



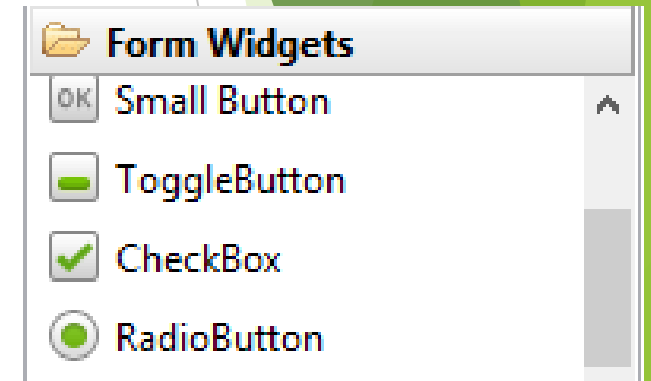
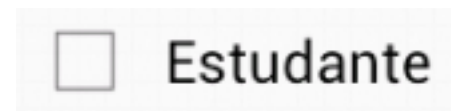
# Programação para Android

Aula 04: Widgets - CheckBox, RadioGroup, ToogleButton, SeekBar, RatingBar, NumberPicker, DatePicker e TimePicker

# Checkbox

- ▶ O Widget Checkbox permite criar uma caixa de marcação múltipla, por padrão acompanhada de uma etiqueta ao lado.
- ▶ Na paleta de Widgets é possível incluir um Checkbox ou podemos adicioná-lo via código XML

```
<CheckBox  
  android:id="@+id/ckEstudante"  
  android:layout_width="fill_parent"  
  android:layout_height="wrap_content"  
  android:text="Estudante"  
>
```



# Exemplo 01 - Checkbox



A screenshot of an Android application interface. At the top, there is a black header bar with a green Android robot icon on the left and the text "Aula04" on the right. Below the header, the text "Nome:" is displayed. Underneath is a text input field with a blue border and a light blue underline, containing the placeholder text "Digite seu nome completo". Below the input field are two checkboxes, each with an empty square box to its left. The first checkbox is followed by the text "Estudante", and the second is followed by "Trabalhador". At the bottom of the form is a wide, light gray button with the text "Verificar" centered on it.

# Manipulando um Checkbox no código Java

- ▶ Para manipular um Checkbox na classe Java, devemos inicialmente fazer a associação deste elemento usando o método `findViewById`, assim como realizado com outros widgets.
- ▶ Baseado no exemplo anterior, iremos programar o evento Click do botão **VERIFICAR** e exibir uma mensagem a partir da seleção da `CheckBox`, para isto criaremos um listener para o click do botão.

```
View.OnClickListener verifica = new View.OnClickListener() {
```

- ▶ Em seguida, programaremos o evento relacionado ao clique. **OBS:** Ao incluir o listener (`onClickListener`) o Netbeans irá incluir um código automaticamente relacionado ao `onClick`. Devemos programar nosso código entre as `{ }`
- ▶ Após programar o clique é necessário alterar a propriedade `onClickListener` do botão (`btVerificar.setOnClickListener(verifica);`)

```
public void onClick(View v) {}
```

# Manipulando um Checkbox no código Java

- ▶ O método `isChecked()` é utilizado para verificar se a checkbox está ou não marcada.

```
CheckBox ckEstudante;  
CheckBox ckTrabalhador;
```

```
ckEstudante = (CheckBox)findViewById(R.id.ckEstudante);  
ckTrabalhador = (CheckBox)findViewById(R.id.ckTrabalhador);
```

```
if (ckEstudante.isChecked() == true)  
{  
    Toast.makeText(getBaseContext(), "Estudante selecionado!", Toast.LENGTH_SHORT).show();  
}  
if (ckTrabalhador.isChecked() == true)  
{  
    Toast.makeText(getBaseContext(), "Trabalhador selecionado!", Toast.LENGTH_SHORT).show();  
}
```

# RadioButton e RadioGroup

- ▶ Estes controles trabalham juntos. O RadioGroup permite agrupar controles RadioButton.
- ▶ Um RadioButton, assim como o CheckBox é uma caixa de marcação única, por padrão acompanhada de uma etiqueta ao lado. O Android faz este processo de marcação/desmarcação de forma automática.
- ▶ Na paleta de Widgets é possível incluir um Checkbox ou podemos adicioná-lo via código XML
- ▶ OBS: Para que um RadioButton tenha funcionalidade de marcação/desmarcação automática este deve estar inserido dentro de um RadioGroup.



RadioGroup



RadioButton

# Radiobutton e Radiogroup

## ► Código XML para criação de Radiobutton

```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <RadioButton
        android:id="@+id/rbMasculino"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Masculino"/>
    <RadioButton
        android:id="@+id/rbFeminino"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Feminino"
    />
</RadioGroup>
```

A white rectangular box with rounded corners containing two radio button options. The top option is "Masculino" with an unselected radio button to its left. The bottom option is "Feminino" with an unselected radio button to its left. The background of the box has a light gray grid pattern.

# Exemplo 02 - Radiobutton

Aula04

Nome:

Estudante

Trabalhador

Sexo:

Masculino

Feminino

Verificar

Feminino selecionado!



# Manipulando um Radiobutton no código Java

- ▶ Para manipular um Radiobutton na classe Java, devemos inicialmente fazer a associação deste elemento usando o método `findViewById`, assim como realizado com outros widgets.
- ▶ Baseado no exemplo anterior, iremos programar o evento Click do botão **VERIFICAR** e exibir uma mensagem a partir da seleção do Radiobutton, para isto criaremos um listener para o click do botão.

```
View.OnClickListener verifica = new View.OnClickListener() {
```

- ▶ Em seguida, programaremos o evento relacionado ao clique. **OBS:** Ao incluir o listener (`onClickListener`) o Netbeans irá incluir um código automaticamente relacionado ao `onClick`. Devemos programar nosso código entre as `{ }`
- ▶ Após programar o clique é necessário alterar a propriedade `onClickListener` do botão (`btVerificar.setOnClickListener(verifica);`)

```
public void onClick(View v) {}
```

# Manipulando um Radiobutton no código Java

- ▶ Assim como no Checkbox, o método `isChecked()` é utilizado para verificar se a checkbox está ou não marcada.

```
RadioButton rbMasculino;  
RadioButton rbFeminino;
```

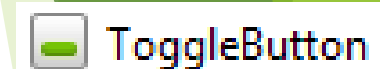
```
rbMasculino = (RadioButton) findViewById(R.id.rbMasculino);  
rbFeminino = (RadioButton) findViewById(R.id.rbFeminino);
```

```
if (rbMasculino.isChecked() == true)  
{  
    Toast.makeText(getBaseContext(), "Masculino selecionado!", Toast.LENGTH_SHORT).show();  
}  
else if (rbFeminino.isChecked() == true)  
{  
    Toast.makeText(getBaseContext(), "Feminino selecionado!", Toast.LENGTH_SHORT).show();  
}
```

# ToogleButton

- ▶ Um ToogleButton é um botão de alternância que permite os usuários alterar a configuração entre dois estado(Sim/Não, Verdadeiro/Falso)
- ▶ Por padrão o texto deste botão é On/Off. Podemos alterar usando as propriedades:
  - ▶ android:textOn="Curtir"
  - ▶ android:textOff="Não curtir"
- ▶ Na paleta de Widgets é possível incluir um Tooglebutton ou podemos adiciona-lo via código XML

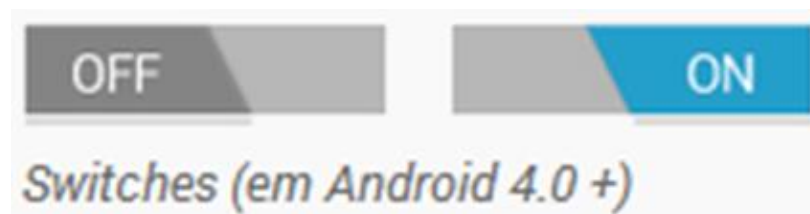
```
<ToggleButton  
    android:id="@+id/toggleButton1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textOn="Curtir"  
    android:textOff="Não curtir"  
>
```



Botões de alternância

# Switches

- ▶ A versão Android 4.0 (API nível 14 em diante) introduz um outro tipo de botão de alternância chamado um Switch que fornece um controle deslizante, que você pode adicionar via XML.



```
<Switch
    android:id="@+id/toggleButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="Curtir"
    android:textOff="Não curtir"
/>
```



# Manipulando um ToggleButton e Switch no código Java

- ▶ Manipulamos ToggleButton e Switch de forma semelhante a um Checkbox ou Radiobutton. Neste caso, criaremos um listener específico para o click do **ToggleButton** ou **Swicth**. Para verificar se valor marcado está **On** ou **Off** usamos o método `isChecked()`.

```
ToggleButton tgCurtir;  
tgCurtir = (ToggleButton) findViewById(R.id.tgCurtir);  
  
if (tgCurtir.isChecked() == true)  
{  
    Toast.makeText(getBaseContext(), "Você curtiu!!", Toast.LENGTH_SHORT).show();  
}  
else  
{  
    Toast.makeText(getBaseContext(), "Que pena!!!", Toast.LENGTH_SHORT).show();  
}
```

# Exemplo 03 - ToggleButton e Switch



Aula04

Nome:

Digite seu nome completo

Estudante

Trabalhador

Sexo:

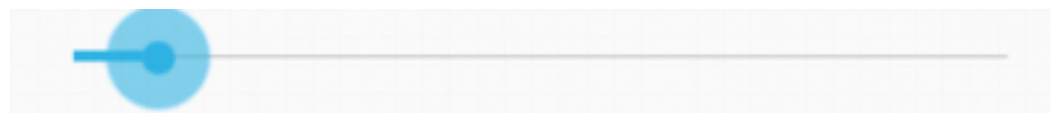
Masculino

Feminino

Não curtir

# SeekBar

- ▶ Uma SeekBar é uma barra de progresso no qual o usuário pode tocar e arrastar para a esquerda ou para a direita para definir o nível de progresso.



- ▶ Na paleta de Widgets é possível incluir um SeekBar ou podemos adicioná-lo via código XML

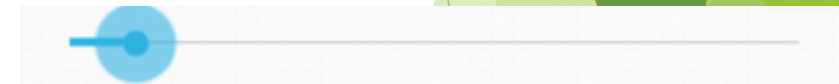


- ▶ Por padrão o valor máximo de uma SeekBar é 100 e o valor mínimo é 0. O intervalo é definido de 1 em 1. Apenas o valor máximo pode ser definido via XML. Os demais atributos só podem ser modificados via código Java.

# SeekBar

- ▶ O código XML para incluir o widget Seekbar em uma tela é semelhante aos demais componentes. OBS: a propriedade android:max define o valor máximo da SeekBar

```
<SeekBar  
    android:id="@+id/skIdade"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:max="120"  
>
```





# Manipulando um Seekbar no código Java

- ▶ Manipulamos um Seekbar de forma semelhante aos demais componentes.
- ▶ Para adicionar funcionalidade quando o usuário arrastar a barra, precisamos criar um listener. Neste caso definimos o `OnSeekBarChangeListener`.

```
SeekBar.OnSeekBarChangeListener avanca = new SeekBar.OnSeekBarChangeListener ()
```

- ▶ O NetBeans irá gerar automaticamente 3 eventos:

```
public void onStopTrackingTouch (SeekBar seekBar)
```

```
public void onStartTrackingTouch (SeekBar seekBar)
```

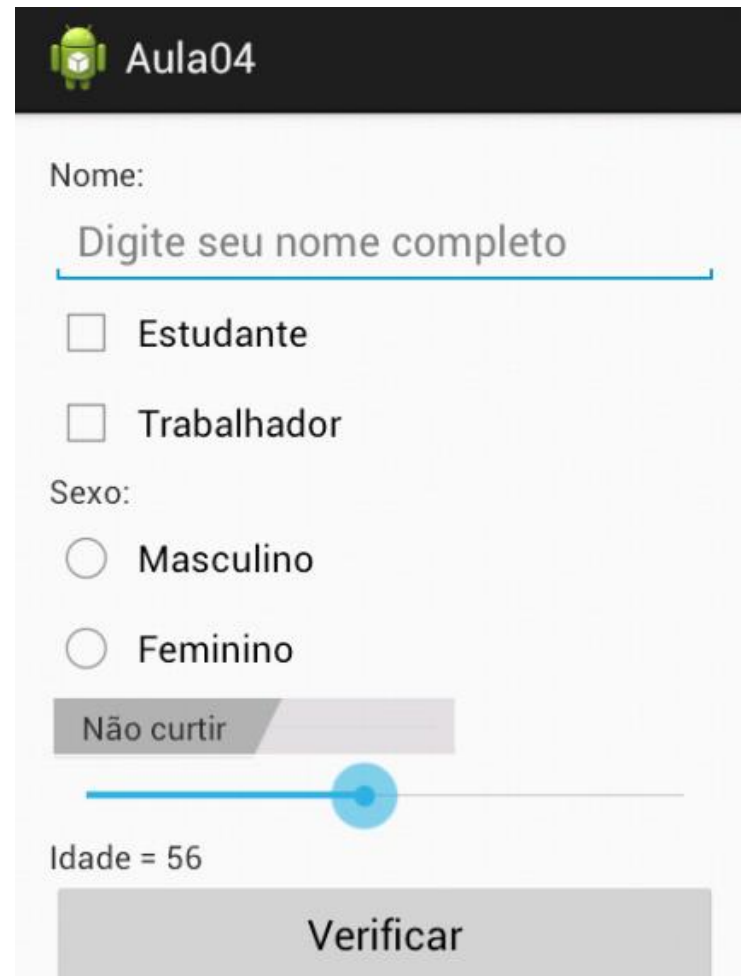
```
public void onProgressChanged (SeekBar seekBar,  
int progress, boolean fromUser)
```

# Manipulando um Seekbar no código Java

- ▶ Usamos o evento `public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)` para capturar as mudanças iniciadas pelo usuário.
- ▶ Para obter o valor atual definido pelo usuário, usamos o parâmetro `progress`.
- ▶ Após programar o evento `onProgressChanger` é necessário “setar” o listener relacionado ao componente inserido na tela, para que a operação seja realizada.
- ▶ Exemplo:

```
SeekBar skIdade;  
skIdade = (SeekBar) findViewById(R.id.skIdade);  
skIdade.setOnSeekBarChangeListener(avanca);
```

# Exemplo 04 - Seekbar



Aula04

Nome:  
Digite seu nome completo

Estudante

Trabalhador

Sexo:

Masculino

Feminino

Não curtir

Idade = 56

Verificar

# Ratingbar

- ▶ Um RatingBar é uma extensão da SeekBar e ProgressBar que mostra uma classificação em estrelas.



- ▶ Na paleta de Widgets é possível incluir um SeekBar ou podemos adicioná-lo via código XML:

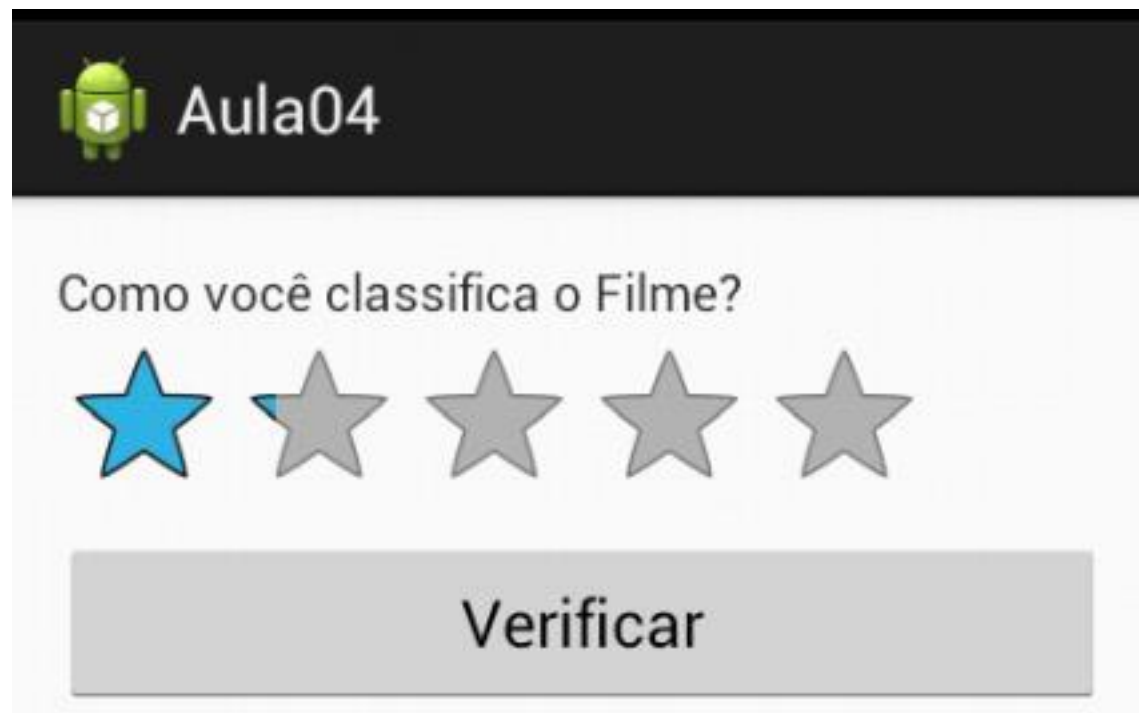


```
<RatingBar
    android:id="@+id/rtNota"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numStars="5"
/>
```

# Ratingbar

- ▶ Os principais atributos de um Ratingbar são:
  - ▶ `android:numStars` = define o número de itens (estrelas)
  - ▶ `android:rating` = define o valor padrão da classificação (OBS: deve ser um número com casa decimal, ex: 1,2)
  - ▶ `android:stepSize` = define o passo (variação) da classificação (OBS: deve ser um número com casa decimal, ex: 1,2)
- ▶ O listener utilizado para programar uma ação quando o usuário alterar a quantidade de estrelas marcada é o : **`OnRatingBarChangeListener`** ()
- ▶ O método: **`getRating`** () retorna a classificação atual (número de estrelas cheias).

# Exemplo 05 - Ratingbar

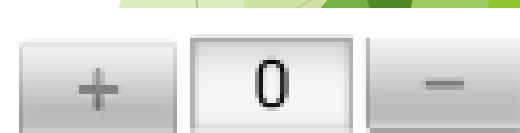
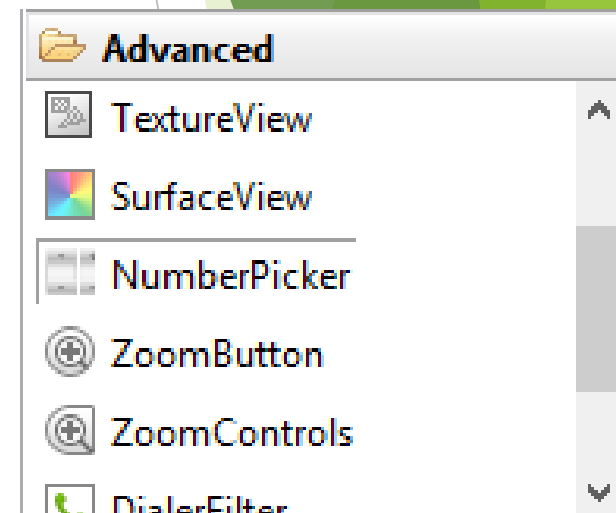


# Numberpicker

- ▶ O Numberpicker é um widget que permite que o usuário selecionar um número a partir de um conjunto pré-definido, ou seja, permite criar um índice de numeração que pode aumentar ou diminuir, com valores fixos determinados pelo programador
- ▶ Este componente fica na aba Advanced da paleta e pode ser adicionado arrastando a tela ou via código XML.

```
<NumberPicker  
    android:id="@+id/npIngressos"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:orientation="horizontal"
```

- ▶ Após incluir o elemento na parte visual, faz-se necessário criar uma associação ao controle no código Java para determinar o valor máximo, mínimo e o incremento.



# Numberpicker

- ▶ No código Java, no método onCreate da activity, devemos definir o valor máximo, mínimo e atual do NumberPicker.

```
NumberPicker npIngressos;  
npIngressos = (NumberPicker) findViewById(R.id.npIngressos);  
  
npIngressos.setMinValue(0);  
npIngressos.setMaxValue(10);  
npIngressos.setValue(0);
```

- ▶ O listener **OnValueChangedListener** é usado para programar a mudança do valor atual
- ▶ O método **getValue()** retorna o valor do seletor

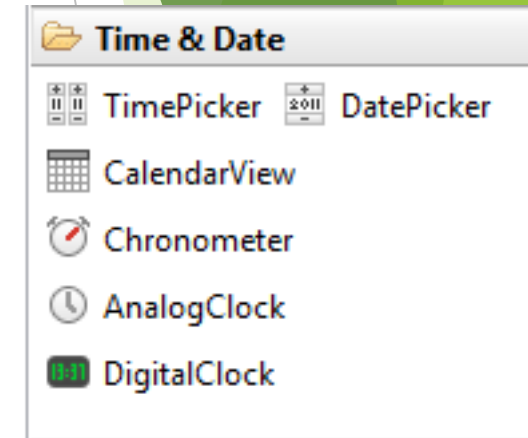


# DatePicker e TimePicker

- ▶ O Android possui alguns widgets específicos para trabalhar com valores de data e hora. Na paleta do Netbeans eles estão organizados na guia Time & Date.
- ▶ Estes elementos podem ser adicionados arrastados ao formulário ou através do código XML.

```
<DatePicker  
    android:id="@+id/datePicker1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```

```
<TimePicker  
    android:id="@+id/timePicker1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```



# DatePicker

- ▶ Os principais atributos de um DatePicker são:
  - ▶ android: calendarViewShown: Se o ponto de vista da agenda será mostrada.
  - ▶ android: maxDate = A data máxima mostrada no formato dd / mm / aaaa.
  - ▶ android: minDate = A data mínima mostrada no formato dd / mm / aaaa.
  - ▶ android: spinnersShown = Se os spinners são mostrados.
  - ▶ android: startYear = O primeiro ano, por exemplo, "1940".
- ▶ O listener **OnDateChangeListener** é usado para realizar uma programação quando o usuário alterar a data
- ▶ Os métodos `GetDayOfMonth ()`, `getMonth ()`, `getYear ()` entre outros são usados para obter os valores definidos pelo usuário

# TimePicker

- ▶ O listener **OnTimeChangeListener** é usado para realizar uma programação quando o usuário alterar o horário
- ▶ Os métodos **getCurrentHour()**, **getCurrentMinute()**, entre outros são usados para obter os valores definidos pelo usuário
- ▶ **OBS:** Para manipular um **DatePicker** ou **TimePicker** na classe Java, devemos inicialmente fazer a associação deste elemento usando o método **findViewById**, assim como realizado com outros widgets já estudados.

# Na próxima aula...

- ▶ Mais widgets: Spinner e ListView

