



## Programação para a Plataforma Android – Aula 5

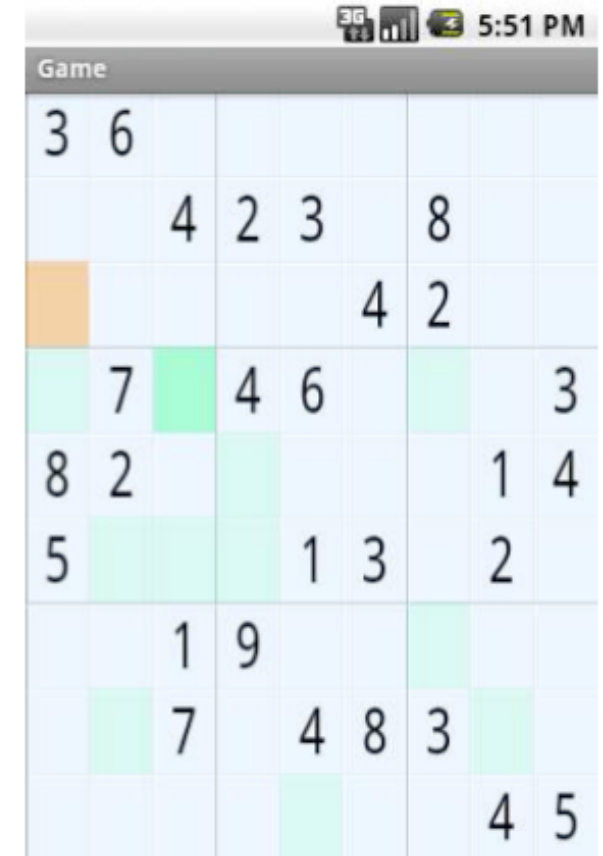
### Sudoku

9	4		1		2		5	8
6				5				4
		2	4		3	1		
	2						6	
5		8		2		4		1
	6						8	
		1	6		8	7		
7				4				3
4	3		5		9		1	2

- Refinando layouts
- Lidando com telas em modo paisagem
- Invocando atividades dentro de atividades
- Usando temas para melhorar layouts
- Depurando via o console de saída
- Carregando uma atividade

# Sudoku

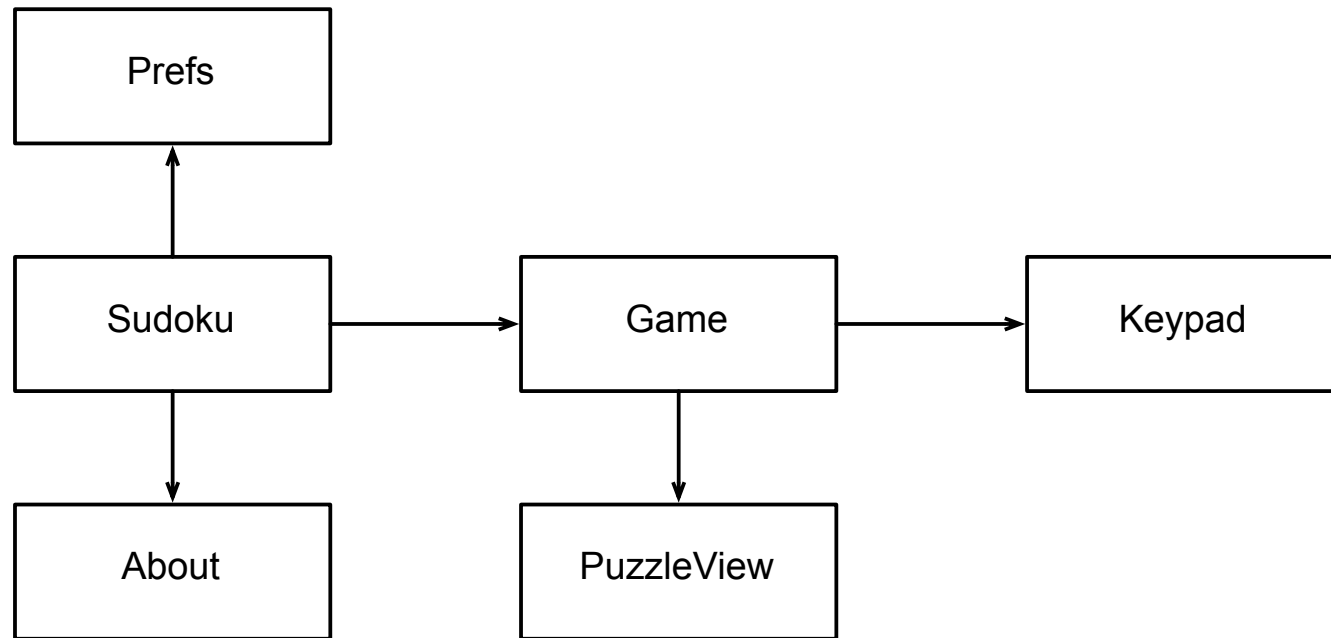
- Sudoku é um jogo de raciocínio, cujo objetivo é preencher uma matriz de 9x9 casas com números inteiros entre 1 e 9 inclusive, de tal forma que nenhuma linha, coluna ou diagonal possua números iguais.



3	6							
		4	2	3		8		
					4	2		
	7		4	6				3
8	2						1	4
5				1	3		2	
		1	9					
		7		4	8	3		
							4	5

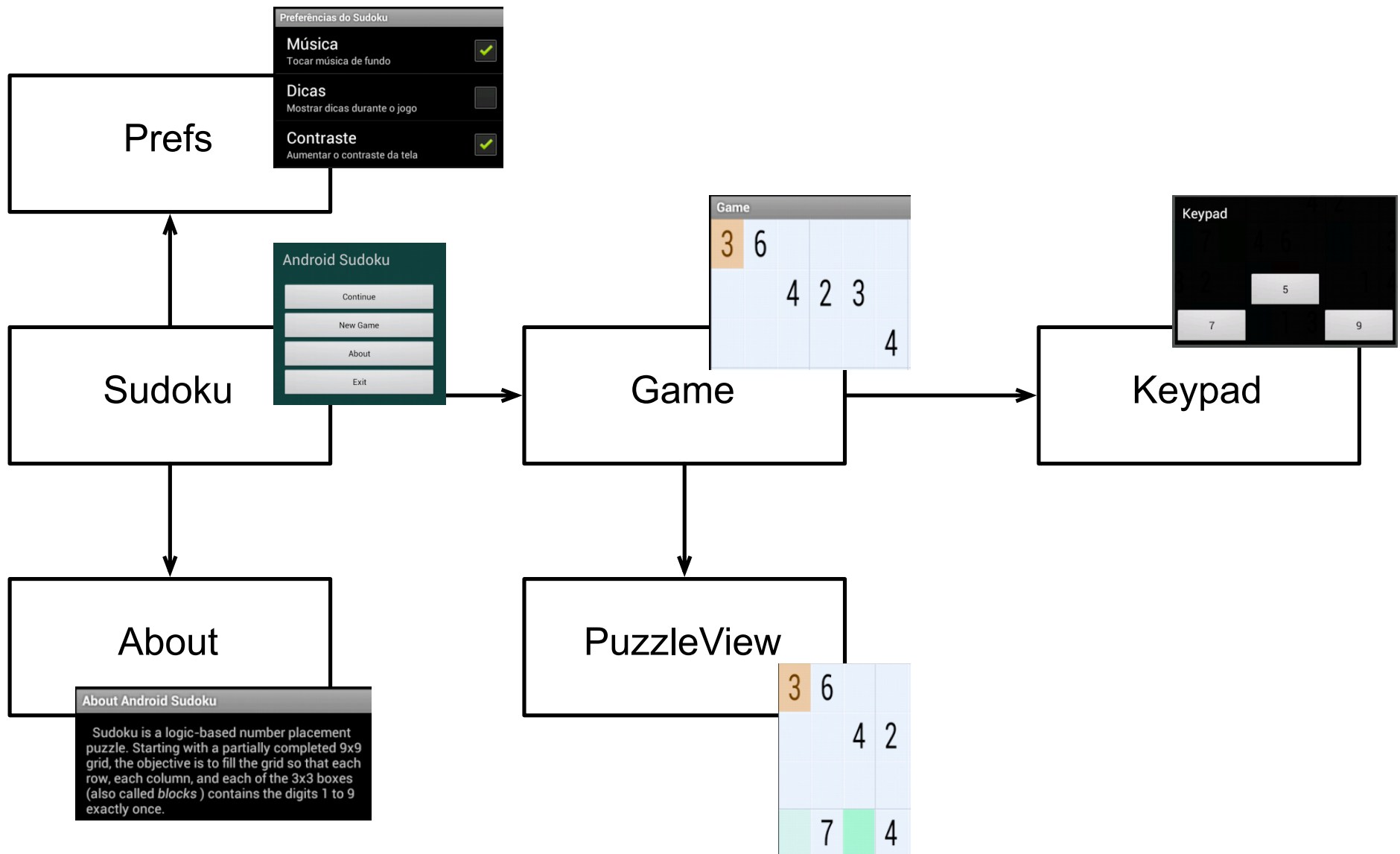
Este exemplo foi retirado do Capítulo 3 do livro  
*Hello, Android (The Pragmatic Programmers)*  
Por Ed Burnette

# Visão Geral do Jogo

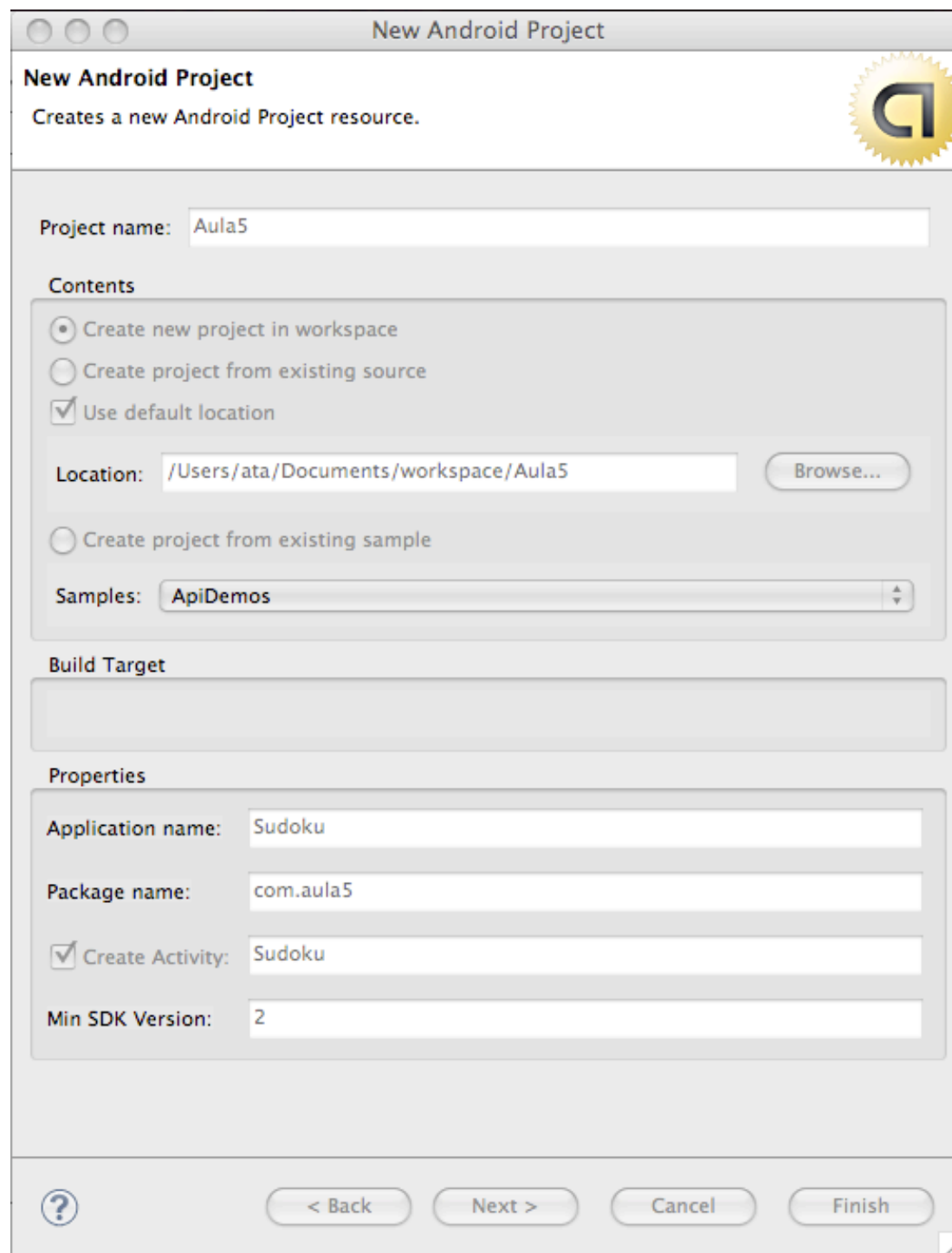


Para que serve  
cada uma dessas  
classes?

# Visão Geral do Jogo



# Criando o projeto



**New Android Project**  
Creates a new Android Project resource.

Project name:

**Contents**

☒ Create new project in workspace  
☐ Create project from existing source  
☒ Use default location

Location:

☐ Create project from existing sample

Samples:

**Build Target**

**Properties**

Application name:

Package name:

☒ Create Activity:

Min SDK Version:

# A Primeira Atividade

```
package com.aula5;

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

public class Sudoku extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

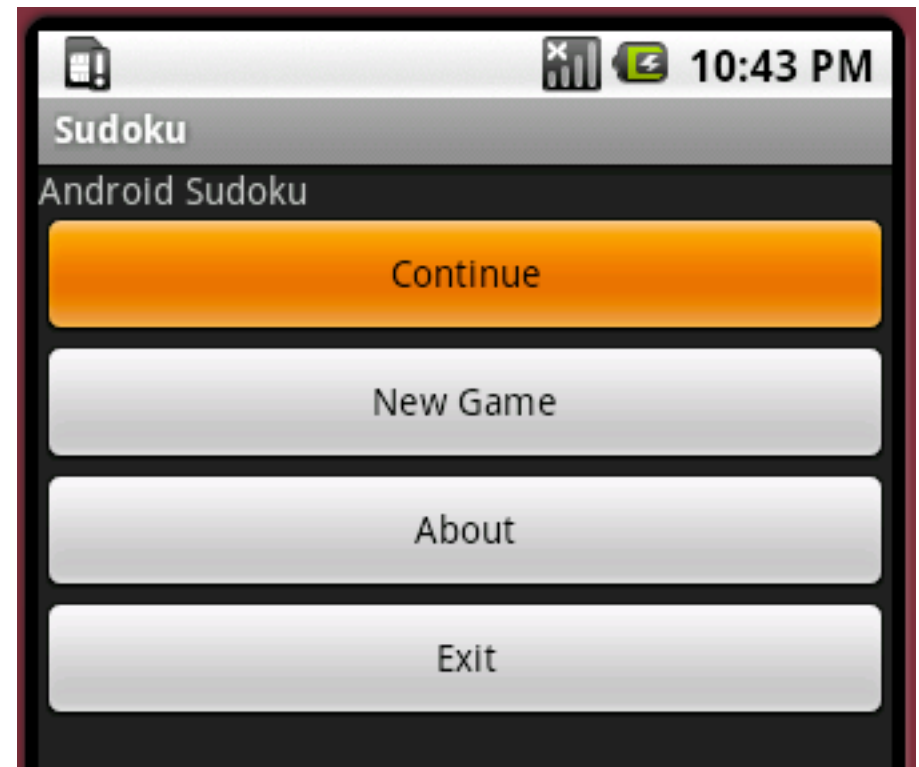
# O Layout Principal

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/main_title" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/continue_label" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/new_game_label" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/about_label" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/exit_label" />
</LinearLayout>
```

Você consegue  
prever como será a  
tela principal de  
nossa aplicação?

# O Layout Principal

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/main_title" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/continue_label" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/new_game_label" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/about_label" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/exit_label" />
</LinearLayout>
```





# O Layout Principal

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/main_title" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/continue_label" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/new_game_label" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/about_label" />
    <Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/exit_label" />
</LinearLayout>
```



# Cores também são recursos

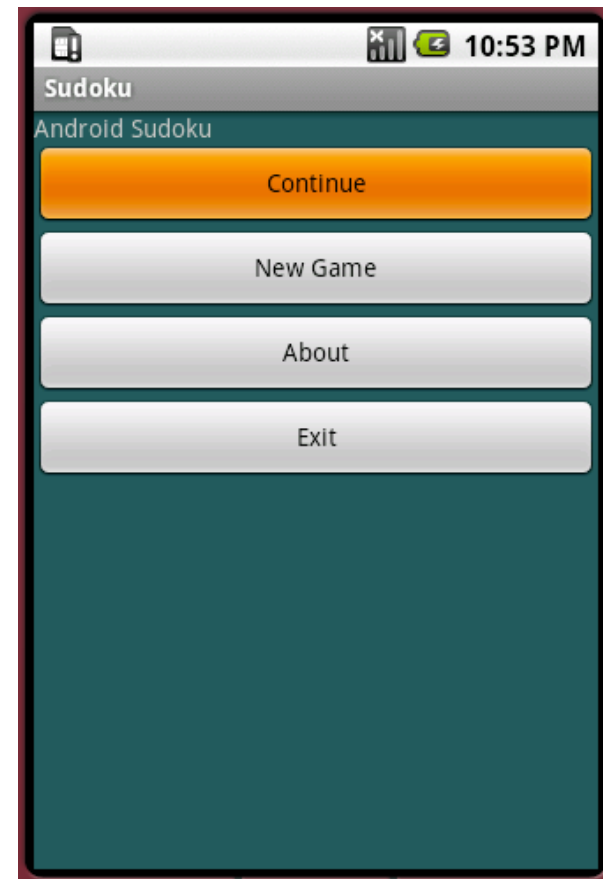
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="background">#3500ffff</color>
</resources>
```

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

# Margens

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

Como adicionar  
uma margem entre  
os botões e o  
perímetro da tela?



# Alinhamento

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
    android:background="@color/background"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="30dip">
    ....
</LinearLayout>
```

O que são  
dip's?

Isto vai dar  
mais  
trabalho...

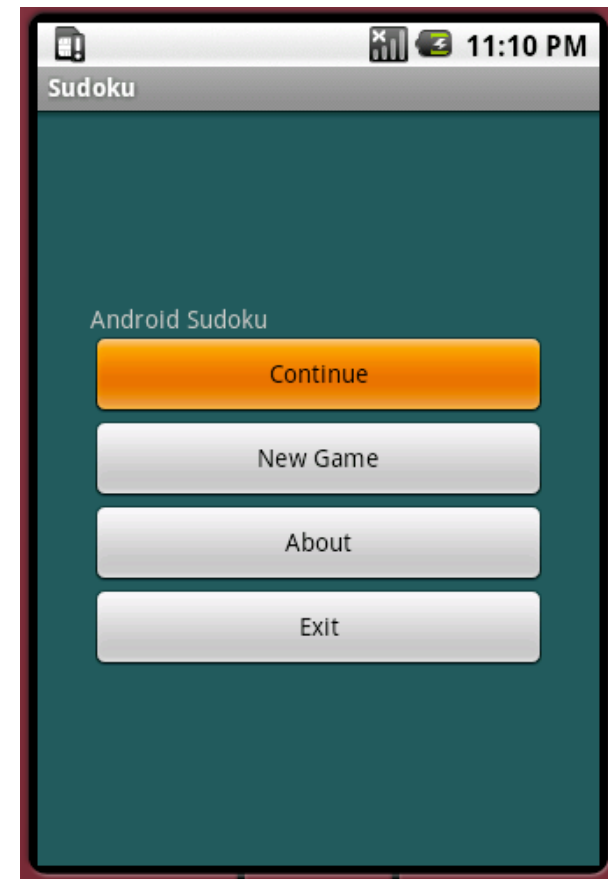
Como centralizar  
os botões  
verticalmente?



# Formatação de Fonte

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background" android:orientation="horizontal"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:padding="30dip">
    <LinearLayout android:orientation="vertical"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_gravity="center">
        <TextView ... />
        <Button ... />
        <Button ... />
        <Button ... />
        <Button ... />
    </LinearLayout>
</LinearLayout>
```

Não dá para  
aumentar o  
tamanho do título  
não?

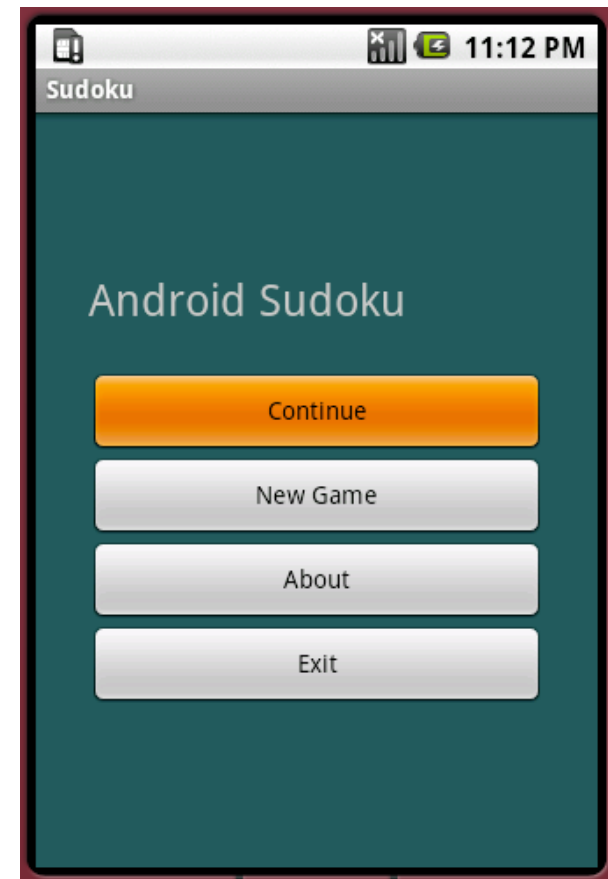


main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout ...>
  <LinearLayout ...>
    <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/main_title"
      android:layout_marginBottom="25dip"
      android:textSize="24.5sp" />
    <Button ... />
    <Button ... />
    <Button ... />
    <Button ... />
  </LinearLayout>
</LinearLayout>
```

O que é  
“sp”?

# Interface Final



# Unidades de Medida

- **px** (pixels): pontos na tela.
- **in** (polegadas): tamanho real (medido por régua).
- **mm** (milímetros): tamanho real.
- **pt** (pontos):  $1/72$  de uma polegada.
- **dp** (*density-independent pixels*): em uma tela com 160 pontos por polegada,  $1dp = 1px$ . Se forem 320 pontos, então  $1dp = 2px$ , etc.
- **sp** (*scale-independent pixels*): mesmo que dp, porém ajustável ao tamanho da fonte.

# Exemplo de Escalas

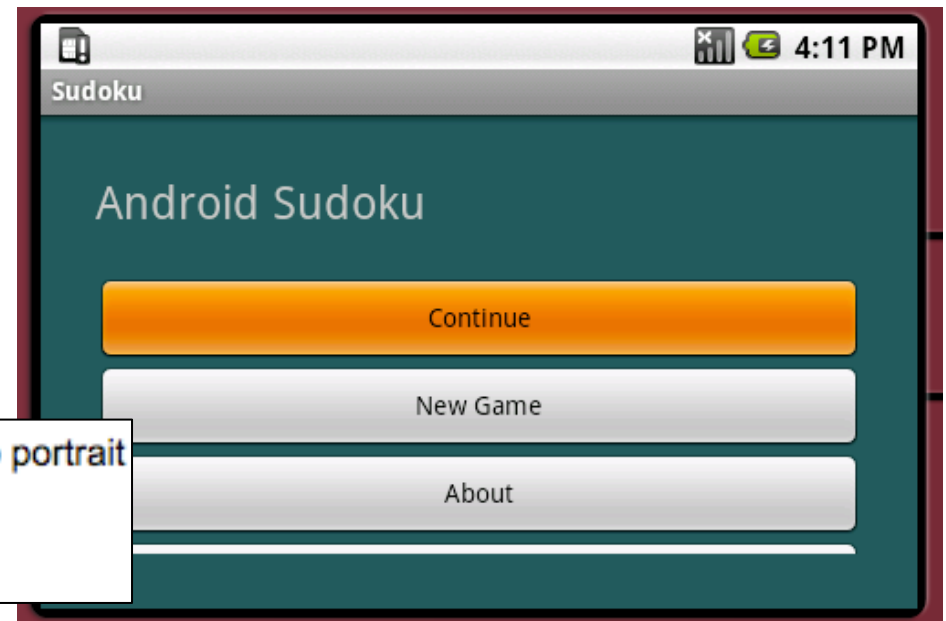
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5sp"
        android:text="160dp"
        android:textSize="22sp" >
    </TextView>
    <View
        android:id="@+id/view1"
        android:layout_width="160dp"
        android:layout_height="20dip"
        android:layout_marginLeft="20sp"
        android:background="#FF22FF22" >
    </View>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5sp"
        android:text="72pt"
        android:textSize="22sp" >
    </TextView>
    <View
        android:id="@+id/view2"
        android:layout_width="72pt"
        android:layout_height="20dip"
        android:layout_marginLeft="20sp"
        android:background="#FF22FF22" >
    </View>
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5sp"
        android:text="1in"
        android:textSize="22sp" >
    </TextView>
    <View
        android:id="@+id/View01"
        android:layout_width="1in"
        android:layout_height="20dip"
        android:layout_marginLeft="20sp"
        android:background="#FF22FF22" >
    </View>
    <TextView
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5sp"
        android:text="160sp"
        android:textSize="22sp" >
    </TextView>
    <View
        android:id="@+id/View04"
        android:layout_width="160sp"
        android:layout_height="20dip"
        android:layout_marginLeft="20sp"
        android:background="#FF22FF22" >
    </View>
    <TextView
        android:id="@+id/TextView02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5sp"
        android:text="160px"
        android:textSize="22sp" >
    </TextView>
    <View
        android:id="@+id/View03"
        android:layout_width="160px"
        android:layout_height="20dip"
        android:layout_marginLeft="20sp"
        android:background="#FF22FF22" >
    </View>
    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5sp"
        android:text="25.4mm"
        android:textSize="22sp" >
    </TextView>
    <View
        android:id="@+id/View02"
        android:layout_width="25.4mm"
        android:layout_height="20dip"
        android:layout_marginLeft="20sp"
        android:background="#FF22FF22" >
    </View>
</LinearLayout>
```





# Layout de Paisagem

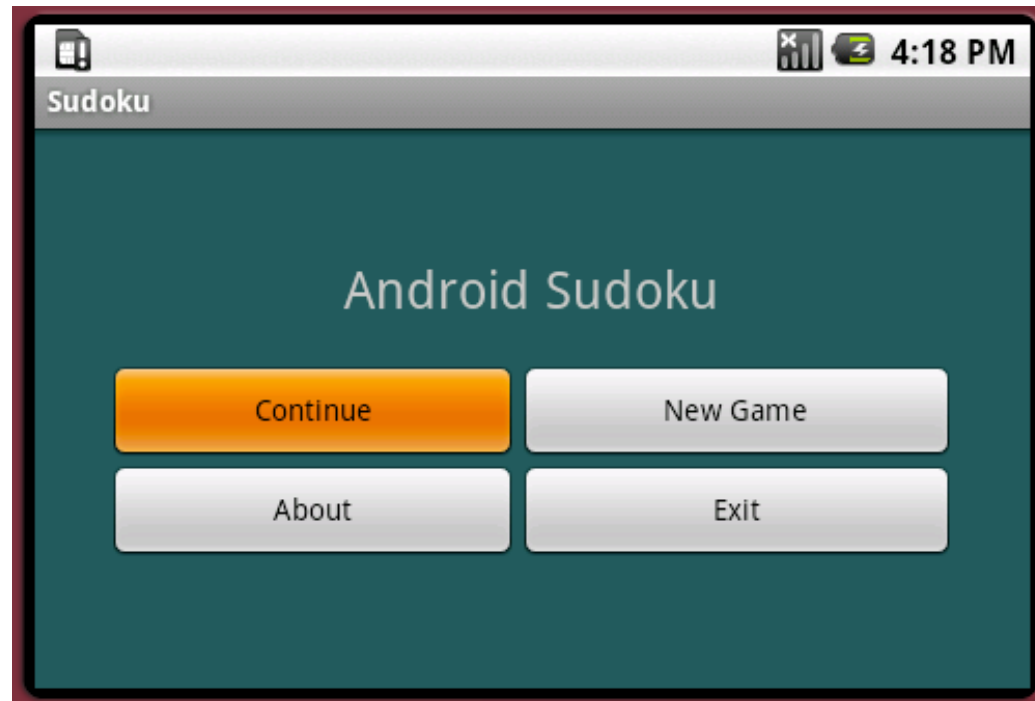
- Muitos smartphones possuem um layout para o modo de paisagem, chamado quando viramos o aparelhos.
- Nosso layout de paisagem ficou assim:
- Qual o problema desse layout?
- E como resolvê-lo?



- `ctrl + fn + F11` on Mac to change the landscape to portrait
- `left-ctrl + F11` on Windows 7.
- `ctrl + F11` on Linux.

# Layout de Paisagem

- É possível definir um layout de paisagem editando o arquivo `res/layout-land/main.xml`.
- Como criar o layout abaixo?



# Tabelas

E se eu  
removesse esse  
pedacinho de  
texto **aqui**?

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:padding="15dip"
    android:orientation="horizontal">
    <LinearLayout
        android:orientation="vertical"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_gravity="center"
        android:paddingLeft="20dip"
        android:paddingRight="20dip">
        <TextView
            android:text="@string/main_title"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_gravity="center"
            android:layout_marginBottom="20dip"
            android:textSize="24.5sp" />
        <TableLayout
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_gravity="center"
            android:stretchColumns="*">
            <TableRow>
                <Button
                    android:id="@+id/continue_button"
                    android:text="@string/continue_label" />
                <Button
                    android:id="@+id/new_button"
                    android:text="@string/new_game_label" />
            </TableRow>
            <TableRow>
                <Button
                    android:id="@+id/about_button"
                    android:text="@string/about_label" />
                <Button
                    android:id="@+id/exit_button"
                    android:text="@string/exit_label" />
            </TableRow>
        </TableLayout>
    </LinearLayout>
</LinearLayout>
```

<TableLayout

```
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_gravity="center"
    android:stretchColumns="*">
```

<TableRow>

<Button

```
    android:id="@+id/continue_button"
    android:text="@string/continue_label" />
```

<Button

```
    android:id="@+id/new_button"
    android:text="@string/new_game_label" />
```

</TableRow>

<TableRow>

<Button

```
    android:id="@+id/about_button"
    android:text="@string/about_label" />
```

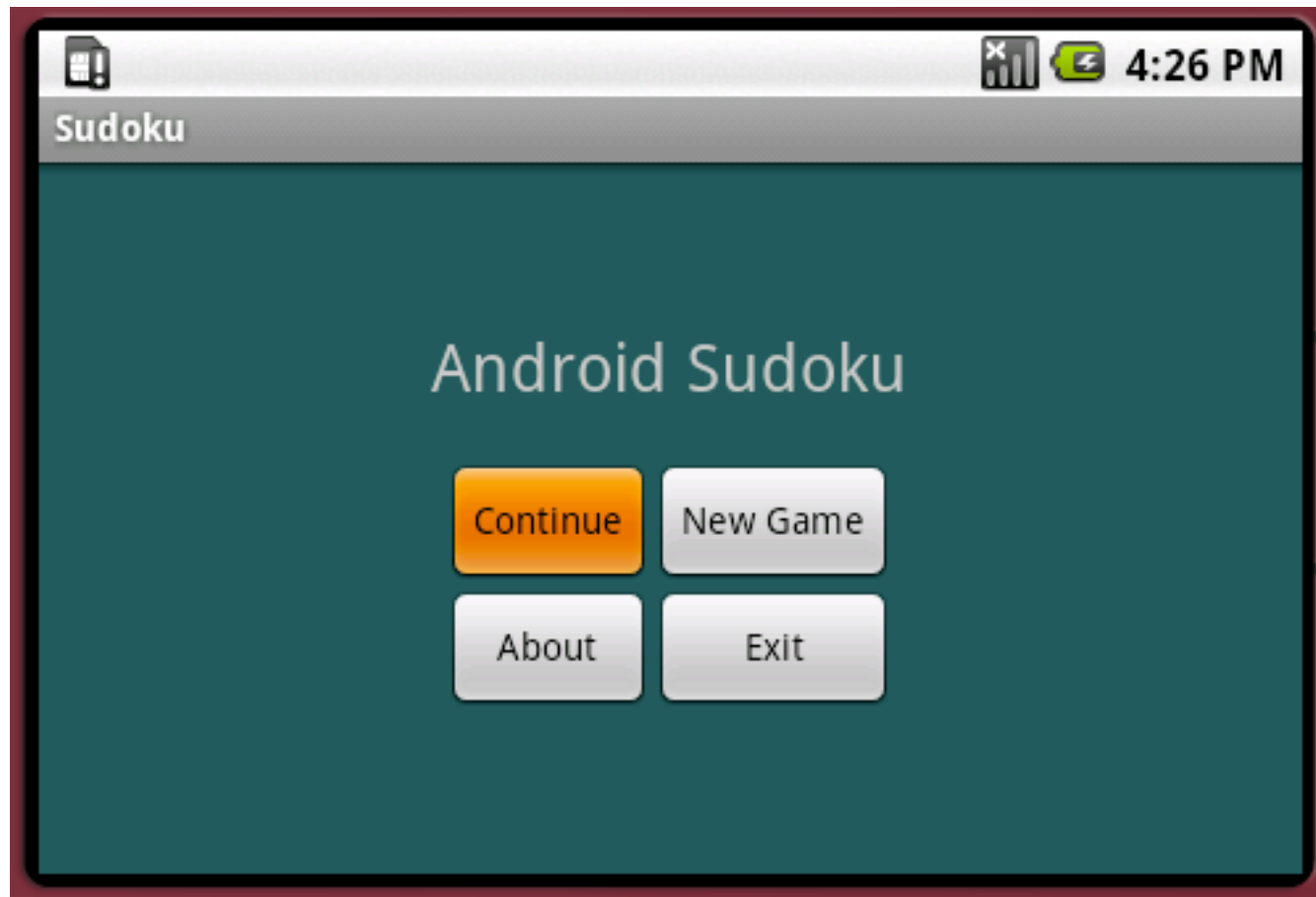
<Button

```
    android:id="@+id/exit_button"
    android:text="@string/exit_label" />
```

</TableRow>

</TableLayout>

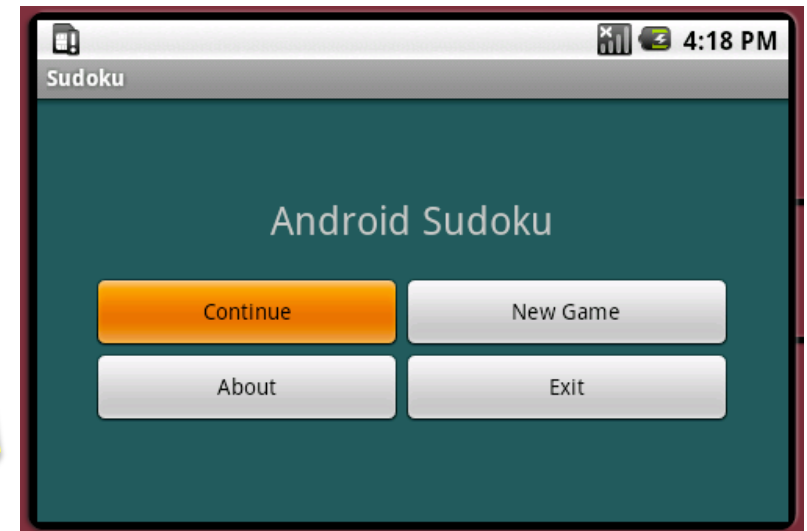
Não fica tão legal...



# Espaçamento

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:padding="15dip"
    android:orientation="horizontal">
    <LinearLayout
        android:orientation="vertical"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_gravity="center"
        android:paddingLeft="20dip"
        android:paddingRight="20dip">
    <TextView
        android:text="@string/main_title"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center"
        android:layout_marginBottom="20dip"
        android:textSize="24.5sp" />
    <TableLayout
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center"
        android:stretchColumns="*">
    <TableRow>
    <Button
        android:id="@+id/continue_button"
        android:text="@string/continue_label" />
    <Button
        android:id="@+id/new_button"
        android:text="@string/new_game_label" />
    </TableRow>
    <TableRow>
    <Button
        android:id="@+id/about_button"
        android:text="@string/about_label" />
    <Button
        android:id="@+id/exit_button"
        android:text="@string/exit_label" />
    </TableRow>
    </TableLayout>
    </LinearLayout>
</LinearLayout>
```

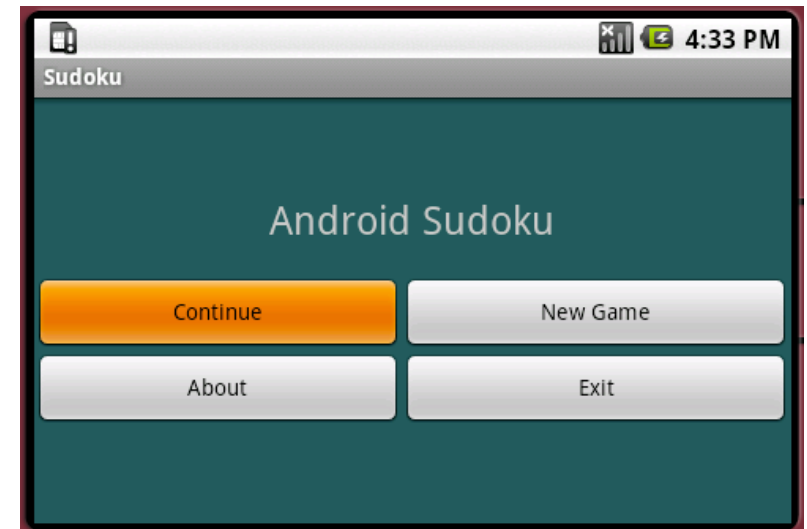
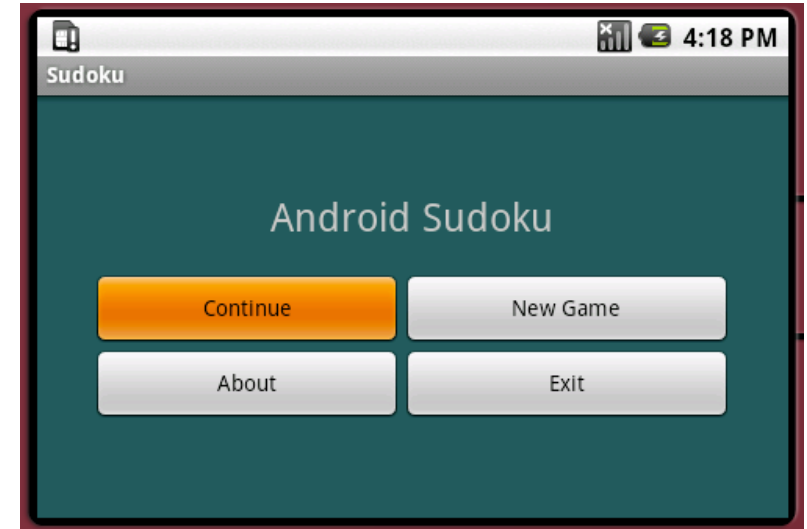
Como ficaria a aplicação se eu removesse os comandos de **espaçamento**?



# Também não ficaria muito legal...

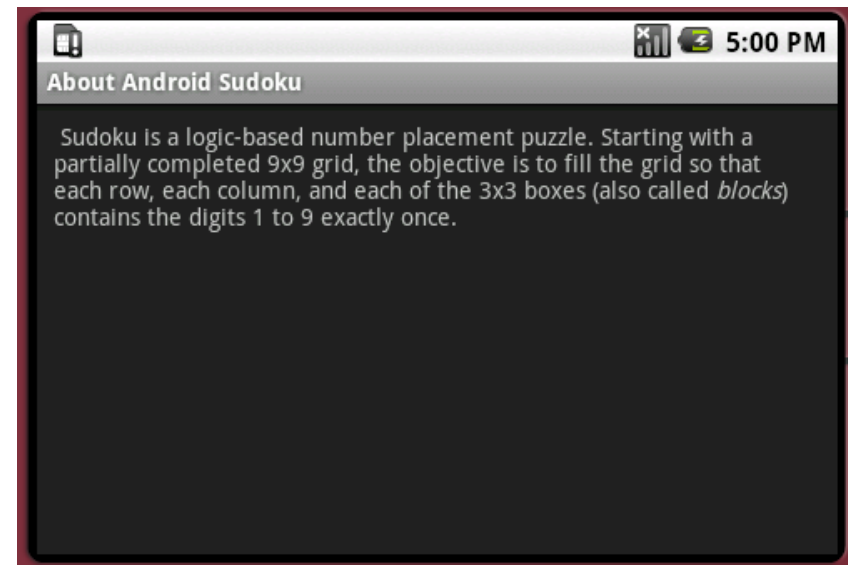
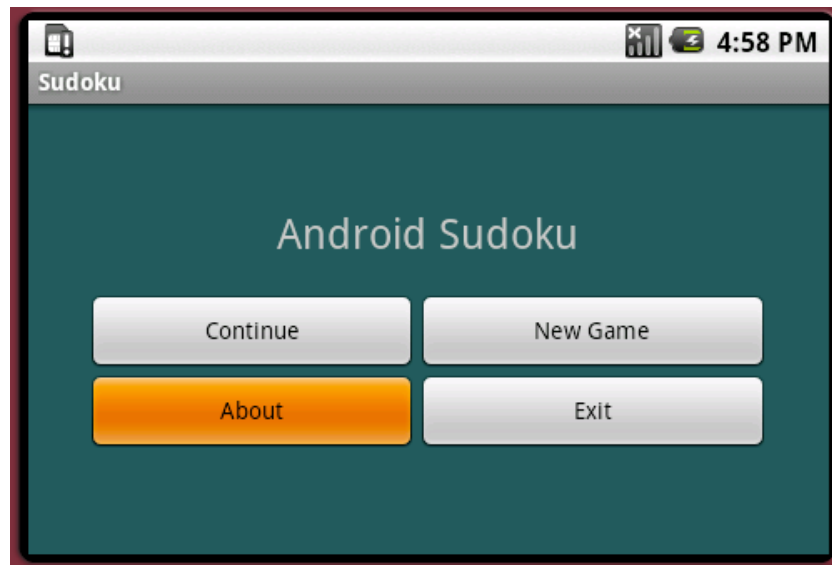
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:orientation="horizontal">
    <LinearLayout
        android:orientation="vertical"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_gravity="center">
        <TextView
            android:text="@string/main_title"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_gravity="center"
            android:layout_marginBottom="20dip"
            android:textSize="24.5sp" />
        <TableLayout
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_gravity="center"
            android:stretchColumns="*">
            <TableRow>
            <Button
                android:id="@+id/continue_button"
                android:text="@string/continue_label" />
            <Button
                android:id="@+id/new_button"
                android:text="@string/new_game_label" />
            </TableRow>
            <TableRow>
            <Button
                android:id="@+id/about_button"
                android:text="@string/about_label" />
            <Button
                android:id="@+id/exit_button"
                android:text="@string/exit_label" />
            </TableRow>
        </TableLayout>
    </LinearLayout>
</LinearLayout>
```

layout-land/main.xml



# Botão *About*

- O que poderíamos fazer com o botão *about*?
  - Outras aplicações possuem uma funcionalidade assim?
  - Como implementá-la?

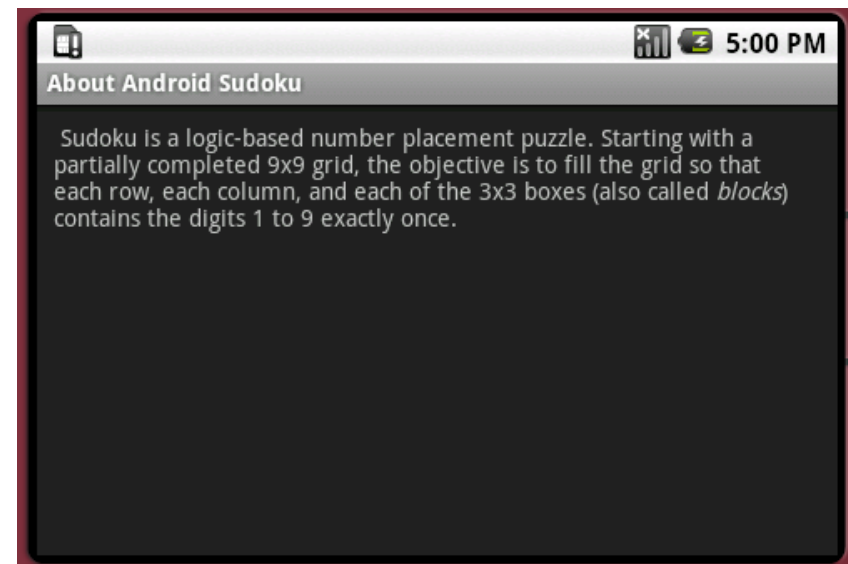


# Activity About

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

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

Como é o código  
do layout de  
nossa atividade?





# Um Layout para About

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dip" >

    <TextView
        android:id="@+id/about_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/about_text" />

</ScrollView>
```

Para que serve  
**esse tipo** de  
layout?

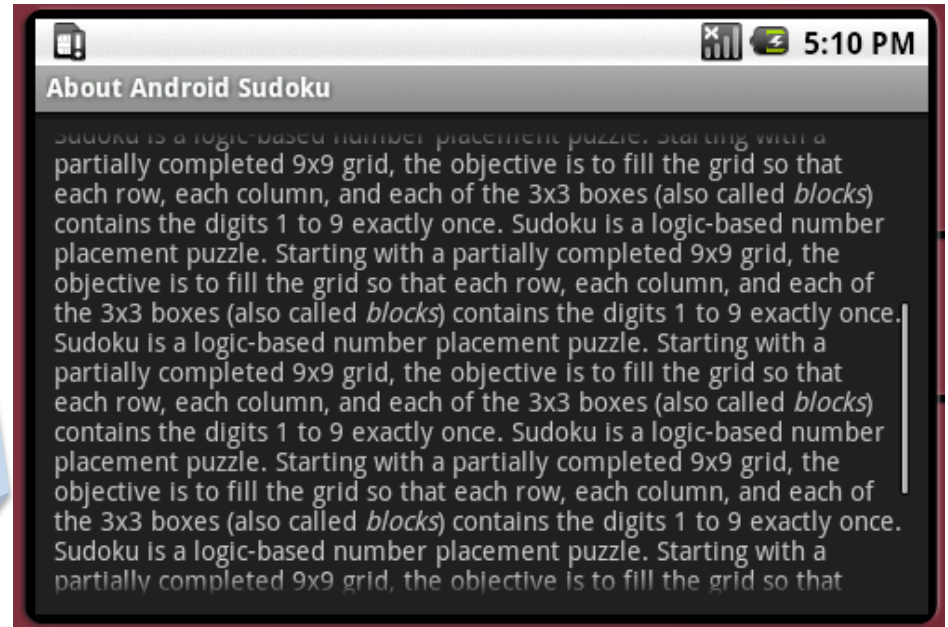
# Um Layout para About

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dip" >

    <TextView
        android:id="@+id/about_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/about_text" />

</ScrollView>
```

Mas onde fica o texto que deixaremos na janela?



# strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">Sudoku</string>
  <string name="main_title">Android Sudoku</string>
  <string name="continue_label">Continue</string>
  <string name="new_game_label">New Game</string>
  <string name="about_label">About</string>
  <string name="exit_label">Exit</string>
  <string name="about_title">About Android Sudoku</string>
  <string name="about_text">\
```

**Sudoku is a logic-based number placement puzzle.**

**Starting with a partially completed 9x9 grid, the objective is to fill the grid so that each row, each column, and each of the 3x3 boxes (also called *blocks*) contains the digits 1 to 9 exactly once.**

```
  </string>
</resources>
```

Certo... mas ainda falta integrar essa nova atividade com o botão About.

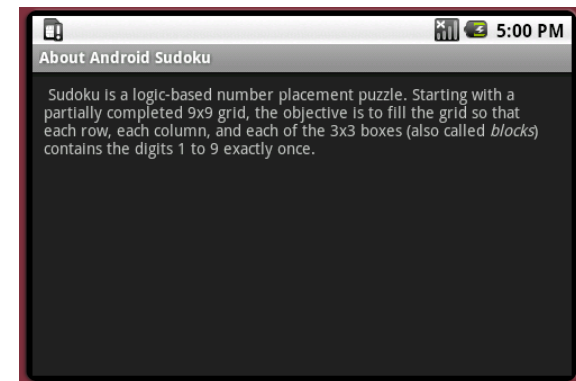
Como a atividade About deve ser invocada?

# O Manifesto

- Para adicionar-se uma nova atividade, é preciso modificar o arquivo `manifest.xml`.

```
<activity android:name=".About" android:label="@string/about_title">  
  <intent-filter>  
    <action android:name="android.intent.action.MAIN" />  
    <category android:name="android.intent.category.LAUNCHER" />  
  </intent-filter>  
</activity>
```

Agora, o que é  
preciso para  
implementar a  
atividade *About*?



# Relacionamento entre Atividades

- Uma atividade pode invocar outra, via objetos chamados Intenções (*Intents*).

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    View aboutButton = findViewById(R.id.about_button);  
    aboutButton.setOnClickListener(this);  
}  
public void onClick(View v) {  
    switch (v.getId()) {  
        case R.id.about_button:  
            Intent i = new Intent(this, About.class);  
            startActivity(i);  
            break;  
    }  
}
```

Pergunta repetida:  
como se dá a  
amarração entre  
botão e evento?

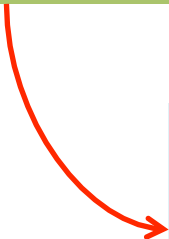
*Lembre-se:*

```
public class Sudoku extends Activity implements OnClickListener
```

# De volta ao Manifesto

- Se a atividade não existir por si mesma, e não quisermos exportá-la para outras aplicações, então podemos simplificar o manifesto:

```
<activity android:name=".About" android:label="@string/app_name">  
  <intent-filter>  
    <action android:name="android.intent.action.MAIN" />  
    <category android:name="android.intent.category.LAUNCHER" />  
  </intent-filter>  
</activity>
```



```
<activity  
  android:name=".About"  
  android:label="@string/about_title">  
</activity>
```

# Caixa de Diálogo

- Seria mais interessante que a atividade About fosse mostrada como uma caixa de diálogo.
- Como podemos modificar nossa aplicação para obtermos essa caixa de diálogo?



# Temas

- Android permite que desenvolvedores criem *Temas*.
- Temas são como arquivos de formato **CSS**.
- Já existem alguns temas prontos.



```
<activity
  android:name=".About"
  android:label="@string/about_title"
  android:theme="@android:style/Theme.Dialog">
</activity>
```





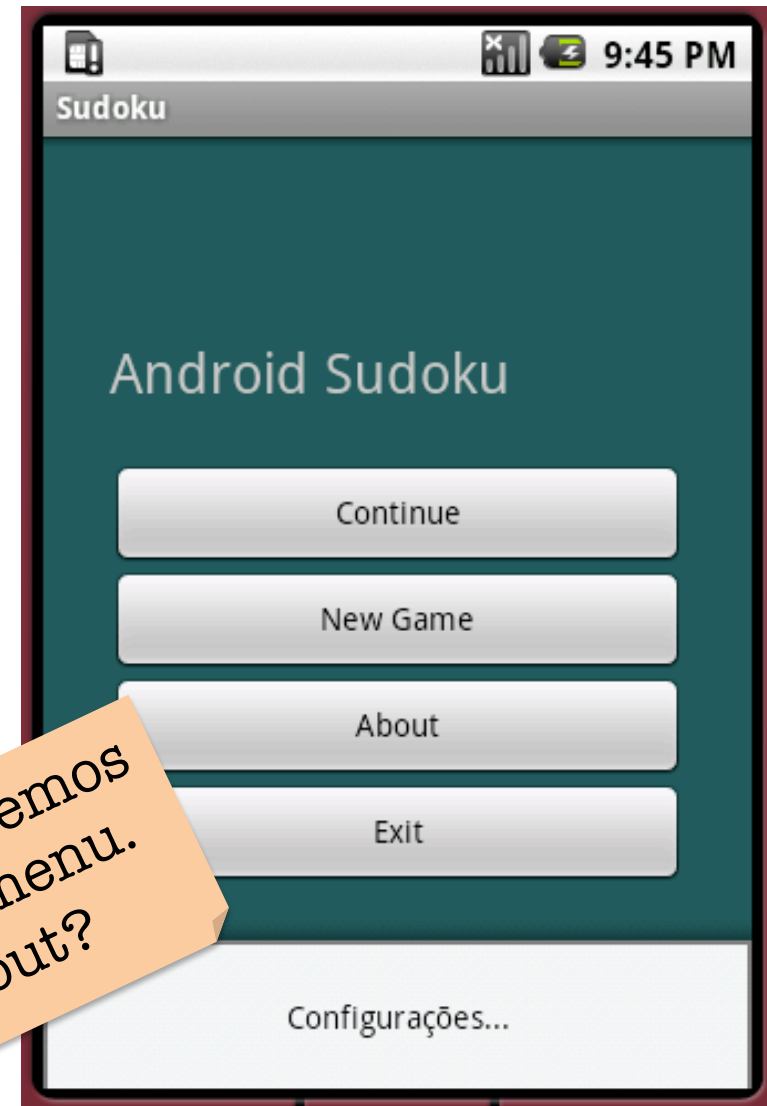
# Adicionando Configurações

## **Menu de preferências:**

Adicione um menu à sua aplicação, de tal forma que o usuário possa configurar o ambiente de jogo.

- O usuário deveria ser capaz de escutar música enquanto brinca.
- E ele poderia pedir dicas para a aplicação.
- Essas opções podem ser configuradas via um menu que dispara a atividade de configuração

Começemos pelo menu.  
Layout?



# string.xml

- Vamos criar algumas strings que nos auxiliarão na construção do menu:

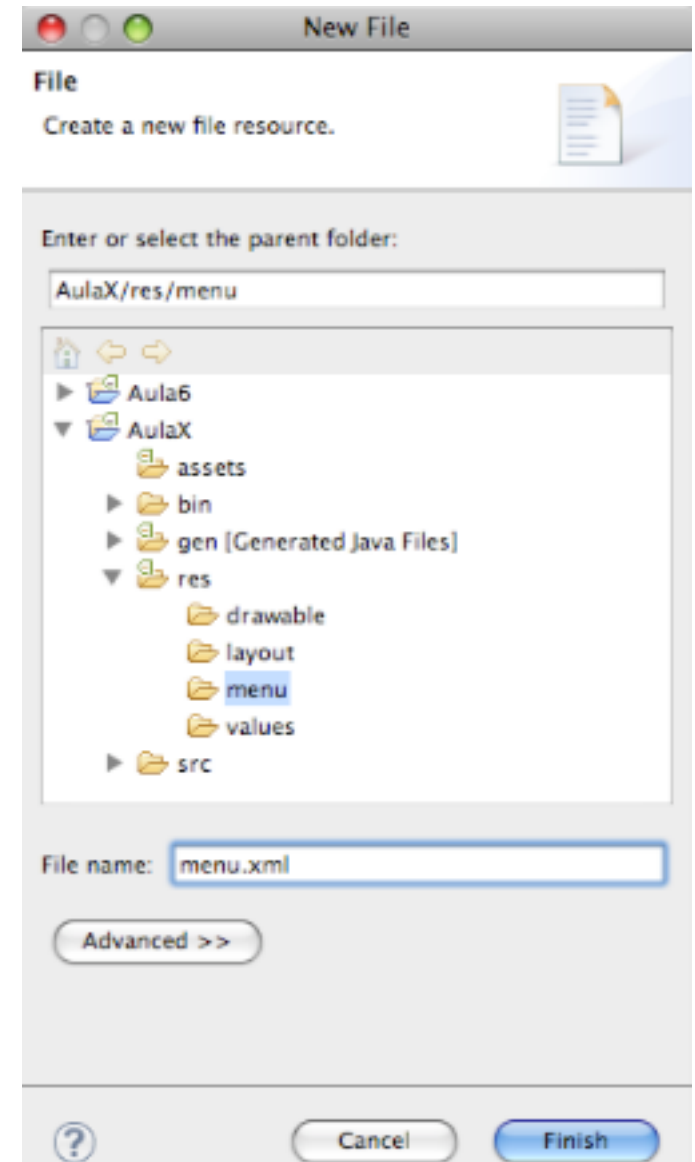
```
<string name="settings_label">Configurações...</string>  
<string name="settings_title">Preferências do Sudoku</string>  
<string name="settings_shortcut">s</string>  
<string name="music_title">Música</string>  
<string name="music_summary">Tocar música de fundo</string>  
<string name="hints_title">Dicas</string>  
<string name="hints_summary">Mostrar dicas durante o jogo</string>
```

Novamente: por que criar strings em strings.xml? Quais strings devem ficar lá?

Tudo bem, mas vocês lembram como inserir o menu em nossa aplicação?

# O Layout do Menu

- O ambiente de desenvolvimento Android define uma pasta menu, em que é definido o layout do menu da aplicação.

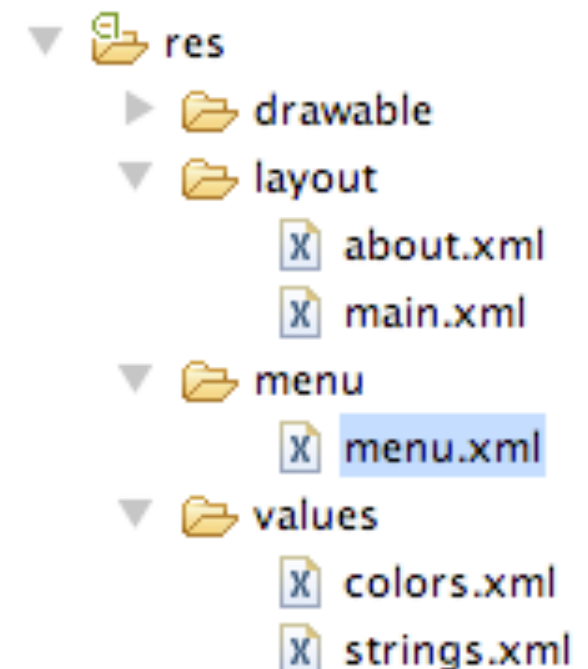


# O Layout do Menu

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item
  android:id="@+id/settings"
  android:title="@string/settings_label"
  android:alphabeticShortcut="@string/settings_shortcut" />
</menu>
```

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android=
"http://schemas.android.com/apk/res/android">
<item
  android:id="@+id/settings"
  android:title="Configurações"
  android:alphabeticShortcut="s" />
</menu>
```

Será que  
**Isso** daria  
certo?



# Ativando o Menu

@Override

```
public boolean onCreateOptionsMenu(Menu menu) {  
    super.onCreateOptionsMenu(menu);  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.menu, menu);  
    return true;  
}
```

@Override

```
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.settings:  
            startActivity(new Intent(this, Prefs.class));  
            return true;  
        }  
        return false;  
    }
```

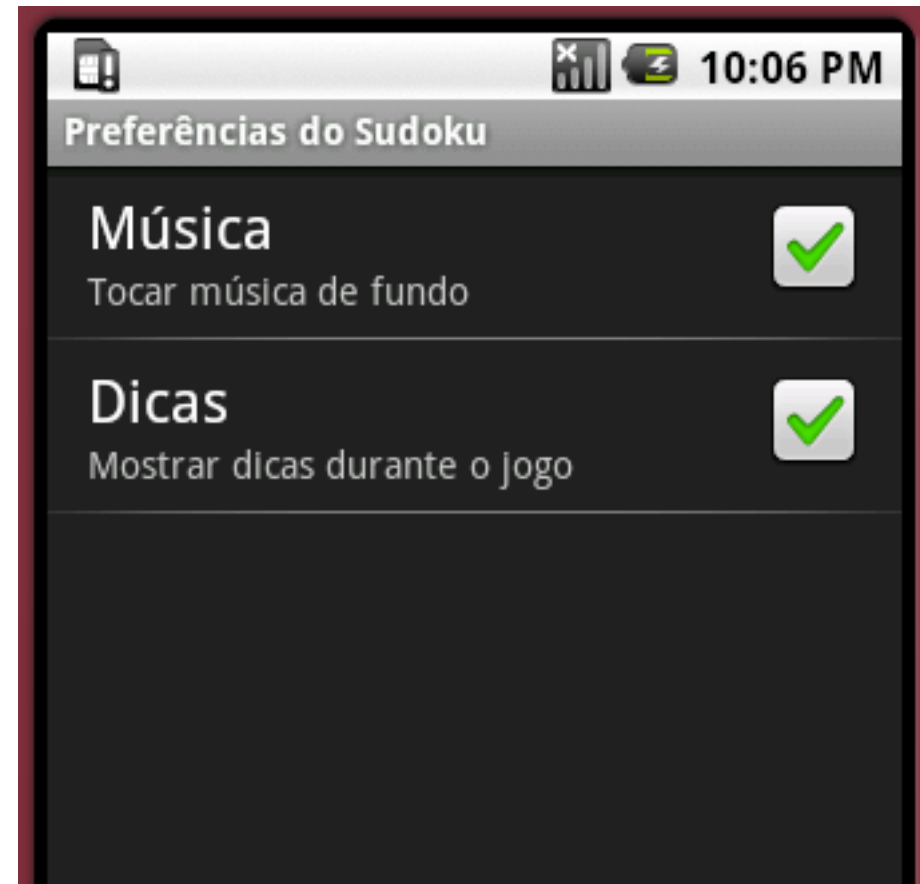
Como deveria  
ser o layout de  
**Prefs**?

E como deveria ser  
essa atividade **Prefs**,  
que modifica as  
preferências do  
usuário?

# A Caixa de Preferências

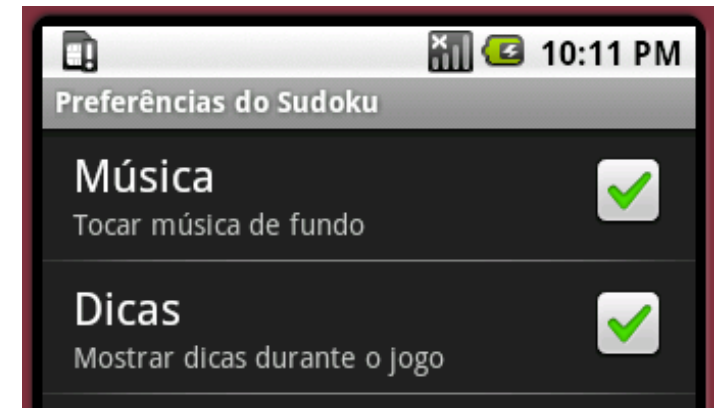
- Podemos usar botões de verificação!

Mas como  
implementar  
esse layout?



# Caixa de Preferências

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
  <CheckBoxPreference
    android:key="music"
    android:title="@string/music_title"
    android:summary="@string/music_summary"
    android:defaultValue="true" />
  <CheckBoxPreference
    android:key="hints"
    android:title="@string/hints_title"
    android:summary="@string/hints_summary"
    android:defaultValue="true" />
</PreferenceScreen>
```



E a implemen-  
tação da ativi-  
dade em si?

# A Atividade Prefs

```
import android.os.Bundle;
import android.preference.PreferenceActivity;

public class Prefs extends PreferenceActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.layout.settings);
    }
}
```

O que é **esse** método aqui?

E o que está faltando para usarmos a atividade?



# Adicionar a Atividade ao Manifesto

```
<activity  
    android:name=".Prefs"  
    android:label="@string/settings_title">  
</activity>
```

## **Níveis de Dificuldade:**

Quando o usuário clicar em “new game”, ele deverá receber um menu de contexto, com três opções de dificuldade: fácil, médio e difícil.

Seria legal um pouco de música não?  
Bom, isso fica para um próximo capítulo.

Por hora, vamos nos preocupar com o botão “new game”

Por onde começamos?

# Registrando um Evento

- Precisamos de um identificador para o botão que começará o jogo:

main.xml

```
<Button android:id="@+id/new_button" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="@string/new_game_label" />
```

- E precisamos de um escutador de eventos para o novo botão:

Sudoku.java

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ...
    View newButton = findViewById(R.id.new_button);
    newButton.setOnClickListener(this);
}
```

# Níveis de Dificuldade

```
public void onClick(View v) {  
    switch (v.getId()) {  
        ...  
        case R.id.new_button:  
            openNewGameDialog();  
            break;  
    }  
}  
  
private void openNewGameDialog() {  
    new AlertDialog.Builder(this)  
        .setTitle(R.string.new_game_title).setItems(  
            R.array.difficulty, new DialogInterface.OnClickListener() {  
                public void onClick(DialogInterface dialoginterface, int i) {  
                    startGame(i);  
                }  
            }).show();  
}  
  
private void startGame(int i) {  
    Log.d(TAG, "clicked on " + i);  
}
```

Sudoku.java



Existem alguns **recursos** que precisamos definir.

E há também uma porção de coisas **misteriosas**.

# Mais Strings...

strings.xml

```
<string name="new_game_title">Dificuldade</string>
<string name="easy_label">Fácil</string>
<string name="medium_label">Médio</string>
<string name="hard_label">Difícil</string>
```

## E um arranjo:

values/array.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <array name="difficulty">
    <item>@string/easy_label</item>
    <item>@string/medium_label</item>
    <item>@string/hard_label</item>
  </array>
</resources>
```

# Console de Saída

Sudoku.java

```
public void onClick(View v) {  
    switch (v.getId()) {  
        ...  
        case R.id.new_button:  
            openNewGameDialog();  
            break;  
    }  
}  
  
private void openNewGameDialog() {  
    new AlertDialog.Builder(this)  
        .setTitle(R.string.new_game_title).setItems(  
            R.array.difficulty, new DialogInterface.OnClickListener() {  
                public void onClick(DialogInterface dialoginterface, int i) {  
                    startGame(i);  
                }  
            }).show();  
}  
  
private void startGame(int i) {  
    Log.d("Sudoku", "clicked on " + i);  
}
```

O que acontece  
quando alguém  
clica no botão  
“new game”?

# Console de Saída

```
public void onClick(View v) {
    switch (v.getId()) {
        ...
        case R.id.new_button:
            openNewGameDialog();
            break;
    }
}

private void openNewGameDialog() {
    new AlertDialog.Builder(this)
        .setTitle(R.string.new_game_title).setItems(
            R.array.difficulty, new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialoginterface, int i) {
                    startGame(i);
                }
            })
        .show();
}

private void startGame(int i) {
    Log.d("Sudoku", "clicked on " + i);
}
```

```
09-25 10:38:38.D 690 Sudoku clicked on 1
09-25 10:38:38.W 560 InputManagerSer Window already
09-25 10:38:42.D 605 dalvikvm GC freed 221 ol
09-25 10:38:49.D 690 Sudoku clicked on 0
09-25 10:38:49.W 560 InputManagerSer Window already
09-25 10:38:53.D 690 Sudoku clicked on 2
```

# Ainda o Manifesto

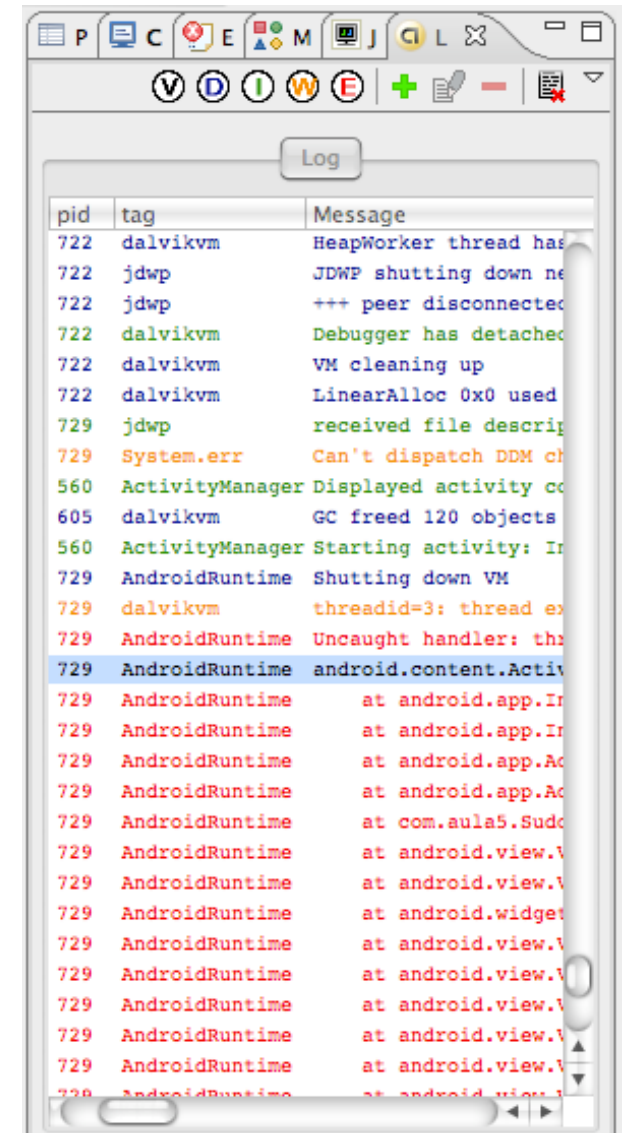
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.aula5" android:versionCode="1" android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Sudoku" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".About" android:label="@string/about_title"
            android:theme="@android:style/Theme.Dialog">
        </activity>
        <activity android:name=".Prefs" android:label="@string/settings_title">
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="2" />
</manifest>
```

O que aconteceria se apagássemos a declaração da atividade no manifesto?

# Usando o Console

- Observaríamos a seguinte mensagem no console:

```
android.content.ActivityNotFoundException: Unable to find explicit activity class {com.aula5/com.aula5.About}; have you declared this activity in your AndroidManifest.xml?
```





# Padrão de Projetos Builder

```
public void onClick(View v) {  
    switch (v.getId()) {  
        ...  
        case R.id.new_button:  
            openNewGameDialog();  
            break;  
    }  
}  
  
private void openNewGameDialog() {  
    new AlertDialog.Builder(this)  
        .setTitle(R.string.new_game_title).setItems(  
            R.array.difficulty, new DialogInterface.OnClickListener() {  
                public void onClick(DialogInterface dialoginterface, int i) {  
                    startGame(i);  
                }  
            }).show();  
}  
  
private void startGame(int i) {  
    Log.d(TAG, "clicked on " + i);  
}
```

O que é o  
padrão de  
projetos  
Builder?



# Padrão de Projetos Builder

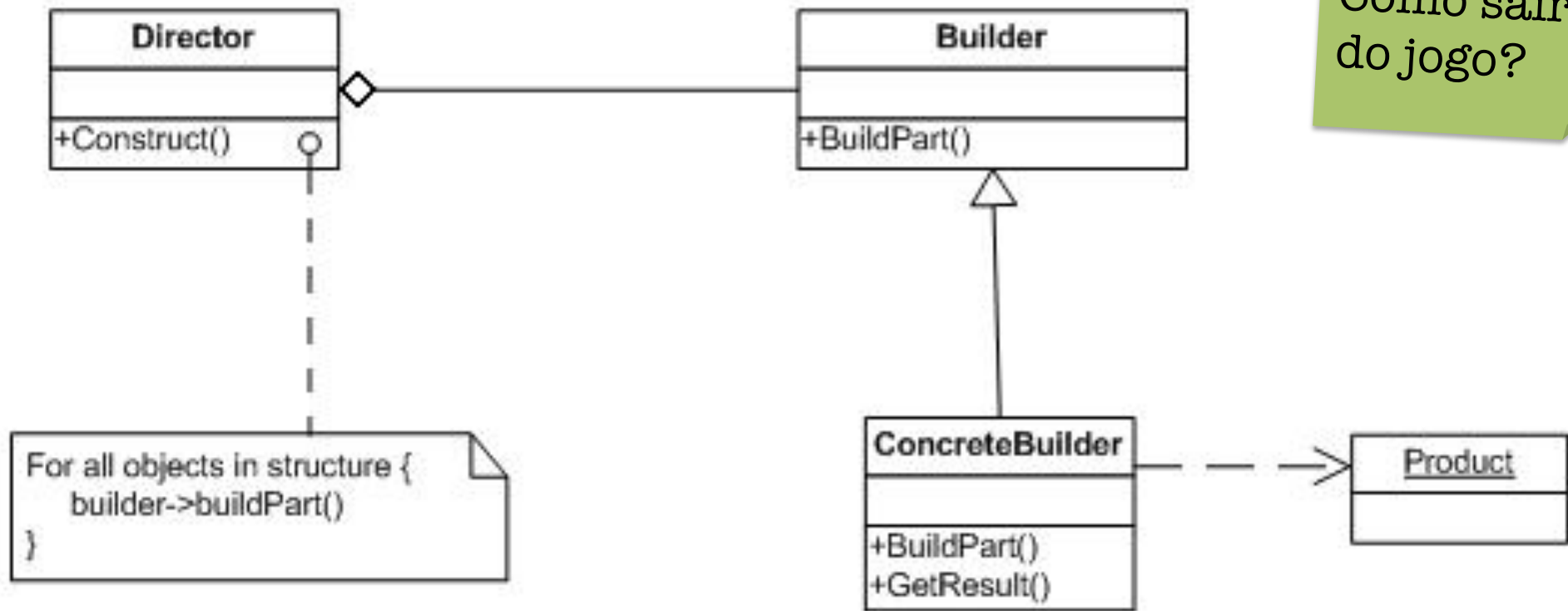
- Builder é uma forma de separar o código de construção do objeto da lógica do programa.
  - Alguns objetos são formados por vários componentes, e a única diferença entre eles é a forma como esses componentes são arranjados.

```
new AlertDialog.Builder(this)
    .setTitle(R.string.new_game_title).setItems(
        R.array.difficulty, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialoginterface, int i) {
                startGame(i);
            }
        })
```



# Padrão de Projetos Builder

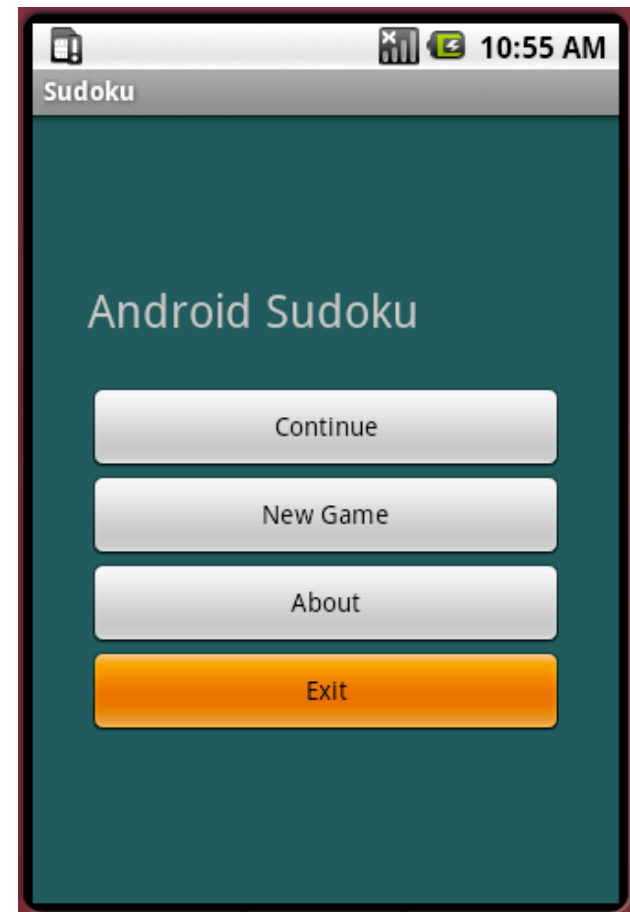
Como sair  
do jogo?



# Terminando o Jogo

- Implemente o tratador de eventos que seria ativado caso clicássemos sobre o botão **Exit**.

Na verdade, esse botão não é muito necessário. Porque?



# Terminando o Jogo

```
public void onClick(View v) {  
    switch (v.getId()) {  
        case R.id.about_button:  
            Intent i = new Intent(this, About.class);  
            startActivity(i);  
            break;  
        case R.id.new_button:  
            openNewGameDialog();  
            break;  
        case R.id.exit_button:  
            finish();  
            break;  
    }  
}
```

E tendo  
terminado a  
atividade, como  
podemos começa-  
la novamente?

# Carregando Atividades

- Todas as atividades testadas estão disponíveis em modo de emulação:

