INSTITUTO FED. DE EDUCAÇÃO, CIÊNC. E TEC. DE PERNAMBUCO
CURSO: TEC. EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS
PROFESSOR: RAMIDE DANTAS
ASSUNTO: LISTVIEW

Prática 03

Parte 1: Criando a atividade principal

Passo 1: Crie uma novo projeto contendo uma atividade em branco:

Application Name:	Pratica03
Company Domain:	pdm.tads.ifpe.edu.br
Template:	Empty Activity
Activity:	MainActivity

Passo 2: No arquivo de layout da atividade principal (**activity_main.xml**), adicione um objeto `ListView` (substituindo `TextView`) com o código abaixo:

```
    <ListView
        android:id="@+id/list_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
```

Passo 3: Adicione a propriedade “cities” a `MainActivity` e ajuste o código do método `onCreate()` como mostrado abaixo:

```
public class MainActivity extends AppCompatActivity {

    private static final String [] cities = {"Recife", "João Pessoa", "Natal",
            "Fortaleza", "Rio de Janeiro", "São Paulo",  "Salvador", "Vitória",
            "Florianópolis", "Porto Alegre", "São Luiz","Teresina",
            "Belém", "Manaus"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        ...

        ListView listView = (ListView)findViewById(R.id.list_view);
        listView.setAdapter(new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_1, cities)
        );
    }

    ...
}
```

No código acima é usado um `ArrayAdapter<String>`, cuja função é fornecer os itens que serão apresentados ao usuário para o `ListView`. Nesse caso é usado um adaptador simples que trabalha com arrays de strings.

Passo 4: Rode e teste a aplicação.

Parte 2: Criando um layout para os items da lista.

Passo 1: Crie um novo arquivo de layout (*res >layout >new*) com nome **city_listitem.xml**, contendo o código abaixo:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/city_name"
        android:textSize="14pt"
        android:textColor="@android:color/black"
        android:layout_width="wrap_content"
        android:layout_height="match_parent" />

    <TextView
        android:id="@+id/city_info"
        android:layout_weight="1"
        android:layout_width="wrap_content"
        android:layout_height="match_parent" />

</LinearLayout>
```

Passo 2: Modifique o método `onCreate()` da MainActivity para ter usar o novo layout:

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...

        ListView listView = (ListView)findViewById(R.id.list_view);
        listView.setAdapter(new ArrayAdapter<String>(this,
                        R.layout.city_listitem,
                        R.id.city_name, cities
                )
        );
    }
```

No construtor do Adapter é passado o layout e o id do view dentro do layout que será usado para exibir as strings.

Passo 3: Ainda no `onCreate()`, adicione ao final um tratador para o evento de toque em um item da lista:

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        ...

        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, final View view,
                                    int position, long id) {
                Toast.makeText(parent.getContext(), "Cidade selecionada: " +
                        cities[position], Toast.LENGTH_SHORT).show();
            }
        });
    }
```

Passo 4: Rode a aplicação e teste a lista.

Parte 3: Melhorando a apresentação da lista

Passo 1: Crie uma classe `City`, com as propriedades abaixo:

```
public class City {
    private String name;
    private String info;

    public City(String name, String info) {
        this.name = name;
        this.info = info;
    }

    public String getName() {
        return this.name;
    }

    public String getInfo() {
        return this.info;
    }
}
```

Passo 2: Crie um novo Adapter a ser usado com o `ListView`, para exibir as cidades usando o layout criado anteriormente:

```
public class CityArrayListAdapter extends ArrayAdapter<City> {

    private City [] cities;

    public CityArrayListAdapter(Context context, int resource, City [] cities) {
        super(context, resource, cities);
        this.cities = cities;
    }

    @Override
    public View getView(int position, View view, ViewGroup parent) {

        LayoutInflater inflater = LayoutInflater.from(getContext());
        View listItem = inflater.inflate(R.layout.city_listitem, null, true);

        TextView cityName = (TextView) listItem.findViewById(R.id.city_name);
        TextView cityInfo = (TextView) listItem.findViewById(R.id.city_info);

        cityName.setText(cities[position].getName());
        cityInfo.setText(cities[position].getInfo());

        return listItem;
    }
}
```

Essa classe é responsável por criar os Views que farão parte da lista que será exibida ao usuário. Esses views podem conter elementos complexos, como imagens. No método `getView()`, os dados dos objetos City são usados para preencher os elemetnos visuais do item da lista.

Passo 3: Modifique a classe MainActivity, criando novos objetos City com as informações adicionais. Adapte o método `onCreate()` para usar o nosso novo Adapter. Modifique também o trecho que trata o toque no item para exibir o nome da cidade no Toast.

```
public class MainActivity extends AppCompatActivity {

    public static final City [] cities = {
            new City("Recife", "Capital de Pernambuco"),
            new City("João Pessoa", "Capital da Paraíba"),
            new City("Natal", "Capital do Rio Grande do Norte"),
            new City("Fortaleza", "Capital do Ceará"),
            new City("Rio de Janeiro", "Capital do Rio de Janeiro"),
            new City("São Paulo","Capital de São Paulo"),
            new City("Salvador","Capital da Bahia"),
            new City("Vitória","Capital do Espirito Santo"),
            new City("Florianópolis","Capital de Santa Catarina"),
            new City("Porto Alegre","Capital do Rio Grande do Sul"),
            new City("São Luiz","Capital do Maranhão"),
            new City("Teresina","Capital do Piauí"),
            new City("Belém","Capital do Pará"),
            new City("Manaus","Capital do Amazonas")};

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        ...

        ListView listView = (ListView)findViewById(R.id.list_view);
        listView.setAdapter(new CityArrayListAdapter(this,
                        R.layout.city_listitem, cities
                )
        );


        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, final View view,
                                    int position, long id) {
                Toast.makeText(parent.getContext(), "Cidade selecionada: " +
                        cities[position].getName(), Toast.LENGTH_SHORT).show();
            }
        });
    }

    ...
}
```

Passo 4: Rode e teste a aplicação.

Parte 4: Melhorando o desempenho do ListView

Passo 1: Na classe `CityArrayListAdapter`, modifique o método `getView()` para refletir o código abaixo.

```
@Override
 public View getView(int position, View view, ViewGroup parent) {
     View listItem = null;

     if (view == null) {
         LayoutInflater inflater = LayoutInflater.from(getContext());
         listItem = inflater.inflate(R.layout.city_listitem, null, true);
     } else {
         listItem = view;
     }

     TextView cityName = (TextView) listItem.findViewById(R.id.city_name);
     TextView cityInfo = (TextView) listItem.findViewById(R.id.city_info);

     cityName.setText(cities[position].getName());
     cityInfo.setText(cities[position].getInfo());

     return listItem;
 }
```

O parâmetro do tipo View que é passado a esse método contém um item da lista que não está mais visível e pode então ser “reciclado” (convertido) em um novo item a ser exibido. Isso economiza processamento e memória no caso de listas muito grandes, tornando também a rolagem da lista mais suave. (Como a lista é pequena, não deve haver diferença visual perceptível). Durante a criação da lista o parâmetro view é null, portanto os items devem ser criados do zero.

Passo 2: Rode e teste a aplicação.

Passo 3: Crie uma classe estática ViewHolder dentro de CityArrayListAdapter com o código abaixo.

```
static class ViewHolder {
    TextView cityName;
    TextView cityInfo;
}
```

Passo 4: Modifique novamente o método `getView()` para refletir o código abaixo.

```
@Override
public View getView(int position, View view, ViewGroup parent) {
    View listItem = null;
    ViewHolder holder = null;

    if (view == null) {
        LayoutInflater inflater = LayoutInflater.from(getContext());
        listItem = inflater.inflate(R.layout.city_listitem, null, true);

        holder = new ViewHolder();
        holder.cityName =  (TextView) listItem.findViewById(R.id.city_name);
        holder.cityInfo = (TextView) listItem.findViewById(R.id.city_info);

        listItem.setTag(holder);
    } else {
        listItem = view;
        holder = (ViewHolder)view.getTag();
    }

    holder.cityName.setText(cities[position].getName());
    holder.cityInfo.setText(cities[position].getInfo());

    return listItem;
}
```

Nessa modificação é usado um objeto do tipo ViewHolder que serve como atalho para os componentes internos do nosso item (no caso, os dois TextViews). O objetivo desse recurso é evitar usar o método `findViewById()` sempre que o view for reciclado, melhorando o desempenho da aplicação. Cada item da lista tem seu objeto “holder”, o qual é associado ao view pelo método `setTag()`, que permite associar objetos arbitrários a view. (Essas duas técnicas combinadas (reciclagem e ViewHolder) pode apresentar ganhos de desempenho na ordem de 300% segundo o Google.)

Passo 5: Rode e teste a aplicação.

Parte 5 (Desafio): Usando RecyclerView

Passo 1: Crie um novo projeto com API level 22 (Lollipop 5.1).

Passo 2: Adicione no arquivo build.gradle do módulo app, na parte dependencies:

```
compile 'com.android.support:recyclerview-v7:25.0.+'
```

Passo 3: Insira o RecyclerView no **activity_main.xml**.

Siga as intruções do passo 2 da parte 1, substituindo `<ListView .../>` por `<android.support.v7.widget.RecyclerView .../>`

Passo 4: Crie a classe `City`, como no passo 1 da parte 3.

Passo 5: Adicione ao `onCreate()` do MainActivity.java:

```
RecyclerView recyclerView = (RecyclerView)findViewById(R.id.list_view);
recyclerView.setLayoutManager(new LinearLayoutManager(this));
recyclerView.setAdapter(new CityArrayAdapter(cities));
```

Nesse passo pegamos o `RecyclerView` e associamos a ele um `LinearLayoutManager` que oferece comportamento semelhante ao `ListView`. Também associamos ao Adapter que criaremos mais a frente.

Passo 6: Na classe MainActivity, adicione o array de cidades do passo 3 da parte 3.

Passo 7: Crie o layout dos items da lista (parte 2, passo 1).

Passo 8: Crie a classe `CityArrayAdapter`:

```
public class CityArrayAdapter extends RecyclerView.Adapter<CityArrayAdapter.CityHolder>
{
    private final City[] cities;

    public CityArrayAdapter(City[] cities) {
        this.cities = cities;
    }

    @Override
    public CityHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View inflatedView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.city_listitem, parent, false);
        return new CityHolder(inflatedView);
    }

    @Override
    public void onBindViewHolder(CityHolder holder, int position) {
        holder.bindCity(cities[position]);
    }

    @Override
    public int getItemCount() {
        return cities.length;
    }

    ...
}
```

Essa classe faz o papel do `Adapter` do `ListView`, porém incorpora por default o padrão `ViewHolder`. O método `onBindViewHolder()` é usado sempre que um view for reciclado e tiver que exibir uma nova cidade, no nosso caso.

Passo 9: Crie a classe `CityHolder` dentro de `CityArrayAdapter`:

```
public static class CityHolder extends RecyclerView.ViewHolder
                                implements View.OnClickListener {

    private final TextView cityName;
    private final TextView cityInfo;

    private City city;

    public CityHolder(View itemView) {
        super(itemView);
        this.cityName =  (TextView) itemView.findViewById(R.id.city_name);
        this.cityInfo = (TextView) itemView.findViewById(R.id.city_info);
        itemView.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        Toast.makeText(v.getContext(), "Cidade selecionada: " +
                                        city.getName(), Toast.LENGTH_SHORT).show();
    }

    public void bindCity(City city) {
        this.city = city;
        cityName.setText(city.getName());
        cityInfo.setText(city.getInfo());
    }
}
```

Essa classe implementa o `ViewHolder` dos itens da lista. Ela também possui a função de tratar o click do item, porém não é obrigatório que isso seja feito nesse ponto. O método `bindCity()` é chamado para atualizar o item da lista com a nova cidade.