Universidade Federal de Ouro Preto
Departamento de Computação e Sistemas - DECSI

Computação Móvel

Interface Gráfica - *View* (Ref. Cap. 7)

Vicente Amorim
vicente.amorim.ufop@gmail.com
www.decom.ufop.br/vicente

Sumário

* Arquivos XML de *resources*

* *View*

* Componentes Gerais

Arquivos XML de *resources*

Arquivos XML de *resources*

- Introdução

✓ Como visto, a classe R é quem gerencia todos os recursos do sistema *Android*.

✓ Outros tipos de arquivos (além de *layout*) também podem ser caracterizados como de *resources*.

✓ Um exemplo: strings.xml.

✓ Arquivo presente em */res/values/strings.xml*, pode conter diversas mensagens de texto a serem utilizadas no sistema.


Arquivos XML de *resources*

- Introdução

```
<resources>
    <string name="msg_verde_e_branco">Texto verde e branco</string>
    <string name="msg_vermelho_e_branco">Texto vermelho e branco</string>
    <string name="texto1">Texto campo 1 azul</string>
</resources>
```

Arquivos XML de *resources*

- Introdução

✓ Outra utilidade dos arquivos de *resources* do Android pode ser armazenar diversos atributos de cores.

✓ Analogamente ao anterior, um arquivo em /res/values/ colors.xml pode ser definido.

✓ Em cada linha um nome para cada valor de cor.

✓ Ao contrário de se endereçar cores por valores numéricos, endereça-se por *strings*.

Arquivos XML de *resources*

- Introdução

```
<resources>
    <color name="vermelho">#ff0000</color>
    <color name="azul">#0000ff</color>
    <color name="verde">#00ff00</color>
    <color name="branco">#ffffff</color>
</resources>
```

Arquivos XML de *resources*

- Introdução

✓ Arquivos de *resources* podem ainda ser utilizados para criação de estilos.

✓ Com um arquivo de estilo é possível definir de uma só vez o padrão de cor, tipo de fonte, etc. de um componente ou conjunto de componentes.

✓ Arquivos de estilo podem ter qualquer nome, desde que sejam formados pelas *tags* <style name=”nomeDoEstilo”> e a *tag* <item>.


Arquivos XML de *resources*

- Introdução

```
<resources xmlns:android="http://schemas.android.com/apk/res/android">
    <style name="estiloExemplo">
        <item name="android:textSize">14sp</item>
        <item name="android:textColor">#ff0000</item>
        <item name="android:background">#00ff00</item>
        <item name="android:textStyle">italic</item>
    </style>
</resources>
```

Arquivos XML de *resources*

- ✓ Exemplo: AppNum39

```
<LinearLayout ... ...

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Texto azul e branco"
        android:background="#0000ff"
        android:textColor="#ffffff"
        android:textStyle="bold"/>

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/msg_verde_e_branco"
        android:background="@color/verde"
        android:textColor="@color/branco"
        android:textStyle="bold"/>

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/msg_vermelho_e_branco"
        style="@style/estiloExemplo"/>

</LinearLayout>
```

View

View

- Características

✓ A classe View é utilizada como base para qualquer componente gráfico.

✓ Toda subclasse de View precisa implementar o método onDraw(Canvas canvas) que é responsável por desenhar os elementos na tela.

✓ Diversas subclasses da classe View já são disponíveis por default no Android.

View

- Características

  ✓ Alguns dos principais métodos inerentes à classe View:

Método	Descrição
requestFocus()	Força a obtenção do foco por determinado componente.
setPadding(esquerda, acima, direita, abaixo)	Informa o espaço em pixels que deve ser inserido à esquerda, direita, acima e abaixo do componente antes de inserí-lo na tela. É utilizado quando se faz necessário ter um componente levemente separado do outro. Atributo correspondente em XML é: android:padding
setVisibility(v)	Informa a visibilidade do elemento. Pode receber três valores distintos: View.VISIBLE, View.INVISIBLE e View.GONE. A constante INVISIBLE somente não exibe o componente na tela - deixando seu espaço reservado. A constante GONE literalmente remove o objeto da tela.

View

- Características

  ✓ Alguns dos principais métodos inerentes à classe View:

Método	Descrição
requestLayout()	Solicita ao Android refazer o *layout* da tela.
invalidate()	Invalida a View. Isso faz com que o redesenho da mesma seja automaticamente requisitado.
onSizeChanged(int largura, int altura, int larguraAntiga, int alturaAntiga)	*Callback* chamado pelo Android sempre que um componente altera seu tamanho, informando os valores novos e antigos para largura e altura.
onDraw(Canvas)	Método responsável por desenhar o componente na tela. Pode ser implementado manualmente para controlar o que é desenhado na tela.

View

- Características

✓ Alguns dos principais métodos inerentes à classe View:

Método	Descrição
onKeyDown(codigoTecla, evento) / onKeyUp(codigoTecla, evento)	Chamados pelo Android quando o usuário pressiona uma tecla.

✓ Definição das dimensões:

✦ Valores de dimensões são frequentemente setados para *fill_parent* e *wrap_content*;

✦ Também é possível fazer uso de valores numéricos

View

- Características
  - ✓ Definição das dimensões:
    - ✦ Valores numéricos possibilitam controlar com mais precisão o tamanho do objeto.

Opção	Descrição
px(pixels)	Correspondente ao número de pixels da tela.
in(polegadas)	Valor em polegadas baseando-se no tamanho físico da tela.
mm (milímetros)	Valor em milímetros baseando-se no tamanho físico da tela.
pt (pontos)	1/72 de uma polegada. Também leva em conta o tamanho físico da tela.
dp	(*Density-independent pixels*). Unidade relativa à resolução da tela. Compilador também aceita a abreviação "dip".
sp	(*Scale-independent pixels*). Idem ao dp, mas também considera o tamanho da fonte que o usuário está utilizando. É recomendado que se faça uso dessa unidade quando especificando o tamanho de uma fonte.

View

- Características

✓ TextView

✦ Mais simples das subclasses de View (android.widget.TextView) e representa um texto na tela.

✓ EditText

✦ Subclasse de View utilizada para que o usuário possa entrar com dados no sistema.

✦ Pode receber texto normal, apenas números ou valores de senha.

View

- Características

  ✓ AutoCompleteTextView

    ✦ Recurso de "*auto complete*" é muito utilizado em vários tipos de aplicações hoje em dia.

    ✦ Mesmo recurso pode ser utilizado no Android através da classe android.widget.AutoCompleteTextView.

    ✦ Para fazer uso de tal componente é necessário chamar o setAdapter(...) passando como argumento uma implementação de ListAdapter.

- Características

  ✓ AutoCompleteTextView

    ✦ Dois parâmetros principais durante a criação da classe:

      1. android:completionThreshold: Recebe o número de letras que o usuário precisa digitar para iniciar o auto-preenchimento do texto.

      2. android:completionHint: Recebe um texto utilizado para exibir uma dica sobre o preenchimento do texto. Tal dica será exibida para o usuário final da aplicação.

View

- Características

✓ AutoCompleteTextView

✦ Exemplo AppNum40:

```
<LinearLayout ...>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Estados" />

    <AutoCompleteTextView android:id="@+id/estados"
        android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:completionThreshold="1"
        android:completionHint="Digite o nome de um estado:" />

    <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="OK"/>

</LinearLayout>
```


View

- Características

✓ AutoCompleteTextView

✦ Exemplo AppNum40:

```
public class MainActivity extends Activity {

    private static final String[] ESTADOS = new String[] {"Acre", "Alagoas", "Amapa",
"Amazonas", "Bahia", "Ceara", "Espírito Santo", "Minas Gerais", "Sao Paulo", "Rio de
Janeiro", "Rio Grande do Sul", "Pernambuco"
    };

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

- Características
  - ✓ AutoCompleteTextView
    - ✦ Exemplo AppNum40:

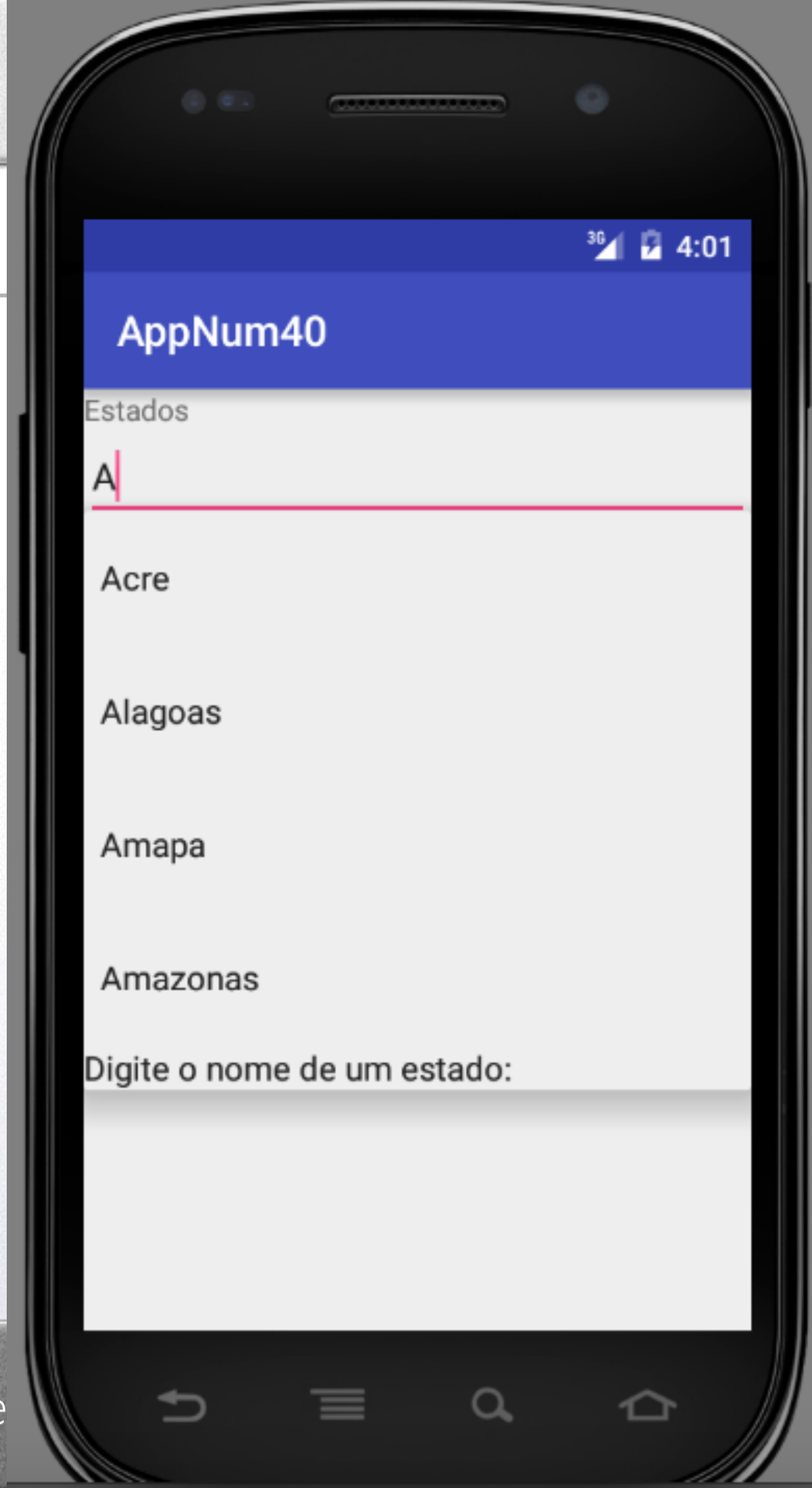
```
private void initEstados() {
        ArrayAdapter<String> adaptador = new
ArrayAdapter<String>(this,android.R.layout.simple_dropdown_item_1line, ESTADOS);
        AutoCompleteTextView estados =
(AutoCompleteTextView)findViewById(R.id.estados);
        estados.setAdapter(adaptador);
    }
```

View

- Características

✓ AutoCompleteTextView

✦ Exemplo AppNum40:



- Características

  ✓ Button e ImageButton

    ✦ Até agora fizemos uso da classe Button para representar um botão na tela.

    ✦ Entretanto, a classe ImageButton (android.widget.ImageButton) permite utilizar uma imagem para desenhar o botão.

View

- Características
  - ✓ Button e ImageButton
    - ✦ Exemplo AppNum41:

```
<LinearLayout ...>
    <ImageButton
        android:id="@+id/img1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/smile1" />

     <ImageButton
        android:id="@+id/img2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

View

- Características

  ✓ Button e ImageButton

    ✦ Exemplo AppNum41:

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ImageButton botaoImagem1 = (ImageButton)findViewById(R.id.img1);
        botaoImagem1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                Toast.makeText(MainActivity.this, "Imagem 1 OK", Toast.LENGTH_SHORT).show();

            }
        });
```

- Características
  - ✓ Button e ImageButton
    - ✦ Exemplo AppNum41:

```
ImageButton botaoImagem2 = (ImageButton)findViewById(R.id.img2);
        botaoImagem2.setImageResource(R.drawable.smile2);
        botaoImagem2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                Toast.makeText(MainActivity.this, "Imagem 2 OK", Toast.LENGTH_SHORT).show();
            }
        });
```

View

- Características
  - ✓ Button e ImageButton
    - ✦ Exemplo AppNum41:

- Características

  ✓ CheckBox e ToggleButton

    ✦ Um CheckBox é um elemento que armazena um valor booleano (*true*/*false*, ligado/desligado).

    ✦ CheckBox pode ser criado no Android através da classe android.widget.CheckBox.

    ✦ Outra classe que pode ser utilizada para selecionar uma opção é a ToggleButton (android.widget.ToggleButton).

    ✦ Ambos os componentes possuem o método isChecked() para verificação do estado corrente.

View

- Características
  - ✓ CheckBox e ToggleButton
    - ✦ Exemplo AppNum42:

```
<LinearLayout ... >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Exemplo de CheckBox e ToggleButton" />

    <CheckBox android:id="@+id/check1"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="Check 1" />

    <CheckBox android:id="@+id/check2"
        style="?android:attr/starStyle"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="Check 2" />
```

- Características
  - ✓ CheckBox e ToggleButton
    - ✦ Exemplo AppNum42:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ToggleButton exibe os textos Ligado ou Desligado..." />

<ToggleButton android:id="@+id/toggle"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:textOn="Ligado"
    android:textOff="Desligado" />

<Button android:id="@+id/btOK"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:text="OK"/>
</LinearLayout>
```


View

- Características
  - ✓ CheckBox e ToggleButton
    - ✦ Exemplo AppNum42:

```
public class MainActivity extends Activity {

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

        final ToggleButton toggle = (ToggleButton)findViewById(R.id.toggle);
        Button b = (Button)findViewById(R.id.btOK);
        b.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                boolean selecionado = toggle.isChecked();
                Toast.makeText(MainActivity.this, "Selecionado: "+selecionado, Toast.LENGTH_SHORT).show();
            }
        });
    }
```

View

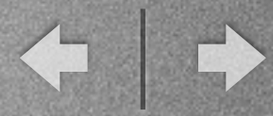
- Características
  - ✓ CheckBox e ToggleButton
    - ✦ Exemplo AppNum42:

- Exercícios

13) Início de ano, hora das compras de materiais escolares. Para facilitar a vida destes consumidores você resolveu criar uma aplicação para o cálculo do valor de todos os itens do material. Sua aplicação possuirá duas atividades:

a) Primeira atividade servirá de cadastro para os itens e valores que irão constar na lista de material escolar. Os possíveis são: lápis, caneta, lápis de cor, canetinha, folhas, tablet 7", notebook, giz de cera.

b) Os itens/valores cadastrados na atividade anterior serão exibidos na nova tela como uma lista de checkboxes/togglebuttons. Ao pressionar o botão finalizar o valor total deverá ser exibido na tela.

View

- Características

  ✓ RadioButton

    ✦ Comum em ambientes *web* e *desktop*.

    ✦ Permite que o usuário selecione apenas uma das opções apresentadas por uma lista.

    ✦ No Android, suportado através das classes android.widget.RadioGroup e android.widget.RadioButton.

    ✦ A classe RadioGroup define o grupo que contém a lista de opções, onde cada opção é apresentada por um RadioButton.


View

- Características
  - ✓ RadioButton
    - ✦ Exemplo AppNum43:

```
<LinearLayout ... >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nome: " />

    <EditText android:id="@+id/textNome"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Concorda?" />
```

- Características

✓ RadioButton

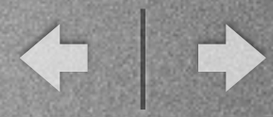
✦ Exemplo AppNum43:

```
<RadioGroup android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:orientation="horizontal" android:id="@+id/group1">
        <RadioButton android:id="@+id/radioSim"
            android:layout_width="wrap_content" android:layout_height="wrap_content"
            android:text="Sim" android:checked="false"/>

        <RadioButton android:id="@+id/radioNao"
            android:layout_width="wrap_content" android:layout_height="wrap_content"
            android:text="Nao" android:checked="false"/>
    </RadioGroup>
    <CheckBox android:id="@+id/checkReceberEmail"
            android:layout_width="wrap_content" android:layout_height="wrap_content"
            android:text="Receber e-mail?" />

    <Button android:id="@+id/buttonEnviar"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="Enviar" />
</LinearLayout>
```

View

- Características
  - ✓ RadioButton
    - ✦ Exemplo AppNum43:

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final EditText textNome = (EditText)findViewById(R.id.textNome);
        final RadioGroup group = (RadioGroup)findViewById(R.id.group1);
        group.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup arg0, int arg1) {
                boolean sim = R.id.radioSim == arg1;
                boolean nao = R.id.radioNao == arg1;
```

View

- Características
  - ✓ RadioButton
    - ✦ Exemplo AppNum43:

```
        if (sim) {
                    Toast.makeText(MainActivity.this, "SIM Selecionado",
Toast.LENGTH_SHORT).show();
                }
                else {
                    Toast.makeText(MainActivity.this, "NAO Selecionado",
Toast.LENGTH_SHORT).show();
                }
            }
        });

        final CheckBox check = (CheckBox)findViewById(R.id.checkReceberEmail);
        check.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton arg0, boolean arg1) {
                Toast.makeText(MainActivity.this, "Check: "+arg1, Toast.LENGTH_SHORT).show();
            }
        });
```

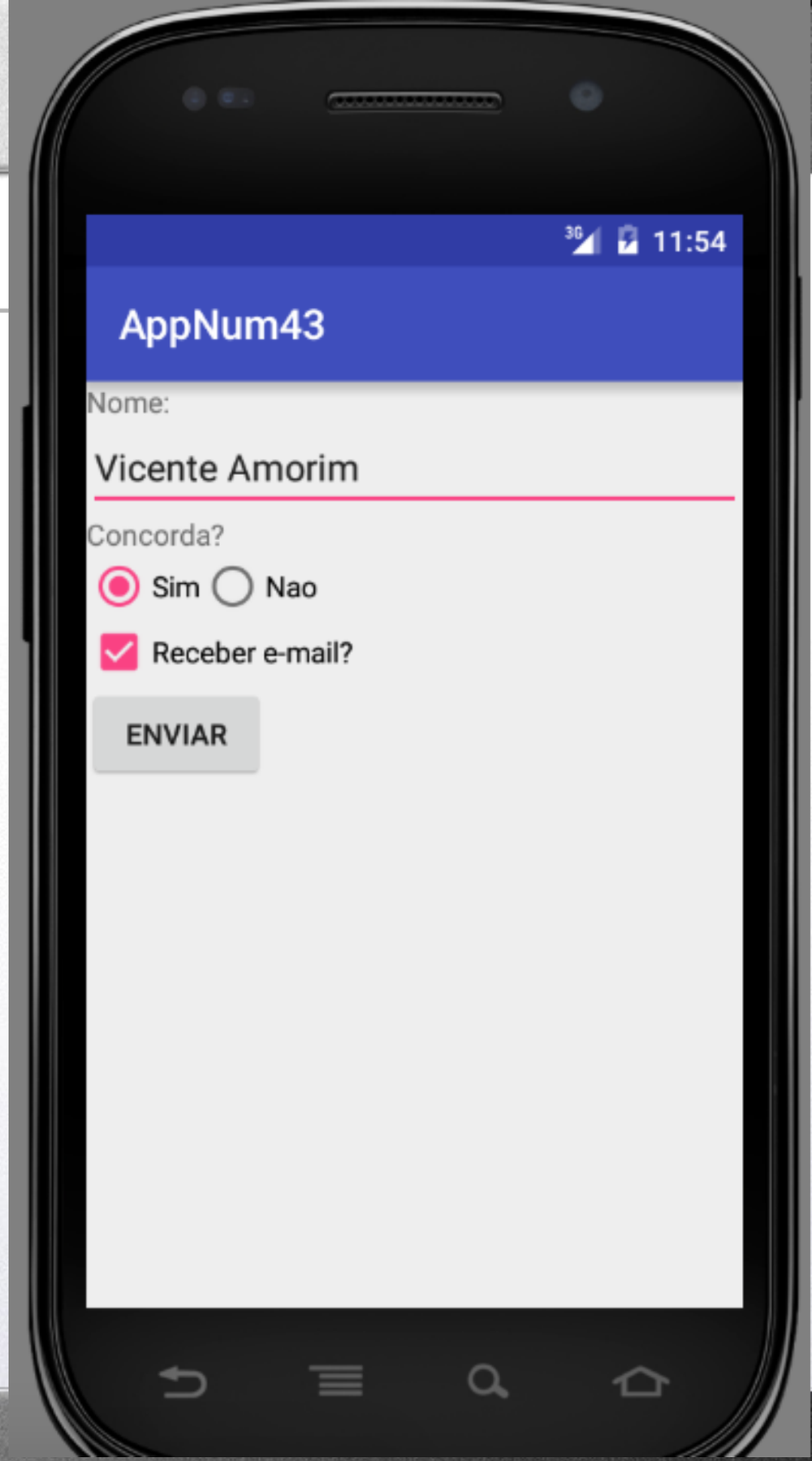

View

- Características
  - ✓ RadioButton
    - ✦ Exemplo AppNum43:

```
Button b = (Button)findViewById(R.id.buttonEnviar);
        b.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View arg0) {
                Log.i("appNum38","CLICOU OK!");
                //Compara id do radioSim
                boolean concorda = R.id.radioSim == group.getCheckedRadioButtonId();
                boolean receberEmail = check.isChecked();
                StringBuffer sb = new StringBuffer();
                sb.append("Nome: ").append(textNome.getText())
                    .append("\nReceber Email: ").append(receberEmail ? "Sim" : "Não")
                    .append("\nConcorda: ").append(concorda ? "Sim" : "Não");
                Toast.makeText(MainActivity.this, sb.toString(),
Toast.LENGTH_SHORT).show();
            }
        });
```

View

- Características
  - ✓ RadioButton



- Características

  ✓ Spinner

    ✦ Utilizado para criar um combo com opções na tela.

    ✦ Equivalente à tag <select> do html.

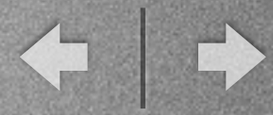
    ✦ No Android, suportado através das classes android.widget.Spinner e android.widget.SpinnerAdapter.

    ✦ Lista de elementos a ser exibida é definida através de um *adapter* (*SpinnerAdapter*).


View

- Características
  - ✓ Spinner
    - ✦ Exemplo AppNum44:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width=“match_parent" android:layout_height="wrap_content"
        android:text="Selecione uma opção: " />

    <Spinner android:id="@+id/comboPlanetas"
        android:layout_width=“match_parent" android:layout_height="wrap_content"
        android:drawSelectorOnTop="true" android:prompt="@string/texto_combo" />

    <TextView android:id="@+id/selectedPlanet"
        android:layout_width=“match_parent" android:layout_height="wrap_content" />
</LinearLayout>
```

View

- Características
  - ✓ Spinner
    - ✦ Exemplo AppNum44:

```
public class MainActivity extends AppCompatActivity {
    private String[] planetas = {"Mercurio", "Venus", "Terra", "Marte", "Jupiter", "Saturno",
"Netuno", "Plutao"};
    private int[] imagens = {R.drawable.mercurio, R.drawable.venus, R.drawable.terra,
R.drawable.marte,
    R.drawable.jupiter, R.drawable.saturno, R.drawable.netuno, R.drawable.plutao};

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

        final ImageView imagem = (ImageView) findViewById(R.id.img);
        final Spinner combo = (Spinner) findViewById(R.id.comboPlanetas);
```

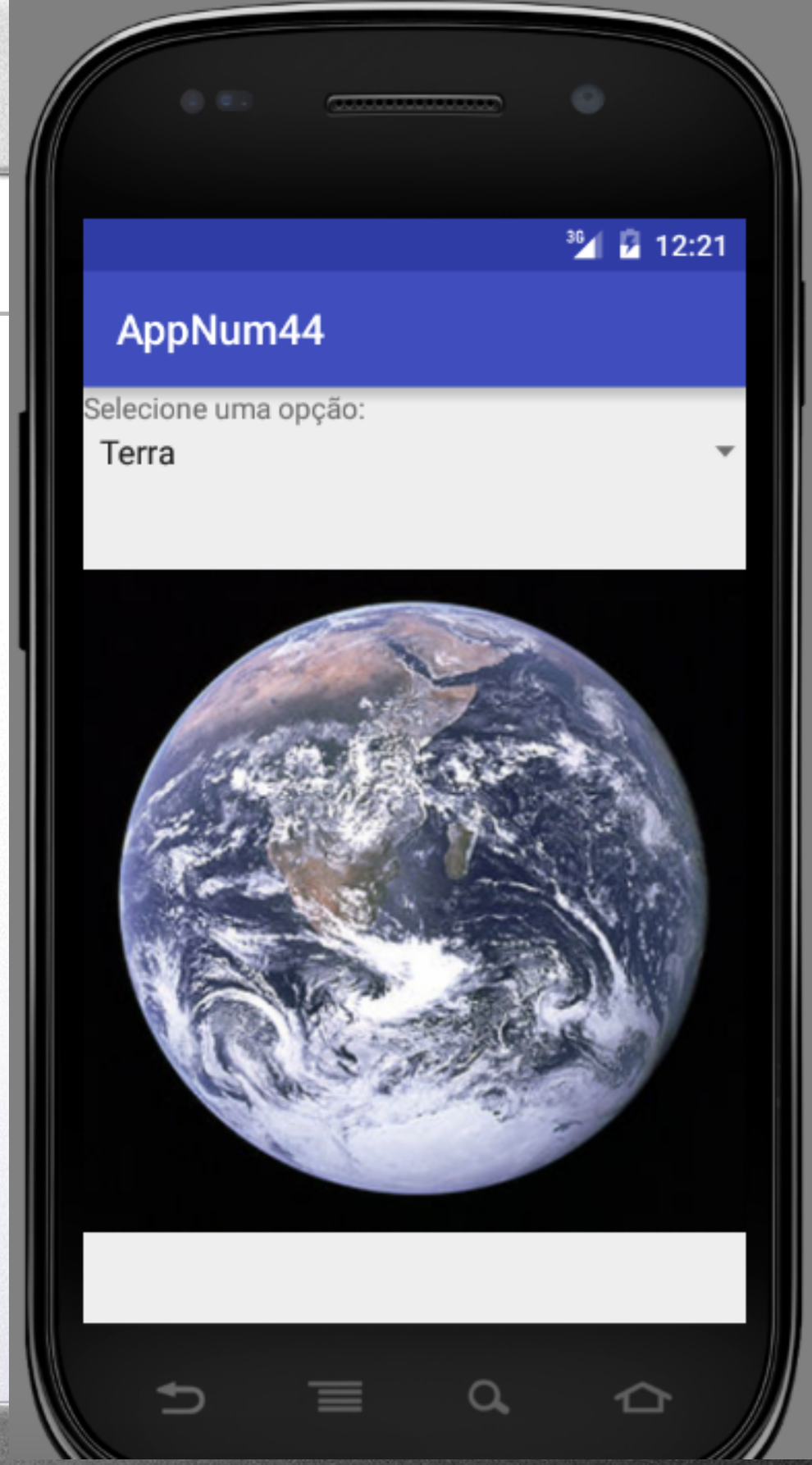

View

- Características
  - ✓ Spinner
    - ✦ Exemplo AppNum44:

```
        ArrayAdapter<String> adaptador = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, planetas);
        adaptador.setDropDownViewResource(android.R.layout.simple_spinner_item);
        combo.setAdapter(adaptador);

        combo.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
                imagem.setImageResource(imagens[arg2]);
            }

            @Override
            public void onNothingSelected(AdapterView<?> arg0) {

            }
        });
```

View

- Características
  - ✓ Spinner



- Características

  ✓ Spinner

    ✦ Para se obter o item selecionado do Spinner, basta utilizar um dos seguintes métodos:

Método	Descrição
Object getSelectedItem()	Retorna o item selecionado.
long getSelectedItem()	Retorna o id do objeto selecionado.
int getSelectedItemPosition().	Retorna a posição do item selecionado. Equivalente ao array fornecido ao Spinner.


View

- Características

  ✓ Spinner

    ✦ Ao se criar o ArrayAdapter foi utilizado um recurso nativo android.R.layout.simple_spinner_item.

✦ Entretanto, é possível trocá-lo para outro padrão. Exemplo:

```
adaptador.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

✦ Um *layout* totalmente customizado pode ainda ser criado baseando-se no nativo fornecido pela plataforma.

View

- Características
  - ✓ Spinner

- Características

  ✓ ProgressDialog

    ✦ Normalmente, em aplicações com alguma tarefa/processamento mais demorado, deve ser provido ao usuário algum *feedback* do que está ocorrendo.

    ✦ No Android já existe uma classe especial para *feedback* de ações aos usuários.

    ✦ A classe android.app.ProgressDialog fornece as primitivas para implementação de uma janela de espera.

- Características

  ✓ ProgressDialog

    ✦ A janela de espera é exibida pelo Android com a mensagem selecionada e uma animação.

    ✦ A janela ficará aberta por um tempo indeterminado até que o método *dismiss()* seja chamado para finalizá-la.

    ✦ Tal janela é extremamente útil principalmente em aplicações que executam requisições assíncronas à serviços externos a app.

View

- Características

✓ ProgressDialog

✦ Exemplo AppNum45:

```
<LinearLayout ... >

    <ImageView android:id="@+id/img"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
</LinearLayout>
```

✦ Permissão de acesso a internet no AndroidManifest.xml:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

View

- Características

  ✓ ProgressDialog

    ✦ Exemplo AppNum45:

```
public class MainActivity extends AppCompatActivity {

    private static final String URL_img = "http://3.bp.blogspot.com/-hHz3vGo9ZGM/TZ2qv-IYIDI/
AAAAAAAADqg/C9ZpsvXZdyU/s1600/apexwallpaper_android-wallpaper11-795482.png";
    private ProgressDialog dialog;

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

        //Abre janela com barra de progresso
        dialog = ProgressDialog.show(this, "Exemplo", "Buscando imagem, por favor aguarde...", false,
true);
        downloadImagem(URL_img);
    }
```

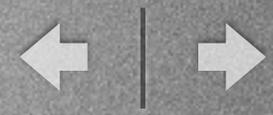
View

- Características

  ✓ ProgressDialog

    ✦ Exemplo AppNum45:

```
private void downloadImagem(final String urlImg) {
    new Thread() {
        @Override
        public void run() {
        try {
            //Download da imagem
            URL url = new URL(urlImg);
            InputStream is = url.openStream();
            //Converte o InputStream do Java para Bitmap
            final Bitmap image = BitmapFactory.decodeStream(is);
            is.close();
            //Atualiza a tela
            atualizaTela(image);
        }catch(MalformedURLException e) {
            Log.e("APPNUM45","Erro ao fazer download. URL inválida."+e.getMessage());
        } catch (IOException e) {
            e.printStackTrace();
        }
        }
    }.start();
}
```

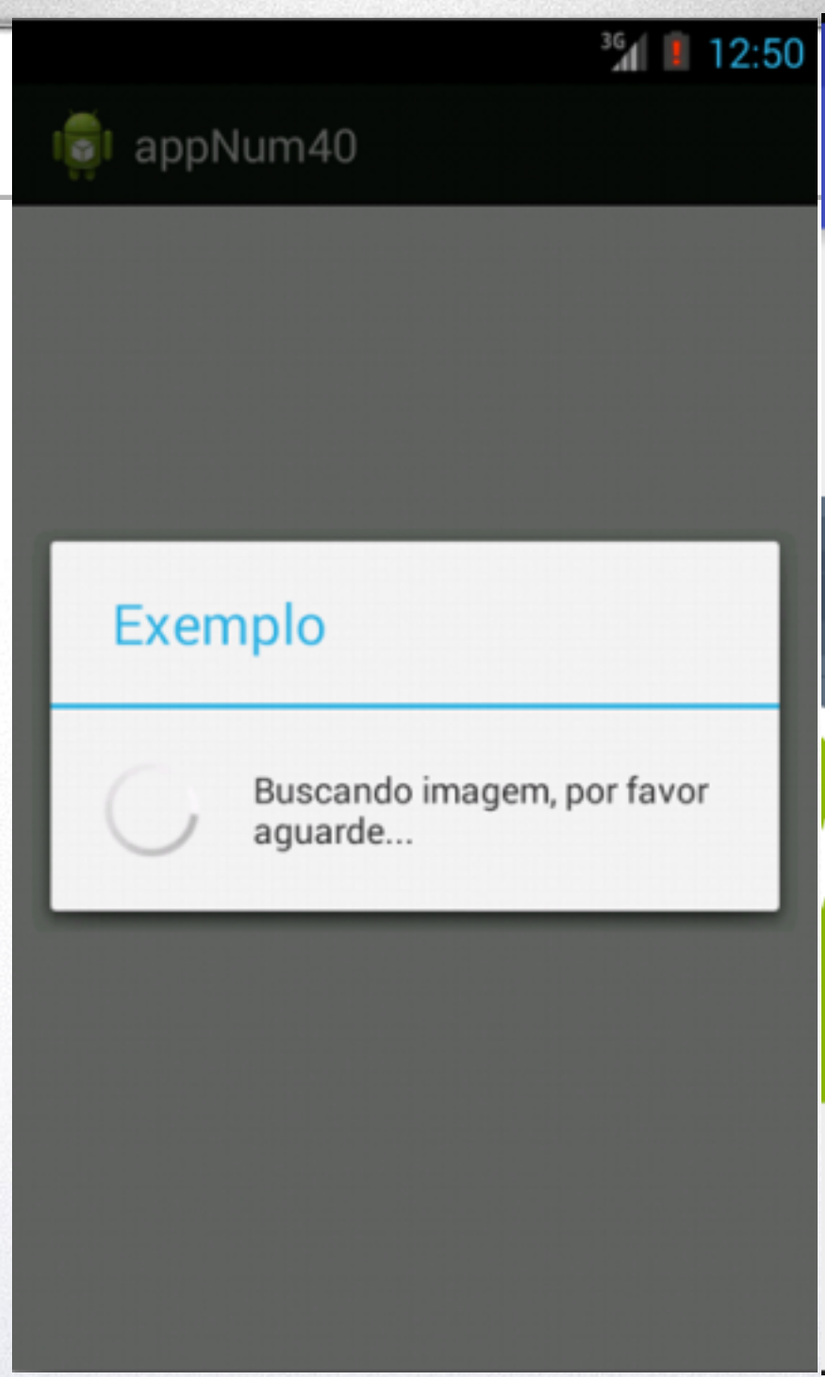
View

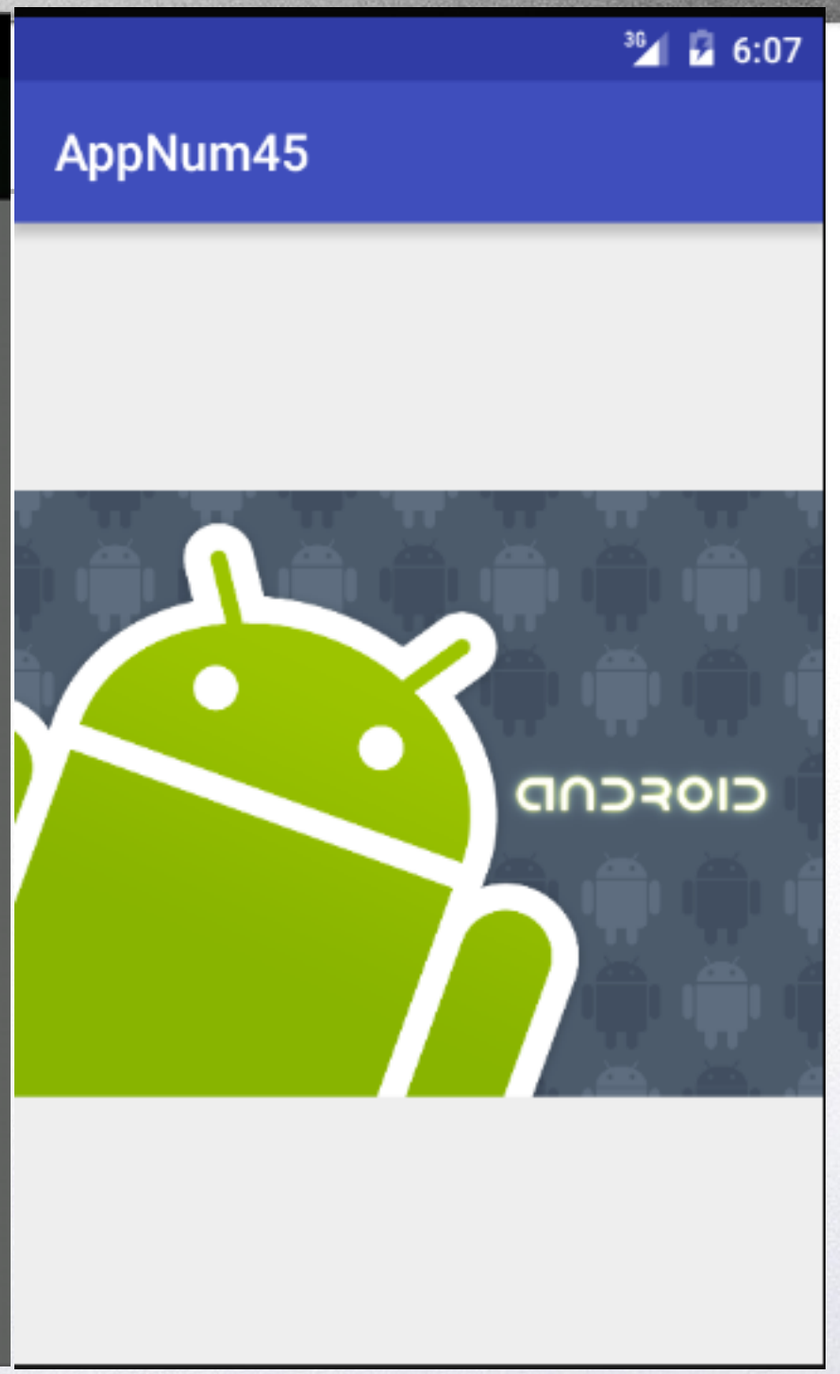
- Características
  - ✓ ProgressDialog
    - ✦ Exemplo AppNum45:

```
private void atualizaTela(final Bitmap imagem) {
    //Trocar por handler() se precisar fazer
    // alguma manipulação durante o download
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            //Fecha a janela de progresso
            dialog.dismiss();
            ImageView imgView = (ImageView)findViewById(R.id.img);
            imgView.setImageBitmap(imagem);
        }
    });
}
```

View

- Características
  - ✓ ProgressDialog
- ✦ Exemplo AppNum45:





- Exercício

14) Criar uma aplicação que contenha um Spinner para a seleção de imagens a serem baixadas pela internet.

- Cada item do spinner deve ser o nome de um clube de futebol diferente. Quando ocorrer a seleção pelo usuário, a app deve baixar o escudo e exibí-lo no campo apropriado.

Obs.: Utilize as imagens dos escudos disponíveis no site da disciplina.

View

- Características

  ✓ ProgressBar

    ✦ Utilizada para exibir uma barra de progresso na tela.

    ✦ Disponibilizada através da classe android.widget.ProgressBar.

    ✦ Pode ser indeterminada ou não. Resultado: Necessário ou não o incremento da barra de progresso.

    ✦ Permite fornecer ao usuário um *feedback* do quanto da tarefa está pronto.

    ✦ Para atualizar a barra de progresso, utiliza-se o método setProgress(valor).

- Características

  ✓ ProgressBar

    ✦ Exemplo AppNum46:

```
<LinearLayout ... >
    <TextView
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="Barra de Progresso" />

    <ProgressBar android:id="@+id/barraProgresso"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="200dip" android:layout_height="wrap_content"
        android:max="100"/>

    <Button android:id="@+id/btOK"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="Simular Tarefa"/>
</LinearLayout>
```

- Características

  ✓ ProgressBar

    ✦ Exemplo AppNum46:

```
public class MainActivity extends AppCompatActivity {

    private static final int MAXIMO = 100;
    private static final int THREAD_SLEEP = 200;
    private static final String CATEGORIA = "CEA436";
    private ProgressBar mProgress;
    private int progresso = 0;

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

        mProgress = (ProgressBar) findViewById(R.id.barraProgresso);
        Button b = (Button) findViewById(R.id.btOK);
        b.setOnClickListener(new Button.OnClickListener() {
```

View

- Características
  - ✓ ProgressBar
    - ✦ Exemplo AppNum46:

```
@Override
public void onClick(View arg0) {
    new Thread(new Runnable() {
        public void run() {
            while (progresso < MAXIMO) {
                simularTarefa();
                //Atualiza a barra de progresso
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Log.d(CATEGORIA,">> Progresso = "+progresso);
                        mProgress.setProgress(progresso);
                    }
                });
            }
            Log.i(CATEGORIA, "FIM!!");
        }
    }).start();
  }
});
}
```


View

- Características
  - ✓ ProgressBar
    - ✦ Exemplo AppNum46:

```
private void simularTarefa() {
    progresso++;
    try {
        Thread.sleep(THREAD_SLEEP);
    }catch(InterruptedException e) {
        e.printStackTrace();
    }
}
```

View

- Características
  - ✓ ProgressBar
    - ✦ Exemplo AppNum46:


View

- Características

  ✓ ProgressBar

    ✦ Uma *thread* é criada para simular o processamento (*sleep* 200ms).

    ✦ Valor máximo da barra é definido através do atributo android:max=”100”.

    ✦ ProgressDialog é uma janela, enquanto que ProgressBar é um *widget* da subclasse View.

View

- Características

  ✓ AlertDialog

    ✦ Algumas vezes aplicações precisam informar seu estado interno aos usuários.

    ✦ A comunicação de tais estados pode ser feita através de uma janela de diálogo.

    ✦ No Android, a classe android.app.AlertDialog é utilizada para o suporte a tal funcionalidade.

    ✦ Várias opções de customização são disponibilizadas pelo Android.


View

- Características

✓ AlertDialog

✦ Exemplo AppNum47:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button android:id="@+id/bt"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Teste Alerta!" />
</LinearLayout>
```

View

- Características

✓ AlertDialog

✦ Exemplo AppNum47:

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

Button bt = (Button)findViewById(R.id.bt);
bt.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        AlertDialog.Builder alerta = new AlertDialog.Builder(MainActivity.this);
        alerta.setIcon(R.drawable.smiley2);
        alerta.setTitle("Título 1");
        alerta.setMessage("Por favor, escolha sim ou não, obrigado.");
        alerta.setPositiveButton("Sim", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface arg0, int arg1) {
                Toast.makeText(MainActivity.this,  "Sim!",  Toast.LENGTH_SHORT).show();
            }
        });
```

View

- Características

✓ AlertDialog

✦ Exemplo AppNum47:

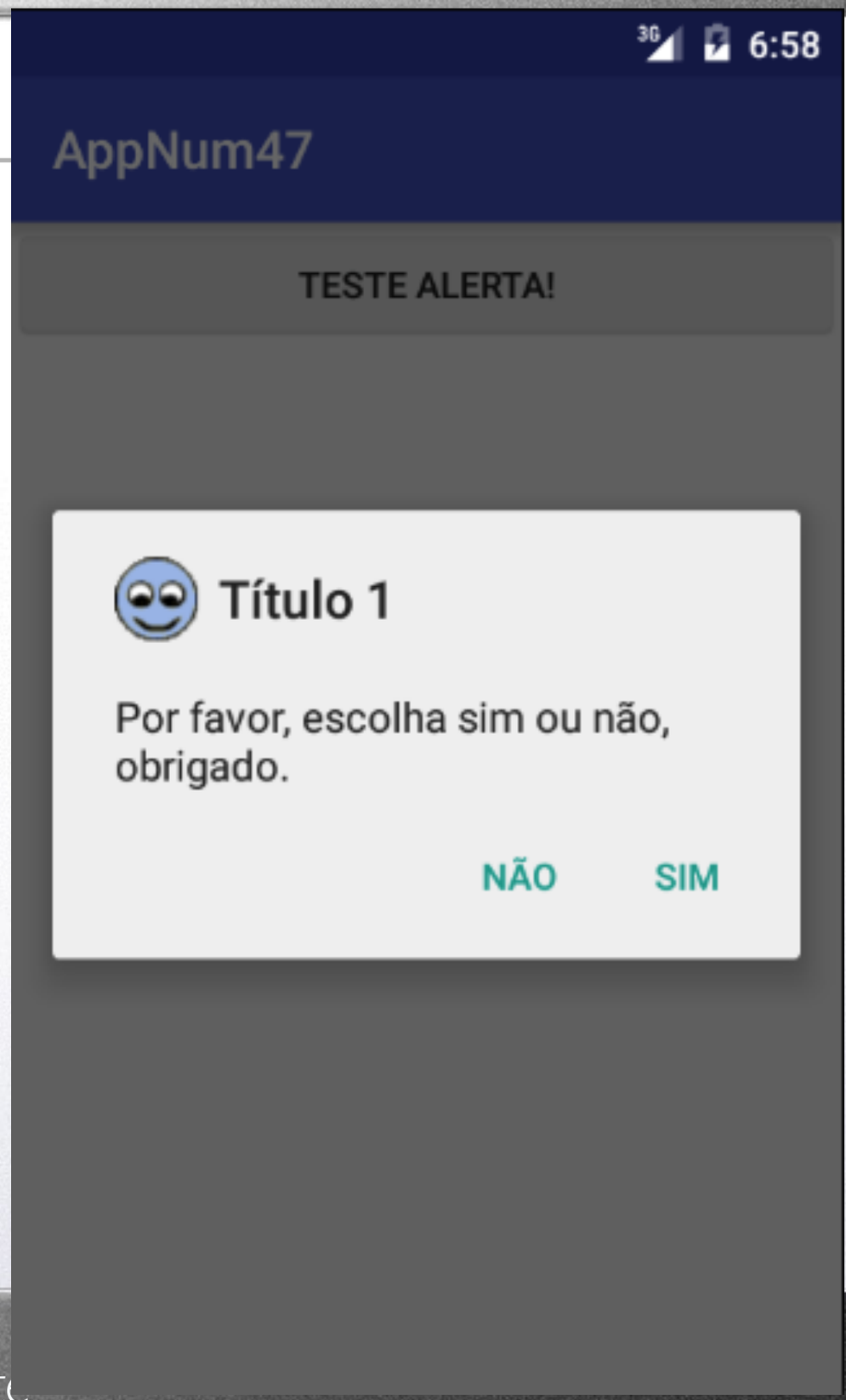
```
        alerta.setNegativeButton("Não", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface arg0, int arg1) {
                Toast.makeText(MainActivity.this, "Não!",
        Toast.LENGTH_SHORT).show();
            }
        });
        alerta.show();
    }
});
}
```

View

- Características

✓ AlertDialog

✦ Exemplo AppNum47:



View

- Características

  ✓ ListView

    ✦ ListActivity implementa internamente uma ListView.

    ✦ Em uma ListActivity, o método setContentView(view) é chamado implicitamente.

    ✦ Problema: ListView do ListActivity ocupa todo o *layout* da tela e não sobra espaço para outros componentes.

    ✦ Solução: Uso do ListView em uma *activity* normal.

View

- Características

✓ ListView

✦ Exemplo AppNum48: activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.cea436.appnum48.MainActivity">

    <ListView android:id="@+id/listview" android:layout_height="0dp"
    android:layout_width="match_parent"
    android:layout_weight="1"
    android:layout_margin="10dp"/>

    <View android:layout_width="match_parent" android:layout_height="20dp"
    android:background="@color/colorPrimary"/>

</LinearLayout>
```


View

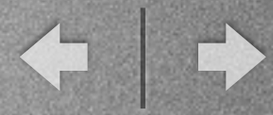
- Características
  - ✓ ListView
    - ✦ Exemplo AppNum48: simple_adapter.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView android:id="@+id/text" android:layout_width="wrap_content"
    android:layout_height="50dp" />

</LinearLayout>
```


View

- Características
  - ✓ ListView
    - ✦ Exemplo AppNum48: SimpleAdapter.java

```
public class SimpleAdapter extends BaseAdapter
{
    private String[] planetas = new String[]
{"Mercurio","Venus","Terra", "Marte",
"Jupiter","Saturno", "Urano",
"Netuno","Plutao"};

    private Context context;

    public SimpleAdapter(Context context) {
        super();
        this.context = context;
    }

    @Override
    public int getCount() {
        return planetas.length;
    }

    @Override
    public Object getItem(int i) {
        return planetas[i];
    }

    @Override
    public long getItemId(int i) {
        return i;
    }
```

View

- Características

  ✓ ListView

  ✦ Exemplo AppNum48:
  SimpleAdapter.java

```
@Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        String planeta = planetas[i];
        TextView t = new TextView(context);

        float dip = 50;
        //Densidade da tela
        float densidade =
context.getResources().getDisplayMetrics().density;
        int px = (int) (dip * densidade + 0.5f);
        t.setHeight(px);
        t.setText(planeta);
        return t;
    }
}
```

View

- Características
  - ✓ ListView
    - ✦ Exemplo AppNum48: MainActivity.java

```
public class MainActivity extends AppCompatActivity implements AdapterView.OnItemClickListener{

    private static final String TAG = "CEA436";
    private ListView listView;

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

        //ListView
        listView = (ListView) findViewById(R.id.listview);
        listView.setAdapter(new SimpleAdapter(this));
        listView.setOnItemClickListener(this);
    }
```

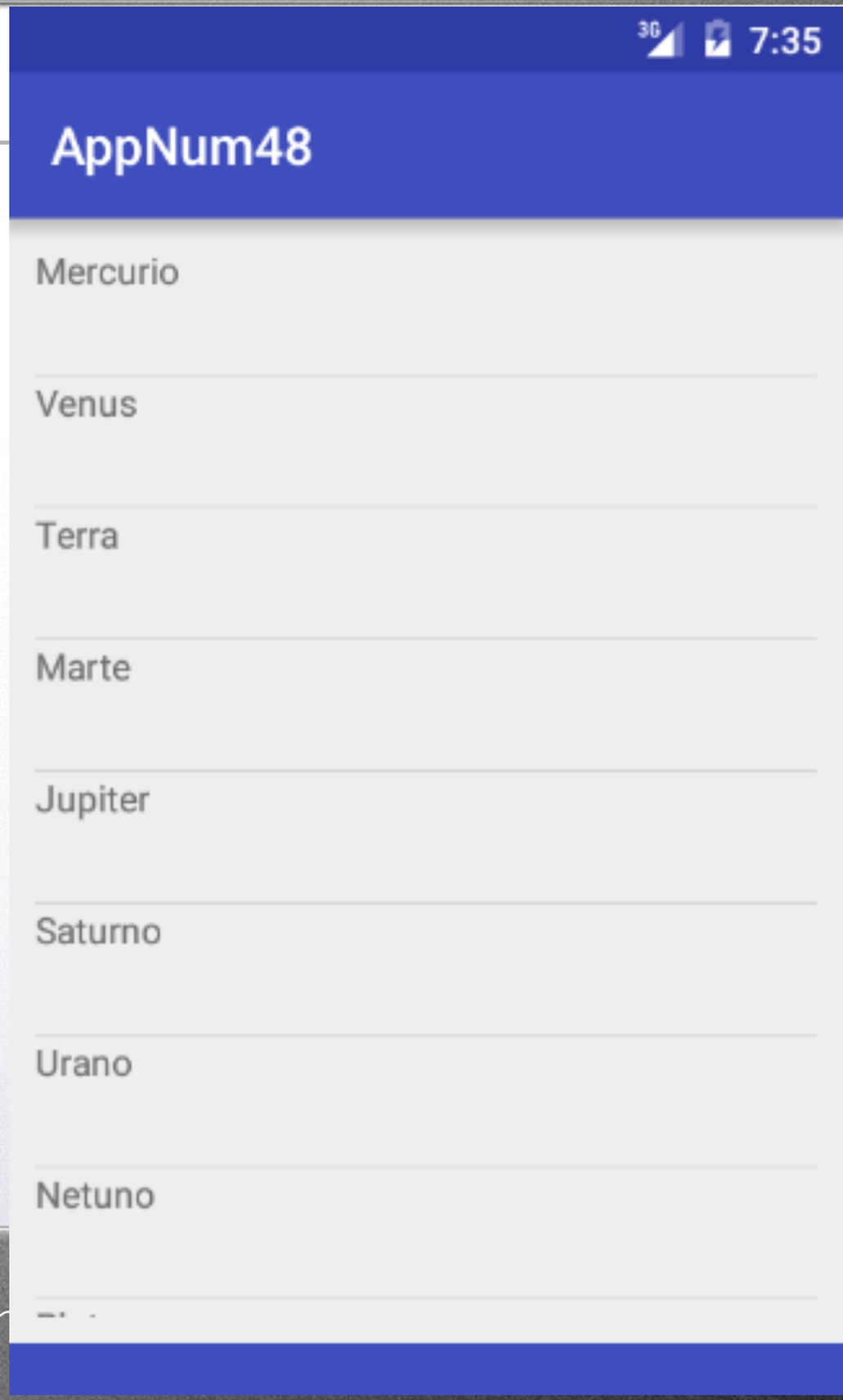
View

- Características
  - ✓ ListView
    - ✦ Exemplo AppNum48: MainActivity.java

```
@Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        //Notifica o objeto selecionado
        String s = (String) adapterView.getAdapter().getItem(i);
        Toast.makeText(this, "Texto selecionado: "+s+", posicao: "+i,Toast.LENGTH_SHORT).show();
    }
}
```

View

- Características
  - ✓ ListView
    - ✦ Exemplo AppNum48



- Características

✓ ListView com adapter customizado

✦ Os adapters utilizados no ListView podem ser customizados de acordo com a necessidade da aplicação.

✦ No próximo exemplo a lista de planetas será composta também por itens visuais.

✦ Modificações são primordialmente feitas no adapter da classe para refletir as necessidades.

✦ Ainda, é utilizado também o conceito de LayoutInflater como padronização do layout a ser apresentado na tela.


View

- Características

✓ ListView

✦ Exemplo AppNum49: adapter_planeta.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:gravity="center_vertical">

    <!-- Planeta imagem -->
    <ImageView android:id="@+id/imgPlaneta"
    android:layout_width="0dp" android:layout_height="wrap_content"
    android:layout_weight="3" android:src="@drawable/terra"/>

    <!-- Planeta nome -->
    <TextView android:id="@+id/tNomePlaneta"
    android:layout_width="0dp" android:layout_height="wrap_content"
    android:layout_weight="7" android:layout_marginLeft="10dp" android:textColor="#000000"/>
</LinearLayout>
```

View

- Características

  ✓ ListView

    ✦ Exemplo AppNum49: activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.cea436.appnum49.MainActivity">

    <ListView android:id="@+id/listview"
    android:layout_width="match_parent"
    android:layout_height="0dp" android:layout_weight="1" android:layout_margin="10dp"/>

</LinearLayout>
```

View

- Características
  - ✓ ListView
    - ✦ Exemplo AppNum49: Planeta.java

```
public class Planeta {
    public String nome;
    public int img; //Referencia para imagem em R.drawable.xxx

    public Planeta(String nome, int img) {
        this.nome = nome;
        this.img = img;
    }

    public static List<Planeta> getPlanetas() {
        List<Planeta> planetas = new ArrayList<Planeta>();

        planetas.add(new Planeta("Mercurio",
R.drawable.mercurio));
        planetas.add(new Planeta("Venus", R.drawable.venus));
        planetas.add(new Planeta("Terra", R.drawable.terra));
        planetas.add(new Planeta("Marte", R.drawable.marte));
        planetas.add(new Planeta("Jupiter",
R.drawable.jupiter));
        planetas.add(new Planeta("Saturno",
R.drawable.saturno));
        planetas.add(new Planeta("Netuno", R.drawable.netuno));
        planetas.add(new Planeta("Plutao", R.drawable.plutao));
        return planetas;
    }
}
```

View

- Características

✓ ListView

✦ Exemplo AppNum49: PlanetaAdapter.java

```
public class PlanetaAdapter extends BaseAdapter {
    private final Context context;
    private final List<Planeta> planetas;

    public PlanetaAdapter(Context context,
List<Planeta> planetas) {
    this.context = context;
    this.planetas = planetas;
}

@Override
public int getCount() {
    return planetas != null ? planetas.size() : 0;
}

@Override
public Object getItem(int i) {
    return planetas.get(i);
}
```

View

- Características
  - ✓ ListView
    - ✦ Exemplo AppNum49: PlanetaAdapter.java

```
@Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int i, View view, ViewGroup
viewGroup_parent) {
        //Infla a view
        View newView =
LayoutInflater.from(context).inflate(R.layout.adapter_planeta,
viewGroup_parent, false);
        //Busca as views que precisa atualiar
        TextView t = (TextView)
newView.findViewById(R.id.tNomePlaneta);
        ImageView img = (ImageView)
newView.findViewById(R.id.imgPlaneta);
        //Atualiza os valores das views
        Planeta planeta = planetas.get(i);
        t.setText(planeta.nome);
        img.setImageResource(planeta.img);
        return newView;
    }
}
```


View

- Características

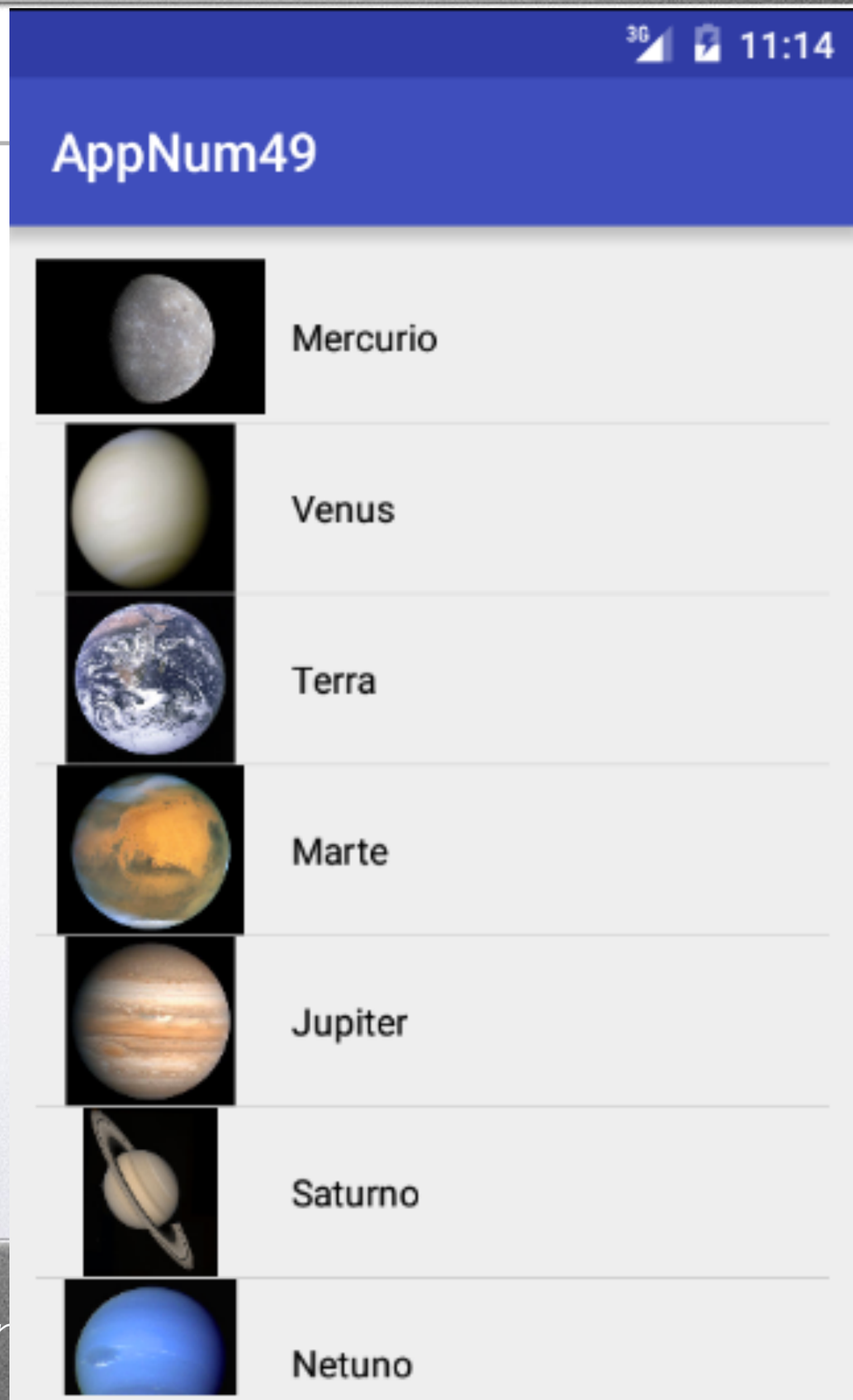
✓ ListView

✦ Exemplo AppNum49: MainActivity.java

```
public class MainActivity extends AppCompatActivity implements AdapterView.OnItemClickListener {
    protected static final String TAG = "CEA436";
    private ListView listView;
    private List<Planeta> planetas;

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

        listView = (ListView) findViewById(R.id.listview);
        planetas = Planeta.getPlanetas();
        listView.setAdapter(new PlanetaAdapter(this, planetas));
        listView.setOnItemClickListener(this);
    }

    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        Planeta p = this.planetas.get(i);
        Toast.makeText(this,"Planeta: "+p.nome, Toast.LENGTH_SHORT).show();
    }
}
```

View

- Características
  - ✓ ListView
    - ✦ Exemplo AppNum49



View

- Características

  ✓ GridView

  ✦ Implementado pela classe android.widget.GridView.

  ✦ Utilizado para exibir os componentes em formato de *grid* com linhas e colunas.

  ✦ É comumente utilizado para exibição de várias imagens como em um álbum de fotos.


View

- Características
  - ✓ GridView
    - ✦ Exemplo AppNum50: adapter_imagem.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <ImageView android:id="@+id/img"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</LinearLayout>
```


View

- Características

  ✓ GridView

    ✦ Exemplo AppNum50: activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context="com.cea436.appnum50.MainActivity"
    android:orientation="vertical">

    <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Exemplod e GridView!" />

    <GridView android:id="@+id/grid1"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:padding="10dip" android:gravity="center" android:verticalSpacing="10dip"
    android:horizontalSpacing="10dip" android:numColumns="auto_fit" android:columnWidth="40dip"/>
</LinearLayout>
```

View

- Características

✓ GridView

✦ Exemplo AppNum50: ImagemAdapter.java

```
public class ImagemAdapter extends BaseAdapter {

    private Context ctx;
    private final int[] imagens;

    public ImagemAdapter(Context c, int[] imagens) {
        this.ctx = c;
        this.imagens = imagens;
    }

    @Override
    public int getCount() {
        return imagens.length;
    }

    @Override
    public Object getItem(int i) {
        return imagens[i];
    }
```


View

- Características
  - ✓ GridView
    - ✦ Exemplo AppNum50: ImagemAdapter.java

```
    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup_parent) {
        View newView = LayoutInflater.from(this.ctx).inflate(R.layout.adapter_imagem,
viewGroup_parent, false);
        ImageView img = (ImageView) newView.findViewById(R.id.img);
        img.setImageResource(imagens[i]);
        return newView;
    }
}
```

View

- Características

  ✓ GridView

  ✦ Exemplo AppNum50: MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    private int[] imagens = {R.drawable.smile1, R.drawable.smile2,
R.drawable.smile1, R.drawable.smile2,
R.drawable.smile1, R.drawable.smile2,
R.drawable.smile1, R.drawable.smile2};

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

    GridView grid = (GridView) findViewById(R.id.grid1);
    grid.setOnItemClickListener(onGridViewitemClick());
    //Informa adapter para preencher o GridView
    grid.setAdapter(new ImagemAdapter(this,imagens));
}
```

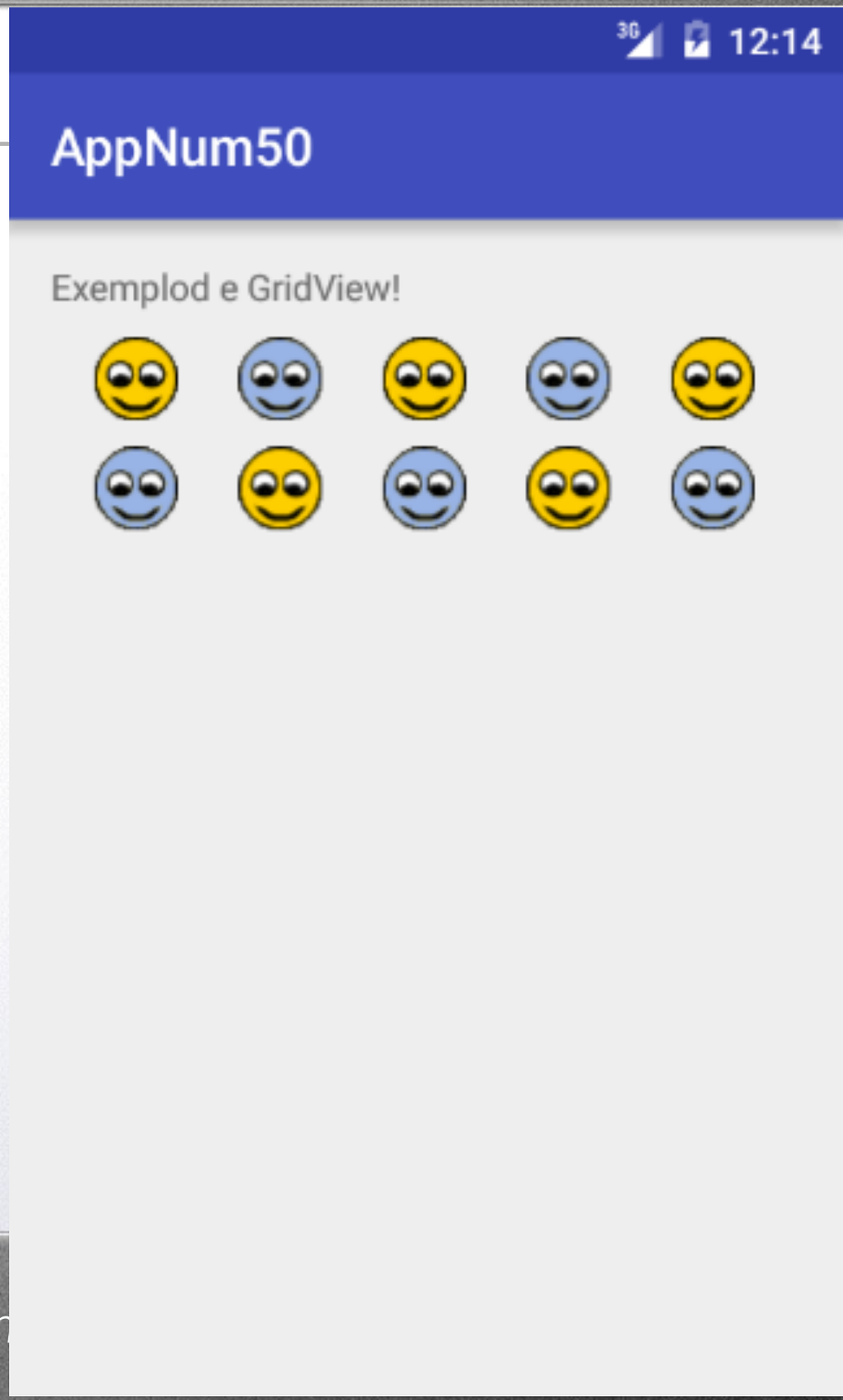

View

- Características
  - ✓ GridView
    - ✦ Exemplo AppNum50: MainActivity.java

```
    private OnItemClickListener onGridViewitemClick() {
            return new OnItemClickListener() {
                public void onItemClick(AdapterView<?> parent, View v, int pos, long id) {
                    Toast.makeText(MainActivity.this, "Imagem selecionada:
"+pos,Toast.LENGTH_SHORT).show();
                }
            };
        }
    }
```

View

- Características
  - ✓ GridView
    - ✦ Exemplo AppNum50



- Características

✓ ImageSwitcher

✦ Implementado pela classe android.widget.ImageSwitcher.

✦ Utilizada para exibir uma imagem após a outra com uma animação.

✦ Se baseia em implementações prontas da classe android.widget.ViewSwitcher.

✦ Para que funcione é necessário chamar o método setFactory(viewFactory) informando uma implementação de ViewFactory.

View

- Características

  ✓ ImageSwitcher

    ✦ Implementação da interface ViewFactory obrigatoriamente deve conter o método makeView().

    ✦ makeView() retorna uma View (imagem a ser exibida).

    ✦ No exemplo a seguir, a imagem é alterada sempre que o botão “Próxima" for clicado.

- Características

✓ ImageSwitcher: Exemplo: AppNum51 - activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:orientation="vertical" android:padding="10dp"
    tools:context="com.cea436.appnum51.MainActivity">

    <Button android:id="@+id/btProxima"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:text="@string/proxima"/>

    <ImageSwitcher android:id="@+id/imageSwitcher"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:layout_margin="10dp"/>
</LinearLayout>
```

View

- Características

✓ ImageSwitcher: Exemplo: AppNum51 - MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    private int[] imagens = {R.drawable.jupiter, R.drawable.plutao, R.drawable.netuno,
    R.drawable.saturno, R.drawable.marte, R.drawable.mercurio,
    R.drawable.terra, R.drawable.venus};

    private ImageSwitcher imageSwitcher;
    private int idx = 0;


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

        //Configura o imageswitcher e seus efeitos
        imageSwitcher = (ImageSwitcher) findViewById(R.id.imageSwitcher);
        imageSwitcher.setFactory(new ImageSwitcher.ViewFactory() {

...
```

View

- Características

✓ ImageSwitcher:

Exemplo:

AppNum51 -

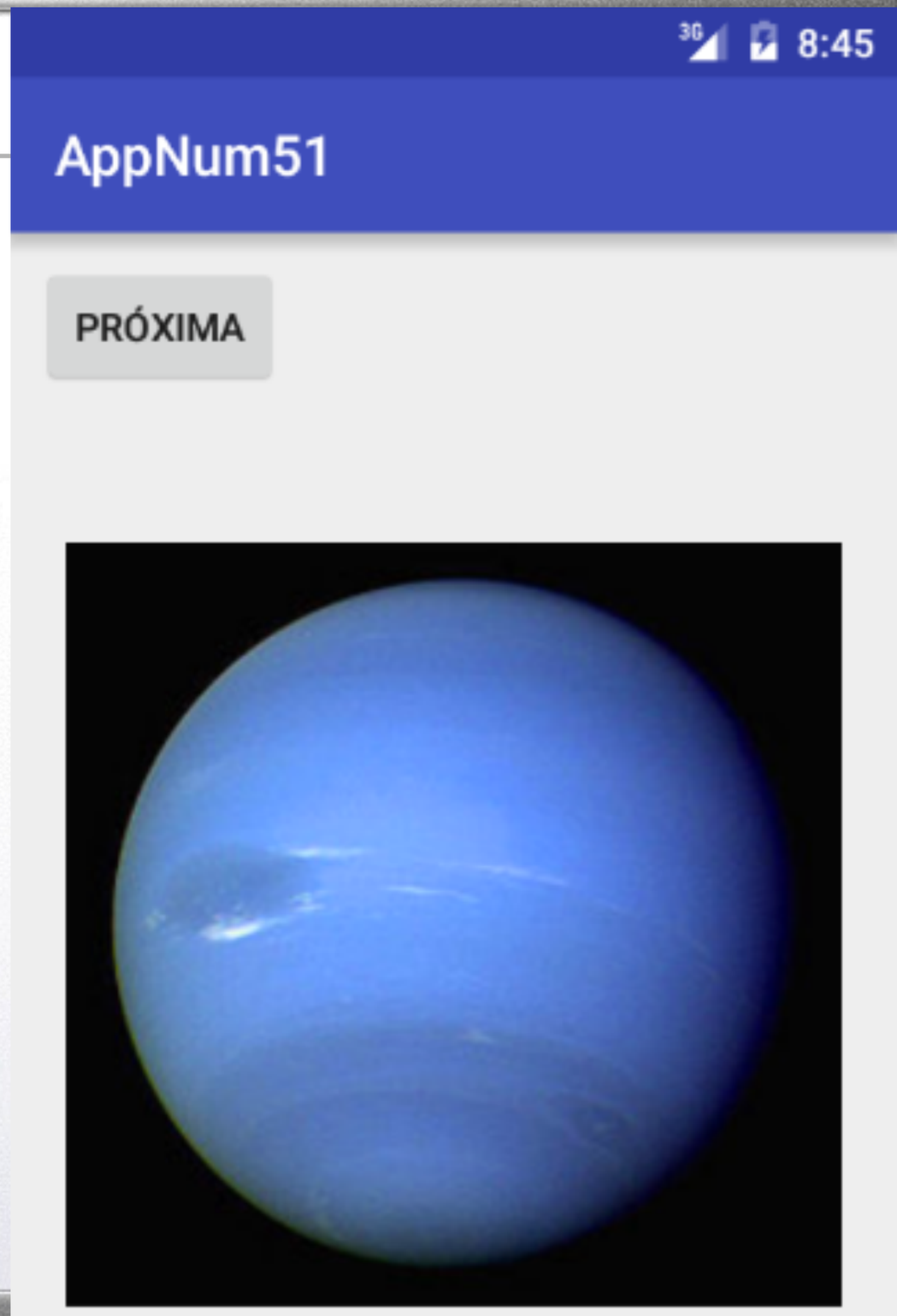
MainActivity.java

```
        //Quando imageSwitcher é chamado, makeView tb é invocado
        @Override
        public View makeView() {
            ImageView img = new ImageView(getBaseContext());
            img.setScaleType(ImageView.ScaleType.FIT_CENTER);
            img.setLayoutParams(new
ImageSwitcher.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT));
            return img;
        }
    });
```

```
        //Configurando as animações
        imageSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
android.R.anim.fade_in));

imageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,android.R.anim
.fade_out));
        Button btProxima = (Button) findViewById(R.id.btProxima);
        btProxima.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (idx == imagens.length) {
                    idx = 0;
                }
                imageSwitcher.setImageResource(imagens[idx++]);
            }
        });
```

View

- Características
  - ✓ ImageSwitcher: Exemplo: AppNum51



- Características

✓ WebView

✦ Possibilita a exibição de uma página web em um aplicativo deve-se utilizar um WebView.

✦ Implementado pela classe android.webkit.WebView.

✦ Muito útil durante a criação de interfaces gráficas que sirvam tanto para o aplicativo quanto para páginas web.

View

- Características

  ✓ WebView

    ✦ Obrigatória a permissão de acesso a internet em AndroidManifest.xml:

```
<uses-permission android:name="android.permission.INTERNET"/>
```


View

- Características

✓ WebView

✦ Exemplo AppNum52: activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.cea436.appnum52.MainActivity">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <WebView android:id="@+id/webView" android:layout_margin="10dp"
        android:layout_width="match_parent" android:layout_height="match_parent"/>

        <ProgressBar android:id="@+id/progress" android:layout_gravity="center"
        android:layout_width="wrap_content" android:layout_height="wrap_content" />
    </FrameLayout>
</LinearLayout>
```

View

- Características

✓ WebView

✦ Exemplo AppNum52: MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    private final String pageURL = "http://www.uol.com.br";

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

        final WebView webView = (WebView) findViewById(R.id.webView);
        final ProgressBar progress = (ProgressBar) findViewById(R.id.progress);
        progress.setVisibility(View.INVISIBLE);
        webView.loadUrl(pageURL);
...
```

View

- Características

✓ WebView

✦ Exemplo AppNum52: MainActivity.java

```
        webView.setWebViewClient(new WebViewClient() {
            //Pagina iniciando a ser carregada
            @Override
            public void onPageStarted(WebView view, String url,
Bitmap favicon) {
                Log.i("CEA436", "onPageStarted()");
                progress.setVisibility(View.VISIBLE);
            }

            //Pagina terminou de ser carregada
            @Override
            public void onPageFinished(WebView view, String url)
{
                Log.i("CEA436", "onPageFinished()");
                progress.setVisibility(View.INVISIBLE);
            }

            //Recebeu um erro durante carregamento
            @Override
            public void onReceivedError(WebView view, int
errorCode, String description, String failingURL) {
                //Emitir erro na tela
                Log.i("CEA436", "onReceivedError()");
            }
        });
```

View

- Características
  - ✓ WebView
    - ✦ Exemplo AppNum52

- Características

✓ Movimentação de elementos utilizando o *touch screen*

✦ *Framework* Android provê maneiras de se utilizar os eventos gerados pelo usuário na tela (*touch screen*).

✦ A classe View possui o método *callback* onTouchEvent(MotionEvent) que é chamada sempre que um toque na tela é efetuado.

✦ Utilizando-se a classe MotionEvent é possível recuperar informações sobre o evento gerado.

- Características
  - ✓ Movimentação de elementos com *TouchScreen*
    - ✦ Exemplo AppNum53: MainActivity.java

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TouchScreenView view = new TouchScreenView(this);
        setContentView(view);
    }
```


View

- Características
  - ✓ Movimentação de elementos com *TouchScreen*
    - ✦ Exemplo AppNum53: TouchScreenView.java

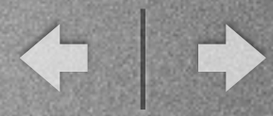
```
public class TouchScreenView extends View {
private static final String CATEGORIA = "AppNum53";
private Drawable img;
int x,y;
private boolean selecionou;
private int larguraTela, alturaTela, larguraImg, alturaImg;

public TouchScreenView(Context context) {
    super(context,null);

    img = context.getResources().getDrawable(R.drawable.android);
    //Recupera dimensoes da imagem
    larguraImg = img.getIntrinsicWidth();
    alturaImg = img.getIntrinsicHeight();
    setFocusable(true);
}
```

View

- Características
  - ✓ Movimentação de elementos com *TouchScreen*
    - ✦ Exemplo AppNum53: TouchScreenView.java

```
@Override
    //Callback para quando a tela é iniciada ou
redimensionada
    protected void onSizeChanged(int width, int height,
int oldw, int oldh) {
        super.onSizeChanged(width, height, oldw, oldh);
        this.alturaTela = height;
        this.larguraTela = width;
        x = width/2 - (larguraImg/2);
        y = height/2 - (alturaImg/2);
        Log.i(CATEGORIA,"onSizeChanged x:y = "+x
+":"+y);
    }
```

```
@Override
    public void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint pincel = new Paint();
        pincel.setColor(Color.WHITE);
        canvas.drawRect(0, 0, larguraTela,
alturaTela, pincel);
        //Definição da área passível de desenho
        img.setBounds(x,y,x+larguraImg,y+alturaImg);
        img.draw(canvas);
    }
```

View

- Características
  - ✓ Movimentação de elementos com *TouchScreen*
    - ✦ Exemplo AppNum53: TouchScreenView.java

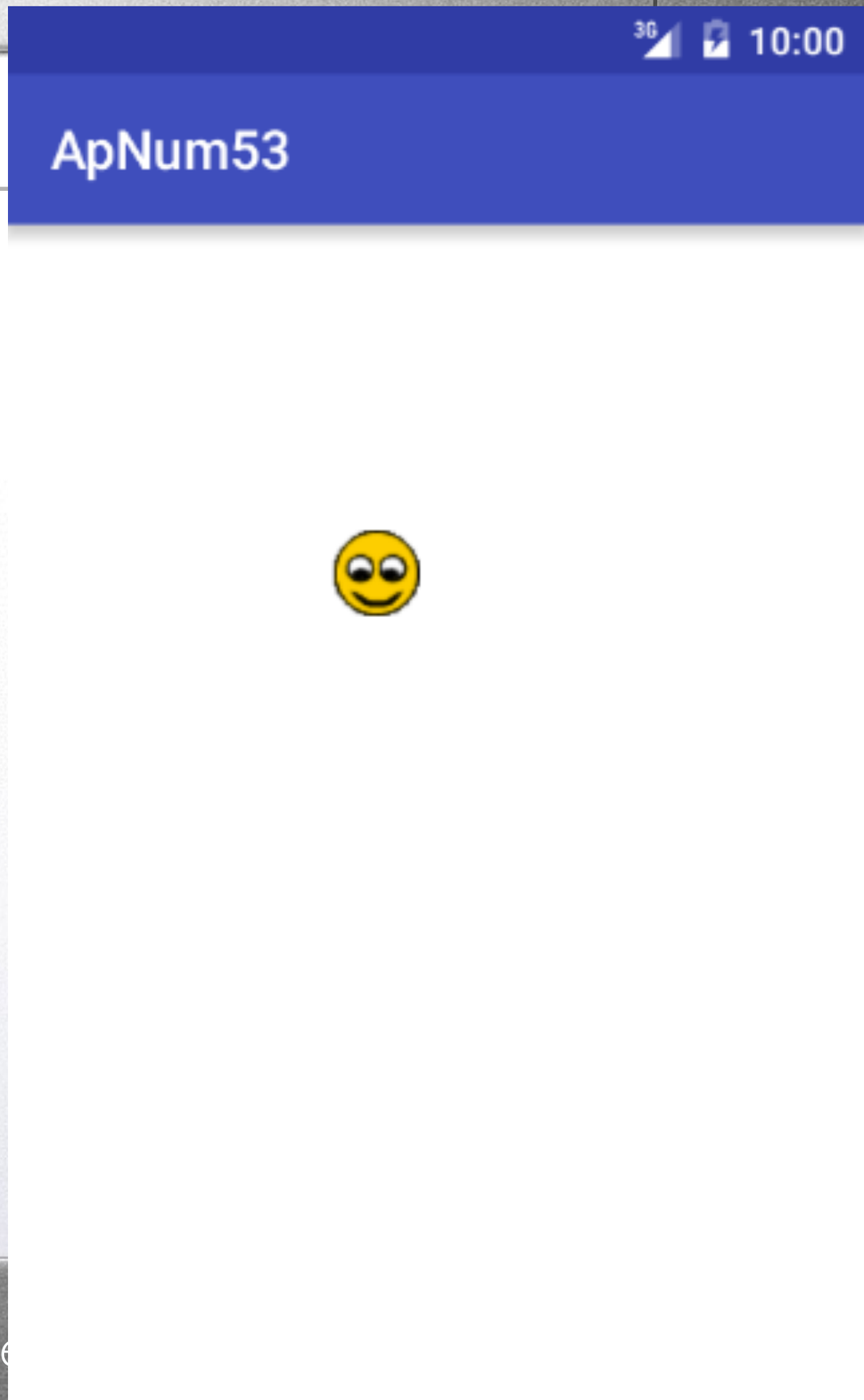
```
@Override
    //Move a imagem
    public boolean
onTouchEvent(MotionEvent event) {
        float x = event.getX();
        float y = event.getY();
        Log.i(CATEGORIA,"onTouchEvent:
x:y = "+x+":"+y);
        switch(event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                //Inicia movimento se pressionou a imagem
                selecionou = img.copyBounds().contains((int)x,(int)y);
                break;
            case MotionEvent.ACTION_MOVE:
                //Arrasta o boneco
                if (selecionou) {
                    this.x = (int)x - (larguraImg/2);
                    this.y = (int)y - (alturaImg/2);
                }
                break;
            case MotionEvent.ACTION_UP:
                //Finaliza movimento
                selecionou = false;
                break;
        }
        invalidate();
        return true;
    }
```

View

- Características

  ✓ Movimentação de elementos com *TouchScreen*

    ✦ Exemplo AppNum53



View

- Características

  ✓ Classe Canvas: Desenho

    ✦ Exemplo anterior cobria a extensão de um componente já existente.

    ✦ Problema: Como criar um componente visual totalmente novo?

    ✦ Solução: Criação de um componente customizado que será uma classe direta de View.

    ✦ Sobrescrever o método onDraw(Canvas).

View

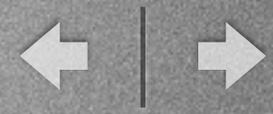
- Características
  - ✓ Classe Canvas: Desenho
    - ✦ Exemplo AppNum54: activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <com.cea436.appnum54.Quadrado1 android:id="@+id/canvas"
        android:layout_width="match_parent" android:layout_height="match_parent"/>
</LinearLayout>
```

View

- Características
  - ✓ Classe Canvas: Desenho
    - ✦ Exemplo AppNum54: Quadrado1.java

```
public class Quadrado1 extends View {

    private Paint pincelVermelho;
    private Paint pincelPreto;
    private Paint pincelAzul;

    public Quadrado1(Context context) {
        this(context,null);
    }

    public Quadrado1(Context context, AttributeSet attrs) {
        super(context,attrs);
        setBackgroundColor(Color.LTGRAY);
        //Pincel vermelho
        pincelVermelho = new Paint();
        pincelVermelho.setARGB(255, 255, 0, 0);
        //Pincel preto
        pincelPreto = new Paint();
        pincelPreto.setARGB(255, 0, 0, 0);
        //Pincel azul
        pincelAzul = new Paint();
        pincelAzul.setARGB(255, 0, 0, 255);
        //Configura a View para receber foco e tratar
eventos de teclado
        setFocusable(true);
    }
```

View

- Características
  - ✓ Classe Canvas: Desenho
    - ✦ Exemplo AppNum54: Quadrado1.java

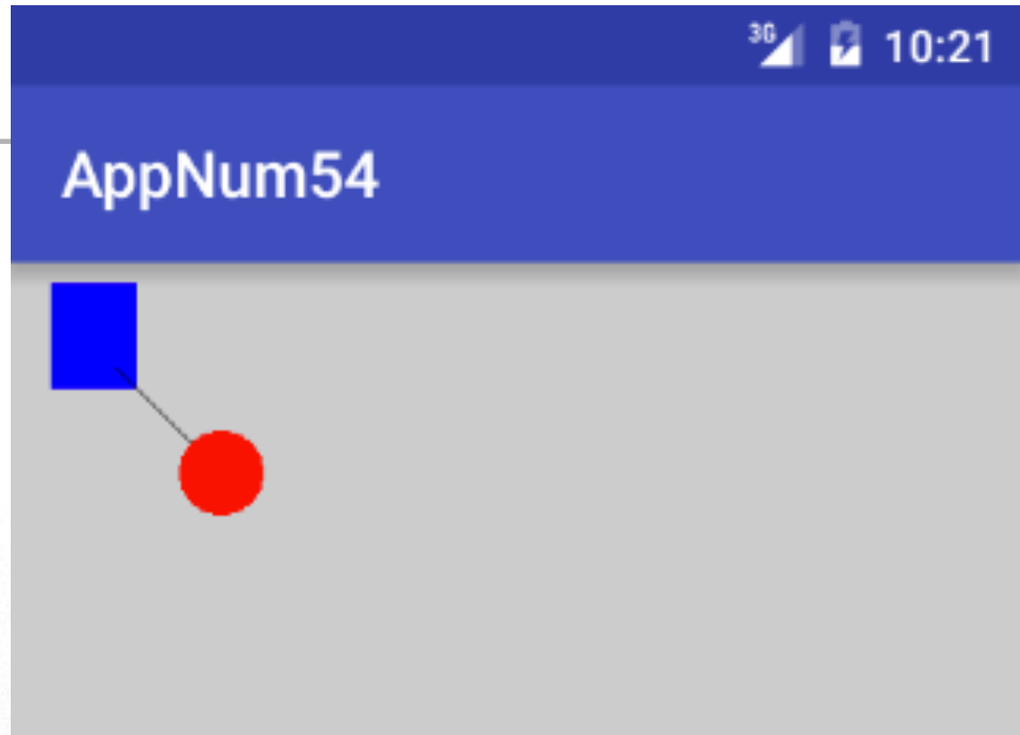
```
@Override
    public void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        int y = 10;

        canvas.drawRect(19, y, 10+50, y+50, pincelAzul);
        canvas.drawLine(50, 50, 100, 100, pincelPreto);
        canvas.drawCircle(100, 100, 20, pincelVermelho);
    }
```

View

- Características

  ✓ Classe Canvas: Desenho

✦ Exemplo AppNum54: Quadrado1.java



- Características

✓ Movimentação de objetos

✦ Característica fortemente presente em dispositivos com interação via *touch-screen*.

✦ Funcionalidade muito utilizada na implementação de jogos.

✦ Suporte também possível de se implementar através do uso de teclas físicas:

. Identificação do evento de tecla através de códigos associados a cada uma delas.

- Características

  ✓ Movimentação de objetos

    ✦ Em dispositivos com teclado físico, implementação do tratamento de eventos pode ser feito pela função onKeyDown(int codigo, KeyEvent evento).

    ✦ Função deve ser implementada dentro da classe que herde da classe View.

    ✦ AppNum53 e AppNum54 podem ser utilizadas como base.

- Características

✓ Exercício 15:

✦ Utilizando os conceitos de implementação e tratamento de eventos relacionados à classe View, crie um jogo de pingue-pong. Este deverá conter dois elementos:

1) Um círculo, significando a bola; e

2) Uma barra horizontal significando o (único) jogador.

* Quando do contato da barra com a bolinha, essa deverá ser rebatida no sentido contrário.