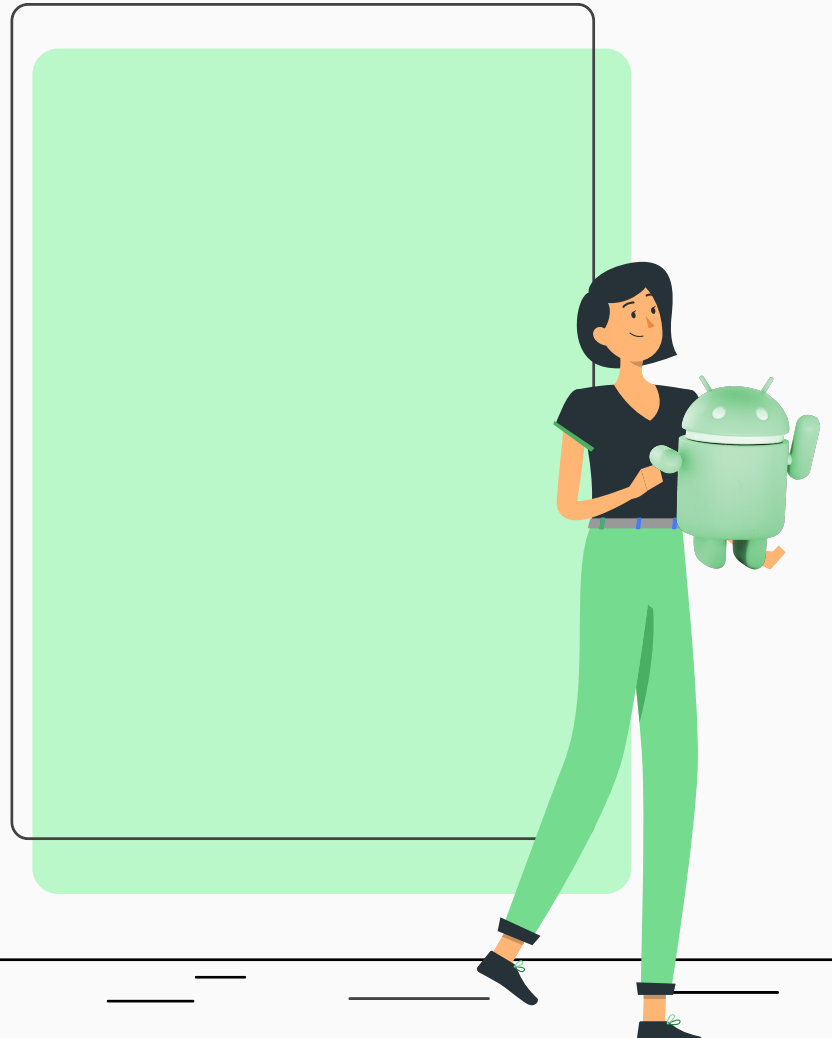
AULA 7

Criando Interfaces &
Refatorando projetos



01
LINEAR LAYOUT

Criar layouts básicos

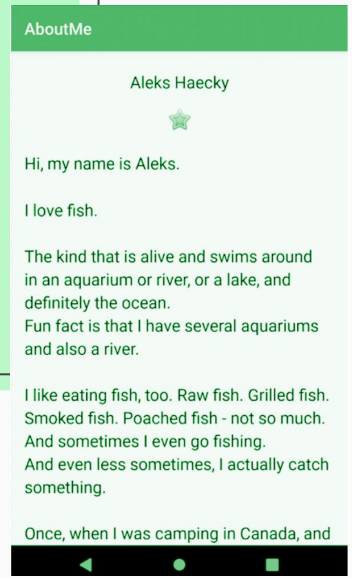
02
SUPER TRUNFO

Orientação à objetos

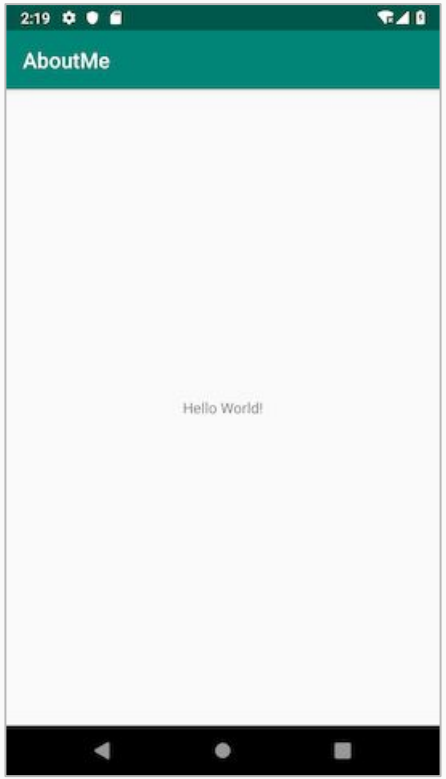
01 LINEAR LAYOUT

ABOUTME

No aplicativo AboutMe, você pode mostrar fatos interessantes sobre você ou pode personalizar o aplicativo para um amigo, membro da família ou animal de estimação. Este aplicativo exibe um nome, um botão CONCLUÍDO, uma imagem de estrela e algum texto rolável.



DOJO - NOVO PROJETO



- Criar novo projeto, AboutMe a partir de uma *activity* vazia
- Executar projeto no emulador
- Iniciar repo Git
- Fazer primeiro push

NOVO PROJETO

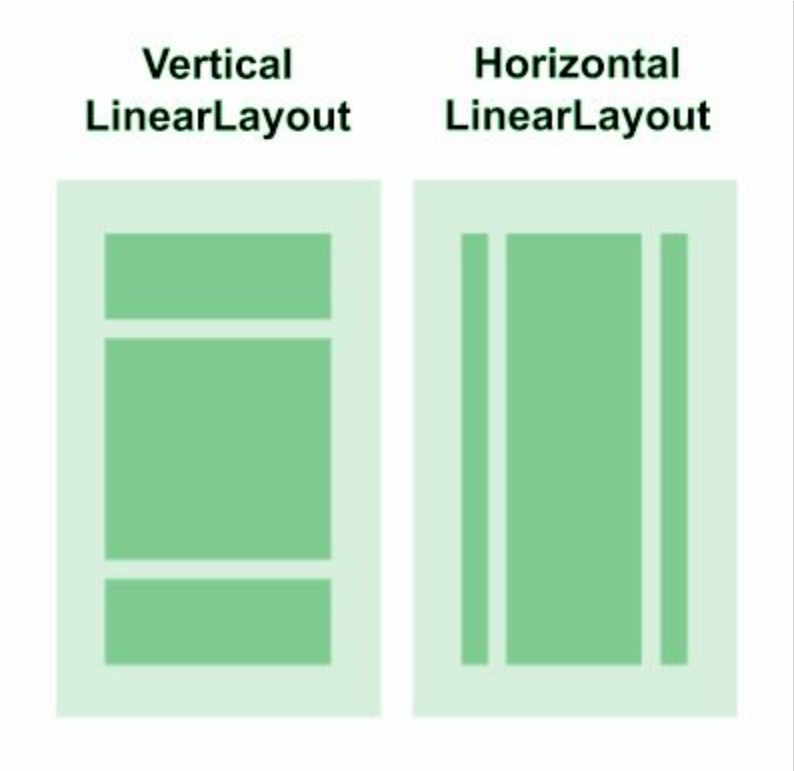
O template de activty vazia cria uma única activty, **Mainactivity.kt**. O template também cria um arquivo de layout chamado **activity_main.xml**. O arquivo de layout tem **ConstraintLayout** como seu **ViewGroup** raiz, e tem um único **TextView** como seu conteúdo.

VIEWGROUPS

Um **ViewGroup** é um elemento de interface que pode conter elementos de interface filhos, que são outros elementos e/ou grupos de elementos. As chamadas *views* que compõem um layout são organizadas como uma hierarquia de *views* com um grupo de *views* como raiz.

Em um grupo de *views* LinearLayout, os elementos da IU são organizados horizontal ou verticalmente.

VIEWGROUPS



DOJO - LINEAR LAYOUT

- Abrir arquivo *activity_main.xml*
- Alterar *rootView* de ConstraintLayout para LinearLayout
- Remover TextView
- Commitar alterações

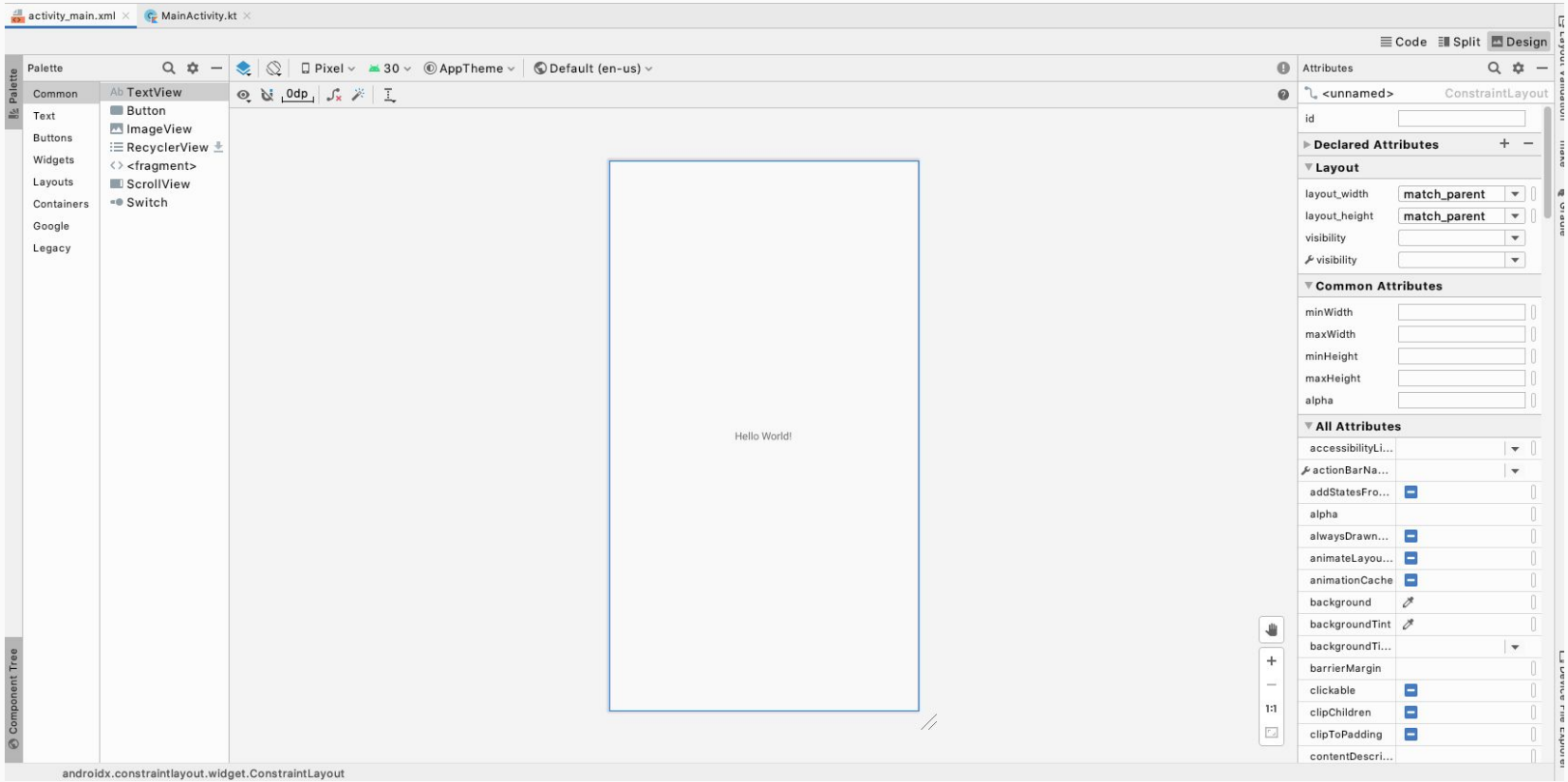
ANTES

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.androi
    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=".MainActivity">

    <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" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

DEPOIS

```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"

</LinearLayout>
```

DOJO - LAYOUT EDITOR



DOJO - LAYOUT EDITOR

1. Editor de design: exibe uma representação visual do layout da tela. O editor de design é a parte principal do Editor de Layout.
2. Barra de ferramentas: fornece botões para configurar a aparência do seu layout no editor de design e para alterar alguns atributos do layout:
    - Use o Design para uma visualização do mundo real do seu layout.
    - Use o Blueprint para ver apenas os contornos de cada visualização.
    - Use Design + Blueprint para ver os dois monitores lado a lado.
3. Paleta: fornece uma lista de *Views* e ViewGroups que você pode arrastar para o seu layout.
4. Atributos: Mostra os atributos da *View* ou *ViewGroups* atualmente selecionado.
5. Exibe a hierarquia de layout como uma árvore de views.

DOJO - LAYOUT EDITOR

1. Mude para o modo *code*

O LinearLayout possui os atributos obrigatórios layout_height, layout_width e orientação, que é vertical por padrão.

2. Mude para o modo design selecionado
3. Adicione um TextView no layout
4. Commitar alterações

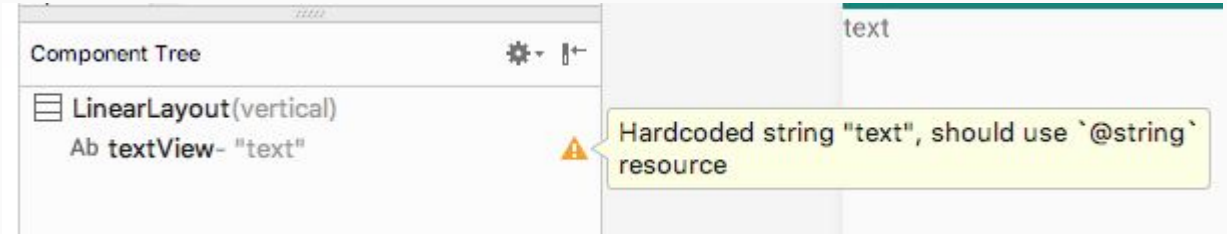
DOJO - LAYOUT EDITOR

1. Defina os seguintes atributos para o TextView através do painel Attributes

    - ID: name_text
    - Text: seu Nome
    - textAppearance > textSize: 20sp
    - textAppearance > textColor: @android:color/black
    - textAppearance > textAlignment: Center
2. Commitar alterações

DOJO - STRING RESOURCE

1. Na Árvore de componentes, próximo a TextView, você notará um ícone de aviso. Para ver o texto de aviso, clique no ícone ou aponte para ele, conforme mostrado na imagem abaixo.
2. Adicione o resource *name*
3. Abra arquivo strings.xml
4. Commitar alterações



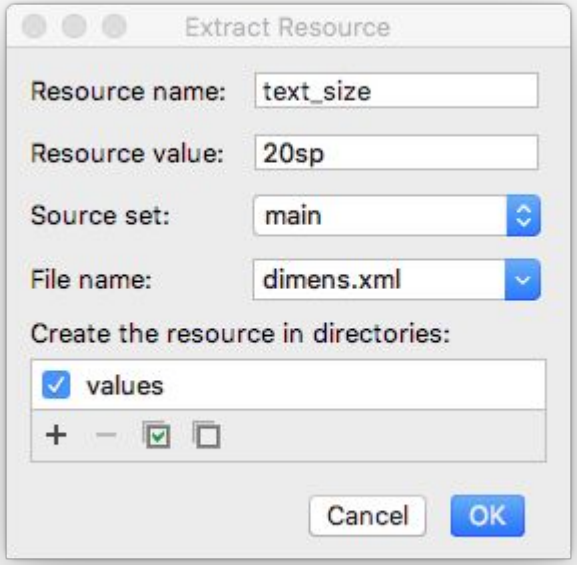
STRINGS RESOURCES

Separar o conteúdo utilizado na sua aplicação do layout/código ajuda muito na gestão desses recursos.

A internacionalização da sua aplicação pode acontecer de forma muito mais simples se você extrair TODOS os textos para o arquivo *strings.xml*
O reuso e a manutenção desse conteúdo também acontece de forma mais fácil.

DOJO - DIMENSION RESOURCE

1. Da mesma forma que extraímos o nome da string, vamos extrair o tamanho do texto.
2. Repare no arquivo dimens.xml
3. Commitar alterações



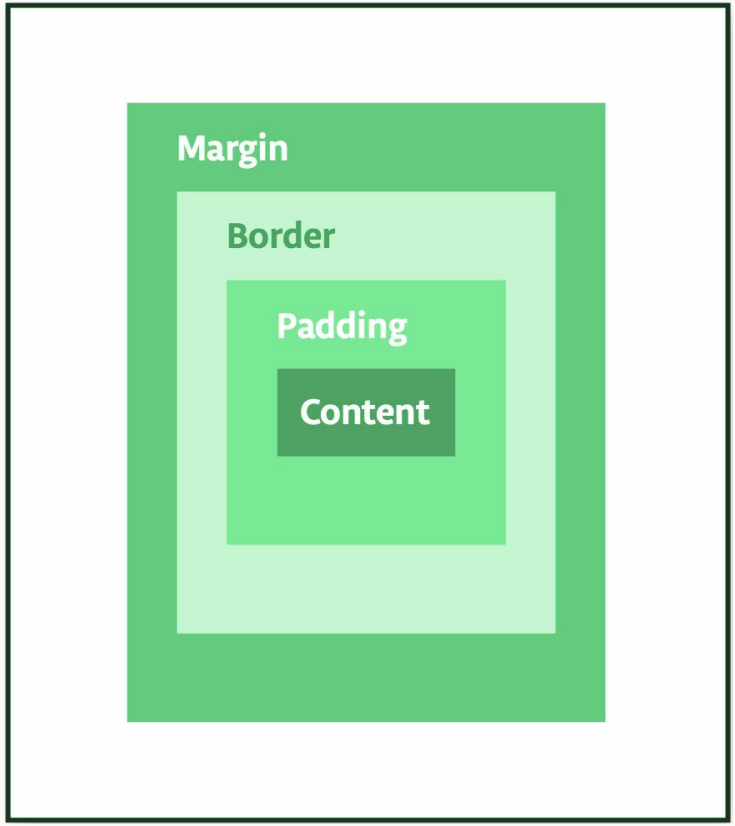
DIMENSIONS RESOURCES

Separar as propriedades de layout utilizadas otimiza o uso e gestão.

Modificar de forma uniforme essas propriedades fica muito mais fácil.
Reaproveitar essas propriedades também

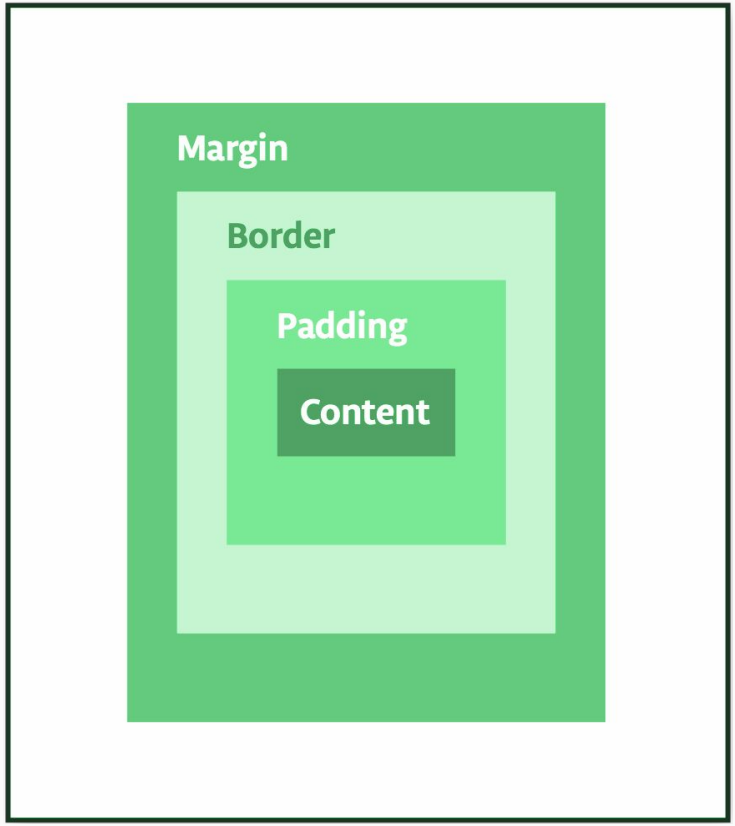
ESTILIZANDO TEXTVIEW

Padding é o espaço dentro dos limites de uma view ou elemento. É o espaço entre as bordas da view e o conteúdo da view.



ESTILIZANDO TEXTVIEW

Margem é o espaço adicionado fora das bordas da view. É o espaço da borda da view até seu pai.

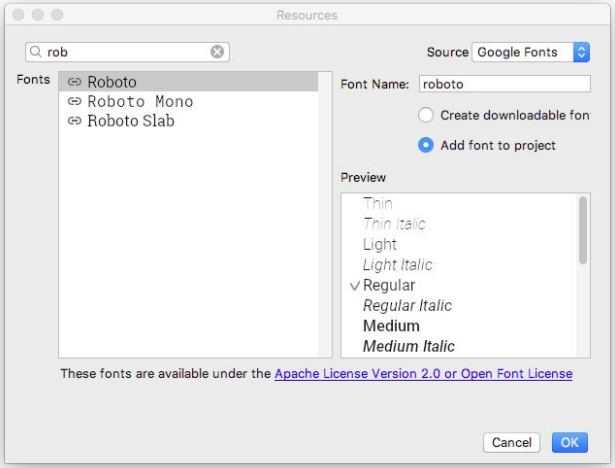


DOJO - PADDING & MARGIN

1. Defina o padding e a margin do TextView a partir dos resources
- Small_padding = 8dp
- Layout_margin = 16dp

DOJO - FONTS

1. Defina a fontFamily como *roboto*



STYLE

Um **style** é uma coleção de atributos que especificam a aparência e o formato de uma view. Um style pode incluir cor da fonte, tamanho da fonte, cor de fundo, preenchimento, margem e outros atributos comuns.

Você pode extrair a formatação da view do texto do nome em um style e reutilizar o estilo para qualquer número de views em seu aplicativo. Reutilizar um style dá ao seu aplicativo uma aparência consistente quando você tem várias views. Usar styles também permite que você mantenha esses atributos comuns em um único local.

DOJO - STYLE

1. Extraia os estilos aplicados no componente TextView
   Clique com o botão direito do mouse em TextView na árvore de componentes e selecione Refatorar> Extrair estilo.
2. Commitar alterações

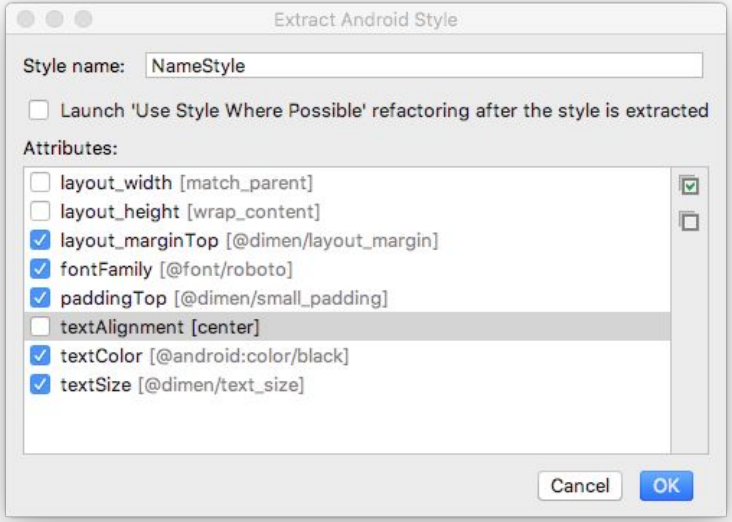


IMAGE VIEW

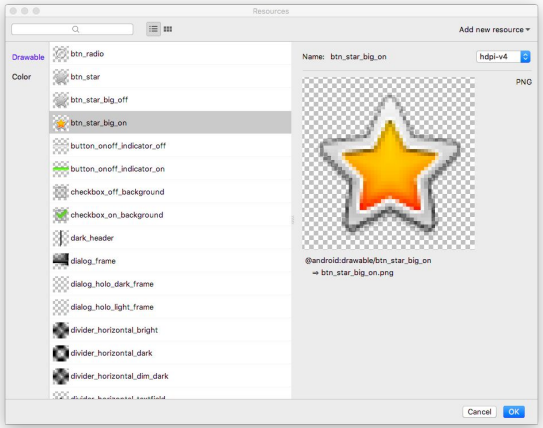
A maioria dos aplicativos Android do mundo real consiste em uma combinação imagens, textos e aceitar entrada do usuário na forma de texto ou eventos de clique.

Um **ImageView** é uma *view* para exibir recursos de imagem. Por exemplo, um ImageView pode exibir recursos de bitmap, como arquivos PNG, JPG, GIF ou WebP, ou pode exibir um recurso Drawable, como um desenho vetorial.

Existem recursos de imagem que vêm com o Android, como exemplos de ícones, avatares e planos de fundo.

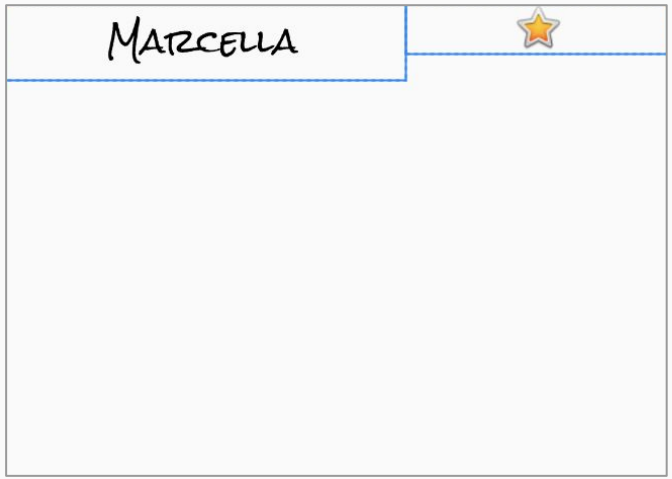
DOJO - IMAGE VIEW

1. Adicione a imagem da estrelinha logo abaixo do TextView



LINEAR LAYOUT

1. Repare no resultado
2. Por que isso aconteceu?
3. O que podemos fazer para a estrela ficar no lugar certo?



LINEAR LAYOUT

Todos os filhos de um LinearLayout são empilhados um após o outro. Portanto, uma lista vertical terá **somente um filho por linha,** independentemente da largura, e uma **lista horizontal terá altura de apenas uma linha (a altura do filho mais alto, mais preenchimento).**

Um LinearLayout respeita margens entre filhos e a gravidade (alinhamento à direita, no centro ou à esquerda) de cada filho.

DOJO - IMAGEVIEW

1. Renomeie o ImageView
2. Para renomear o id do ImageView, clique com o botão direito em "@ + id / imageView" e selecione Refactor> Rename.
3. Na caixa de diálogo Renomear, defina o id como @+id/star_image. Clique em Refatorar.
4. Commitar alterações

CONTENT DESCRIPTION

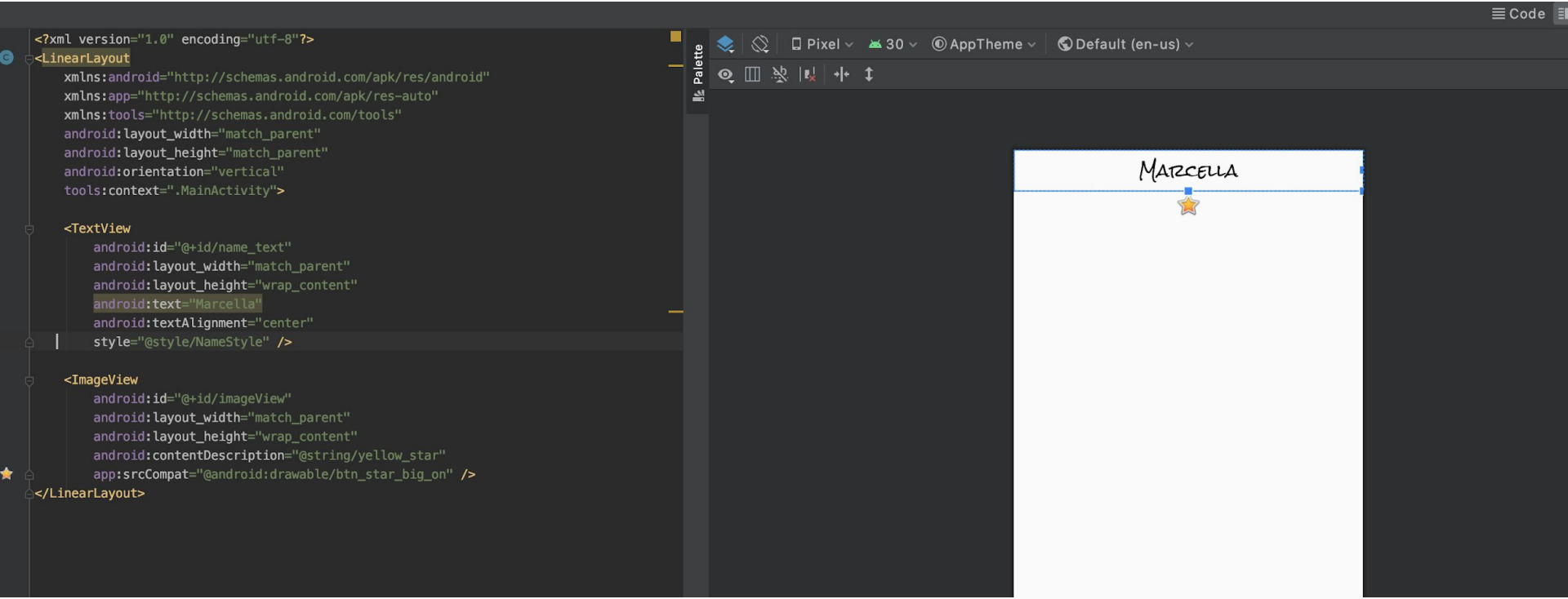
Na guia Design, na Árvore de componentes, clique no ícone de aviso ao lado de star_image. O aviso é para uma **contentDescription** ausente, que os leitores de tela usam para descrever as imagens para o usuário.

DOJO - CONTENT DESCRIPTION

1. No painel Atributos, clique nos três pontos ... ao lado do atributo contentDescription. A caixa de diálogo Recursos é aberta.
2. Na caixa de diálogo Recursos, selecione Adicionar novo recurso> Novo valor de string. Defina o campo Nome do recurso como yellow_star e defina o campo valor do recurso como estrela amarela. Clique OK.
3. Use o painel Atributos para adicionar uma margem superior de 16 dp (que é @ dimen / layout_margin) à yellow_start, para separar a imagem da estrela do nome.
4. Execute seu aplicativo. Seu nome e a imagem da estrela são exibidos na IU do seu aplicativo.
5. Ative o leitor de tela e veja o resultado
6. Commitar alterações

DOJO - CONTENT DESCRIPTION

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/name_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Marcella"
        android:textAlignment="center"
        style="@style/NameStyle" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:contentDescription="@string/yellow_star"
        app:srcCompat="@android:drawable/btn_star_big_on" />
</LinearLayout>
```



SCROLLVIEW

Um **ScrollView** é um GroupView que permite que a hierarquia de **visualização colocada dentro dele seja rolada**. Uma visualização de rolagem pode conter apenas uma outra View, ou GroupView, como filho. A view filho é comumente um LinearLayout. Dentro de um LinearLayout, você pode adicionar outras views.

SCROLLVIEW

Um **ScrollView** é um GroupView que permite que a hierarquia de **visualização colocada dentro dele seja rolada**. Uma visualização de rolagem pode conter apenas uma outra View, ou GroupView, como filho. A view filho é comumente um LinearLayout. Dentro de um LinearLayout, você pode adicionar outras views.

DOJO - SCROLLVIEW

Nesse app, a área com a descrição da sua biografia será "rolável".

1. Adicione um ScrollView logo abaixo da estrela
2. Adicione um TextView, de id: text_bio e crie uma string de resource para preencher o conteúdo
3. Adicione nessa TextView um *style* com as seguintes propriedades
   Id: style_bio
   fontFamily: Roboto
   textSize: 14sp

DOJO - SCROLLVIEW

Preencha a string bio_text com uma breve descrição da sua carreira profissional (pelo menos 3 linhas)

Dicas:

- Use \n para indicar uma quebra de linha.
- Se você usar um apóstrofo, deverá escapar dele com uma barra invertida.
  Para texto em negrito use ... e para texto em itálico use ... . *Por exemplo:"Este texto está negrito e este texto está itálico ."*

DOJO - SCROLLVIEW

1. Execute o app e observe o resultado
2. Existe algum outro detalhe visual que podemos melhorar?
3. Commitar alterações

PALAVRAS CHAVE

- LINEAR LAYOUT
- SCROLLVIEW
- VIEWGROUP
- CONTENT DESCRIPTION
- RESOURCES
- STYLES

REVIEW

Questão 1
Qual das opções a seguir é um grupo de exibição?

- ▢ EditText
- ▢ LinearLayout
- ▢ TextView
- ▢ Botão

REVIEW

Questão 2
Qual das seguintes hierarquias de visualização não é válida?

- ▢ LinearLayout> TextView, TextView, ImageView
- ▢ ScrollView> LinearLayout> TextView, Botão, Botão, ScrollView> TextView
- ▢ TextView> TextView, ImageView, ScrollView

REVIEW

Questão 3

Estilos são recursos definidos em styles.xml. Usando estilos, você pode definir cores, fontes, tamanho do texto e muitas outras características de uma visualização. Verdadeiro ou falso?

- ▢ Verdadeiro
- ▢ Falso

Questão 4

Um ScrollView é um grupo de visualização que pode conter qualquer número de visualizações ou grupos de visualização como seus filhos. Verdadeiro ou falso?

- ▢ Verdadeiro
- ▢ Falso

REVIEW

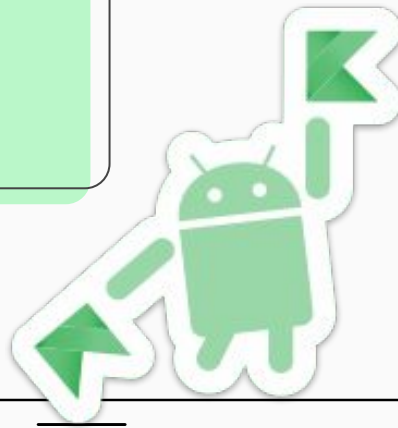
Questão 5
Qual elemento da IU pode ser usado para exibir imagens em seu aplicativo?

- ▢ TextView
- ▢ ImageView
- ▢ Botão
- ▢ ScrollView

SUPER TRUNFO

Super trunfo é um jogo de cartas. Veja o código do aplicativo Super trunfo para entender como o jogo funciona.

https://github.com/marcellalcs/supertrunfo



O QUE É PROGRAMAÇÃO ORIENTADA À OBJETOS

Objeto: coisa material ou abstrata que pode ser percebida pelos sentidos e descrita por meio das suas **características**, **comportamento** e **estado** atual.

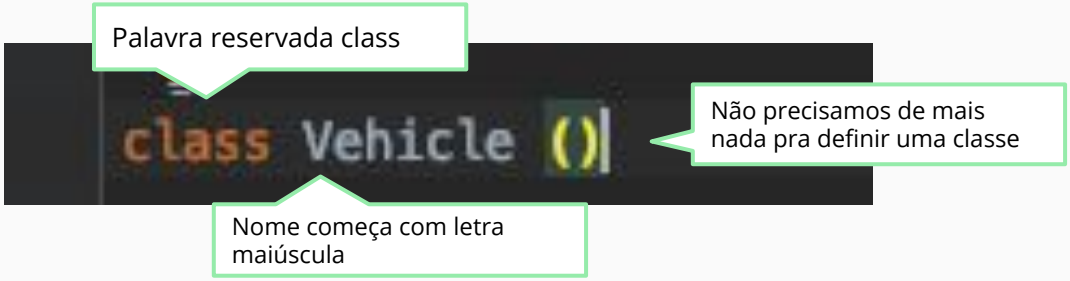
QUAIS OBJETOS EXISTEM NO APP SUPER TRUNFO?

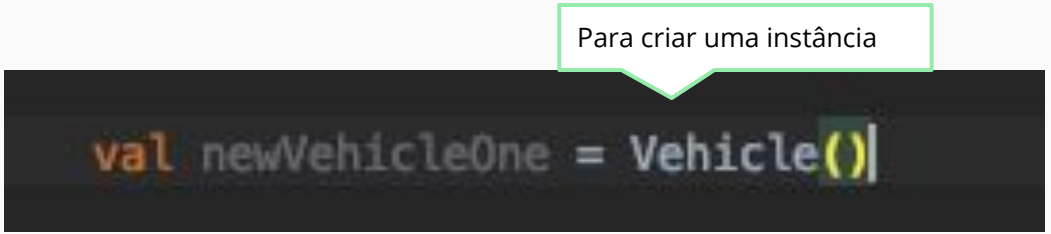
- JOGADOR
- VEÍCULO
- CONDUTOR
- CARTAS
- ...

CLASS - KOTLIN

É um modelo de código de programa extensível para criar objetos, fornecendo valores iniciais para o estado (variáveis de membro) e implementações de comportamento (funções ou métodos de membro)

VEHICLE CLASS - KOTLIN





VEHICLE CLASS - KOTLIN

```
class Vehicle () {
    var maxAcceleration: Int
}
```

Para definir um atributo desta forma, precisamos inicializá-lo

```
class Vehicle () {
    var maxAcceleration: Int = 0
}
```

VEHICLE CLASS - KOTLIN

```
val newVehicleOne = Vehicle()
newVehicleOne.maxAcceleration = 100
```

Getters e setters são definidos automaticamente para qualquer propriedade

```
class Vehicle () {
    var maxAcceleration: Int = 0
        get() = field * 3
        set(value) {
            field = value/3
        }
}
```

É possível sobre escrever os getters e setters, adicionando comportamentos personalizados

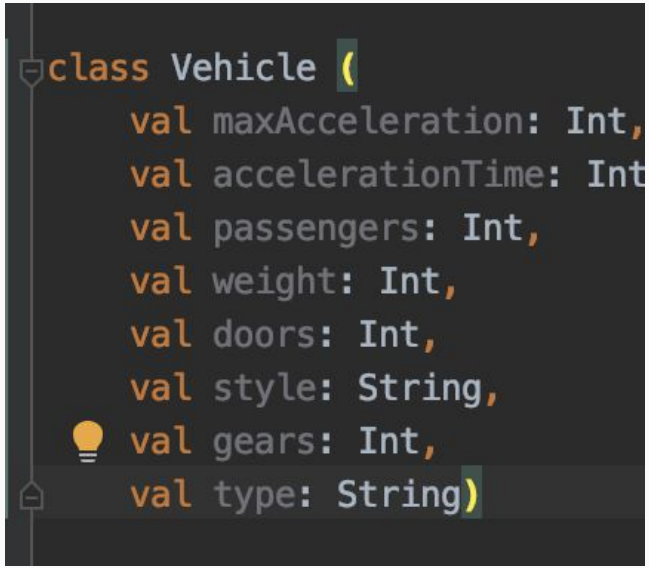
DESAFIO

```
val newVehicleOne = Vehicle()
newVehicleOne.maxAcceleration = 100
val maxAcceleration : Int  = newVehicleOne.maxAcceleration
```

Qual o valor de maxAcceleration?

Podemos trabalhar com as propriedades dos objetos declarando-as como no exemplo anterior. Mas essa **não é a forma mais comum de declarar propriedades de classes em Kotlin.**

CLASS CONSTRUCTORS - KOTLIN



CLASS CONSTRUCTORS - KOTLIN

```
val newVehicleOne = Vehicle(
    maxAcceleration: 100,
    accelerationTime: 120,
    passengers: 5,
    weight: 120,
    doors: 2,
    style: "sedã",
    gears: 5,
    type: "car")
```

CLASS CONSTRUCTORS - KOTLIN

```
val testAcceleration : Int  = newVehicleOne.accelerationTime
newVehicleOne.maxAcceleration = 10
```

Por que o erro?

DATA CLASS - KOTLIN

dataclass.kt

```kotlin
fun main(args: Array<String>) {
    val vehicleOne = Vehicle( name: "Uno",  accelerationTime: 10)
    val vehicleTwo = Vehicle( name: "Uno",  accelerationTime: 10)

    if(vehicleOne == vehicleTwo){
        print("mesmo carro")
    } else{
        print("carros diferentes")
    }

}

class Vehicle (var name: String, var accelerationTime: Int)
```

O que será impresso no console?

DATA CLASS - KOTLIN

A linguagem Kotlin introduz o conceito de classes de dados, que representam classes simples usadas como contêineres de dados e não encapsulam nenhuma lógica adicional.

DESAFIO - CLASS DRIVER

Crie a class Driver

VISIBILIDADE

- **Public**: Se você não especificar nenhum modificador de visibilidade, public é usado por padrão, o que significa que suas declarações estarão visíveis em qualquer lugar;
- **Private**: Se você marcar uma declaração como privada, ela só ficará visível dentro do arquivo que contém a declaração;
- **Internal:** Se você marcá-lo interno, ele ficará visível em qualquer lugar no mesmo módulo;
- **Protected:** protegido não está disponível para declarações de nível superior.