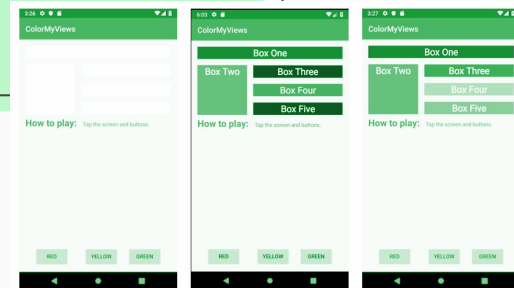


02 CONSTRAINT LAYOUT

COLORMYVIEWS

O aplicativo ColorMyViews é inspirado no artista holandês Piet Mondrian. Ele inventou um estilo de pintura chamado neoplasticismo, que usa apenas linhas verticais e horizontais e formas retangulares em preto, branco, cinza e cores primárias.



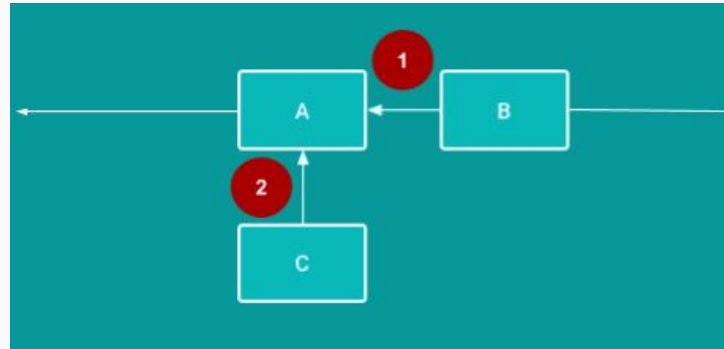
CONSTRAINT LAYOUT

Um **ConstraintLayout** é um **ViewGroup** que permite posicionar e dimensionar views de uma forma flexível. Um **ConstraintLayout** permite criar layouts grandes e complexos com hierarquias de views plana (sem grupos de views aninhados).

CONSTRAINTS

Uma constraint é uma conexão ou alinhamento entre dois elementos da IU. Cada constraint conecta ou alinha uma view a outra, ao layout pai ou a uma guideline invisível. Em um `ConstraintLayout`, você posiciona uma view definindo pelo menos uma constraint horizontal e uma vertical.



- 1) Constraint horizontal: B é restringido para ficar à direita de A.
- 2) Constraint vertical: C é restringido para ficar abaixo de A.



INICIANDO PROJETO

1. Crie o projeto ColorMyViews a partir de uma activity vazia
2. Crie um repo no GitHub para o projeto
3. Faça o primeiro Push

DOJO - LAYOUT INSPECTOR

1. Abrir arquivo layout
2. Remover constraints existentes 
3. Rodar aplicação, verificar posição do TextView
4. Mantenha a posição do TextView centralizada e aplique a varinha mágica 
5. Rode a aplicação e veja o resultado

CONSTRAINT BIAS

As constraint bias posiciona a view ao longo dos eixos horizontal e vertical. Por padrão, a view é centralizada entre as duas restrições com uma polarização de 50%.

Para ajustar o bias, você pode arrastar os controles deslizantes de polarização no inspetor de visualização. Arrastar um controle deslizante de polarização altera a posição da visualização ao longo do eixo.



CONSTRAINT BIAS

As constraint bias posiciona a view ao longo dos eixos horizontal e vertical. Por padrão, a view é centralizada entre as duas restrições com uma polarização de 50%.

Para ajustar o bias, você pode arrastar os controles deslizantes de polarização no inspetor de visualização. Arrastar um controle deslizante de polarização altera a posição da visualização ao longo do eixo.



TIPO DE CONSTRAINT

No inspetor de visualização, as setas dentro do quadrado representam o tipo de constraint:



1. **wrap_content:** a visualização se expande apenas o necessário para conter seu conteúdo.



2. **fixo:** você pode especificar uma dimensão como a margem da vista na caixa de texto ao lado das setas de restrição fixa.



3. **match_constraint:** a view se expande o máximo possível para atender às restrições de cada lado, após considerar as próprias margens da visualização. Essa constraint é muito flexível, pois permite que o layout se adapte a diferentes tamanhos e orientações de tela. Ao permitir que a visualização corresponda às restrições, você precisa de menos layouts para o aplicativo que está construindo.

DOJO - TIPO DE CONSTRAINT

1. Deixe o TextView com as constraints do tipo: `match_constraint`
2. Remova a constraint de baixo do TextView
3. Adicione margem de 16dps para as outras constraints
4. Extraia o resource de dimensão 16dp com o nome *margin_wide*



1. Defina a fontFamily do TextView com o estilo *Roboto*
2. Adicione a dimen abaixo no arquivo `dimens.xml`

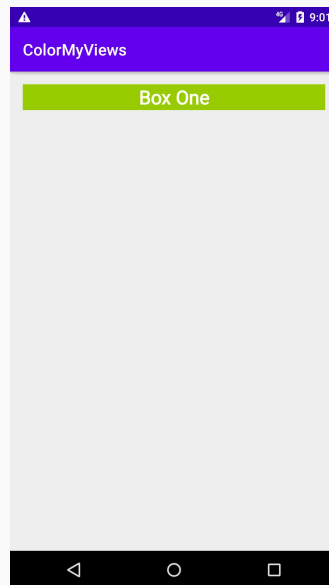
```
<dimen name="box_text_size">24sp</dimen>
```

3. Adicione o estilo abaixo para aplicarmos no TextView

```
<style name="whiteBox">  
  <item name="android:background">@android:color/holo_green_light</item>  
  <item name="android:textAlignment">center</item>  
  <item name="android:textSize">@dimen/box_text_size</item>  
  <item name="android:textStyle">bold</item>  
  <item name="android:textColor">@android:color/white</item>  
  <item name="android:fontFamily">@font/roboto</item>  
</style>
```

DOJO - ESTILO BOX ONE

1. Crie um string resource para o conteúdo do TextView com nome *box_one* e conteúdo Box One
2. Defina o ID do TextView como *box_one_text*
3. Aplique o estilo *whiteBox* nesse TextView

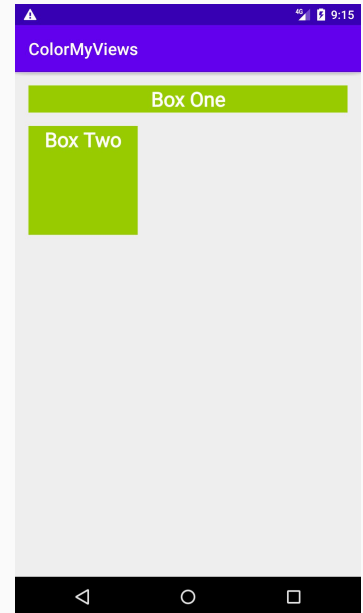


DOJO - BOX TWO

1. Adicione um novo TextView na tela
2. Posicione o TextView a partir de duas constraints: seu topo alinhado com a base da Box One, seu lado esquerdo alinhado com a borda da tela
3. Crie um resource de string com nome box_two e conteúdo Box Two
4. Estilize o TextView com os valores abaixo:

Attribute	value
id	box_two_text
layout_height	130dp
layout_width	130dp
style	@style/whiteBox
text	@string/box_two

5. Execute seu app, ele deverá estar assim:

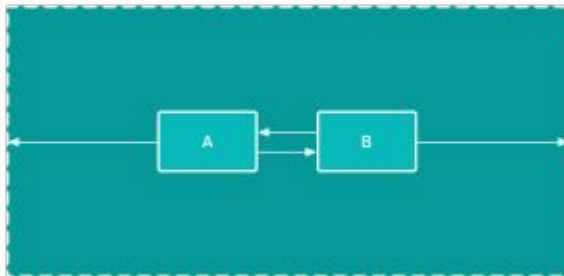


Ao desenvolver aplicativos do mundo real, use **restrições flexíveis para a altura e largura** de seus elementos de interface do usuário, sempre que possível.

Por exemplo, use **match_constraint** ou **wrap_content**. Quanto mais elementos de IU de tamanho fixo você tiver em seu aplicativo, menos adaptável será seu layout para diferentes configurações de tela.

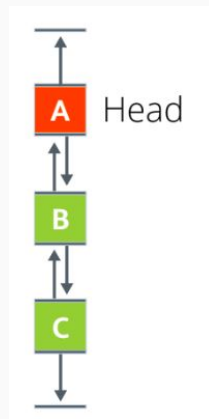
Uma chain é um grupo de views vinculadas umas às outras com restrições bidirecionais. As views dentro de uma chain podem ser distribuídas vertical ou horizontalmente.

Por exemplo, o diagrama a seguir mostra duas visualizações restritas uma à outra, o que cria uma cadeia horizontal.



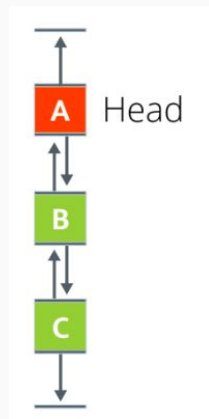
Head da chain

A primeira view em uma cadeia é chamada de cabeça da cadeia. Os atributos definidos no topo da cadeia controlam, posicionam e distribuem todas as visualizações na cadeia. Para chains horizontais, a cabeça é a vista mais à esquerda. Para correntes verticais, a cabeça é a vista mais superior. Em cada um dos dois diagramas abaixo, "A" é o chefe da cadeia.



Head da chain

A primeira view em uma cadeia é chamada de cabeça da cadeia. Os atributos definidos no topo da cadeia controlam, posicionam e distribuem todas as visualizações na cadeia. Para chains horizontais, a cabeça é a vista mais à esquerda. Para correntes verticais, a cabeça é a vista mais superior. Em cada um dos dois diagramas abaixo, "A" é o chefe da cadeia.



Chain style

Os estilos de cadeia definem a maneira como as vistas encadeadas são espalhadas e alinhadas. Você estiliza uma corrente atribuindo um atributo de estilo de corrente, adicionando peso ou definindo bias nas views. Existem quatro estilos de cadeias:

Spread: este é o estilo padrão. As views são distribuídas uniformemente no espaço disponível, após as margens serem contabilizadas.



Spread inside: A primeira e a última views são anexadas ao pai em cada extremidade da cadeia. O resto das visualizações são distribuídas uniformemente no espaço disponível.



Packed: As views são empacotadas juntas, após as margens serem contabilizadas. Você pode então ajustar a posição de toda a corrente mudando o bias da visão da cabeça da corrente.



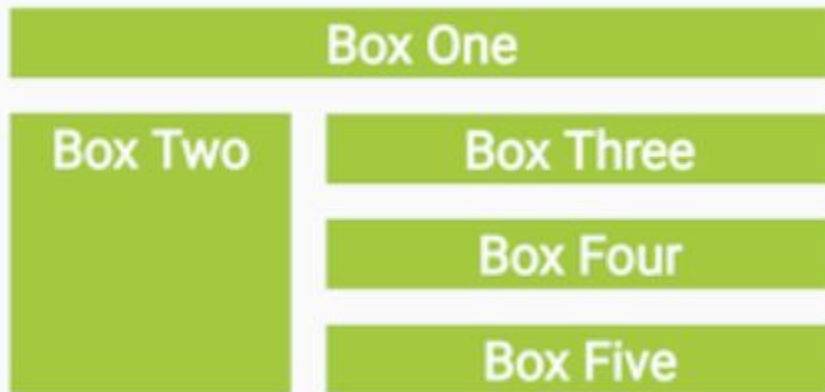
Packed: As views são empacotadas juntas, após as margens serem contabilizadas. Você pode então ajustar a posição de toda a corrente mudando o bias da visão da cabeça da corrente.



Weighted: as views são redimensionadas para preencher todo o espaço, com base nos valores definidos nos atributos `layout_constraintHorizontal_weight` ou `layout_constraintVertical_weight`. Por exemplo, imagine uma cadeia contendo três visualizações, A, B e C. A visualização A usa um peso de 1. As visualizações B e C usam cada uma um peso de 2. O espaço ocupado pelas visualizações B e C é o dobro da visualização A , como mostrado abaixo.



O efeito que queremos criar é o seguinte:

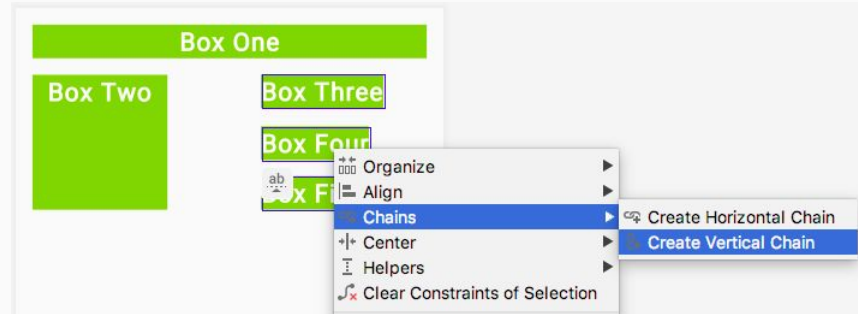


Repare no posicionamento das caixas três, quatro e cinco.

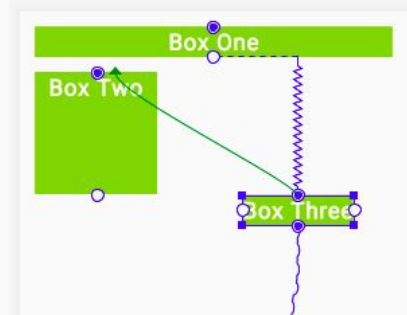
1. Crie essas três TextViews, extraia o conteúdo de texto para strings.xml e estilize

Attribute	Top text view	Middle text view	Bottom text view
ID	box_three_text	box_four_text	box_five_text
text	@string/box_three	@string/box_four	@string/box_five
style	@style/whiteBox	@style/whiteBox	@style/whiteBox

1. Selecione os três TextViews, clique botão direito, selecione: **Chains > Create Vertical Chain**.

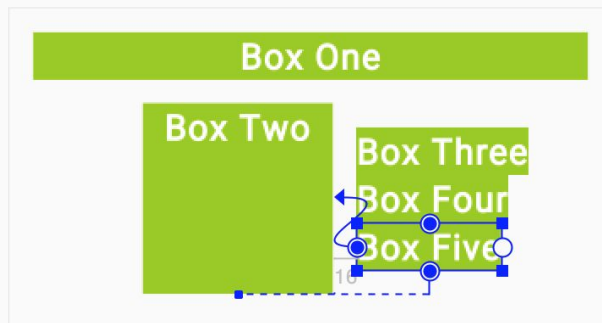


Adicione uma constraint que se estende do topo da Caixa Três até o topo da Caixa Dois. Isso remove a constraint superior existente e a substitui pela nova constraint. Você não precisa excluir a constraint explicitamente.



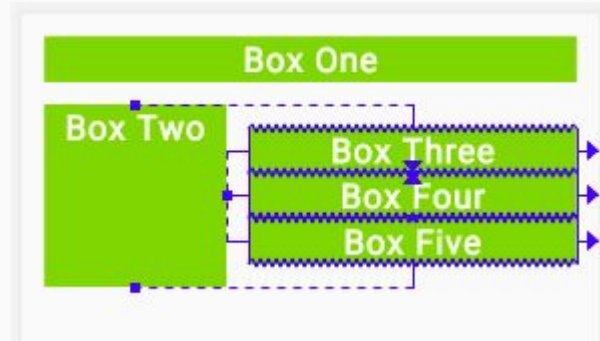
Adicione uma constraint da parte inferior da Caixa Cinco até a parte inferior da Caixa Dois.

Crie uma constraint conectando o lado esquerdo da Caixa Três ao lado direito da Caixa Dois. Repita para o Quadro Quatro e o Quadro Cinco, restringindo o lado esquerdo de cada um ao lado direito do Quadro Dois.



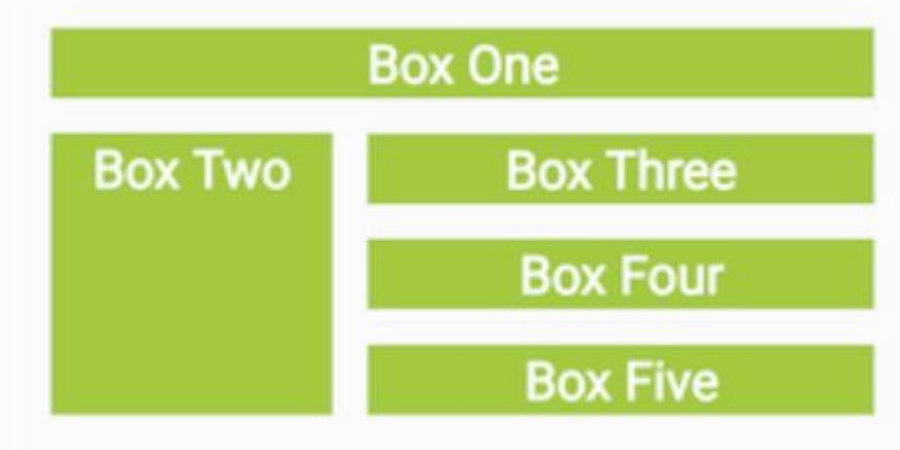
Restrinja o lado direito de cada uma das três visualizações de texto ao lado direito do layout.

Para cada uma das três TextViews, altere o atributo `layout_width 0dp`, que é equivalente a alterar o tipo de restrição para Match Constraints.



DOJO - DESAFIO CHAINS

Usando margins, deixe o posicionamento final das caixas assim:



PALAVRAS CHAVE

CONSTRAINT LAYOUT

CHAINS

MATCH_CONSTRAINT

WRAP_CONTENT

MARGIN

CHAINS TYPE

Questão 1

Em um `ConstraintLayout`, qual das opções a seguir descreve as constraints necessárias para manter uma visualização no lugar durante o tempo de execução?

- ☐ Duas restrições horizontais.
- ☐ Uma restrição vertical.
- ☐ Pelo menos uma restrição horizontal e uma vertical.
- ☐ Não há necessidade de restringir a visão.

Questão 2

Qual dos seguintes tipos de constraint expande uma visualização apenas o necessário para ajustar seu conteúdo?

- ☐ Wrap content
- ☐ Match constraint
- ☐ Fixed constraint
- ☐ Baseline constraint

Questão 3

O inspetor de visualização está disponível apenas para visualizações em _____.

- ☐ Um ConstraintLayout
- ☐ Um ConstraintLayout ou LinearLayout
- ☐ Qualquer ViewGroup
- ☐ Um LinearLayout

Questão 4

Uma chain é um grupo de views vinculadas entre si por _____.

- ☐ Restrições superior e inferior
- ☐ Restrições bidirecionais
- ☐ Restrições direita e esquerda
- ☐ Restrições de linha de base