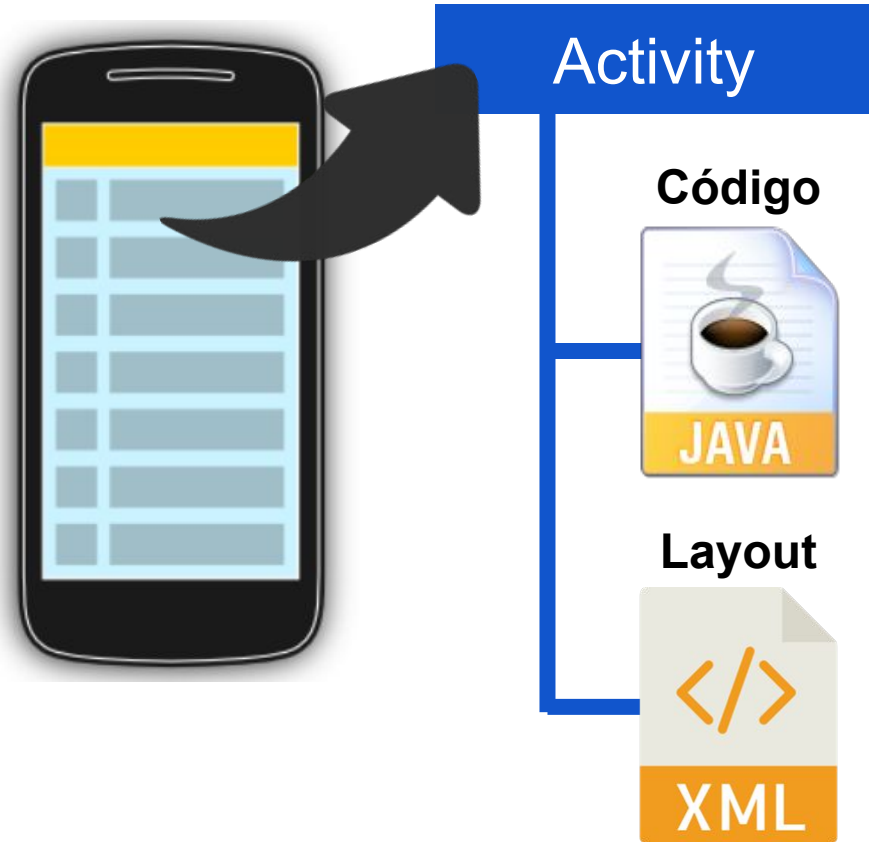


# Desenvolvimento Mobile

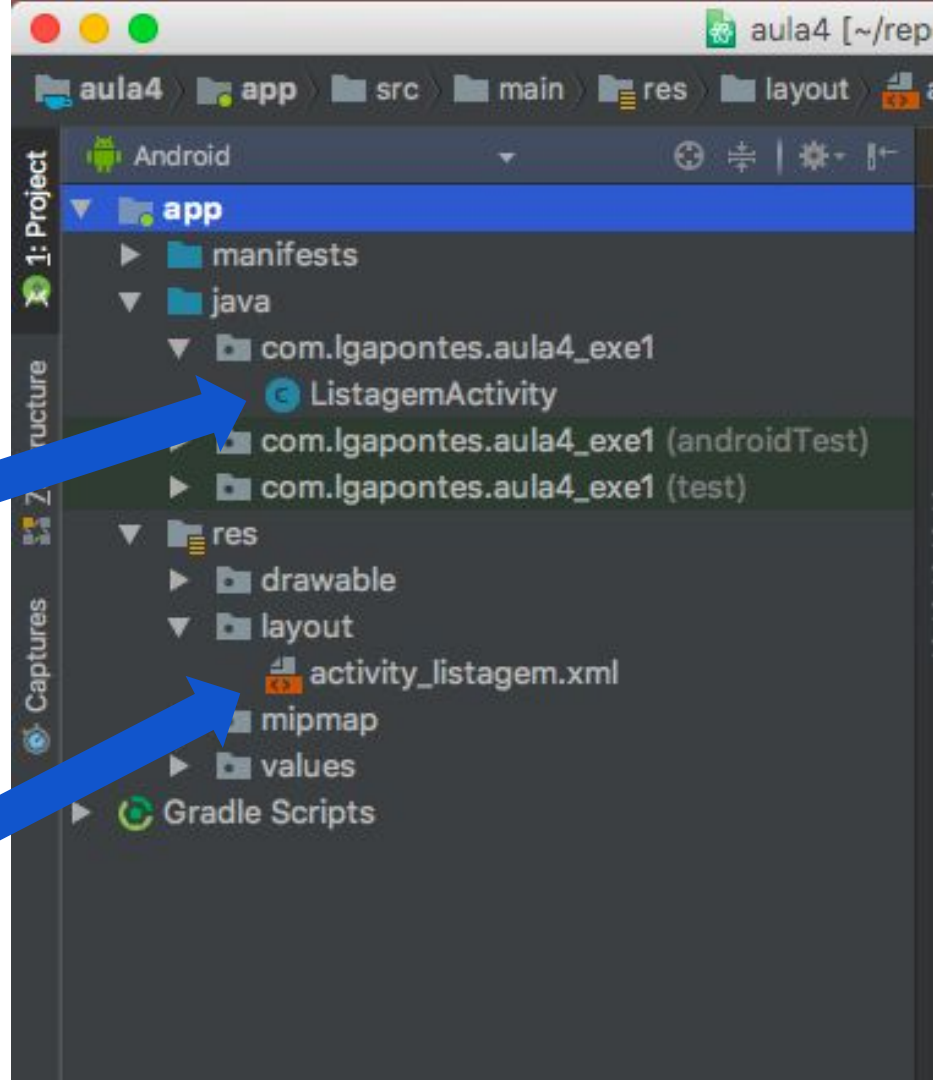
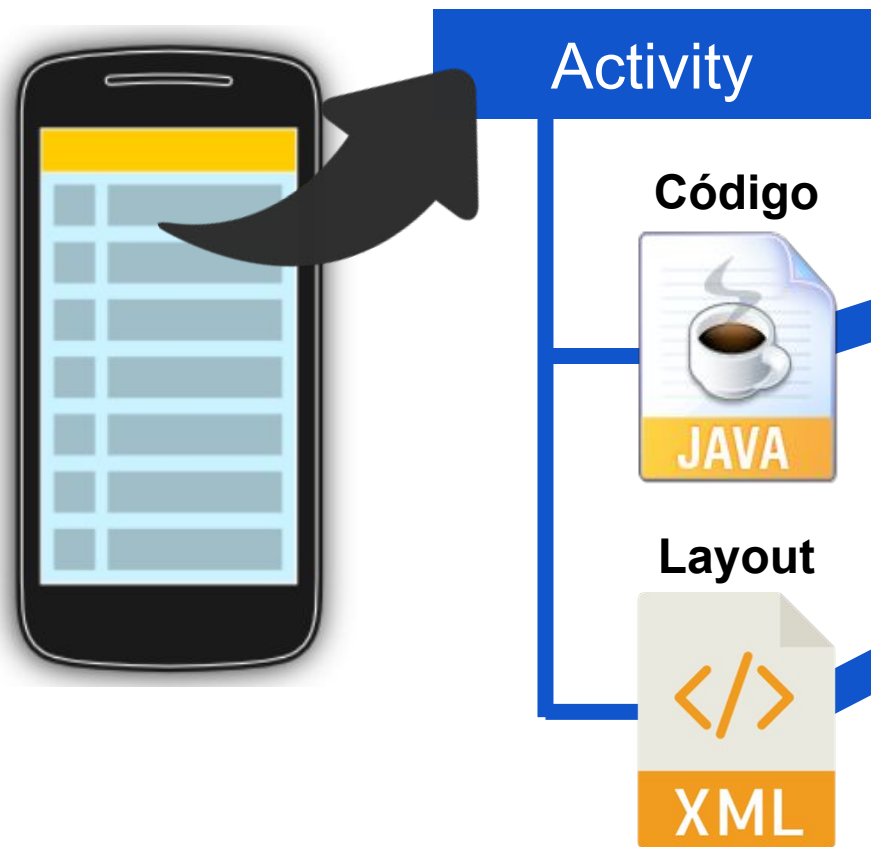
## Aula 4

# Terminologia Android

# Terminologia Android



# Terminologia Android



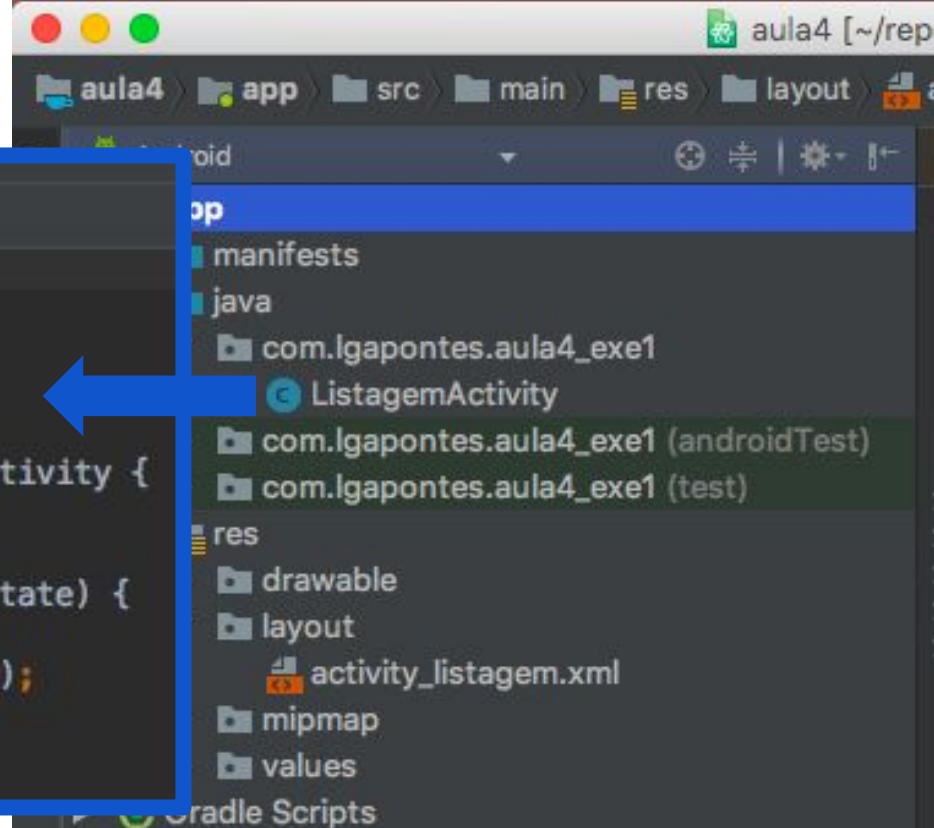
# Terminologia Android

```
vity_listagem.xml x ListagemActivity.java x
package com.lgapontes.aula4_exe1;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class ListagemActivity extends AppCompatActivity {

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



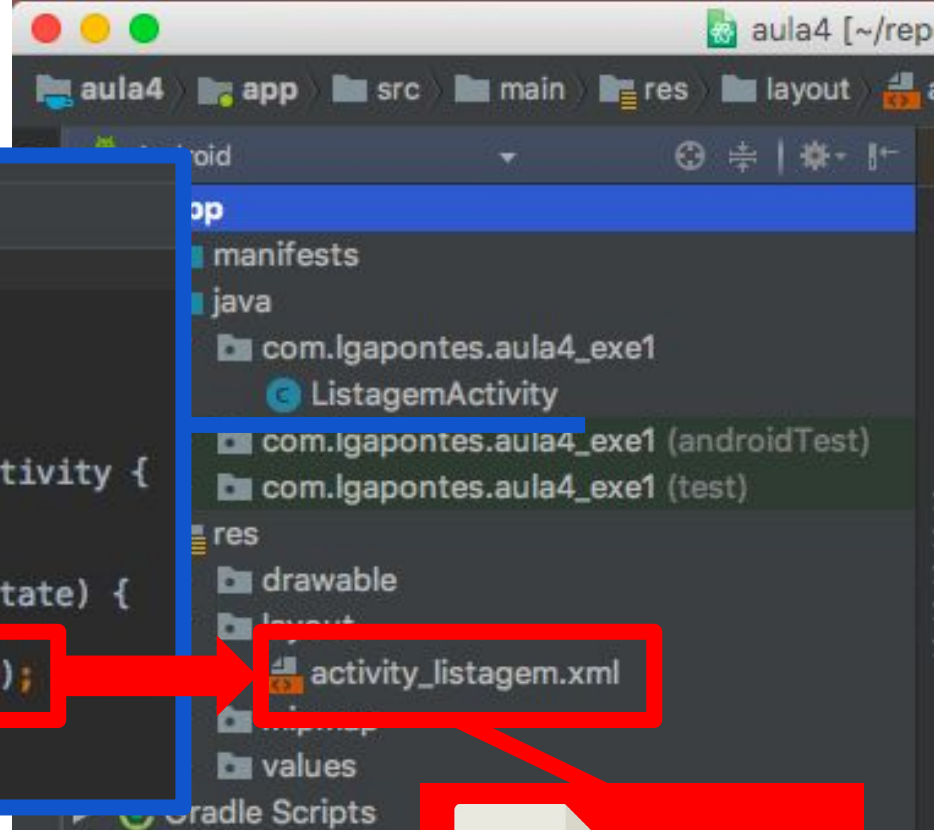
# Terminologia Android

```
vity_listagem.xml x ListagemActivity.java x
package com.lgapontes.aula4_exe1;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

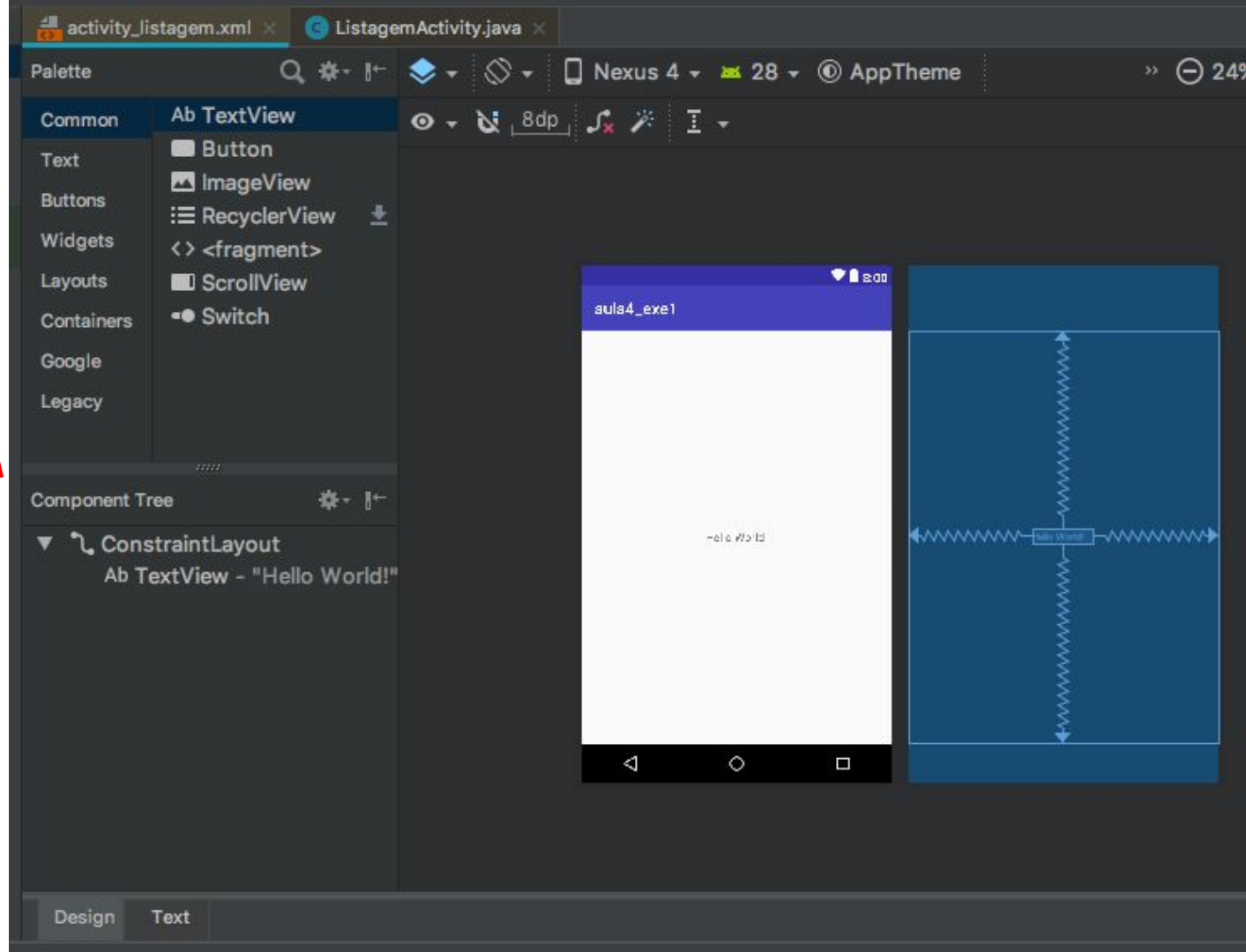
public class ListagemActivity extends AppCompatActivity {

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

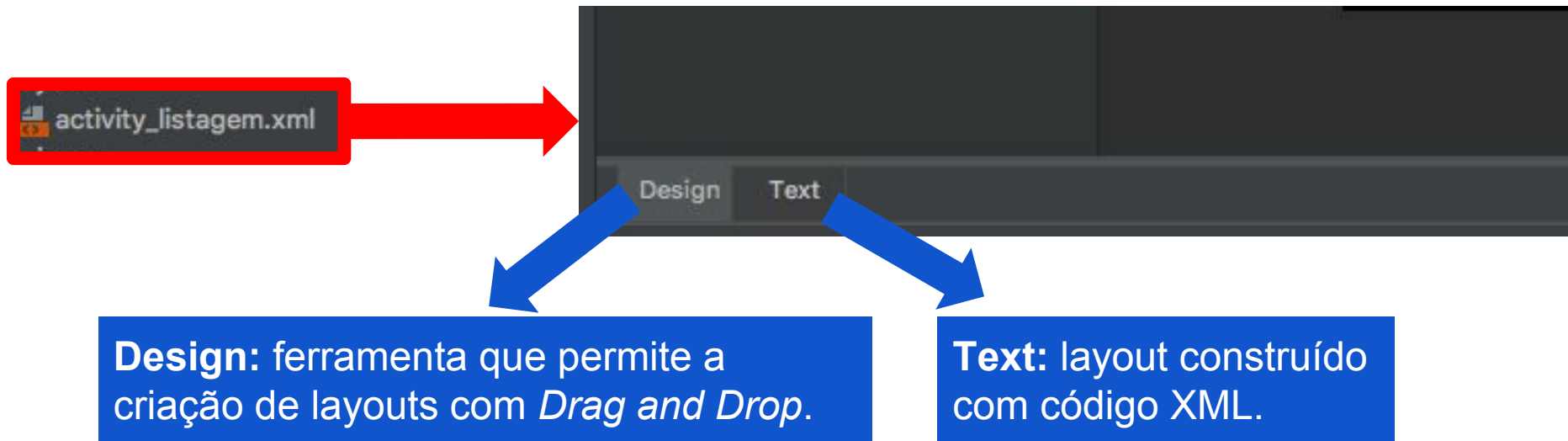


Layout

activity\_listagem.xml



# Terminologia Android



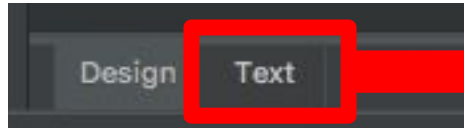
Criar layouts pelo *Design* é mais simples. **Porém**, é muito importante entender o que ocorre com o layout XML em código, pois isso vai nos ajudar a entender como manipulá-lo pelo código Java, quando necessário.



O melhor layout para usar a aba Design é o **ConstraintLayout**.



# Terminologia Android

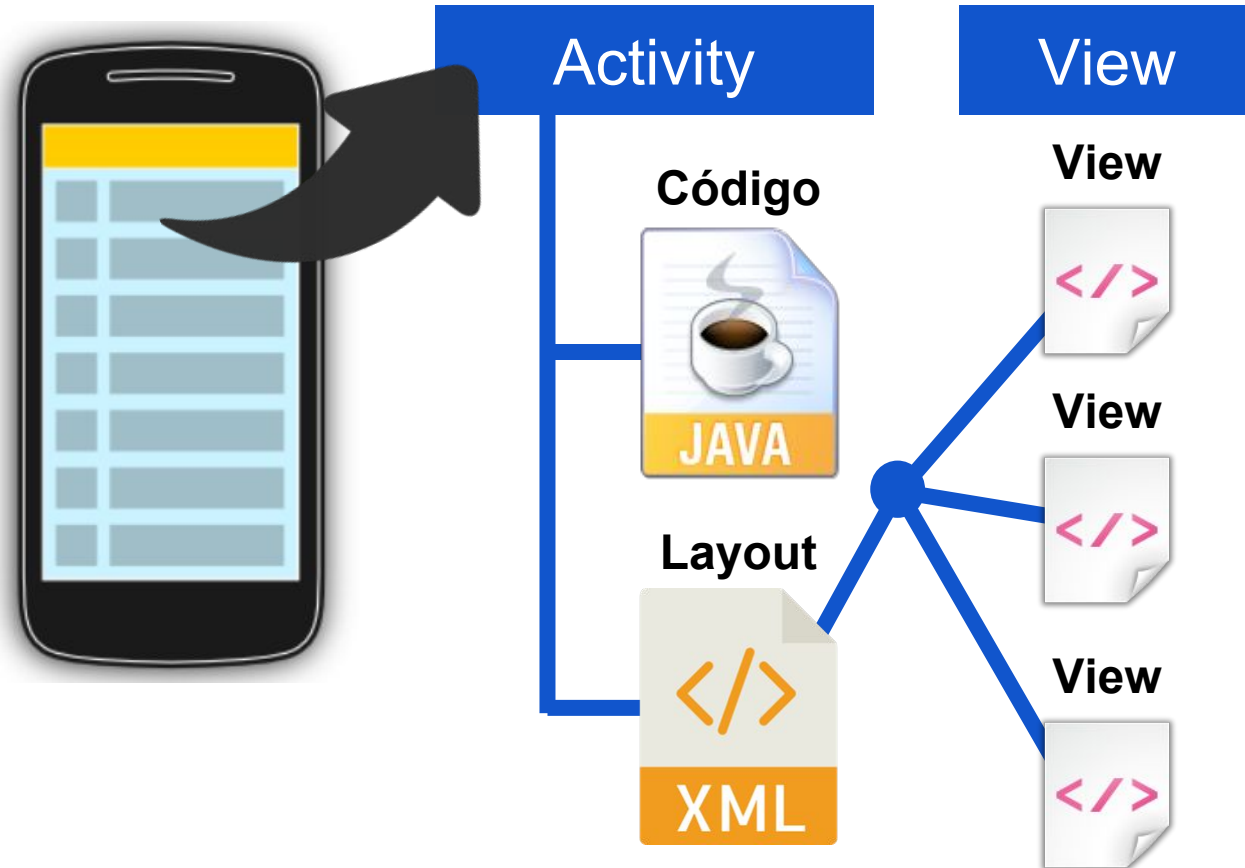


```
ivity_listagem.xml x ListagemActivity.java x
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ListagemActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

# Terminologia Android



# Terminologia Android

```
ivity_listagem.xml x ListagemActivity.java x
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ListagemActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Layout



View



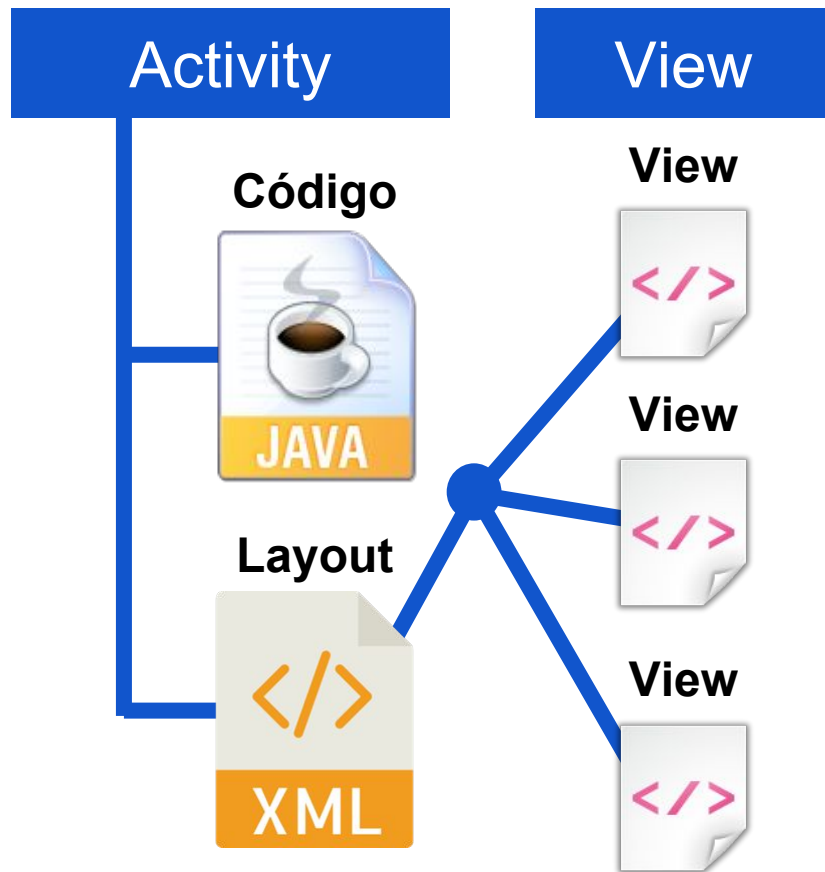
# Resumo

**Activity:** seria o equivalente às telas no desenvolvimento desktop.

**Código Java:** Relacionado a um (ou mais) arquivo de layout XML. Serve como ponto de partida para o Android inicializar a Activity, além de permitir manipulação dos dados do layout.

**Código XML:** representação do layout em formato XML. Ele organiza os **Views** (componentes de tela) de forma hierárquica e é responsável por *receber* os eventos de interação com a tela.

**View:** são os componentes da tela



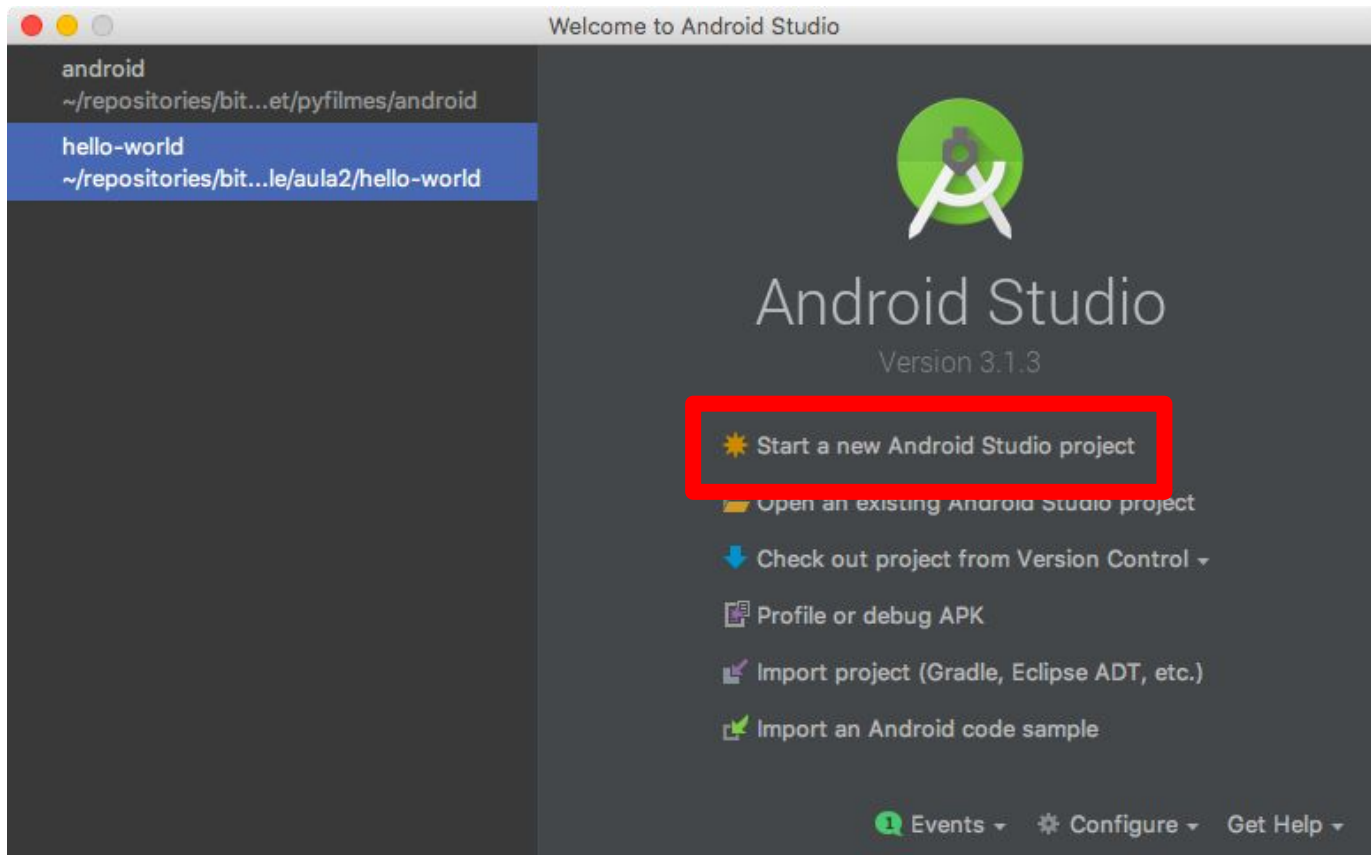
# Exercício em Sala

Vamos criar um projeto para listar o nome de pessoas.

Crie um novo projeto no Android Studio chamado *ListagemPessoas*

- Ele precisa ser compatível com versões anteriores do Android
- Crie-o com uma *Empty Activity*
- Execute o projeto em um emulador

# Resolvendo o Exercício



# Resolvendo o Exercício

Application name

ListagemPessoas

Company domain

lgapontes.com

Project location

/Users/lgapontes/repositories/bitbucket/aulas/desenvolvimento-mobile/aula4/ListagemPessoas

Package name

com.lgapontes.aula4\_listagempessoas

Done

☐ Include C++ support

☐ Include Kotlin support

# Resolvendo o Exercício

## Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

### ☒ Phone and Tablet

API 15: Android 4.0.3 (IceCreamSandwich)

By targeting **API 15 and later**, your app will run on approximately **100%** of devices. [Help me choose](#)

☐ Include Android Instant App support

### ☐ Wear

API 21: Android 5.0 (Lollipop)

### ☐ TV

API 21: Android 5.0 (Lollipop)

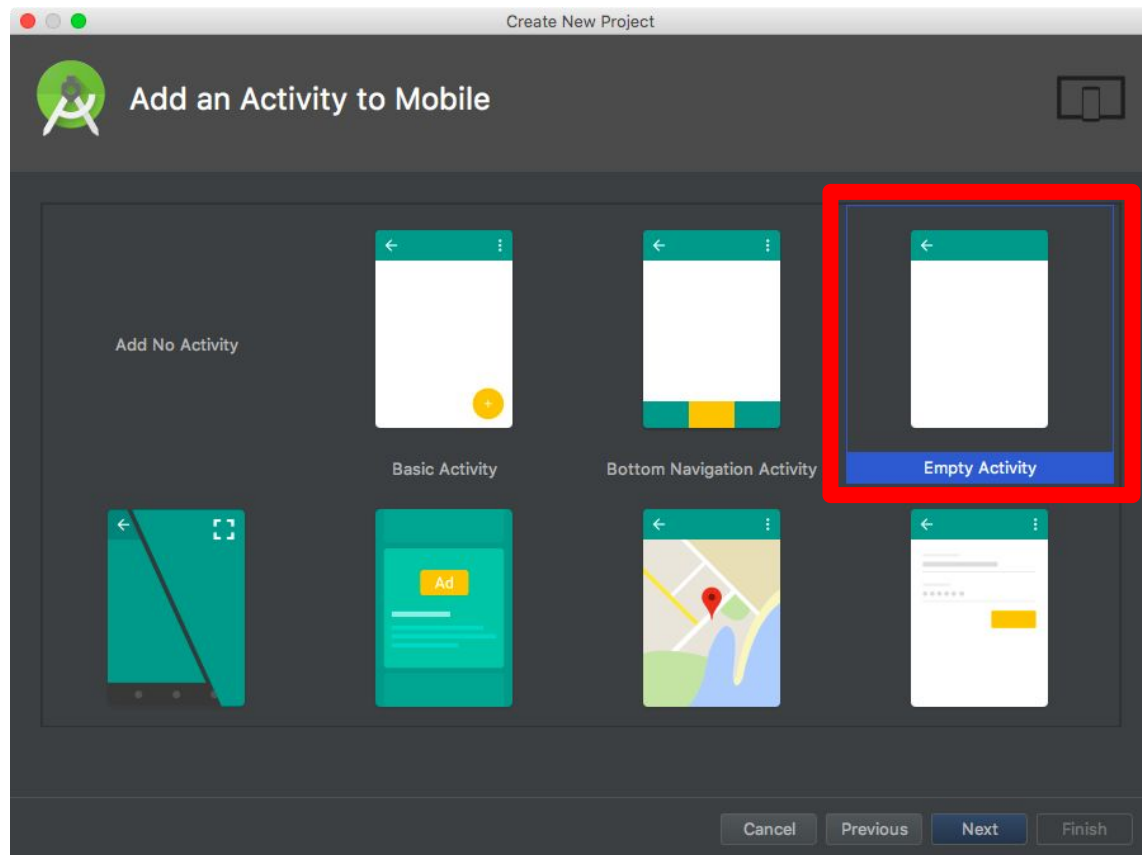
### ☐ Android Auto

### ☐ Android Things

API 24: Android 7.0 (Nougat)



# Resolvendo o Exercício



# Resolvendo o Exercício

Código



Creates a new empty activity

Activity Name:

ListagemActivity

☒ Generate Layout File

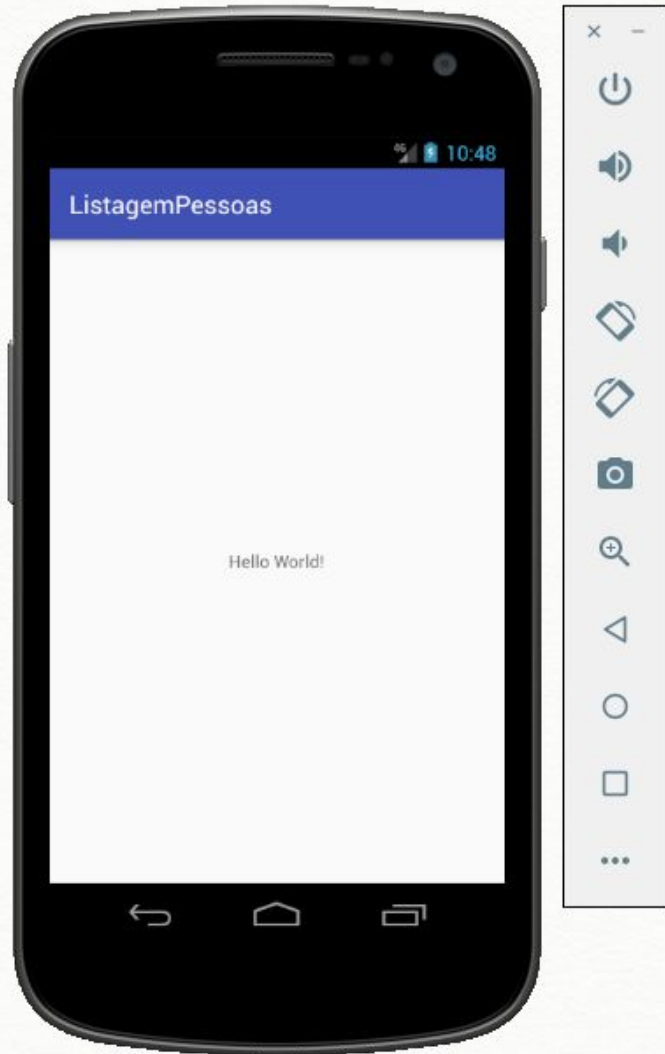
Layout Name:

activity\_listagem

☒ Backwards Compatibility (AppCompat)

Layout





# Listagem de Pessoas

# Arquivo *activity\_listagem.xml*

Vamos começar limpando o arquivo (removendo os namespaces **app** e **tools** e retirando o atributo *context*) e trocando o *ConstraintLayout* para *LinearLayout*. Aproveite e altere o atributo *text* do **TextView** para um nome qualquer.



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

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

</LinearLayout>
```



## Arquivo *activity\_listagem.xml*

```
ml version="1.0" encoding="utf-8"?>
nearLayout
xmlns:android="http://schemas.android.com"
android:layout_width="match_parent"
android:layout_height="match_parent">

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

linearLayout>
```

Os atributos **layout\_width** e **layout\_height** são utilizados para definir, respectivamente, o comprimento e altura que o elemento (Layout ou View) possuem na tela.

O valor **match\_parent** faz com que o elemento ocupe todo o espaço disponível no pai. Como o *LinearLayout* não tem pai, na prática ele vai ocupar toda a tela.

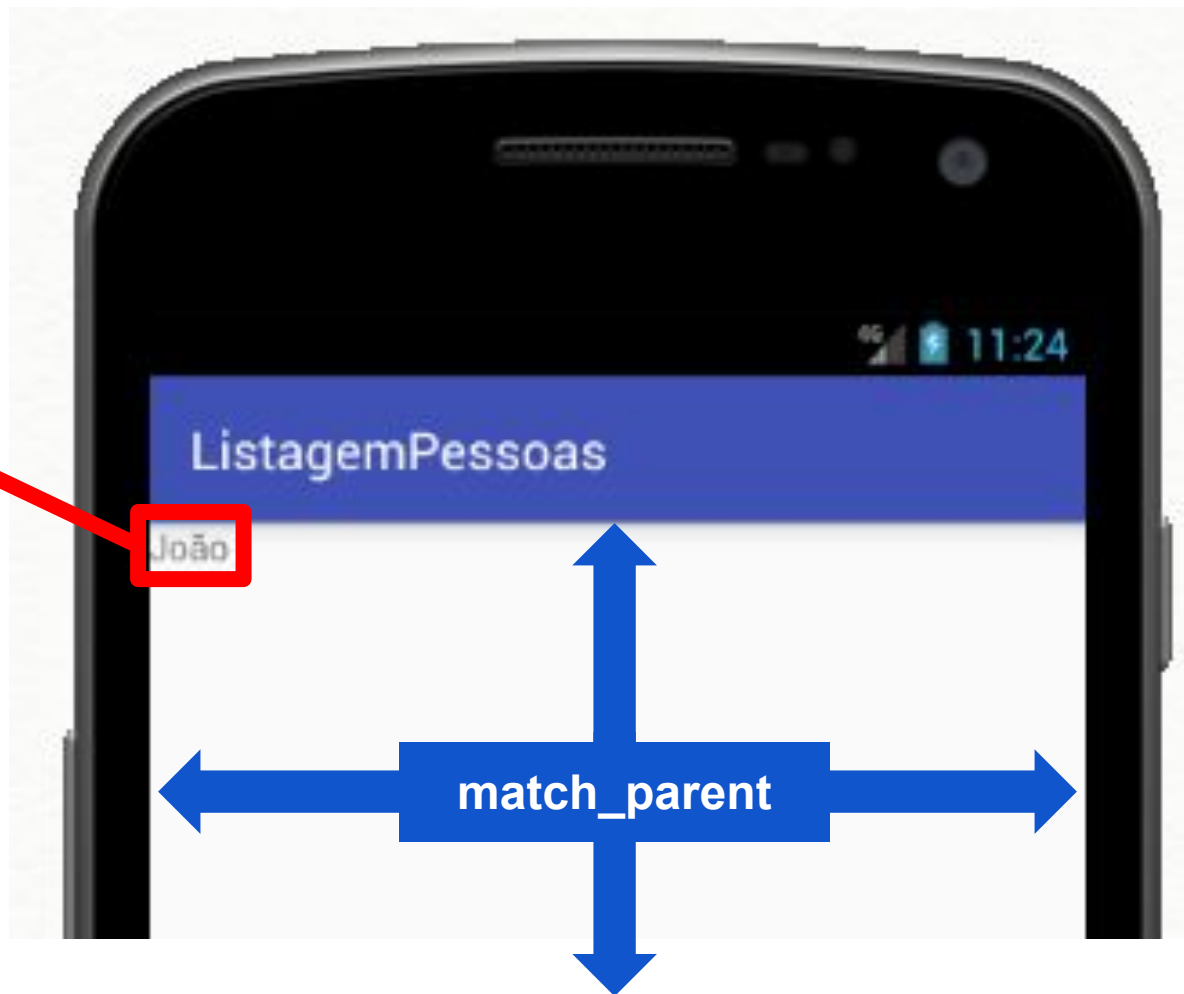
O valor **wrap\_content** faz com que o elemento ocupe o espaço necessário para exibir seu conteúdo.

**wrap\_content**

João

ListagemPessoas





**match\_parent**





# Medidas utilizadas no Android

Além do **match\_parent** e **wrap\_content**, existem medidas que podem ser utilizadas para definir o comprimento e altura dos elementos.

px	Pixels que o elemento terá. <b>Atenção:</b> usar essa medida provocará problemas de layout em dispositivos com tamanhos e densidades de telas diferentes.	
dp	Esta unidade, cujo significado é <b>Density-independent Pixels</b> , é baseada na densidade da tela. Um dp é um pixel independente de densidade que corresponde ao tamanho físico de um pixel em 160 dpi.	
sp	Com significado <b>Scale-independent Pixels</b> , esta unidade é equivalente à dp, porém leva em consideração o tamanho da fonte escolhida pelo usuário. Deve ser usada para textos.	
in	Polegadas baseadas no tamanho físico da tela.	
mm	Milímetros baseados no tamanho físico da tela.	
pt	1/72 de uma polegada. Também é baseado no tamanho físico da tela.	

## Arquivo *activity\_listagem.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="João"
        android:textSize="40sp" />
</LinearLayout>
```

Utilize o atributo **textSize** para definir o tamanho da fonte.



# Arquivo *activity\_listagem.xml*

Vamos incluir outro **TextView** para exibir o nome "Maria".

```
activity_listagem.xml ListagemActivity.java
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="João"
        android:textSize="40sp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Maria"
        android:textSize="40sp" />

</LinearLayout>
```



## Arquivo *activity\_listagem.xml*

Lembre-se que o atributo **layout\_width** definido com o valor *wrap\_content* faz com que o elemento tenha o tamanho de seu conteúdo.

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="João"  
    android:textSize="40sp" />  
  
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Maria"  
    android:textSize="40sp" />
```



## Arquivo *activity\_listagem.xml*

Vamos ajustá-los para *match\_parent* para que ocupem todo o espaço disponível no componente pai.

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="João"  
    android:textSize="40sp"/>  
  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Maria"  
    android:textSize="40sp"/>
```



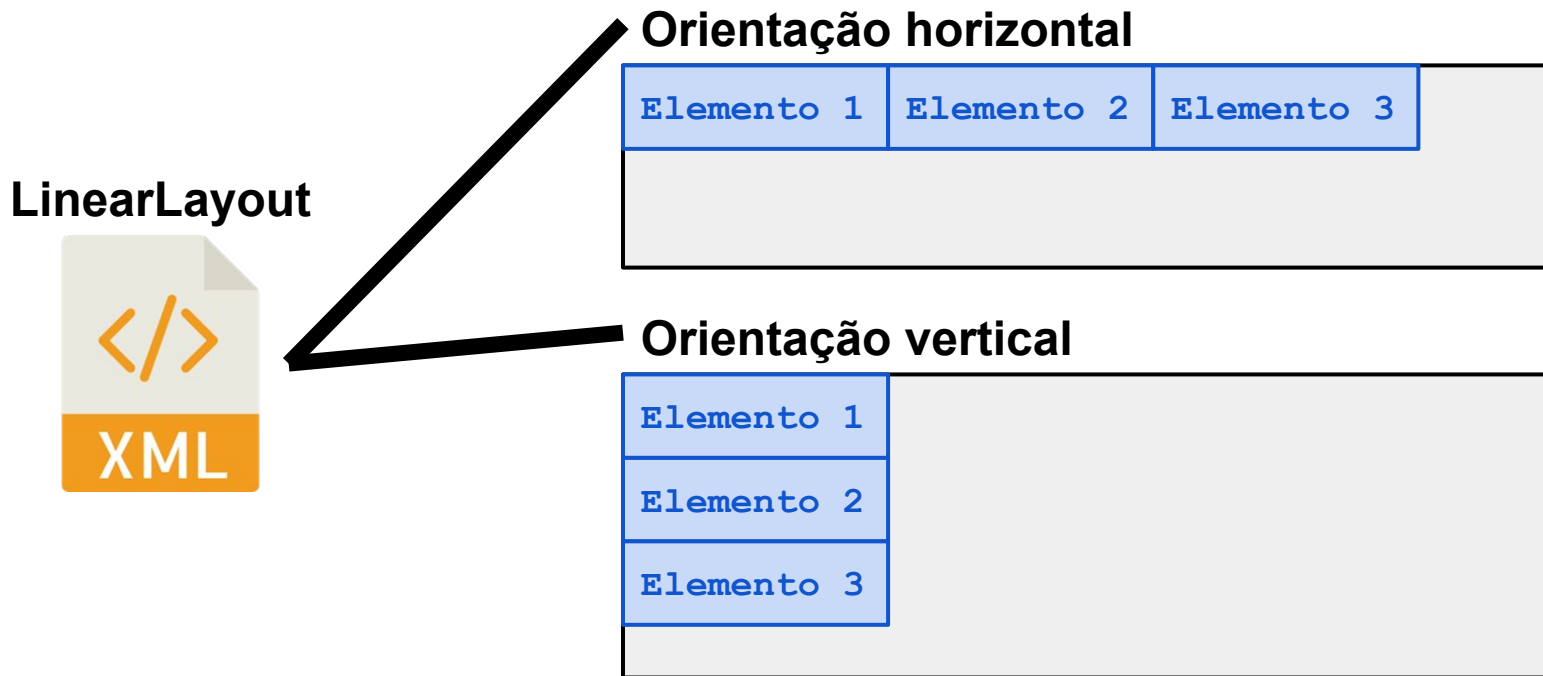
?

E a Maria?



# LinearLayout

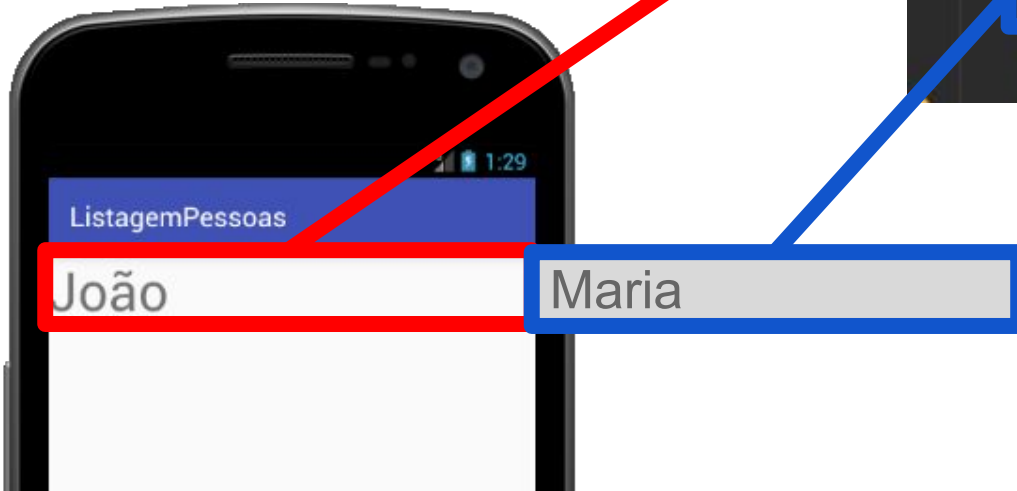
O *LinearLayout*, por default, exibe seus componentes com uma orientação horizontal. Através do atributo *orientation*, podemos alterar esse comportamento fazendo-o exibir seus elementos na vertical.



# LinearLayout

O nome "Maria" não foi exibido porque como ambos estão ocupando todo o conteúdo disponível no elemento pai, o nome "João" *empurrou* o nome "Maria" para fora da tela.

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="João"  
    android:textSize="40sp"/>  
  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Maria"/>
```



# Arquivo *activity\_listagem.xml*

Vamos alterar a orientação do **LinearLayout** para resolver o problema.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="João"
        android:textSize="40sp"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Maria"
        android:textSize="40sp"/>

</LinearLayout>
```





# Exercício em Sala

Altere o projeto *ListagemPessoas* com os seguintes passos:

- Coloque 5 nomes: *Seiya, Shiryu, Hyoga, Shun, Ikki*
- Os nomes devem ser exibidos verticalmente com tamanho de texto definido como 25sp
- Execute o projeto em um emulador

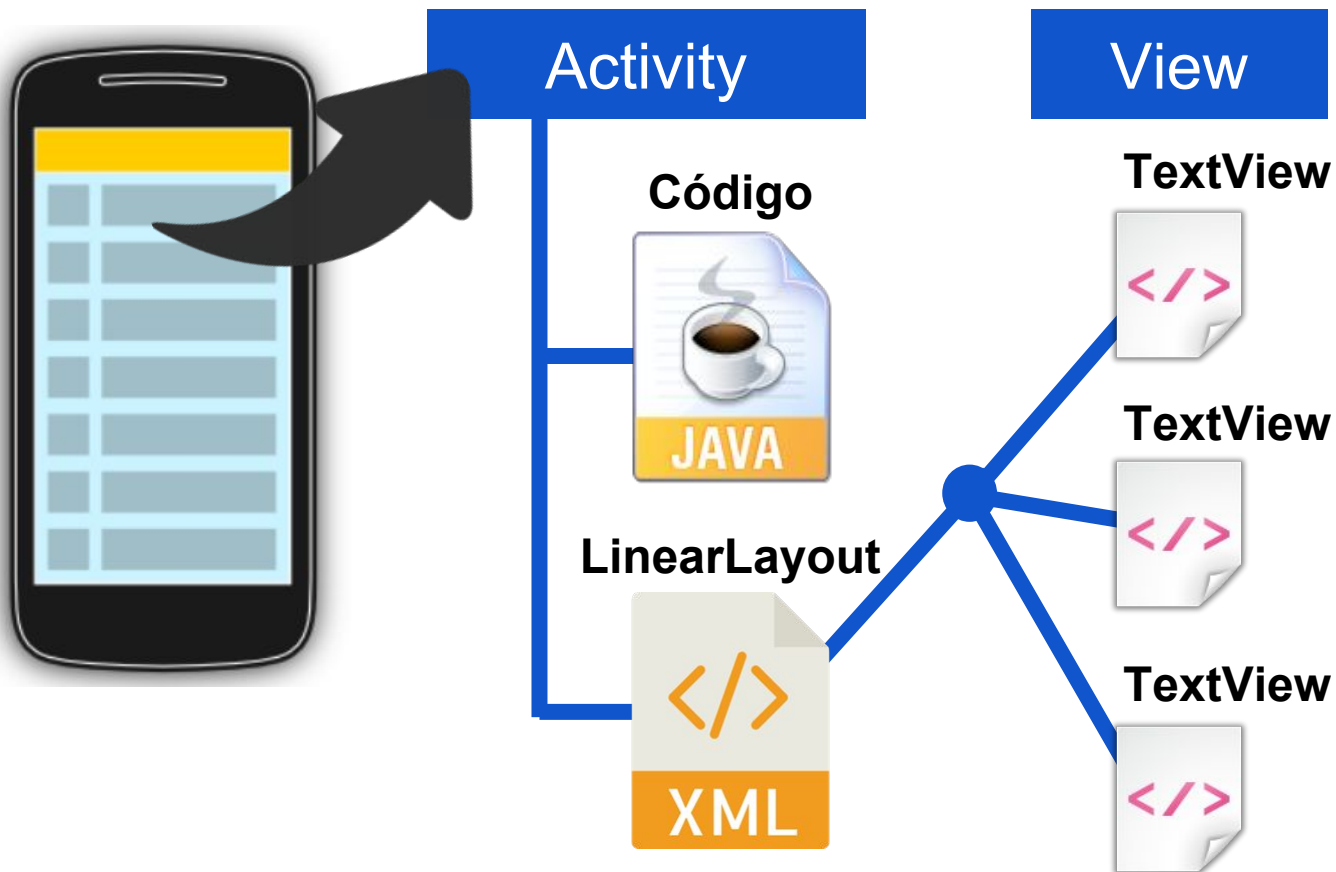
# Resolvendo o Exercício

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Seiya"
    android:textSize="25sp"/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Shiryu"
    android:textSize="25sp"/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Hyoga"
    android:textSize="25sp"/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Shun"
    android:textSize="25sp"/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Ikki"
    android:textSize="25sp"/>
```



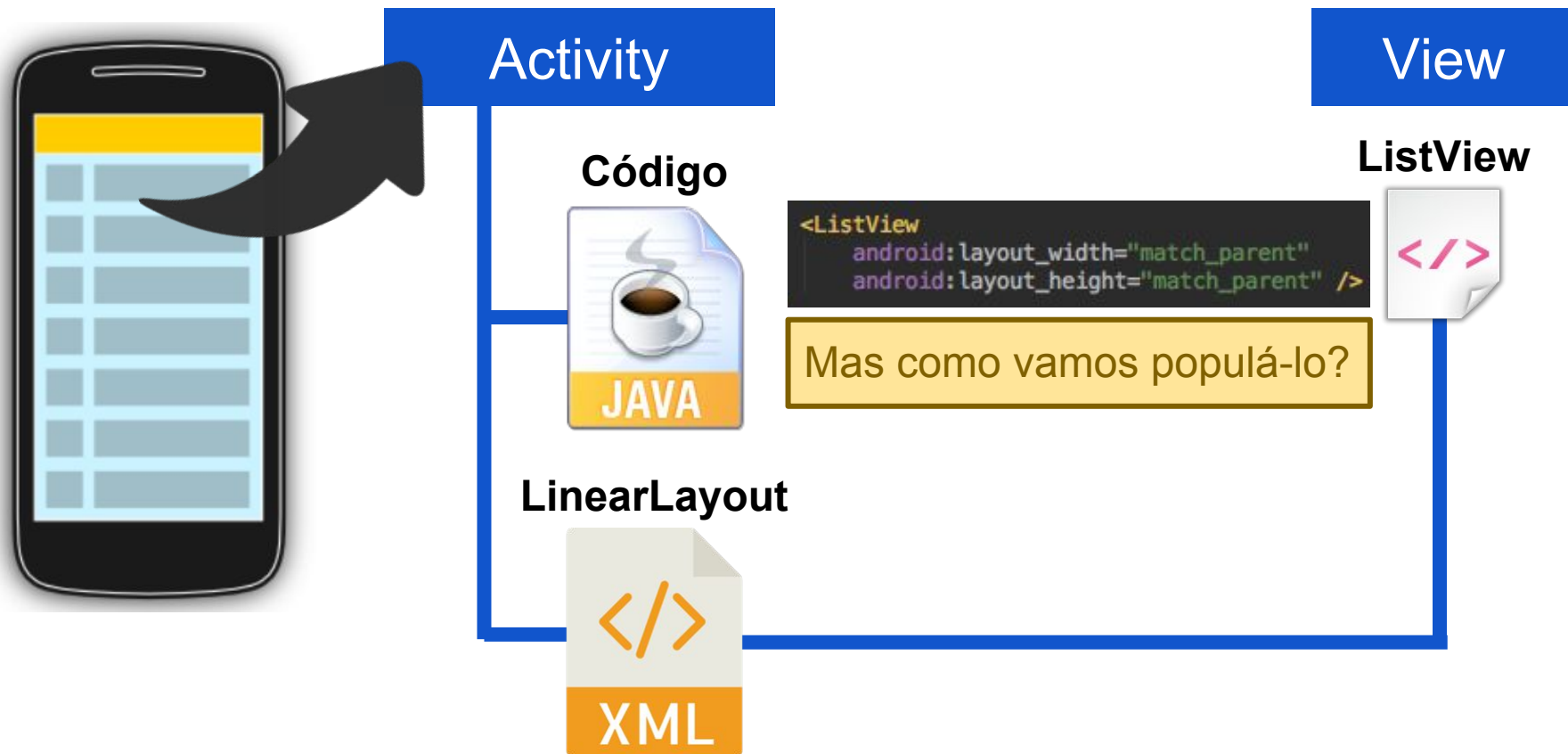
**E se fossem 100 nomes?**

# Devemos usar vários elementos TextView para uma lista?

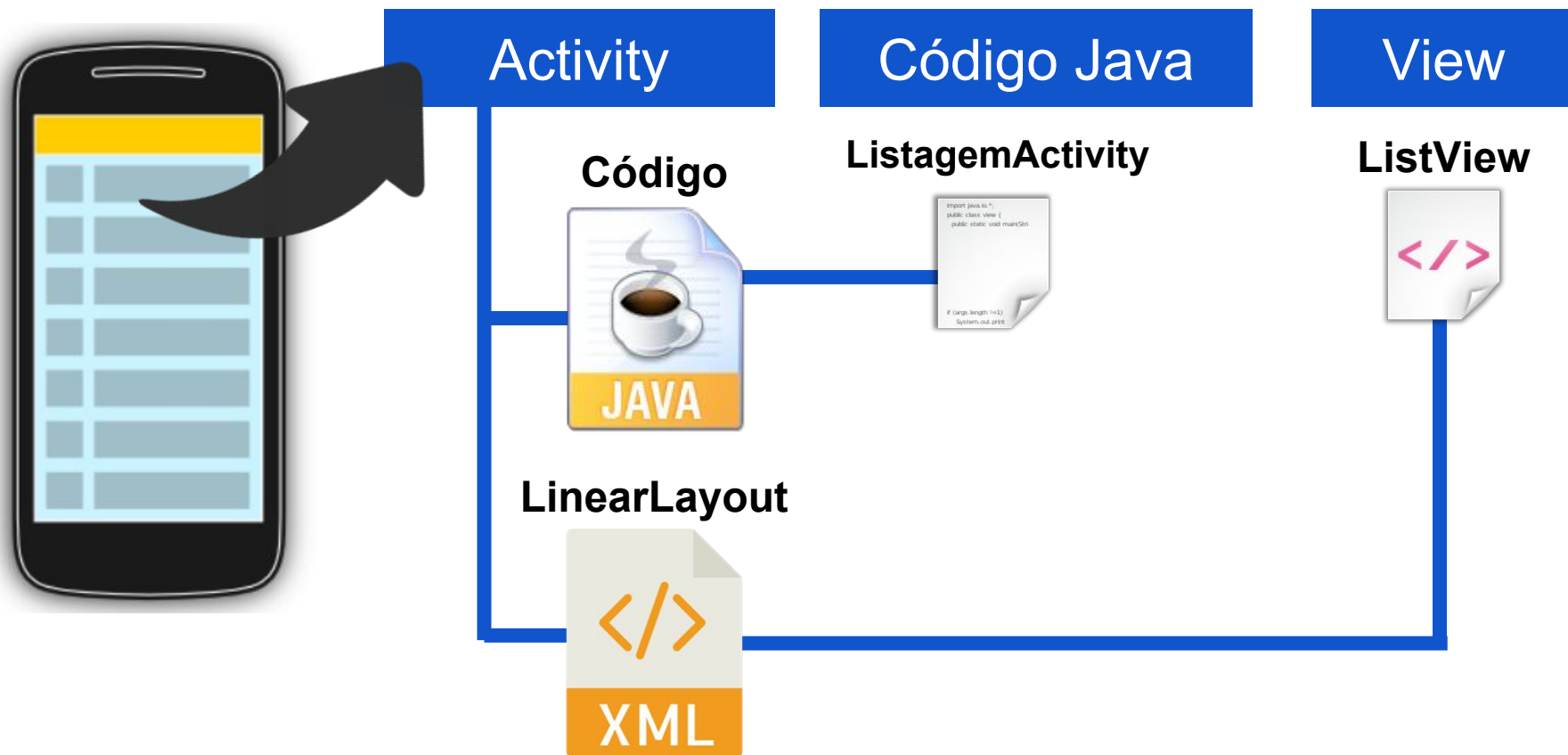


Da forma como fizemos, empilhando os elementos *TextView* um sobre o outro, é uma má prática.

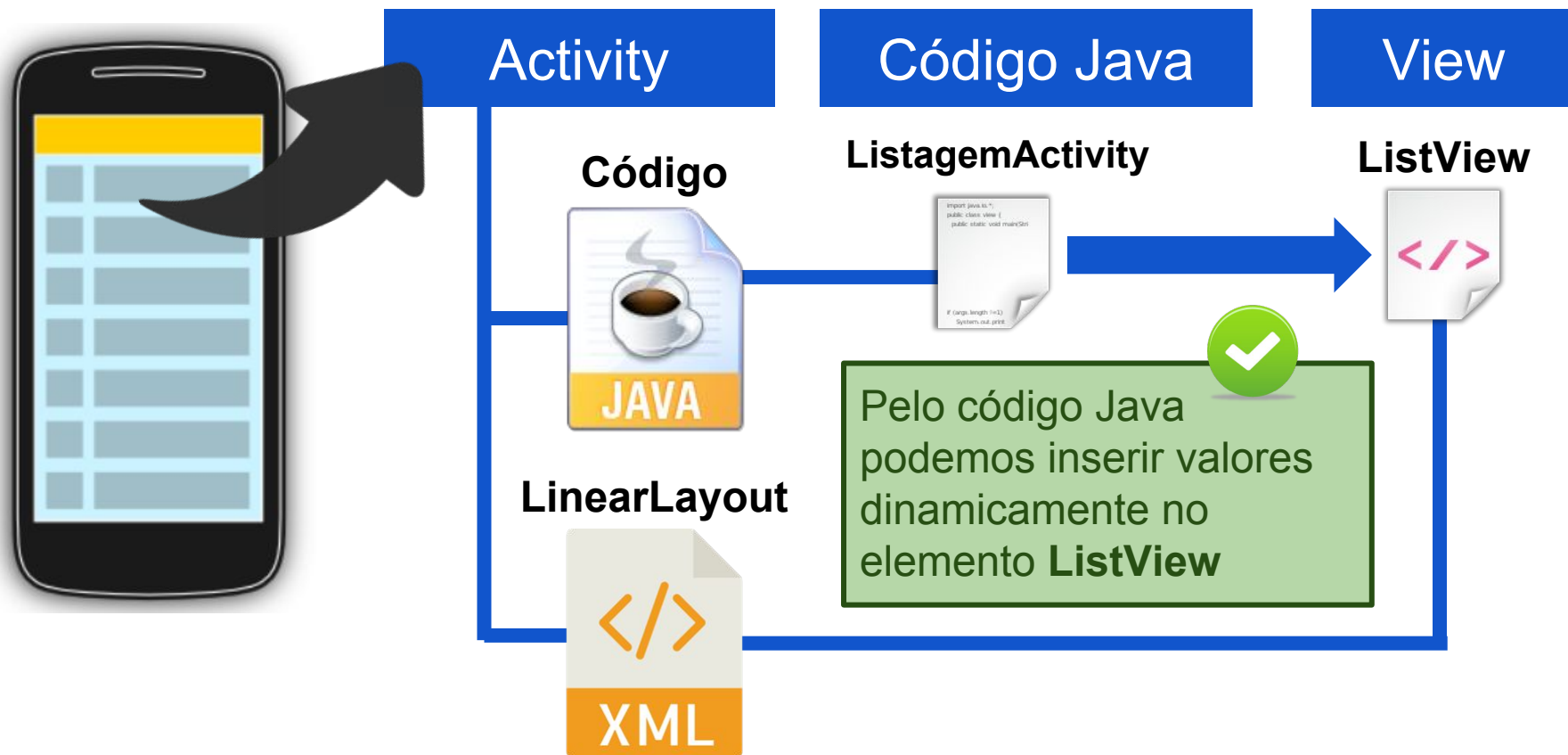
# Usando e populando o elemento ListView



# Usando e populando o elemento ListView

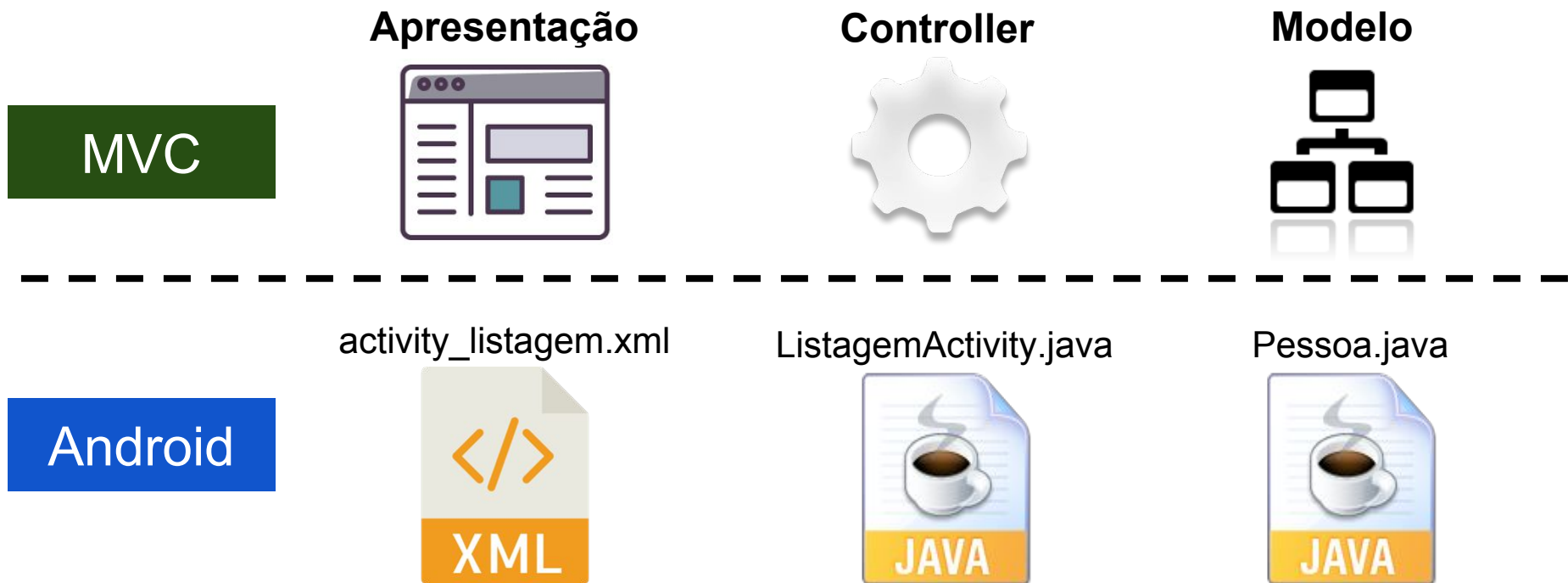


# Usando e populando o elemento ListView



# Como organizar os arquivos do projeto?

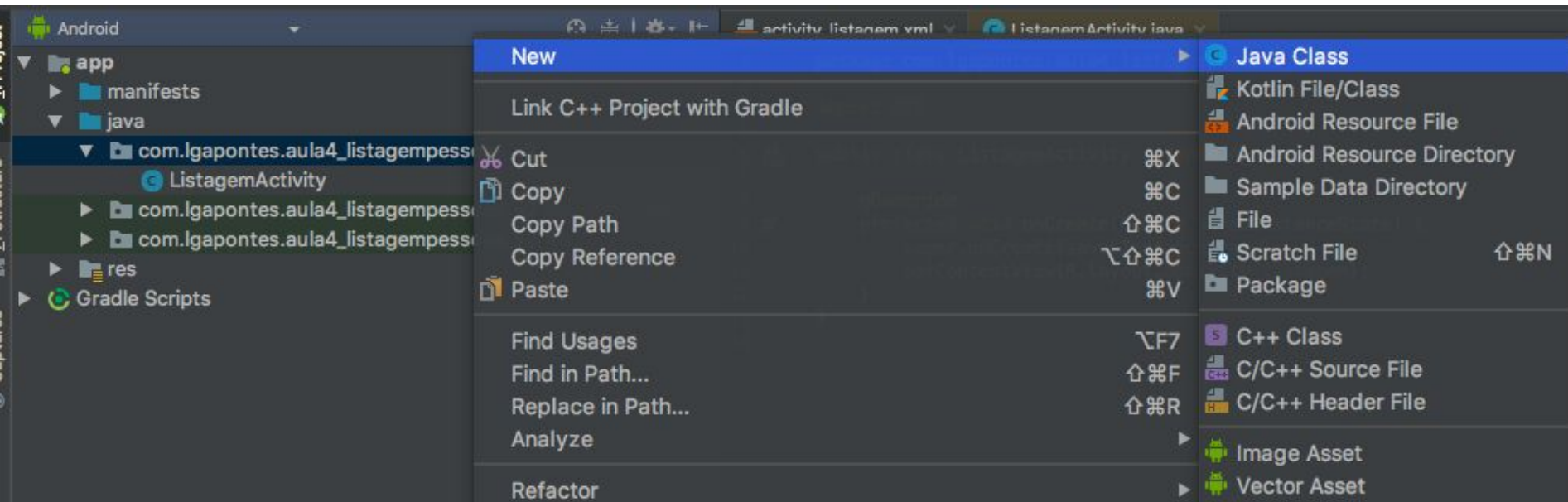
As boas práticas de desenvolvimento consistem em organizar o código-fonte em responsabilidades. No Android também podemos organizar as responsabilidades em **Modelo de Domínio, Controller e Apresentação**.





# Criando o arquivo *Pessoa.java*

Vamos criar uma nova classe Java para representar as pessoas.



Botão direito no  
pacote principal

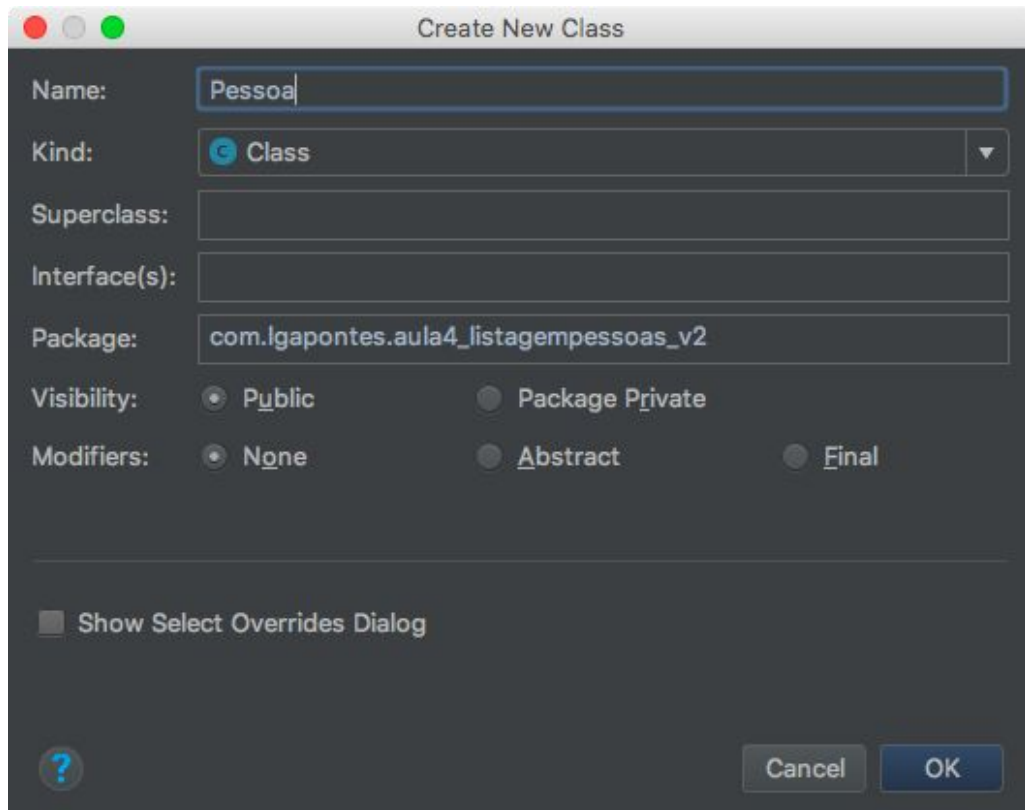


Opção *New*



Opção *Java Class*

# Criando o arquivo *Pessoa.java*



Create New Class

Name:

Kind: ☒ Class ▼

Superclass:

Interface(s):

Package:

Visibility: ☒ Public ☐ Package Private

Modifiers: ☒ None ☐ Abstract ☐ Final

☐ Show Select Overrides Dialog

? Cancel OK

Defina o nome da classe como *Pessoa* e clique em *OK*

# Arquivo *Pessoa.java*

```
activity_listagem.xml × ListagemActivity.java ×  
package com.lgapontes.aula4_listagem;  
  
public class Pessoa {  
    private String nome;  
  
    public Pessoa(String nome) {  
        this.nome = nome;  
    }  
  
    @Override  
    public String toString() {  
        return this.nome;  
    }  
}
```

O código do arquivo *Pessoa.java* deve conter um atributo *nome*, um *constructor* para definir o valor do atributo, e um método para sobrescrever o método *toString()* da superclasse.

# Arquivo *ListagemActivity.java*

Dentro do objeto **ListagemActivity.java**, vamos criar um método *criarPessoas()* que será responsável por gerar uma *List* de nomes.

```
private List<Pessoa> criarPessoas() {  
    List<Pessoa> lista = new ArrayList<Pessoa>();  
  
    String[] nomes = new String[]{  
        "Seiya", "Shiryu", "Hyoga", "Shun", "Ikki"  
    };  
  
    for (String nome : nomes) {  
        Pessoa p = new Pessoa(nome);  
        lista.add(p);  
    }  
  
    return lista;  
}
```

## Dica Extra

Se estivéssemos usando a API 24, poderíamos utilizar recursos mais avançados do Java 8, reduzindo esse método para uma linha.

```
private List<Pessoa> criarPessoas() {  
    List<Pessoa> lista = new ArrayList<Pessoa>();  
  
    String[] nomes = new String[]{  
        "Seiya", "Shiryu", "Hyoga", "Shun", "Ikki"  
    };  
  
    for (String nome : nomes) {  
        Pessoa p = new Pessoa(nome);  
        lista.add(p);  
    }  
  
    return lista;  
}
```

```
List<Pessoa> pessoas = Arrays  
    .asList(new String[]{"Seiya", "Shiryu", "Hyoga", "Shun", "Ikki"})  
    .stream().map(x -> { return new Pessoa(x); }).collect(Collectors.toList());
```

# Arquivo *activity\_listagem.xml*

Agora o arquivo *activity\_listagem.xml* deve conter apenas o elemento **ListView**.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

# Arquivo *ListagemActivity.java*

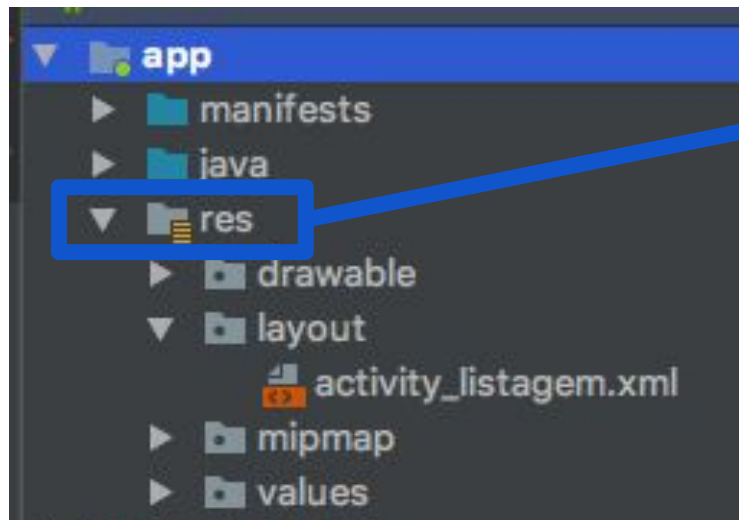
No método `onCreate()` vamos invocar o método `criarPessoas()` e guardar em uma variável local.

```
public class ListagemActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_listagem);  
  
        List<Pessoa> pessoas = criarPessoas();  
  
        // Como pegar o ListView???  
    }  
  
    private List<Pessoa> criarPessoas() {...}  
}
```

Mas e agora, como vamos referenciar o elemento **ListView**?



# Estrutura do projeto Android



Como vimos, nossos layouts estão organizados na pasta *layout* que fica disponível em uma pasta chamada *res*.

Este **res** (de *resources*) organiza todos os recursos que vamos utilizar em nossa aplicação Android.

**drawable**



**layout**



**mipmap**



**values**








**menu**





# Estrutura do projeto Android

 <b>drawable</b>	Pasta com imagens (arquivos ou definições de imagens) da aplicação. Podem conter imagens de diferentes densidades.
 <b>layout</b>	Pasta que contém os arquivos XML de organização dos layouts da aplicação.
 <b>mipmap</b>	Pasta com os ícones ( <i>launcher</i> ) da aplicação, organizados em subpastas de acordo com a densidade.
 <b>values</b>	Pasta para configuração de estilos, valores, cores, entre outras configurações.
 <b>menu</b>	Contém os arquivos XML para organização dos menus da aplicação.

# Utilizando a class **R**

O ambiente de desenvolvimento do Android fornece uma classe gerada automaticamente que permite referenciar os recursos. Esta classe chama-se *R*

ListagemActivity



**R**



**drawable**



**layout**



**mipmap**



**values**



**menu**



# Utilizando a class **R**

Ela permite também acessar os **views** presentes nos layouts a partir de IDs. Podemos definir esses IDs no arquivo XML através do atributo *id*.

ListagemActivity



**R**



`findViewById()`



Dessa forma podemos buscar nossa *ListView* e adicionar os dados.

**layout**



**View** ID=abc



**View** ID=xyz



...

# Arquivo *activity\_listagem.xml*

Vá no elemento **ListView** e defina seu ID através do atributo *id*.

```
<ListView  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:id="@+id/lista_pessoas" />
```

@+id/lista\_pessoas

@+id

Informa ao Android para incluir um novo ID.

lista\_pessoas

ID do elemento. Depois veremos boas práticas para nomenclatura dos IDs da aplicação como um todo.

# Arquivo *ListagemActivity.java*

```
public class ListagemActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_listagem);  
  
        List<Pessoa> pessoas = criarPessoas();  
        ListView listView = (ListView) findViewById(R.id.lista_pessoas);  
  
        // Agora falta adicionar a lista...  
    }  
  
    private List<Pessoa> criarPessoas() {...}  
}
```

Veremos agora como adicionar uma lista no **ListView**

# Trabalhando com a classe *ArrayAdapter*

A classe **ListView** não possui métodos para incluir valores de uma lista. Para isso, o Android oferece uma classe chamada **ArrayAdapter**.

O **ArrayAdapter** nada mais é do que um adaptador que permite definir os valores do **ListView** baseando-se em um padrão de exibição.

O Android oferece alguns padrões prontos. Neste exemplo vamos utilizar o padrão chamado *simple\_list\_item\_1*

```
ArrayAdapter<Pessoa> adapter = new ArrayAdapter<Pessoa>(
    context: this,
    android.R.layout.simple_list_item_1,
    pessoas
);
listView.setAdapter(adapter);
```

# Arquivo *ListagemActivity.java*

```
public class ListagemActivity extends AppCompatActivity {

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

        List<Pessoa> pessoas = criarPessoas();
        ListView listView = (ListView) findViewById(R.id.lista_pessoas);

        ArrayAdapter<Pessoa> adapter = new ArrayAdapter<Pessoa>(
            context: this,
            android.R.layout.simple_list_item_1,
            pessoas
        );
        listView.setAdapter(adapter);
    }

    private List<Pessoa> criarPessoas() {...}

}
```

Ao executar o projeto, veremos o *ListView* populado!





# Exercício em Sala

Altere o projeto *ListagemPessoas* com os seguintes passos:

- Crie a classe Pessoa com os atributos *nome* e *sobrenome*.
- Crie uma lista com 15 pessoas.
- Exiba essa lista na tela do aplicativo através de um **ListView**, onde cada linha deve mostrar o *nome* e o *sobrenome* da Pessoa.
- Execute o projeto. Veja que o **ListView** já habilita o scroll da tela por default.

# Resolvendo o Exercício

```
public class Pessoa {  
  
    private String nome;  
    private String sobrenome;  
  
    public Pessoa(String nome, String sobrenome) {  
        this.nome = nome;  
        this.sobrenome = sobrenome;  
    }  
  
    @Override  
    public String toString() {  
        return this.nome + " " + sobrenome;  
    }  
  
}
```

# Resolvendo o Exercício

```
private List<Pessoa> criarPessoas() {  
    Pessoa[] lista = new Pessoa[]{  
        new Pessoa( nome: "João", sobrenome: "da Silva"),  
        new Pessoa( nome: "Maria", sobrenome: "da Silva"),  
        new Pessoa( nome: "José", sobrenome: "da Silva"),  
        new Pessoa( nome: "Pedro", sobrenome: "da Silva"),  
        new Pessoa( nome: "João", sobrenome: "de Oliveira"),  
        new Pessoa( nome: "Maria", sobrenome: "de Oliveira"),  
        new Pessoa( nome: "José", sobrenome: "de Oliveira"),  
        new Pessoa( nome: "Pedro", sobrenome: "de Oliveira"),  
        new Pessoa( nome: "Tiago", sobrenome: "de Mattos"),  
        new Pessoa( nome: "Manoel", sobrenome: "de Mattos"),  
        new Pessoa( nome: "José", sobrenome: "de Mattos"),  
        new Pessoa( nome: "Joaquim", sobrenome: "de Mattos"),  
        new Pessoa( nome: "Alex", sobrenome: "de Mattos"),  
        new Pessoa( nome: "Fulano", sobrenome: "de Mattos"),  
        new Pessoa( nome: "Ciclano", sobrenome: "de Mattos")  
    };  
  
    return Arrays.asList(lista);  
}
```

# Resolvendo o Exercício



Obrigado!