long click).

Atividade – Verificar Idade

- Para exemplificar os listeners relacionados aos eventos, construa um layout semelhante ao apresentado ao lado.
- Este layout contém um TextView, um EditText e um Button
- Neste aplicativo chamaremos dois listener:
  - **onClick** para o botão verificar idade

Atividade – Verificar Idade

- Para referenciar os objetos: configure o nome da variáveis para :
- txtIdade -> EditText
- btVerificarIdade -> Button

- Assim, nosso código para acessar os componentes fica:

```
EditText txtIdade = (EditText)
findViewById(R.id.txtIdade);

Button bt = (Button)
findViewById(R.id.btVerificarIdade);
```

Atividade – Verificar Idade

- Após a criação do layout mostrado no exemplo anterior será programado um listener referente ao botão VERIFICAR IDADE. O listener com o evento deverá ser programado fora do método onCreate.
- **Para tanto usaremos classe Anônima e o método setOnClickListener:**

```
bt.setOnClickListener( new View.OnClickListener()
{
    @Override
    public void onClick(View v) {
        //Codigo
    }
});
```

Atividade – Verificar Idade

- Dentro do método `public void onClick(View v)` escreveremos o código associado ao clique. Neste caso, faremos um código para verificar a idade e classificar o valor informado e faixa etária(criança, adolescente, adulto)
- Usaremos a estrutura if...else para verificar uma faixa de idade(criança, adolescente e adulto)
- Ao final, exibiremos a faixa etária utilizando a classe Toast.

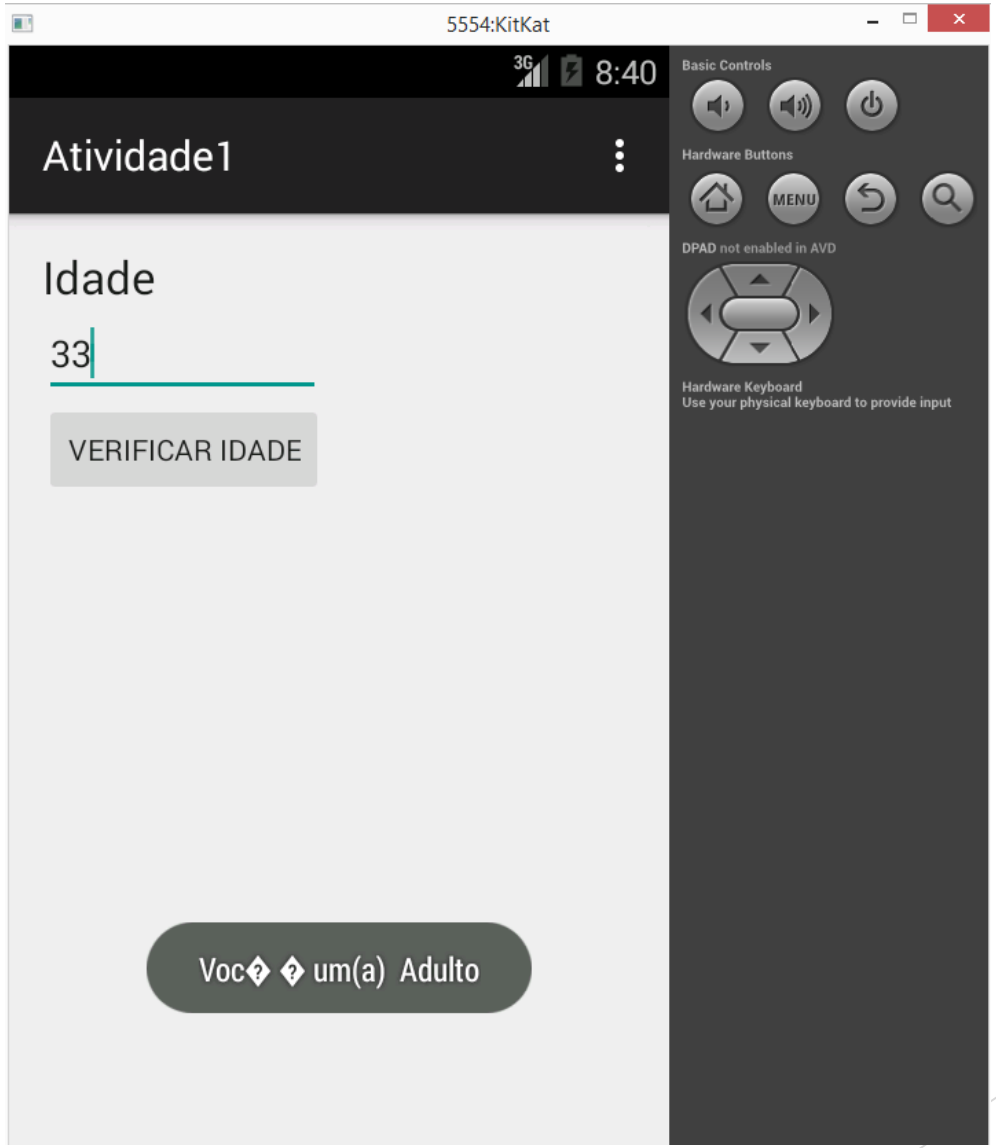
Atividade – Verificar Idade

- Para exemplificar os listeners relacionados aos eventos, construa um layout semelhante ao apresentado ao lado.
- Este layout contém um TextView, um EditText e um Button
- Neste aplicativo chamaremos dois listener:
  - **onClick** para o botão verificar idade

Atividade - Verificar Idade

```
bt.setOnClickListener( new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        EditText txtIdade = (EditText) findViewById(R.id.txtIdade);
        int idade = Integer.parseInt(txtIdade.getText().toString());
        String faixaEtaria;
        if ((idade > 0) && (idade <= 12))
            faixaEtaria = "Criança";
        else if ((idade > 12) && (idade <= 18))
            faixaEtaria = "Adolescente";
        else
            faixaEtaria = "Adulto";

        //A classe Toast é usada para mostrar uma mensagem na tela
        Toast.makeText(getBaseContext(), "Você é um(a)  " +
                                faixaEtaria, Toast.LENGTH_SHORT).show();
    }
});
```

Atividade – Verificar Idade



Exercício - IMC

- Desenvolver um programa em Android para calcular o IMC. Os dados de entrada serão o peso e a altura. Um botão mostrará o resultado do cálculo da seguinte fórmula:

- IMC = PESO/ALTURA^2

Resultado	Situação
Abaixo de 17	Muito abaixo do *peso*
Entre 17 e 18,49	Abaixo do *peso*
Entre 18,5 e 24,99	*Peso* normal
Entre 25 e 29,99	Acima do *peso*
Entre 30 e 34,99	*Obesidade* I
Entre 35 e 39,99	*Obesidade* II (severa)
Acima de 40	*Obesidade* III (mórbida)