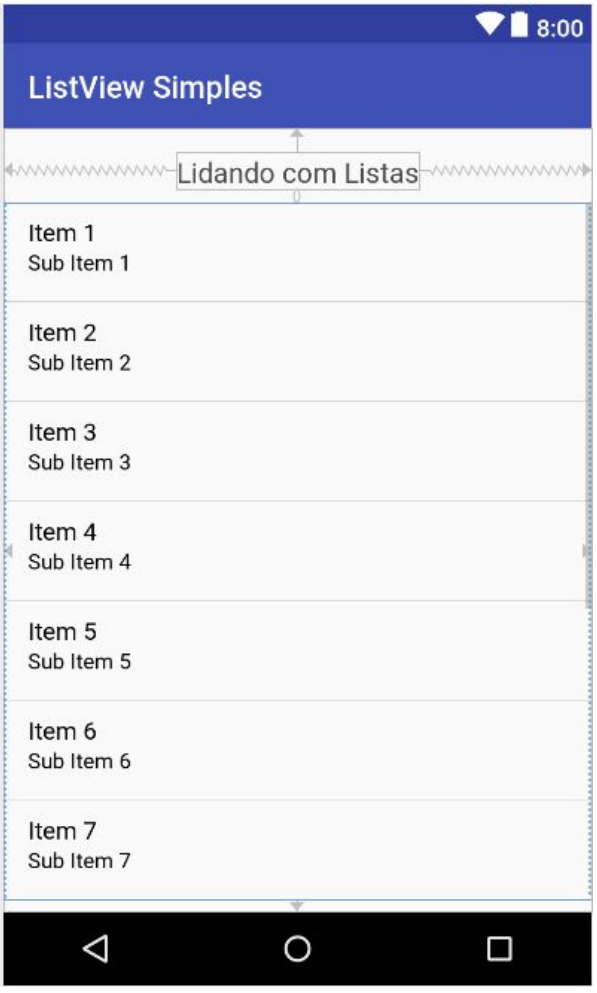
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CURSO: Técnico em Informática Integrado - 3º Ano
DISCIPLINA: LAM

PROFESSOR: Wendell Fioravante da Silva Diniz

Tutorial ListView simples e com adaptador personalizado

O objeto *ListView* do android SDK oferece algumas facilidades para criação de interfaces de listas, muito comuns em aplicativos Android. Existem várias formas de montar tanto listas fixas, com conteúdos pré-definidos, quanto listas dinâmicas, com conteúdos gerados de forma procedural.

Como primeiro exercício, vamos montar uma lista simples com dois itens. Crie um novo projeto com o nome **Lista Simples**, com uma atividade vazia. No layout da *MainActivity*, adicione um componente do tipo **ListView.** O layout deve ficar parecido com o exemplo abaixo.



Crie também um novo arquivo de layout, que chamaremos de *item_lista.xml*, contendo dois *TextViews*, que usaremos para exibir o nome e o aniversário. Sempre que criarmos listas, teremos que criar o layout da linha desta forma.

Agora, vamos programar o comportamento da atividade. Neste app, mostraremos dois itens na lista, um nome e um aniversário. Para passar mais de um item, podemos usar um objeto *SimpleAdapter*, que recebe como parâmetros, dois vetores, um com os valores e outro com os IDs dos elementos de interface onde serão renderizados. Usaremos um *HashMap*, para gravar os valores:

```
1  package com.example.alunos.listviewsimples;
2
3  import android.support.v7.app.AppCompatActivity;
4  import android.os.Bundle;
5  import android.widget.ListView;
6  import android.widget.SimpleAdapter;
7
8  import java.util.ArrayList;
9  import java.util.HashMap;
10
11 public class MainActivity extends AppCompatActivity {
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17
18         String[] nomes = {"João", "Maria", "José", "Ana"};
19         String[] aniversarios = {"12/01", "07/09", "05/04", "23/07"};
20
21         ListView lista = findViewById(R.id.listView);
22
23         ArrayList<HashMap<String, String>> valores = new ArrayList<>();
24         for (int i = 0; i < nomes.length; i++) {
25             HashMap<String, String> item = new HashMap<>();
26             item.put("nome", nomes[i]);
27             item.put("aniv", aniversarios[i]);
28             valores.add(item);
29         }
30
31         String[] chaves = {"nome", "aniv"};
32         int[] labels = {R.id.lblFirst,R.id.lblSecond};
33
34         SimpleAdapter adapter = new SimpleAdapter(getApplicationContext(),
35              valores,R.layout.item_lista,chaves,labels);
36
37         lista.setAdapter(adapter);
38
39     }
40 }
```

Na linha 23 declaramos um *ArrayList*, onde cada item será um *HashMap*, contendo um par chave -> valor, identificando os campos que queremos exibir na lista, neste caso, nomes e aniversários. Na linha 34, usamos um *SimpleAdapter*, que é um adaptador nativo do Android. Rode o aplicativo e corrija os possíveis erros.

Muitas vezes, queremos exibir dados de forma mais personalizada. Por exemplo, durante nossa aula, vimos um exemplo usando uma imagem e duas etiquetas de texto.

Agora, vamos construir um app simples de contatos, usando uma lista dinâmica com adaptador personalizado. Inicie um novo projeto no Android Studio chamado *ListaDinamica*

e adicionar uma atividade vazia como nossa *Main Activity*. Abra o arquivo *MainActivity.java* e acrescente um método chamado *mostrarLista(View v)*, onde criaremos um Intent que passará os dados e iniciará a atividade de lista que programaremos na classe *mostraListaDinamica.class.* O código deve ficar desta forma:

```
1  package com.example.wendell.listadinmica;
2
3  import android.content.Intent;
4  import android.os.Bundle;
5  import android.support.v7.app.AppCompatActivity;
6  import android.view.View;
7
8  public class MainActivity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14     }
15
16     public void mostrarLista(View v) {
17         Intent it = new Intent(this, mostraListaDinamica.class);
18         startActivity(it);
19     }
20 }
```

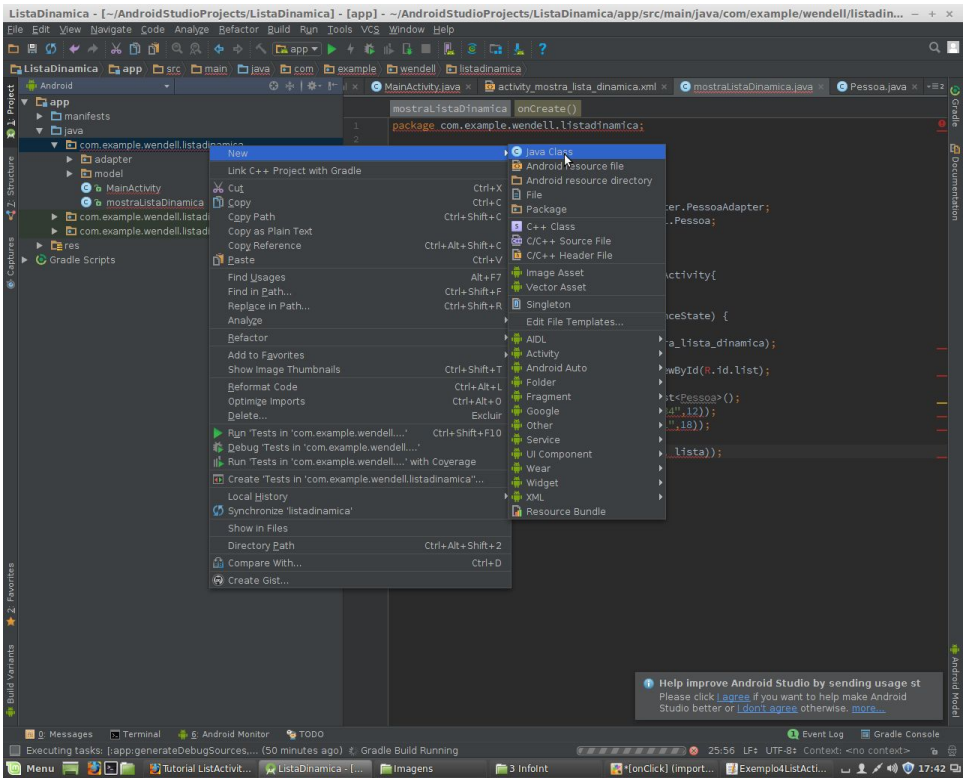
No arquivo activity_main.xml, insira um botão que iniciará uma nova atividade que exibirá nossa lista dinâmica. A interface deve ficar parecida com a figura abaixo:

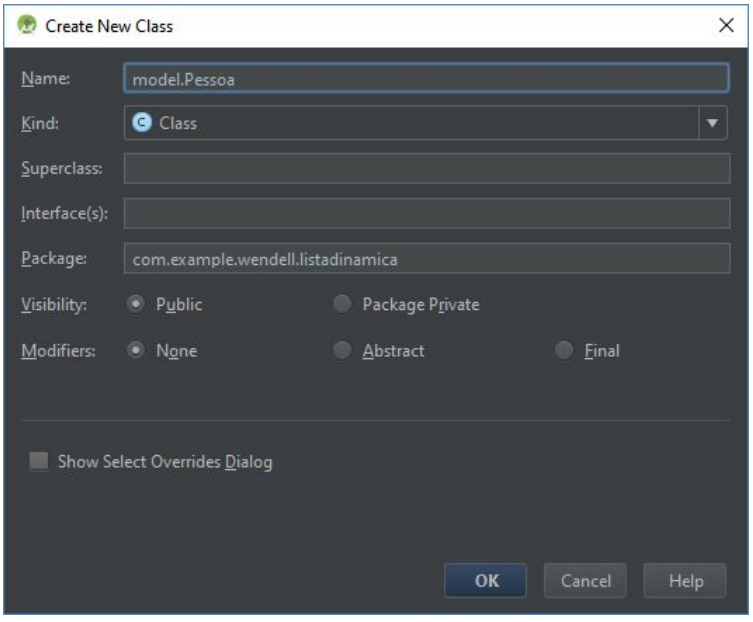
Na parte de design, associe o callback *onClick* do botão com o método *mostrarLista* que criamos no código da *MainActivity*.

No Android SDK, para popular as listas com valores, precisamos de uma classe específica chamada *Adapter*. Existem alguns adapters prontos, que ajudam a mostrar layouts de lista

pré-determinados. Entretanto, se necessário, podemos usar layouts personalizados que podem mostrar outros tipos de valores conforme as necessidades de nossa aplicação. Vamos aprender a construir um destes adaptadores.

Agora, vamos construir um objeto que será o modelo de dados da aplicação. Clique com o botão direito no pacote de seu projeto e escolha criar uma nova classe. Dê o nome *model.Pessoa* a esta classe.



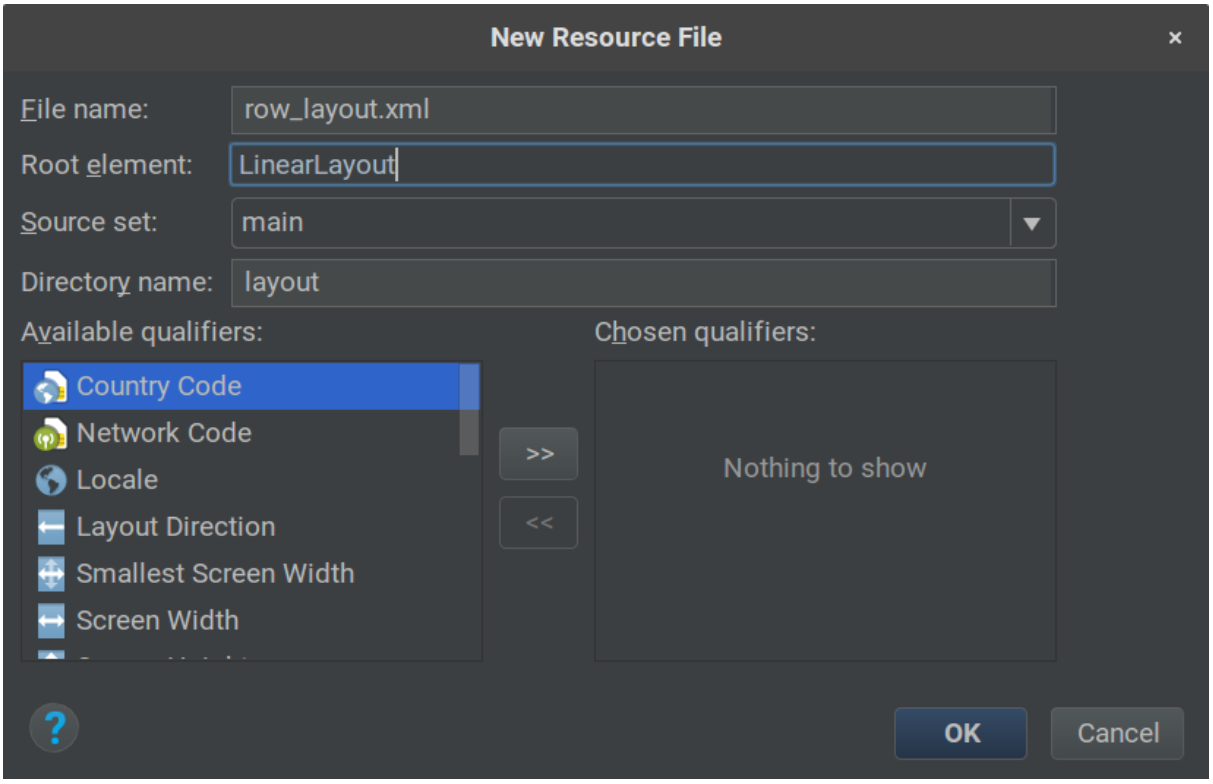


Insira o código abaixo para a classe *Pessoa*. Os dados poderiam vir, por exemplo, de um banco de dados, então esta classe cumpre a função de *Data Holder,* ou seja é uma classe

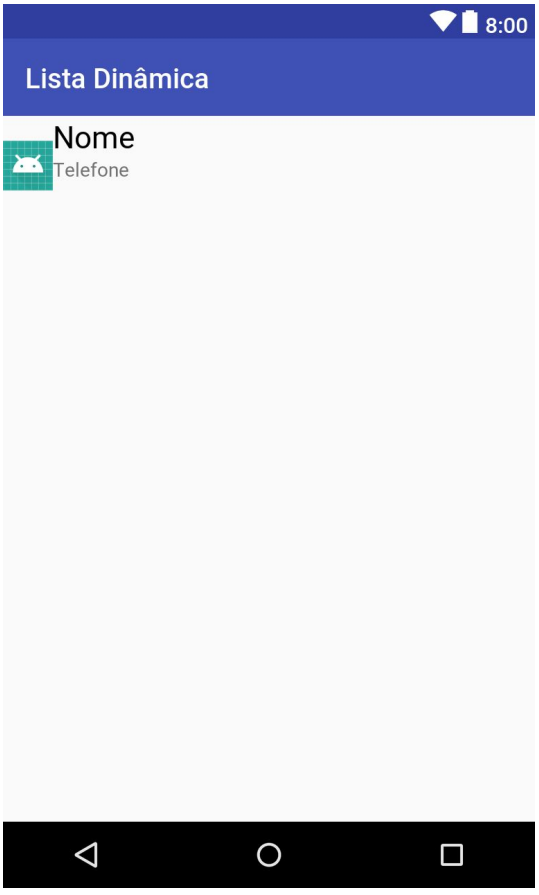
que serve para armazenar dados que vem de uma fonte (Banco de dados, arquivo, servidor remoto, etc…).

```java
package com.example.wendell.listadinmica.model;

public class Pessoa {

    private String nome;
    private String telefone;
    private int imagem;

    public Pessoa(String nome, String telefone, int idImagem) {
        this.nome = nome;
        this.telefone = telefone;
        this.imagem = idImagem;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public int getImagem() {
        return imagem;
    }

    public void setImagem(int id) {
        this.imagem = id;
    }
}
```

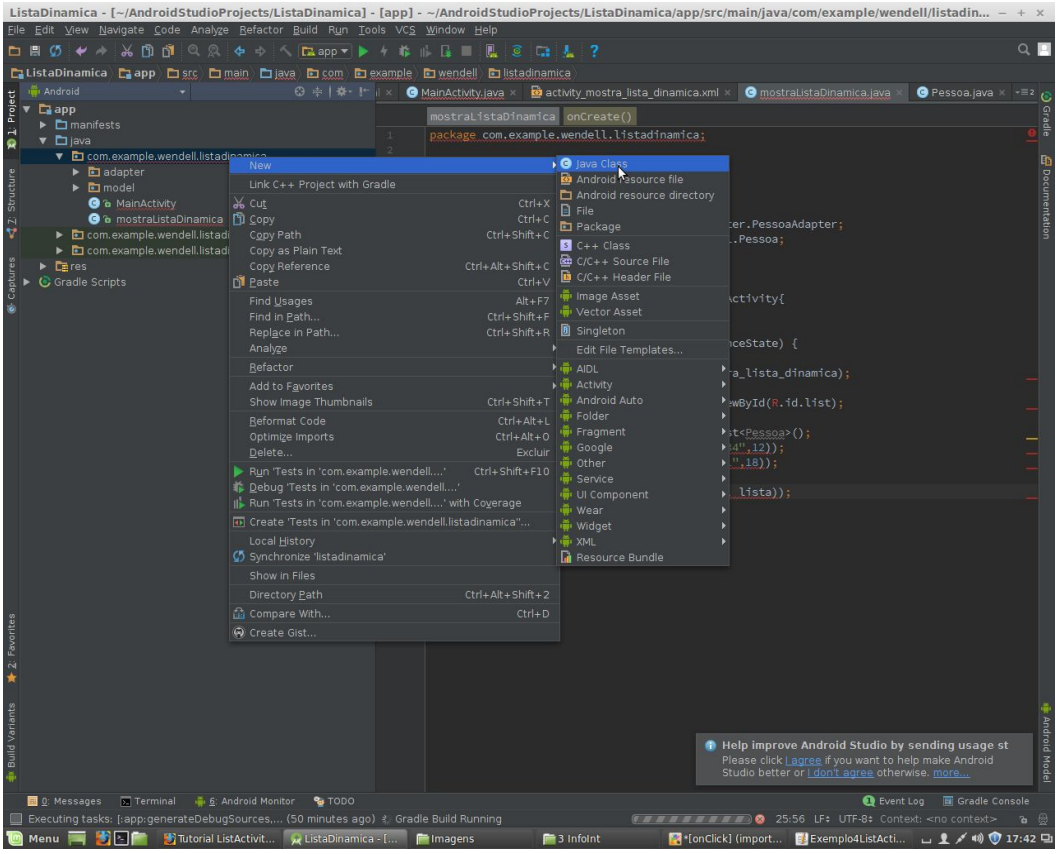
Antes de criar o adaptador personalizado, precisamos de um arquivo de layout com a formatação de cada item da lista. Crie um novo layout na pasta *res/layout* e dê a ele o nome *row_layout.xml*. Para isso, clique com o botão direito sobre a pasta *res/layout* e escolha a opção *New layout resource file.* Na caixa que aparece, mude o valor do elemento raiz (Root element) para *LinearLayout.*

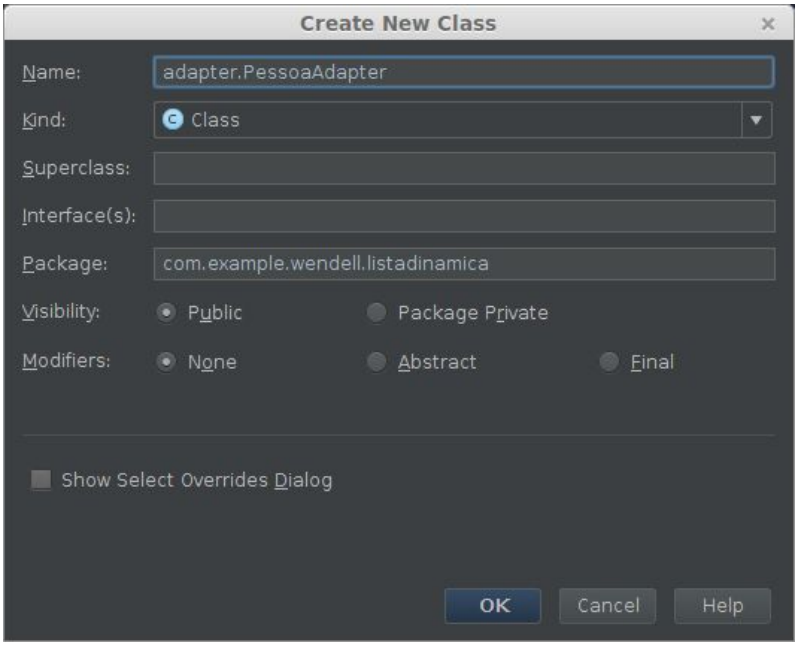


Mude a orientação do *LinearLayout* para horizontal (no lado direito, na aba de propriedades) e a seguir, insira um *ImageView* como primeiro item e um novo *LinearLayout* como segundo item. Neste, inserimos dois *TextViews* para mostrar os campos da classe do modelo de dados. Deixe o layout parecido com o abaixo.



Agora, repita o procedimento para criar uma nova classe e crie a classe *adapter.PessoaAdapter*.





Vamos criar o código responsável por receber os dados provenientes do modelo e processá-lo para exibição. O SDK fornece uma classe base, chamada *BaseAdapter,* que contém a funcionalidade mínima de um adaptador. Nossa classe será baseada nela. Observe também o construtor da classe:

```java
package com.example.wendell.listadinmica.adapter;

import android.app.Activity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import com.example.wendell.listadinmica.R;
import com.example.wendell.listadinmica.model.Pessoa;
import java.util.List;

public class PessoaAdapter extends BaseAdapter {

    private Activity atividade;
    private List<Pessoa> lista;

    public PessoaAdapter(Activity atividade, List<Pessoa> lista) {
        this.atividade = atividade;
        this.lista = lista;
    }
```

Veja que declaramos dois atributos privados, do tipo *Activity* e *List<Pessoa>*. Esses dois atributos representam a atividade que usou o adapter e a lista de objetos da classe *Pessoa* que queremos exibir.

Vimos que a classe *BaseAdapter* possui um conjunto de métodos virtuais, os quais devemos obrigatoriamente, implementar em nossa classe derivada. Vamos implementá-los de acordo com nossa aplicação:

```java
    @Override
    public int getCount() {
        return lista.size();
    }

    @Override
    public Object getItem(int position) {
        return lista.get(position);
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Pessoa obj = lista.get(position);

        View v = atividade.getLayoutInflater().inflate(R.layout.row_layout,
         parent, false);

        TextView textNome = v.findViewById(R.id.txtNome);
        textNome.setText(obj.getNome());

        TextView textTelefone = v.findViewById(R.id.txtTelefone);
        textTelefone.setText(obj.getTelefone());

        ImageView imgImagem = v.findViewById(R.id.imgImagem);
        imgImagem.setImageResource(obj.getImagem());

        return v;
    }
}
```

Repare no método getView. Nele, usamos o método *getLayoutInflater(),* que é um método da classe *Activity*, que pega o layout da linha que criamos e preenche com os dados de cada item na lista. Ao final, o objeto expandido é devolvido para ser exibido no *ListView*.

Como vamos passar uma lista de objetos para a atividade da lista, precisamos modificar o código da classe *Pessoa*, para que ele implemente a interface da classe *Parcelable*, do SDK. Esta classe determina como o objeto é empacotado e desempacotado, quando é passado entre atividades. Acrescente a linhas abaixo à classe *Pessoa:*

```
1 public class Pessoa implements Parcelable {
2
3     private String nome;
4     private String telefone;
5     private int imagem;
```

Aqui, indicamos que nossa classe *Pessoa* implementa a interface *Parcelable.* A seguir, precisamos implementar alguns métodos:

```
1  public class Pessoa implements Parcelable {
2
3      ...
4
5      protected Pessoa(Parcel in) {
6          nome = in.readString();
7          telefone = in.readString();
8          imagem = in.readInt();
9      }
10
11     @Override
12     public int describeContents() {
13         return 0;
14     }
15
16     @Override
17     public void writeToParcel(Parcel dest, int flags) {
18         dest.writeString(nome);
19         dest.writeString(telefone);
20         dest.writeInt(imagem);
21     }
22
23     @SuppressWarnings("unused")
24     public static final Parcelable.Creator<Pessoa> CREATOR =
25       new Parcelable.Creator<Pessoa>() {
26         @Override
27         public Pessoa createFromParcel(Parcel in) {
28             return new Pessoa(in);
29         }
30
31         @Override
32         public Pessoa[] newArray(int size) {
33             return new Pessoa[size];
34         }
35     };
36 }
```

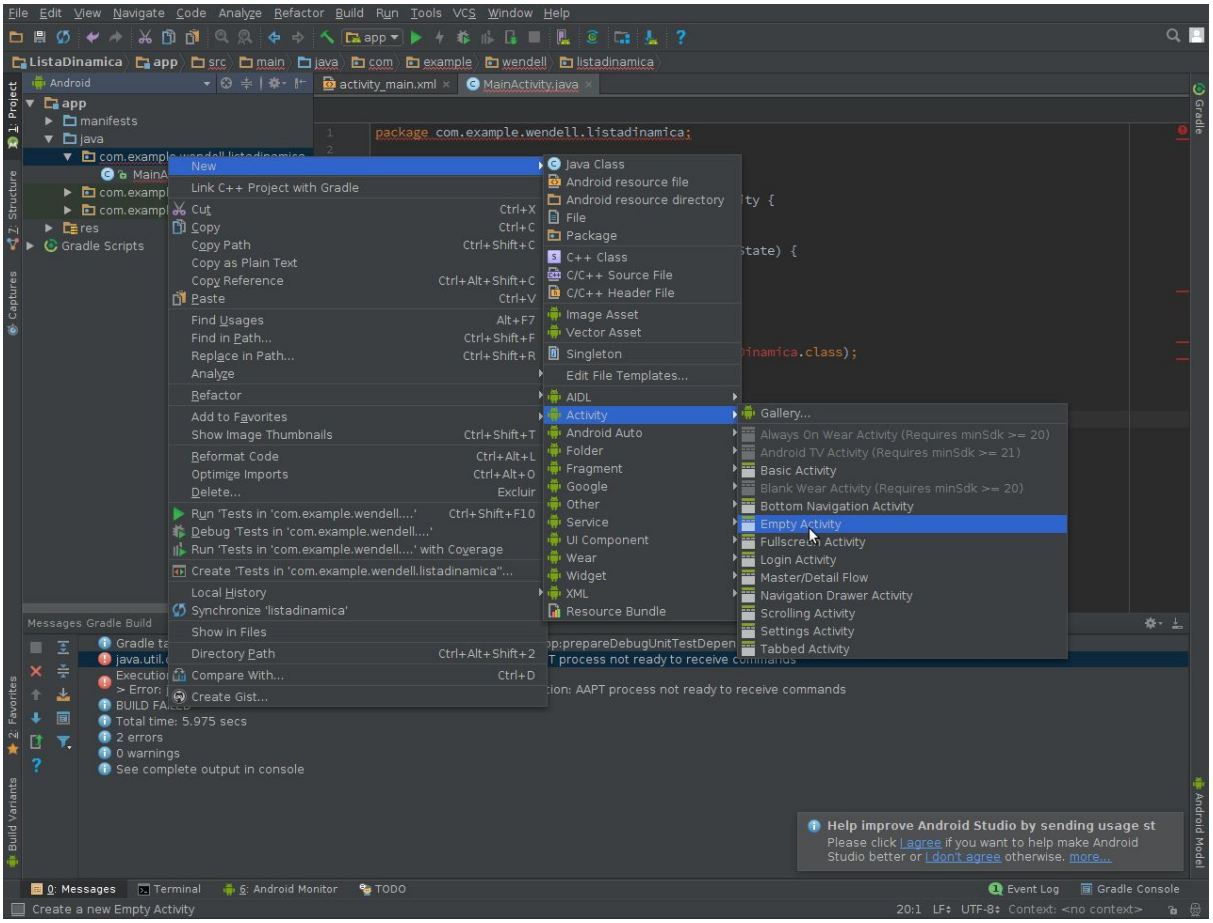
Esses métodos vêm da interface *Parcelable* e são eles que indicam como os dados serão empacotados. Repare que inserimos um novo construtor, que indica como construir a classe quando recebe um objeto do tipo *Parcel* como parâmetro. Este objeto é que é o resultado do empacotamento da classe. Código completo:

```java
package com.example.wendell.listadinmica.model;

import android.os.Parcel;
import android.os.Parcelable;

public class Pessoa implements Parcelable {

    private String nome;
    private String telefone;
    private int imagem;

    public Pessoa(String nome, String telefone, int idImagem) {
        this.nome = nome;
        this.telefone = telefone;
        this.imagem = idImagem;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public int getImagem() {
        return imagem;
    }

    public void setImagem(int id) {
        this.imagem = id;
    }

    protected Pessoa(Parcel in) {
        nome = in.readString();
        telefone = in.readString();
        imagem = in.readInt();
    }

    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(nome);
        dest.writeString(telefone);
        dest.writeInt(imagem);
    }

    @SuppressWarnings("unused")
    public static final Parcelable.Creator<Pessoa> CREATOR =
      new Parcelable.Creator<Pessoa>() {
        @Override
        public Pessoa createFromParcel(Parcel in) {
            return new Pessoa(in);
        }

        @Override
        public Pessoa[] newArray(int size) {
            return new Pessoa[size];
        }
    };
}
```

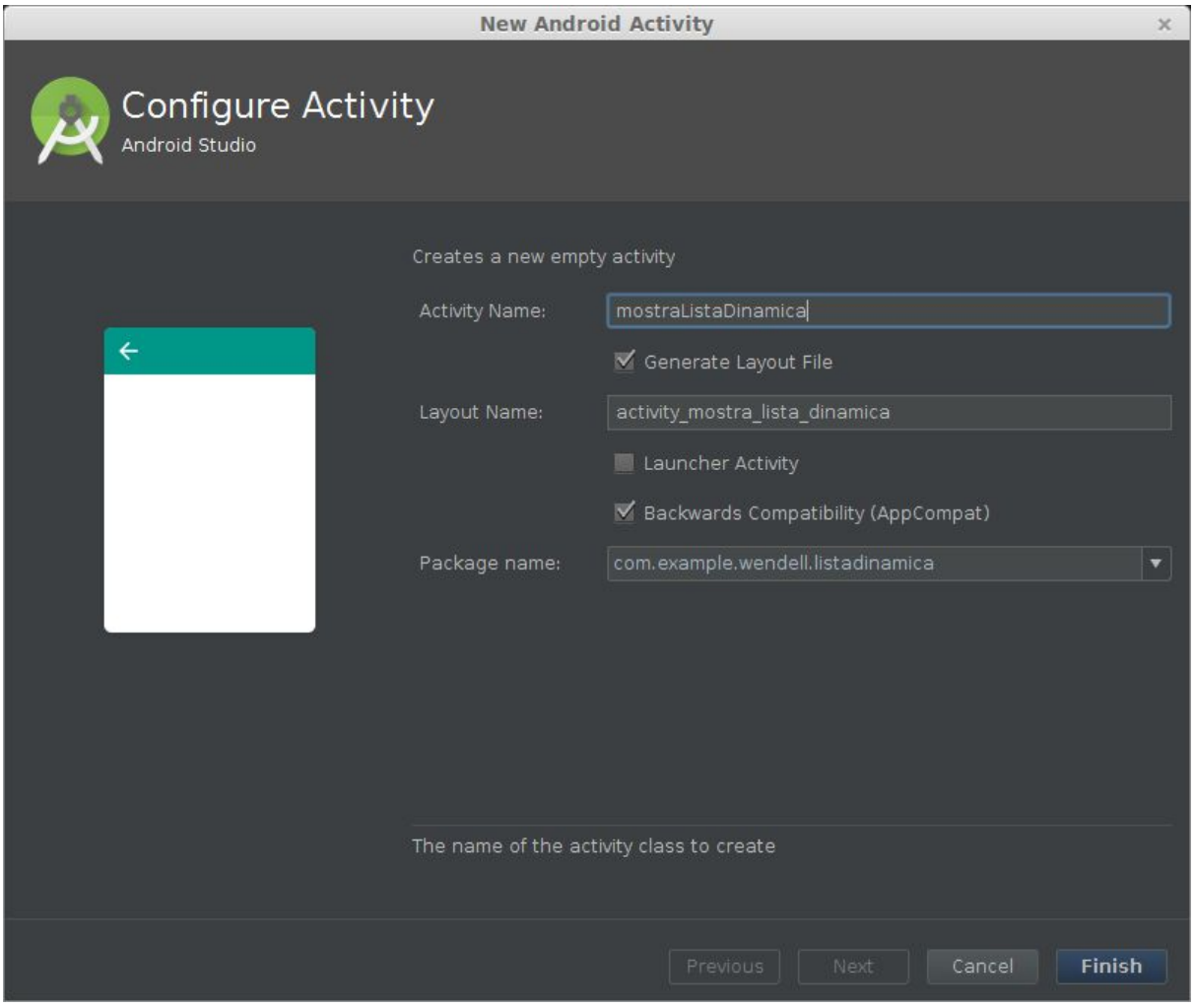
Vamos agora criar, na *MainActivity*, um *ArrayList* com vários objetos do tipo *Pessoa* e popular um *Bundle* para passá-los à atividade que exibirá a lista:

```
package com.example.wendell.listadinmica;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;

import com.example.wendell.listadinmica.model.Pessoa;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    ArrayList<Pessoa> lista = new ArrayList<>();

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

        lista.add(new Pessoa("Maria de Oliveira", "99323-1234",
            R.mipmap.ic_launcher_round));
        lista.add(new Pessoa("Pedro da Silva", "3690-1234",
            R.mipmap.ic_launcher_round));
        lista.add(new Pessoa("João de Souza", "3690-4321",
            R.mipmap.ic_launcher_round));
    }

    public void mostrarLista(View v) {
        Intent it = new Intent(this, mostraListaDinamica.class);
        Bundle bundle = new Bundle();
        bundle.putParcelableArrayList("contatos", lista);
        it.putExtras(bundle);
        startActivity(it);
    }
}
```

Agora, vamos criar a atividade da lista, que será invocada ao clicar no botão. Clique com o botão direito do mouse na pasta de sua aplicação, selecione *New -> Activity -> Empty Activity.*



Nomeie a nova atividade como *mostrarListaDinamica*. Observe se a opção *Generate Layout File* está marcada e clique em Finish.



Agora, vamos trabalhar na interface desta atividade, editando o arquivo activity_mostrar_lista_dinamica.xml, que foi gerado automaticamente. Para exibir a nossa lista dinâmica, precisamos acrescentar um objeto do tipo *ListView* à interface.

Agora, vamos criar a funcionalidade da lista editando o arquivo *mostrarListaDinamica.java*, para receber os dados que foram passados pela atividade principal, ao clicar no botão:

```java
package com.example.wendell.listadinmica;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.ListView;

import com.example.wendell.listadinmica.adapter.PessoaAdapter;
import com.example.wendell.listadinmica.model.Pessoa;

import java.util.ArrayList;

public class mostraListaDinamica extends AppCompatActivity {

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

        ListView listview = findViewById(R.id.listView);

        Intent intent = getIntent();
        Bundle bundle = intent.getExtras();

        try {
            ArrayList<Pessoa> lista = bundle.getParcelableArrayList(
                    "contatos");
            PessoaAdapter adapter = new PessoaAdapter(
                    mostraListaDinamica.this, lista);
            listview.setAdapter(adapter);
        }
        catch (Exception e){
            Log.d(e.getClass().toString(), e.getMessage());
        }

    }
}
```

Pronto! Veja se o app rodará normalmente e corrija-o, se necessário.

Agora é sua vez, futuro desenvolvedor. Você aprendeu a exibir uma lista montada dinamicamente, mas usando dados que colocamos de forma estática no código. Vamos consertar isso… Modifique a atividade principal deste exercício para exibir, além do botão “Mostrar lista”, duas caixas de texto para o usuário inserir o nome e o telefone do contato e um botão para salvar o texto digitado. Quando o usuário pedir para exibir a lista, a versão atualizada da mesma deve ser mostrada.