Desenvolvimento de Aplicações para Dispositivos Móveis

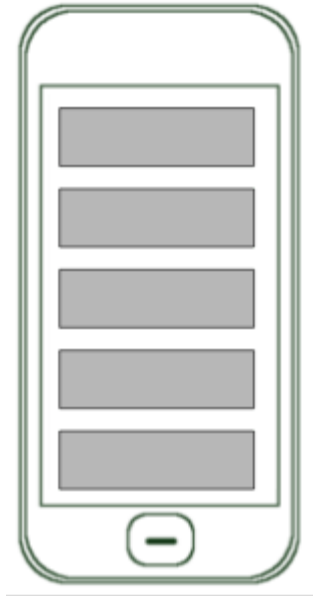
Aula 04 – Trabalhando ListViews

- O Que São ListViews
- Listviews
- Listviews Customizadas

Prof. Fernando Gonçalves Abadia

O que são ListViews

- A **ListView** é um dos componentes mais utilizados para listar dados na plataforma Android de forma eficiente. Basicamente ela organiza os dados um seguido do outro em formato vertical com a opção de rolagem.



O que são ListViews

- Existem 2 tipos de **ListViews** simples que são mais utilizados pelos desenvolvedores no desenvolvimento Android.
- A primeira é a classe **ListView,** que exibe uma lista vertical rolável de itens. A segunda, é a classe **GridView**, que organiza os itens de forma vertical, rolável e também separa em colunas como se fosse uma galeria.
- A utilização de cada umas dessas 2 classes varia de acordo com a sua necessidade em mostrar os dados para o usuário.

Trabalhando ListViews

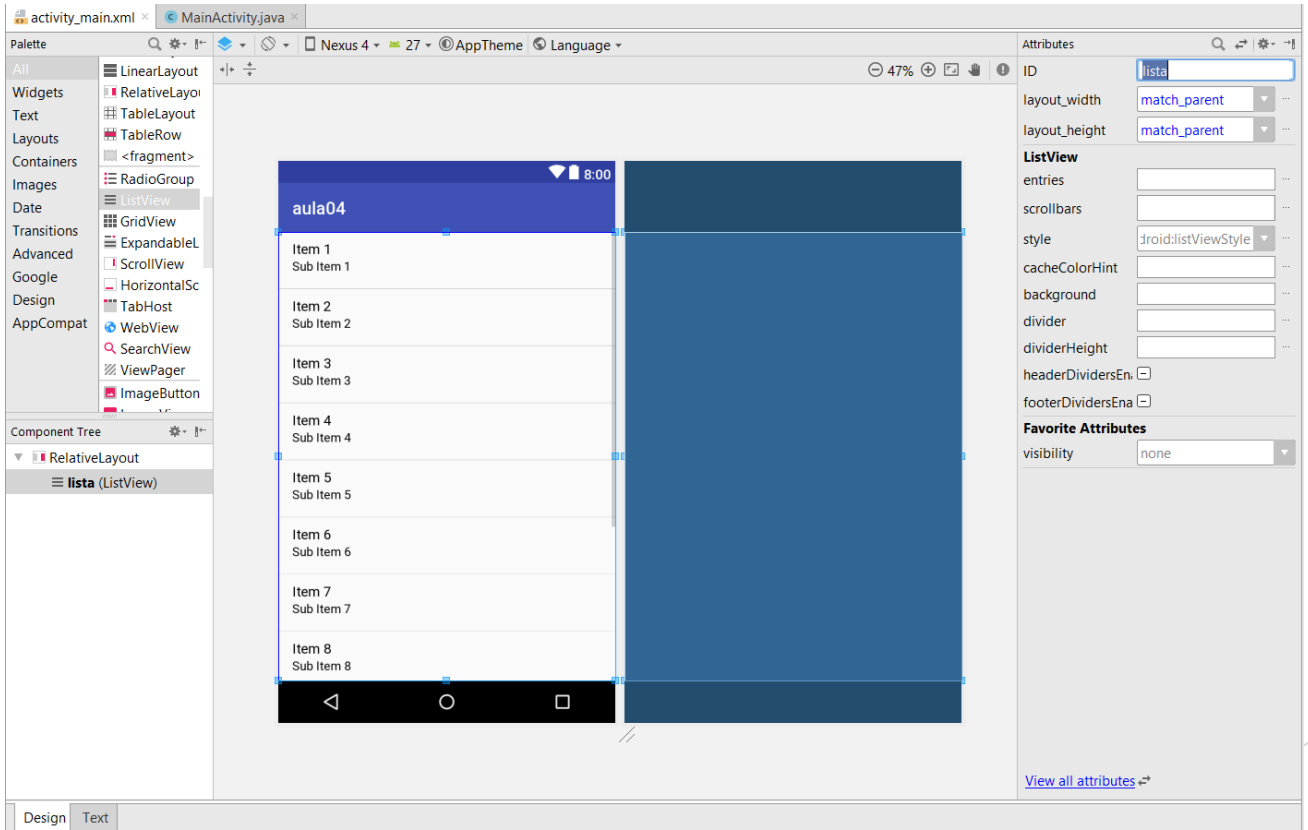
- Para trabalhar um exemplo simples e conciso, precisaremos iniciar um novo projeto, aula04 com uma atividade em branco.
- E no XML da atividade principal, apagar o TextView do Hello World de Entrada.

activity_main.xml | MainActivity.java

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context="com.example.arfso.aula04.MainActivity">
8
9  </android.support.constraint.ConstraintLayout>
10
```

Trabalhando ListViews

- Troque o Layout para um **RelativeLayout** e, na ferramenta de Design, arraste um **ListView** para a área de trabalho. Adicione um ID, como por exemplo: “lista”



Trabalhando ListViews

- No arquivo Java, podemos criar uma classe ListView e adicionar a ListView “lista” que criamos no XML pelo findViewById.

activity_main.xml | MainActivity.java

MainActivity

```
1  package com.example.arfso.aula04;
2
3  import ...
6
7  public class MainActivity extends AppCompatActivity {
8
9      @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.activity_main);
13         ListView listView = (ListView) findViewById(R.id.lista);
14     }
15 }
16
```

Trabalhando ListViews

- Precisamos agora criar um log de atividades para ser colocado na lista. E da mesma forma, criar uma lista de valores seja vetor ou lista de vetor (ArrayList).

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ListView listView = (ListView) findViewById(R.id.lista);

    Log.d( tag: "MainActivity", msg: "onCreate: Started.");

    ArrayList <String> nomes = new ArrayList<>();
    nomes.add("ALEXANDRE MARQUES DE OLIVEIRA");
    nomes.add("BRUNO PEREIRA RAMOS");
    nomes.add("CARLOS GEANINNE AQUINO SILVA");
    nomes.add("CECÍLIA DE BRITO PALHANO");
    nomes.add("DANRLEY CORREIA RAMOS");
```

Atividade corrente

Mensagem a ser criada

Trabalhando ListViews

- Para o próximo passo, será necessário criar um ArrayAdapter. Para isto precisaremos passar o contexto que está sendo trabalhado, o recurso de lista que se encontra em android.R.layout e a própria lista.
- Após a criação de um ArrayAdaptor, podemos setá-lo na listView.

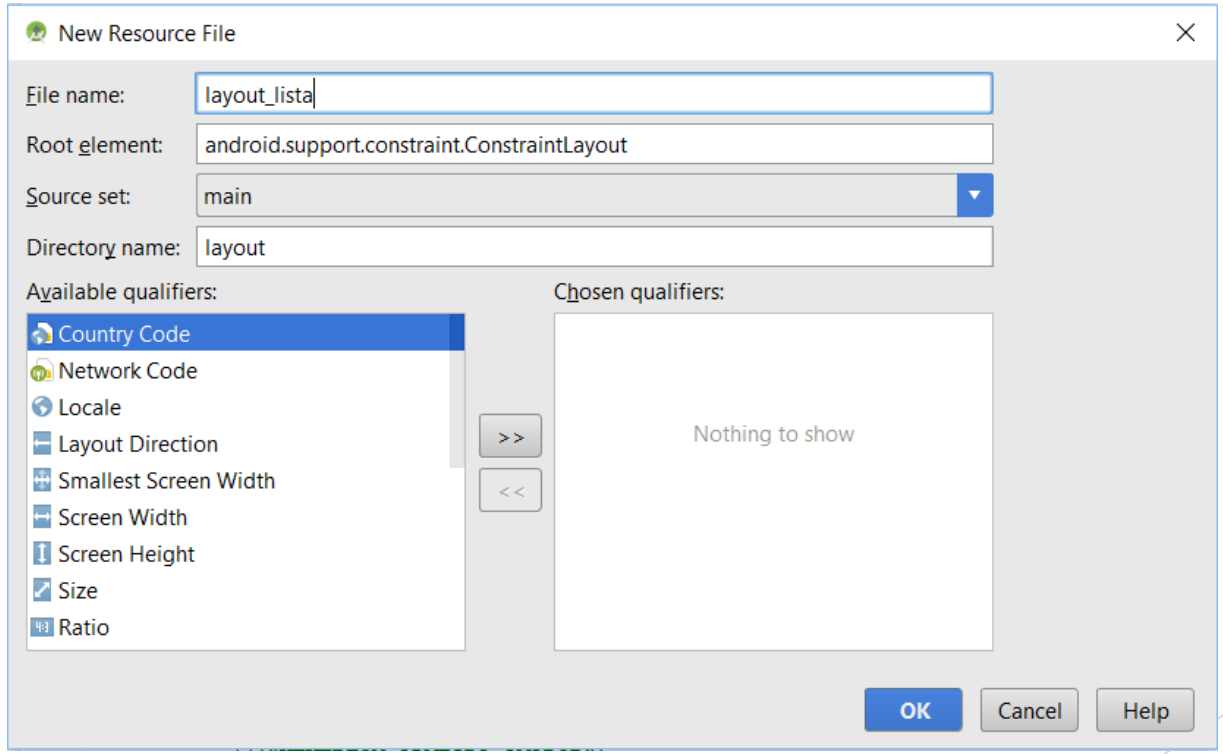
Teste a aplicação

```
    nomes.add("VITOR PIMENTA MARQUES");
    nomes.add("WEVERTON ALMEIDA AMADOR");
    ArrayAdapter adapter = new ArrayAdapter( context: this, android.R.layout.simple_list_item_1, nomes);
    listView.setAdapter(adapter);
}
```

Teste outros layouts

Trabalhando ListViews

- Podemos criar um novo XML para personalizar nossa ListView. Para isso, vamos criar um novo arquivo XML na pasta Layout e definí-lo como layout_lista



Trabalhando ListViews

- Neste novo Layout, vamos trocar o Layout descrito para um TextView e modifica-lo de acordo com o gosto e preferência.

activity_main.xml | MainActivity.java | layout_lista.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <TextView xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="50dp"
5      android:textSize="20sp"
6      android:textColor="@color/colorPrimaryDark"
7      android:gravity="center"
8      android:textAlignment="center">
9  </TextView>
```

Trabalhando ListViews

- Será necessário configurar esta Lista no arquivo java principal.

activity_main.xml | MainActivity.java | layout_lista.xml

```java
package com.example.arfso.aula04;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ListView listView = (ListView) findViewById(R.id.lista);

        Log.d( tag: "MainActivity", msg: "onCreate: Started.");

        ArrayList <String> nomes = new ArrayList<>();
        nomes.add("ALEXANDRE MARQUES DE OLIVEIRA");
        nomes.add("BRUNO PEREIRA RAMOS");
        nomes.add("CARLOS GEANINNE AQUINO SILVA");
        nomes.add("CECÍLIA DE BRITO PALHANO");
        nomes.add("DANRLEY CORREIA RAMOS");
        nomes.add("DIEGO VIANA DA SILVA");
        nomes.add("JONATAS TORRES VIEIRA");
        nomes.add("JULIO ALVES XAVIER");
        nomes.add("JÚLIO CESAR PEREIRA JÚNIOR");
        nomes.add("LUCAS DIOGO FRANÇA");
        nomes.add("LUCAS GABRIEL FERREIRA RODRIGUES");
        nomes.add("PAULO HENRIQUE TOMAZ DOS SANTOS");
        nomes.add("VINÍCIUS VIEIRA ABREU");
        nomes.add("VITOR PIMENTA MARQUES");
        nomes.add("WEVERTON ALMEIDA AMADOR");
        ArrayAdapter adapter = new ArrayAdapter( context: this, R.layout.layout_lista,nomes);
        listView.setAdapter(adapter);
    }
}
```

Customizando uma ListView

- Para adicionar colunas e linhas para cada item de uma lista, precisaremos modificar apenas o arquivo layout_lista.
  - Adicionar um LinearLayout com orientação horizontal e wheightSum 100
  - Modificar o TextView feito anteriormente e colocar dentro do layout linear

activity_main.xml | MainActivity.java | layout_lista.xml

LinearLayout | TextView

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="horizontal"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:weightSum="100">
7
8      <TextView
9          android:gravity="center"
10         android:textAlignment="center"
11         android:layout_width="match_parent"
12         android:layout_height="60dp"
13         android:id="@+id/texto1"
14         android:text="Teste Aula 04"
15         android:textColor="@color/colorPrimaryDark"
16         android:layout_weight="66.6">
17     </TextView>
18
19 </LinearLayout>
20
```

Customizando uma ListView

- Abaixo do TextView podemos colocar um novo layout Linear

```
<TextView
    android:gravity="center"
    android:textAlignment="center"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:id="@+id/texto1"
    android:text="Teste Aula 04"
    android:textColor="@color/colorPrimaryDark"
    android:layout_weight="66.6">
</TextView>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_weight="33.3">


</LinearLayout>
```

Customizando uma ListView

- Finalizando este XML, podemos acrescentar dois textView dentro deste novo layout linear de modo que o tamanho dos dois seja do mesmo tamanho do primeiro textView colocado.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_weight="33.3">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="30dp"
        android:gravity="center"
        android:id="@+id/texto2"
        android:text="Texto 2 aqui"
        />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="30dp"
        android:gravity="center"
        android:id="@+id/texto3"
        android:text="Texto 3 aqui"
        android:layout_below="@+id/texto2"
    />
```

Customizando uma ListView

- Para este novo layout, podemos criar uma nova classe java no mesmo local da atividade principal para armazenar alguns valores.
- No caso, está sendo criado a variável alunos. Crie um construtor, os setters e getters de maneira automática apenas com a função **generate** (alt+insert).

```java
package com.example.arfso.aula04;

/**
 * Created by arfso on 29/08/2018.
 */

public class Aluno {
    private String nome;
    private String matricula;
    private String status;

    public Aluno(String nome, String matricula, String status) {
        this.nome = nome;
        this.matricula = matricula;
        this.status = status;
    }

    public String getNome() {
        return nome;
    }
```

Customizando uma ListView

- E desta forma, precisaremos atualizar a lista feita na atividade principal para que aceite esta nova classe Aluno.

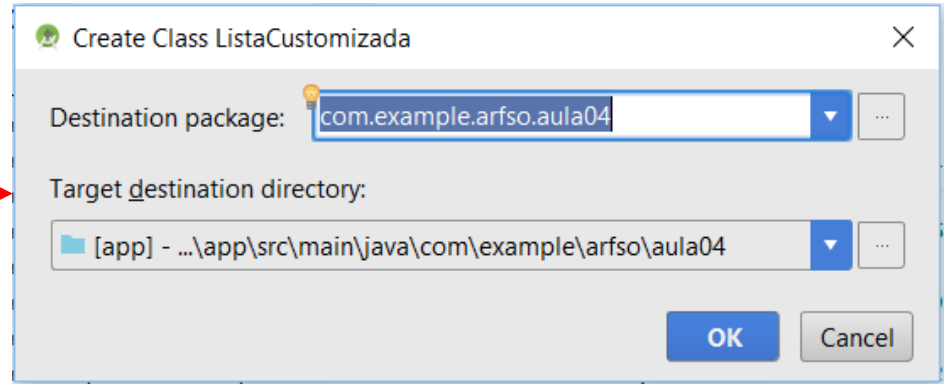
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ListView listView = (ListView) findViewById(R.id.lista);

    Log.d( tag: "MainActivity", msg: "onCreate: Started.");

    ArrayList <Aluno> alunos = new ArrayList<>();
    alunos.add(new Aluno( nome: "ALEXANDRE MARQUES DE OLIVEIRA", matricula: "2016.2.0120.0035-9", status: "ok"));
    alunos.add(new Aluno( nome: "BRUNO PEREIRA RAMOS", matricula: "2016.2.0120.0038-3", status: "ok"));
    alunos.add(new Aluno( nome: "CARLOS GEANINNE AQUINO SILVA", matricula: "2015.2.0120.0012-2", status: "ok"));
    alunos.add(new Aluno( nome: "CECÍLIA DE BRITO PALHANO", matricula: "2016.1.0120.0016-4", status: "ok"));
    alunos.add(new Aluno( nome: "DANRLEY CORREIA RAMOS", matricula: "2016.1.0120.0016-4", status: "ok"));
    alunos.add(new Aluno( nome: "DIEGO VIANA DA SILVA", matricula: "2018.1.0120.0007-0", status: "ok"));
    alunos.add(new Aluno( nome: "JONATAS TORRES VIEIRA", matricula: "2014.2.0120.0029-0", status: "ok"));
    alunos.add(new Aluno( nome: "JULIO ALVES XAVIER", matricula: "2017.1.0120.0025-0", status: "ok"));
    alunos.add(new Aluno( nome: "JÚLIO CESAR PEREIRA JÚNIOR", matricula: "2016.1.0120.0035-0", status: "ok"));
    alunos.add(new Aluno( nome: "LUCAS DIOGO FRANÇA", matricula: "2016.2.0120.0023-5", status: "ok"));
    alunos.add(new Aluno( nome: "LUCAS GABRIEL FERREIRA RODRIGUES", matricula: "2017.1.0120.0026-9", status: "ok"));
    alunos.add(new Aluno( nome: "PAULO HENRIQUE TOMAZ DOS SANTOS", matricula: "2016.1.0120.0027-0", status: "ok"));
    alunos.add(new Aluno( nome: "VINÍCIUS VIEIRA ABREU", matricula: "2017.1.0120.0049-8", status: "ok"));
    alunos.add(new Aluno( nome: "VITOR PIMENTA MARQUES", matricula: "2017.1.0120.0037-4", status: "ok"));
    alunos.add(new Aluno( nome: "WEVERTON ALMEIDA AMADOR", matricula: "2016.2.0120.0012-0", status: "ok");

    ArrayAdapter adapter = new ArrayAdapter( context: this, R.layout.layout_lista,alunos);
    listView.setAdapter(adapter);
}
```

Obsoleto

Customizando uma ListView

- Uma vez que o **ArrayAdapter** deixou de funcionar, podemos criar nossa própria classe **Adapter** com o nome ListaCustomizada.
- Crie a classe ListaCustomizada de maneira automática apenas com as opções fornecidas pelo Android Studio.

```
    ListaCustomizada lista = new ListaCustomizada(this, R.layout.layout_lista,alunos);
    listView.setAdapter(lista);
}
}
```



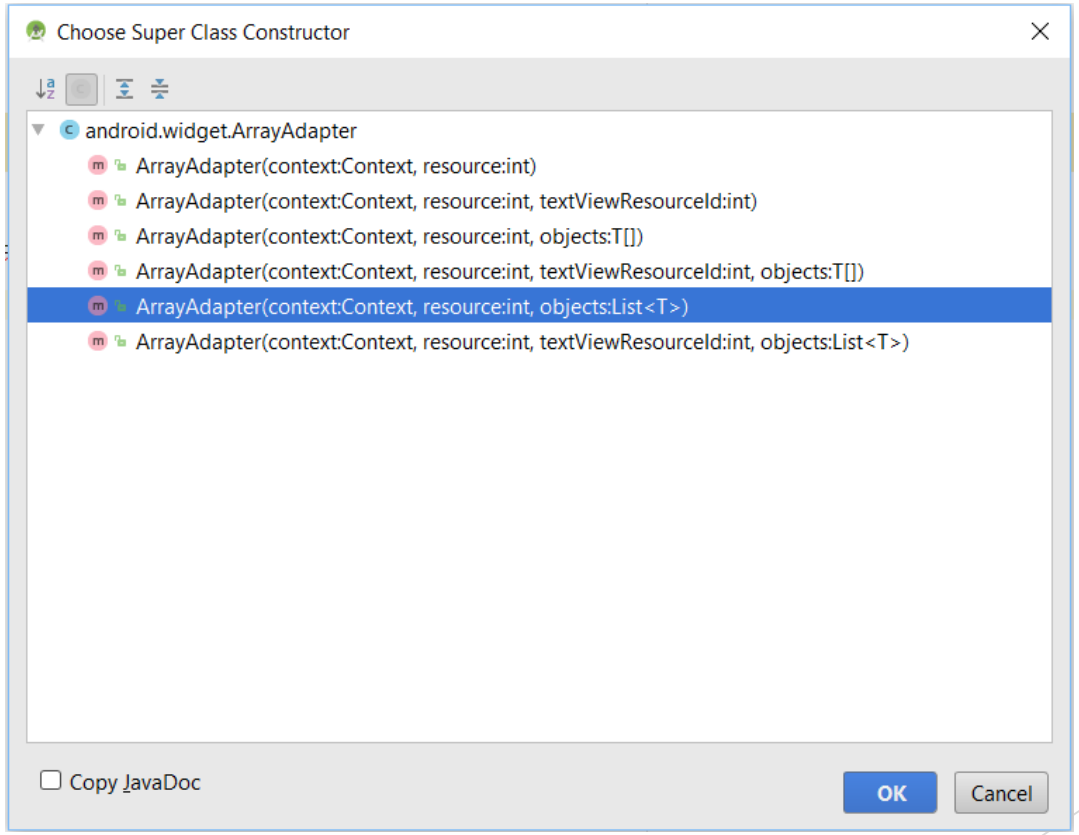
Customizando uma ListView

- Precisaremos modificar a classe recém criada de forma a aceitar a classe Aluna feita anteriormente. Para isso, herdaremos a classe **ArrayAdaptor**.

```
package com.example.arfso.aula04;

import android.widget.ArrayAdapter;

/**
 * Created by arfso on 29/08/2018.
 */

class ListaCustomizada extends ArrayAdapter <Aluno> {

}
```

Customizando uma ListView

- Ao criar o construtor, precisaremos de um que receba todas as informações que precisamos: contexto, layout e a lista de alunos.



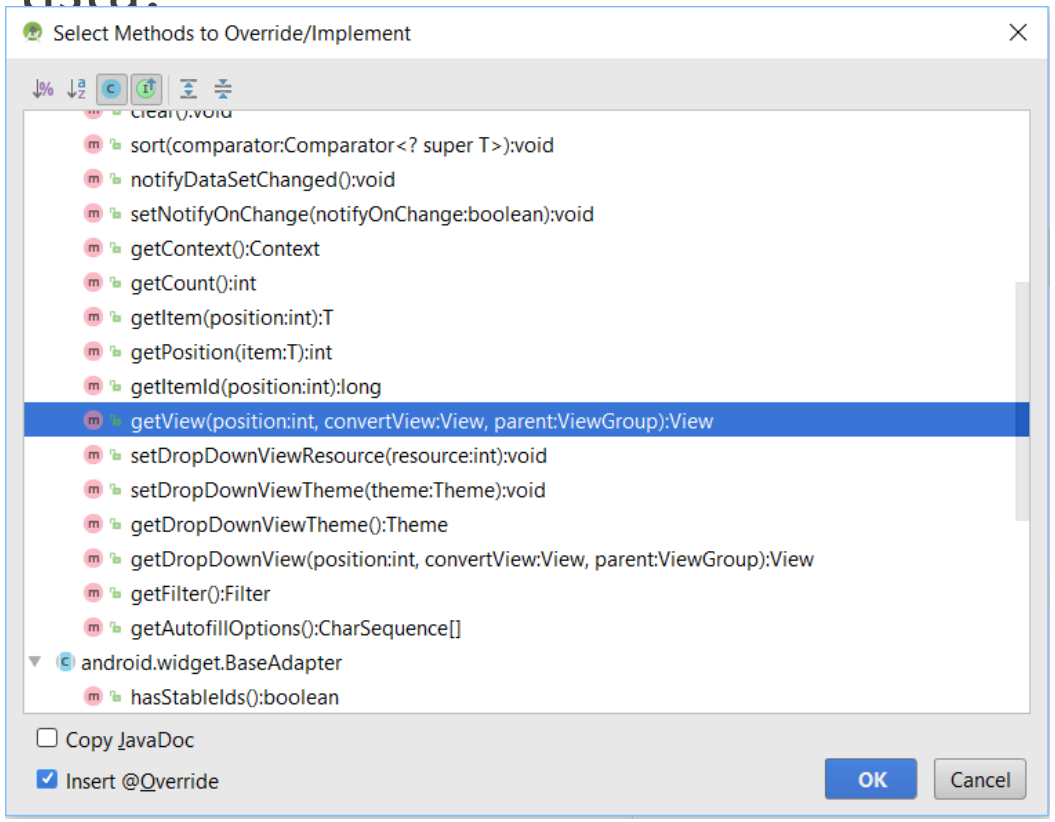
Customizando uma ListView

- E ao criar o construtor, precisaremos trocar a **List** por uma **ArrayList**.

```
class ListaCustomizada extends ArrayAdapter <Aluno> {
    private Context contexto;

    public ListaCustomizada(@NonNull Context context, int resource, @NonNull ArrayList <Aluno> objects) {
        super(context, resource, objects);
        this.contexto = context;
    }
}
```

Customizando uma ListView

- No **Generate**, precisaremos inserir a função **getView** através de um método **Override** para que seja possível trabalhar a escrita da lista.



Customizando uma ListView

- Precisamos passar os parâmetros de Aluno para a View através da classe Aluno e precisaremos trabalhar item por item. Desta forma, foi criado cada **string** separadamente e adicionado à variável através da função **getItem** do java.

```
class ListaCustomizada extends ArrayAdapter <Aluno> {
    private Context contexto;

    public ListaCustomizada(@NonNull Context context, int resource, @NonNull ArrayList <Aluno> objects) {
        super(context, resource, objects);
        this.contexto = context;
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        String nome = getItem(position).getNome();
        String matricula = getItem(position).getMatricula();
        String status = getItem(position).getStatus();

        Aluno aluno = new Aluno(nome,matricula,status);
    }
}
```

Customizando uma ListView

- Com cada item separado da classe alunos, podemos agora colocar cada item no layout através da classe **LayoutInflater.**

```
@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    String nome = getItem(position).getNome();
    String matricula = getItem(position).getMatricula();
    String status = getItem(position).getStatus();

    Aluno aluno = new Aluno(nome,matricula,status);
    LayoutInflater inflater = LayoutInflater.from(contexto);
}
```

Customizando uma ListView

- Para mandar os nomes para o layout, precisaremos do parâmetro de layout, o resource (resourceLayout). Podemos acrescentá-los à lista de variáveis da classe ListaCustomizada.

```
class ListaCustomizada extends ArrayAdapter <Aluno> {
    private Context contexto;
    int resourceLayout;

    public ListaCustomizada(@NonNull Context context, int resource, @NonNull ArrayList <Aluno> objects) {
        super(context, resource, objects);
        this.contexto = context;
        this.resourceLayout = resource;

    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        String nome = getItem(position).getNome();
        String matricula = getItem(position).getMatricula();
        String status = getItem(position).getStatus();

        Aluno aluno = new Aluno(nome,matricula,status);
        LayoutInflater inflater = LayoutInflater.from(contexto);
    }
}
```

Customizando uma ListView

- Precisaremos trabalhar a variável convertView que nos foi fornecida através da getView. Para isso precisaremos do nosso inflater com o resourceLayout.

```
@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    String nome = getItem(position).getNome();
    String matricula = getItem(position).getMatricula();
    String status = getItem(position).getStatus();

    Aluno aluno = new Aluno(nome,matricula,status);
    LayoutInflater inflater = LayoutInflater.from(contexto);
    convertView = inflater.inflate(resourceLayout,parent, attachToRoot: false);
}
```

Customizando uma ListView

- Finalmente precisaremos passar os ID's de cada textView que se encontram no Layout para que possa ser adaptado. E retornar a view modificada.

```
@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    String nome = getItem(position).getNome();
    String matricula = getItem(position).getMatricula();
    String status = getItem(position).getStatus();

    Aluno aluno = new Aluno(nome,matricula,status);
    LayoutInflater inflater = LayoutInflater.from(contexto);
    convertView = inflater.inflate(resourceLayout,parent, attachToRoot: false);
    TextView tvNome = (TextView) convertView.findViewById(R.id.texto1);
    TextView tvMatricula = (TextView) convertView.findViewById(R.id.texto2);
    TextView tvStatus = (TextView) convertView.findViewById(R.id.texto3);

    tvNome.setText(nome);
    tvMatricula.setText(matricula);
    tvStatus.setText(status);
    return convertView;
}
```

Customizando uma ListView

- Finalmente precisaremos passar os ID's de cada textView que se encontram no Layout para que possa ser adaptado. E retornar a view modificada.

```
@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    String nome = getItem(position).getNome();
    String matricula = getItem(position).getMatricula();
    String status = getItem(position).getStatus();

    Aluno aluno = new Aluno(nome,matricula,status);
    LayoutInflater inflater = LayoutInflater.from(contexto);
    convertView = inflater.inflate(resourceLayout,parent, attachToRoot: false);
    TextView tvNome = (TextView) convertView.findViewById(R.id.texto1);
    TextView tvMatricula = (TextView) convertView.findViewById(R.id.texto2);
    TextView tvStatus = (TextView) convertView.findViewById(R.id.texto3);

    tvNome.setText(nome);
    tvMatricula.setText(matricula);
    tvStatus.setText(status);
    return convertView;
}
```

Customizando uma ListView

- Podemos ainda inserir um método que mostrará em qual item foi clicado. Para isso, podemos adicionar uma mensagem **toast** na atividade principal.

```
ListaCustomizada lista = new ListaCustomizada( context: this, R.layout.layout_lista,alunos);
listView.setAdapter(lista);

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(getApplicationContext(), text: "Clicou no item "+position,Toast.LENGTH_SHORT).show();
    }
});
```

Customizando uma ListView

- Agora basta construir o aplicativo e verificar os resultados.

Exercício

Crie uma forma de inserir valores para uma ListView