

Aula 02 – Views

Programação em Java para a Plataforma Android



Agenda

- O que é modelo, visão e controle
- Como definir layouts gráficos
- Como separar visão de modelo
- Como criar layouts em arquivos XML
- Como adicionar eventos à aplicação
- O padrão *observer*



Model-View-Controller

- *Model*: banco de músicas + código para tocá-las, lista de contatos + código de chamada, etc
- *View*: o conjunto de interfaces gráficas que compõem uma aplicação
- *Controller*: o código que faz a ligação entre a visão e o modelo
 - Tratamento de eventos



Criando interfaces de usuário

- Há duas estratégias básicas para criar-se uma visão para o usuário:
 1. Programação do código da interface em Java, diretamente na aplicação
 2. Criação de um arquivo XML que descreve a interface gráfica
- Ambas as abordagens possuem vantagens
 - Vamos dar uma olhada na abordagem 1 primeiro

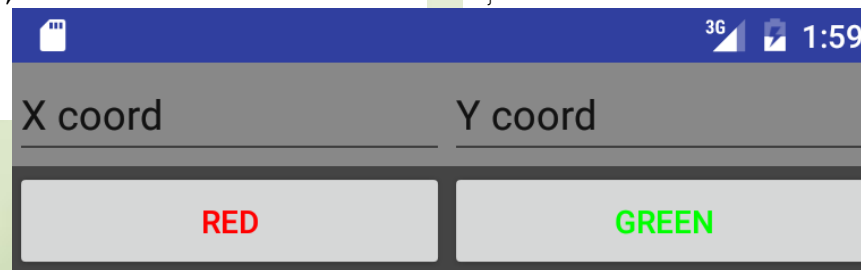


Programação do código da interface em Java

MainActivity.java

```
public final void onCreate(final Bundle state) {
    super.onCreate(state);
    LinearLayout.LayoutParams containerParams = new
    LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.WRAP_CONTENT, 0.0F);
    LinearLayout.LayoutParams widgetParams = new
    LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT,
        1.0F);
    LinearLayout root = new LinearLayout(this);
    root.setOrientation(LinearLayout.VERTICAL);
    root.setBackgroundColor(Color.LTGRAY);
    root.setLayoutParams(containerParams);
    LinearLayout ll = new LinearLayout(this);
    ll.setOrientation(LinearLayout.HORIZONTAL);
    ll.setBackgroundColor(Color.GRAY);
    ll.setLayoutParams(containerParams);
    root.addView(ll);
    EditText tb = new EditText(this);
    tb.setText(R.string.defaultLeftText);
    tb.setFocusable(false);
    tb.setLayoutParams(widgetParams);
    ll.addView(tb);
```

```
tb = new EditText(this);
tb.setText(R.string.defaultRightText);
tb.setFocusable(false);
tb.setLayoutParams(widgetParams);
ll.addView(tb);
ll = new LinearLayout(this);
ll.setOrientation(LinearLayout.HORIZONTAL);
ll.setBackgroundColor(Color.DKGRAY);
ll.setLayoutParams(containerParams);
root.addView(ll);
Button b = new Button(this);
b.setText(R.string.labelRed);
b.setTextColor(Color.RED);
b.setLayoutParams(widgetParams);
ll.addView(b);
b = new Button(this);
b.setText(R.string.labelGreen);
b.setTextColor(Color.GREEN);
b.setLayoutParams(widgetParams);
ll.addView(b);
setContentView(root);
}
```



Programação do código da interface em Java

```
...

LinearLayout.LayoutParams containerParams =
new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.WRAP_CONTENT, 0.0F);

...

LinearLayout root = new LinearLayout(this);

root.setOrientation(LinearLayout.VERTICAL);
root.setBackgroundColor(Color.LTGRAY);
root.setLayoutParams(containerParams);

...

root.addView(ll);

...
```

- O que querem dizer essas linhas de código?

O painel raiz:

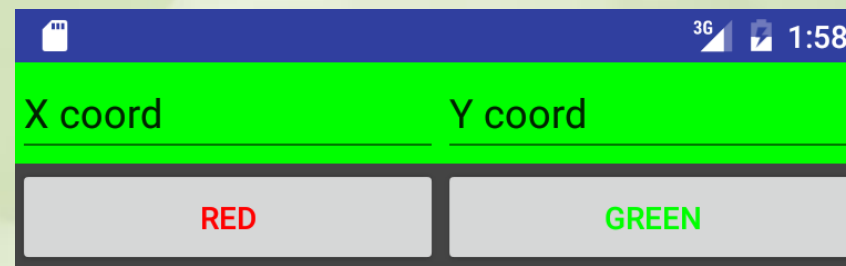
O painel raiz serve de suporte para os outros componentes



Programação do código da interface em Java

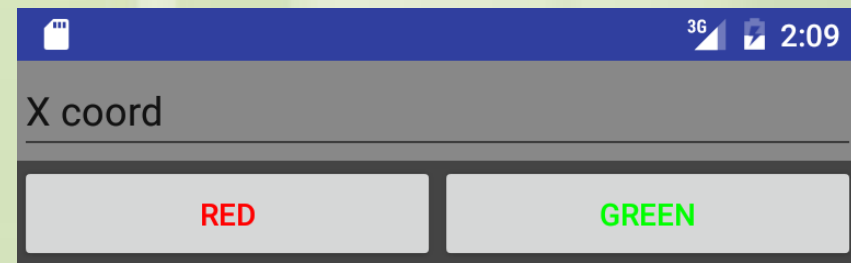
```
...  
    LinearLayout root = new  
    LinearLayout(this);  
  
    root.setOrientation(LinearLayout.VERTICAL);  
    root.setBackgroundColor(Color.LTGRAY);  
    root.setLayoutParams(containerParams);  
    LinearLayout ll = new  
    LinearLayout(this);  
  
    ll.setOrientation(LinearLayout.HORIZONTAL);  
    ll.setBackgroundColor(Color.GRAY);  
    ...
```

- E se eu mudasse **isto** para GREEN?



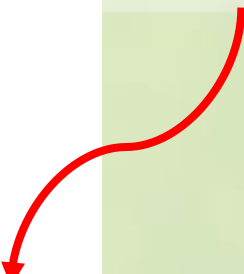
```
tb = new EditText(this);  
tb.setText(R.string.defaultRightText);  
tb.setFocusable(false);  
tb.setLayoutParams(widgetParams);  
ll.addView(tb); ←  
ll = new LinearLayout(this);  
ll.setOrientation(LinearLayout.HORIZONTAL);  
ll.setBackgroundColor(Color.DKGRAY);
```

- E se eu removesse **esta** linha?




```
tb.setLayoutParams(widgetParams);  
ll.addView(tb);  
ll = new LinearLayout(this);  
ll.setOrientation(LinearLayout.HORIZONTAL);  
ll.setBackgroundColor(Color.DKGRAY);  
ll.setLayoutParams(containerParams);  
root.addView(ll);
```

- E se fosse **VERTICAL**?



X coord		Y coord	
		RED	
		GREEN	



Discussão

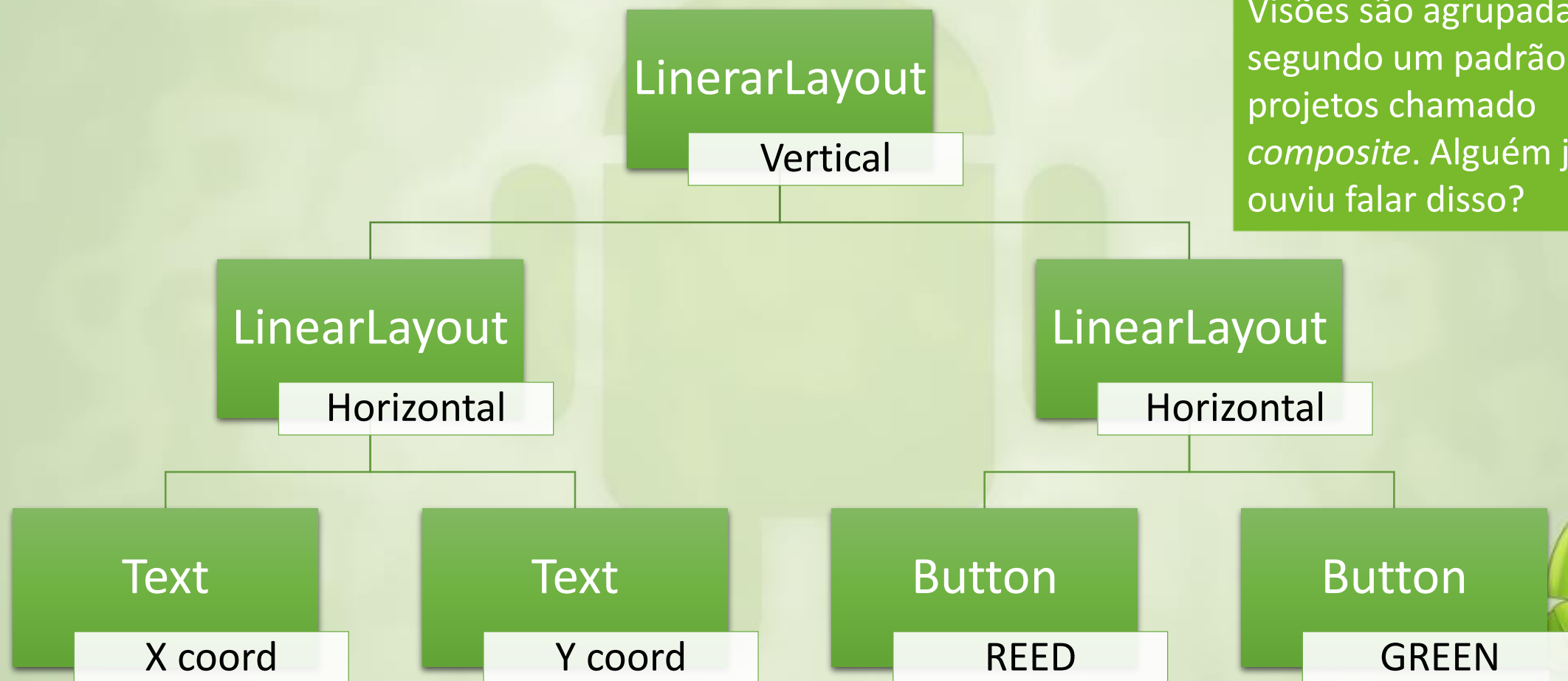
- Alguém aqui já programou interfaces gráficas dessa forma?
- Quais as desvantagens desse tipo de abordagem?
- E quais seriam as alternativas?



Árvores de componentes

Composite:

Visões são agrupadas segundo um padrão de projetos chamado *composite*. Alguém já ouviu falar disso?



Padrões de projeto:
O que é isto mesmo?



Padrões de projeto

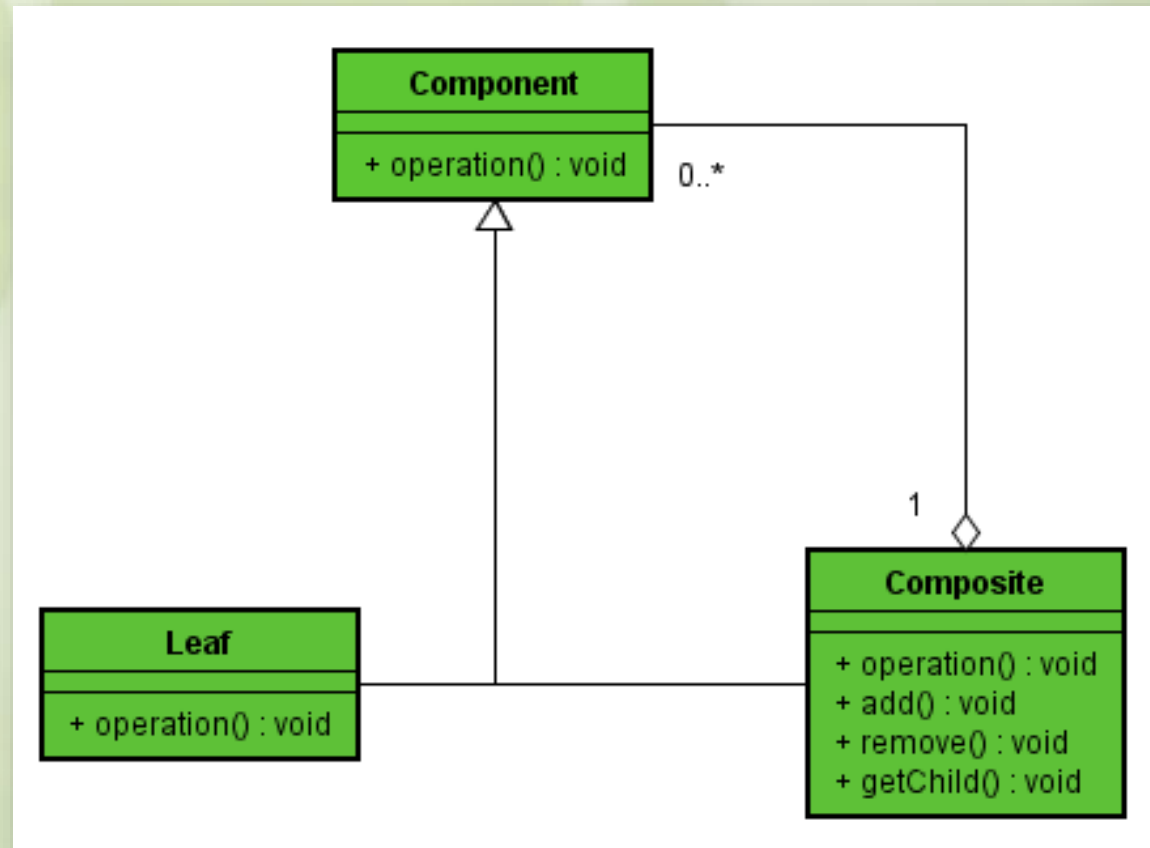
- Soluções comuns para problemas recorrentes em programação
 - Padrões criacionais
 - Padrões estruturais
 - Padrões comportamentais



Composite

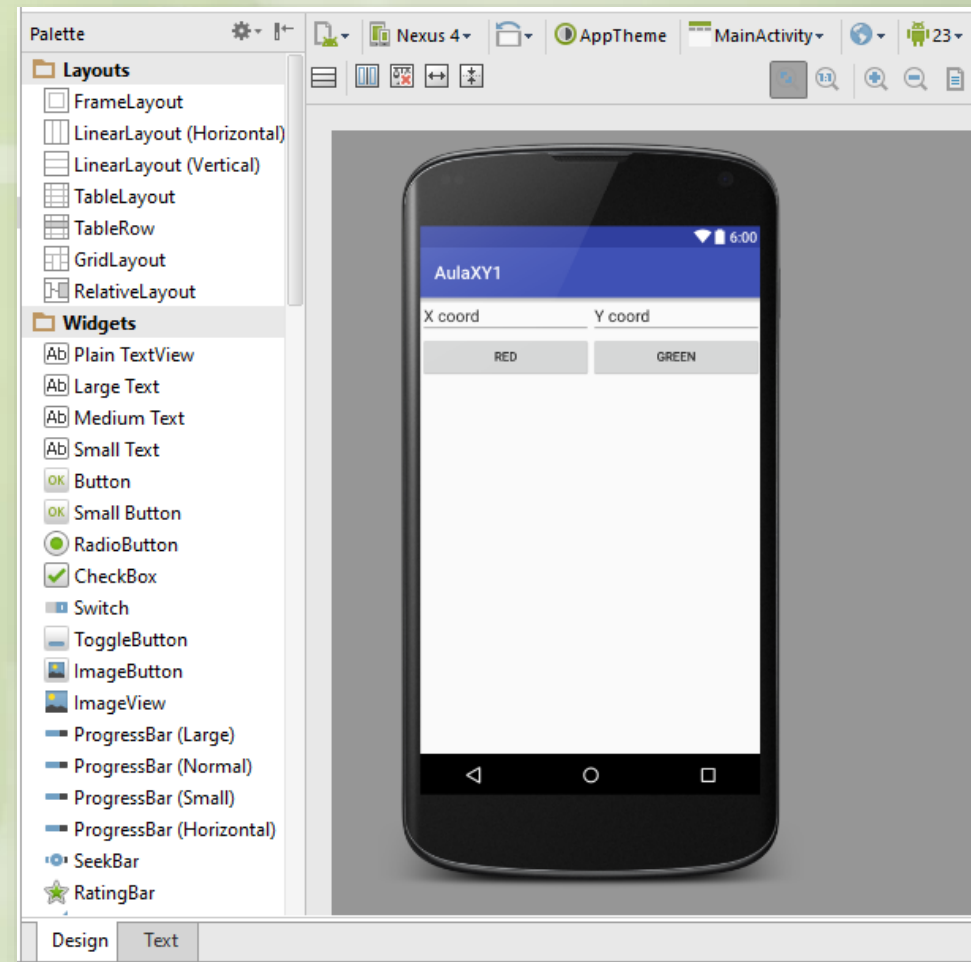
- A ideia desse padrão de projetos é tratar objetos compostos e objetos primitivos da mesma forma

No caso do Android, quais as classes fazem o papel de Leaf e Composite?



Separando visão e modelo

- O Android permite que as interfaces gráficas sejam projetadas em XML
 - XML permite que árvores de componentes gráficos sejam construídas com facilidade
- O Android SDK disponibiliza um editor de interfaces gráficas



Separando visão e modelo

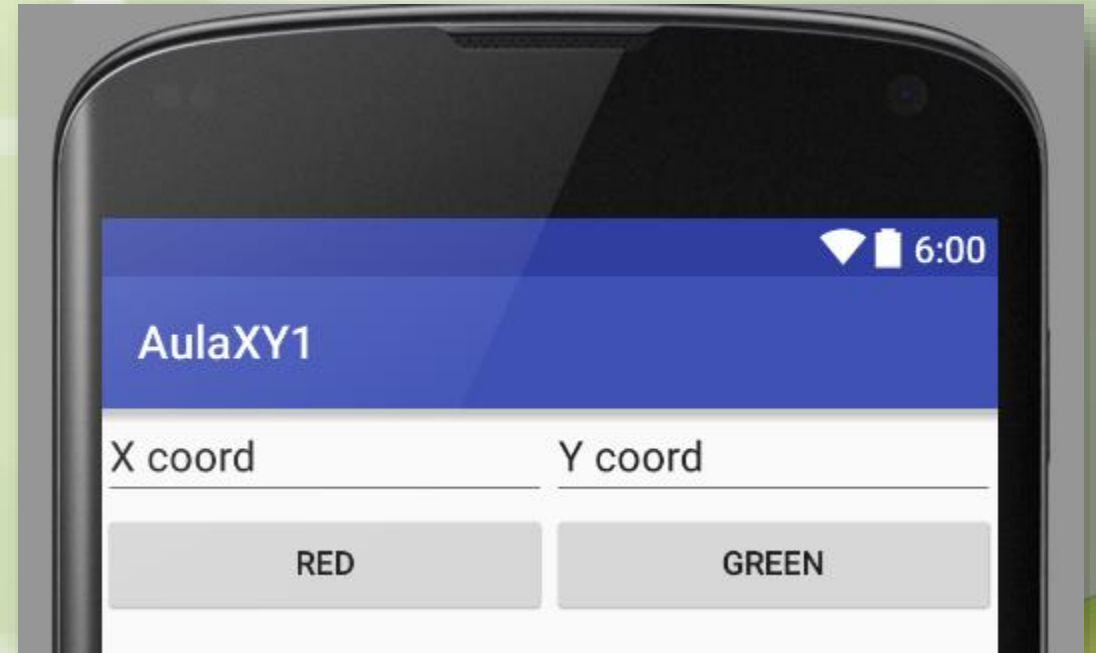
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:orientation="vertical"
    android:id="@+id/root"
    xmlns:android=
"http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <EditText
            android:id="@+id/text1"
            android:text="@string/defaultLeftText"
            android:focusable="false"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
        <EditText
            android:id="@+id/text2"
            android:text="@string/defaultRightText"
            android:focusable="false"
            android:layout_width="fill_parent"
```

```
        android:layout_height="fill_parent"
            android:layout_weight="1"/>
    </LinearLayout>
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <Button
            android:id="@+id/buRon1"
            android:text="@string/labelRed"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
        <Button
            android:id="@+id/buRon2"
            android:text="@string/labelGreen"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
    </LinearLayout>
</LinearLayout>
```



Separando visão e modelo

```
<LinearLayout>
  <LinearLayout>
    <EditText/>
    <EditText/>
  </LinearLayout>
  <LinearLayout>
    <Button/>
    <Button/>
  </LinearLayout>
</LinearLayout>
```



Alguém viu uma
árvore aqui?



Ligando os fios

```
package br.com.android.helloandroid;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

    public final void onCreate(final Bundle state) {
        super.onCreate(state);
        setContentView(R.layout.activity_main);
    }
}
```

- A separação entre visão e modelo permite que o código de modelo fique menor e mais claro



Alguns testes

Como adicionar cores ao
nosso layout?

```
<EditText  
    android:id="@+id/text1"  
    android:text="@string/defaultLevText"  
    android:focusable="false"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:layout_weight="1"/>
```

- Pra que serve esse identificador?

- E se fosse 1.5?



Cores como recursos

- Podemos definir cores em um arquivo XML em
`/res/values/colors.xml`

Como as cores são codificadas?

Como usar esses recursos em nosso arquivo de layout?

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="red">#ffff0000</color>
    <color name="green">#ff00ff00</color>
</resources>
```



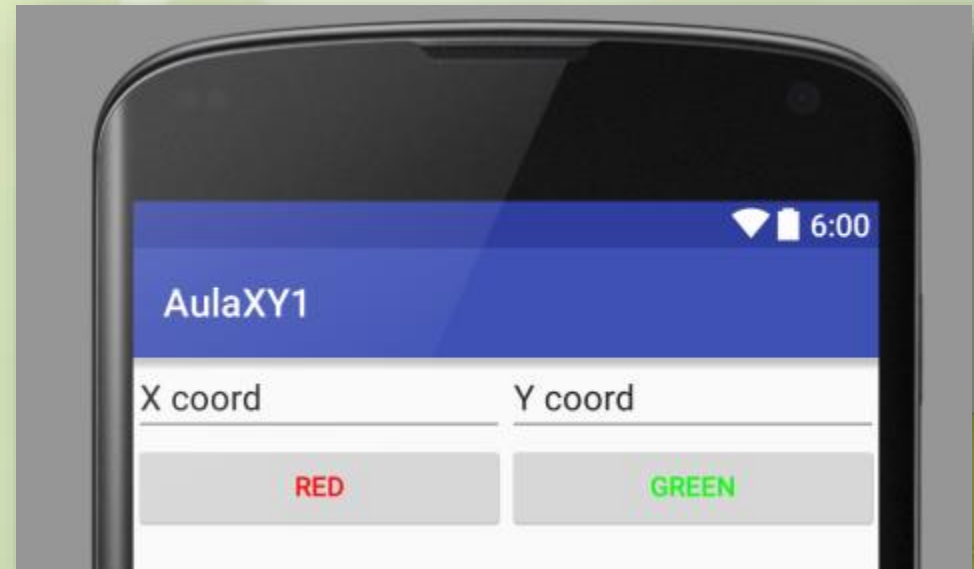
Atributos de cores

<Button

```
    android:id="@+id/button1"  
    android:text="@string/labelRed"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:layout_weight="1"  
    android:textColor="@color/red"/>
```

<Button

```
    android:id="@+id/button2"  
    android:text="@string/labelGreen"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:layout_weight="1"  
    android:textColor="@color/green"/>
```



Adicionando eventos

- Interfaces gráficas interagem com usuários via eventos

```
final EditText tb1 = (EditText) findViewById(R.id.text1);  
final EditText tb2 = (EditText) findViewById(R.id.text2);  
( (Button) findViewById(R.id.button2) )  
    .setOnClickListener(new Button.OnClickListener() {  
        public void onClick(View arg0) {  
            tb1.setText(String.valueOf(numClicks++));  
            tb2.setText(String.valueOf(numClicks++));  
        }  
    });
```



Identificadores

- Interfaces gráficas interagem com usuários via eventos

```
final EditText tb1 = (EditText) findViewById(R.id.text1);  
final EditText tb2 = (EditText) findViewById(R.id.text2);  
( (Button) findViewById(R.id.button2) )  
    .setOnClickListener(new Button.OnClickListener() {  
        public void onClick(View arg0) {  
            tb1.setText(String.valueOf(numClicks++));  
            tb2.setText(String.valueOf(numClicks++));  
        }  
    });
```



Observadores

- O botão agora é um observador de eventos
 - Que observa eventos de clique
- Um observador pode observar eventos em vários elementos

```
private int numClicks = 0;
public void onCreate(Bundle state) {
    super.onCreate(state);
    setContentView(R.layout.activity_main);
    final EditText tb1 = (EditText) findViewById(R.id.text1);
    final EditText tb2 = (EditText) findViewById(R.id.text2);
    Button.OnClickListener listener = new Button.OnClickListener() {
        public void onClick(View arg0) {
            tb1.setText(String.valueOf(numClicks++));
            tb2.setText(String.valueOf(numClicks++));
        }
    };
    ((Button) findViewById(R.id.button1)).setOnClickListener(listener);
    ((Button) findViewById(R.id.button2)).setOnClickListener(listener);
}
```



Exercício: anagramas

- Escreva uma atividade AnagramActivity que determine se duas strings são anagramas
- Use um “Toast” para reportar se as strings são anagramas ou não

Um anagrama (do grego ana = "voltar" ou "repetir" + graphein = "escrever") é uma espécie de jogo de palavras, resultando do rearranjo das letras de uma palavra ou frase para produzir outras palavras, utilizando todas as letras originais exatamente uma vez.



Dúvidas

