



**METODOLOGIA**

# **DESENVOLVEDOR DE APLICATIVOS ANDROID**

## Sumário

<b>Aula 1 - Introdução e configuração de ambiente de desenvolvimento .....</b>	<b>4</b>
Open handset Alliance.....	5
Android .....	5
Arquitetura .....	6
Applications .....	6
Montando o Ambiente.....	8
Instalação do SDK do Java.....	9
Exercício de Fixação.....	24
<b>Aula 2 – Configurando o Eclipse .....</b>	<b>29</b>
Entendendo o projeto Android .....	31
Executando o arquivo .....	32
Adicionando recursos no xml.....	35
Exercício de Fixação.....	41
<b>Aula 3 – Trabalhando com eventos .....</b>	<b>46</b>
Adicionando as alterações.....	48
Associando o botão ao evento.....	50
Exercício de Fixação.....	52
<b>Aula 4 – Criando uma nova tela .....</b>	<b>47</b>
Criando um novo arquivo de layout .....	57
Criando uma Classe Java.....	59
Executando o projeto visualizando a nova tela.....	62
Voltando para a tela anterior.....	62
Ciclo de vida das Activities .....	65
Entendendo o ciclo de vida.....	66
Passando dados entre as telas.....	67
Exercício de Fixação.....	70
<b>Aula 5.....</b>	<b>75</b>
Layouts .....	75
Adicionando linhas ao Layout .....	77
Criando a Activity .....	77
Estilos de layout.....	79
Exercício de Fixação.....	82

<b>Aula 6.....</b>	<b>87</b>
Usando Widgets para a seleção.....	87
Exemplos de ListView.....	87
CheckBox .....	90
RadioButton .....	92
Spinner .....	95
Recurso Color.....	98
Adicionando um recurso de cor .....	99
Exercício de Fixação.....	101
<b>Aula 7.....</b>	<b>106</b>
Tipos de Recursos.....	106
Menus e submenus.....	107
Recursos de menu.....	107
Preparando o exemplo.....	109
Carregando o menu.....	110
Exercício de Fixação.....	119
<b>Aula 8 – Trabalhando com Intent .....</b>	<b>124</b>
O que é Intent? .....	124
Intent Filter.....	125
Realizando uma chamada .....	132
Exercício de Fixação.....	142
<b>Aula 9 – BroadcastReceiver .....</b>	<b>131</b>
Exercício de Fixação.....	162

***Introdução e configuração de ambiente de desenvolvimento***

Olá, seja bem-vindo à primeira aula do curso para desenvolvedor de Android, neste curso você irá aprender a criar aplicativos para dispositivos móveis Android, estes que são escritos em linguagem Java, para criar esses aplicativos você isso irá aprender a trabalhar com os programas do **Kit de desenvolvimento para Android e Eclipse**.



Cada vez mais a quantidade de smartphones vem aumentando, hoje em dia os smartphones são o produto de consumo mais utilizado no mundo, atualmente existem o dobro de celulares com acesso à internet em comparação com o número de computadores.

Isso tende a aumentar de acordo com que os países em desenvolvimento renovam seus aparelhos por modelos mais recentes.



Em um mercado tão promissor fica inevitável que inovações nesta área aconteçam. Tendo em vista as inovações nesta área, criou-se uma parceria entre várias empresas do ramo de telefonia e tecnologia, com o apoio do Google e com isso originou-se a **Open Handset Alliance**.

### **Open handset Alliance**

Esta empresa tem como líder dos projetos o Google, a intenção inicial era construir um sistema operacional de código aberto, onde todos os aparelhos com este software terão os mesmos privilégios que os aplicativos criados e/ou licenciados pelo fabricante. Assim todos os aparelhos poderão ser integrados e customizados através dos sistemas e aparelhos. O primeiro software que surgiu deste conjunto de empresas é o nosso protagonista deste curso o **Android**.

### **Android**

O Android, um sistema operacional móvel completo e aberto foi desenvolvido com base no sistema operacional Linux, foi desenvolvido com a intenção de permitir aos desenvolvedores criar aplicações móveis que possam tirar total proveito do que um aparelho móvel pode oferecer, como efetuar chamadas, enviar mensagens e até mesmo ativar câmera.

Este sistema pode ser adaptado com pretensões de incorporar as novas tecnologias a medida que forem surgindo.

O sistema é muito seguro, isto é, ele impede que alguém crie um software malicioso e acesse informações pessoais no seu dispositivo, além disso toda vez que este sistema for instalado ele cria um novo usuário para este, deixando o aplicativo isolado dos outros e qualquer tentativa de acessar será necessária a autorização do usuário podendo negar a autorização.

Para você desenvolver aplicações para Android utiliza-se o Kit de desenvolvimento SDK, este que disponibiliza as ferramentas e APIs\* necessárias utilizando a linguagem Java.

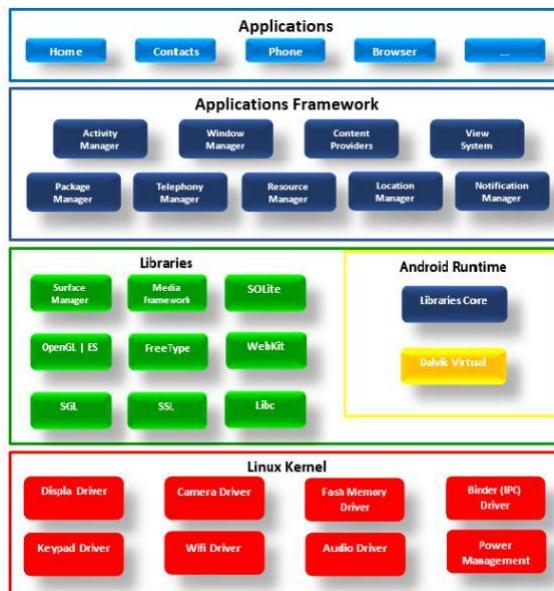
\*API é o acrônimo de *Application Programming Interface* ou, em português, Interface de Programação de Aplicativos.

## Arquitetura

Na imagem abaixo você pode analisar a estrutura arquitetônica de um Android.

Ela é organizada em cinco camadas, são elas:

- Applications**
- Applications Framework**
- Libraries / Android Runtime**
- Linux Kernel**



## Applications

Nesta camada se encontram todos os aplicativos fundamentais escritos em Java como: um cliente de e-mail, mapas, navegadores, calendários, programas SMS, entre outros.

Ou seja, para desenvolver um programa para a plataforma androide, os aplicativos devem ser escritos em Java para serem executados na máquina virtual Dalvik.

## Applications Framework

Nesta camada estão todas as APIs e recursos utilizados pelos aplicativos, como: Classes visuais que incluem listas, grades, caixas de texto, botões, e até um navegador web embutido, **View system** este que é um componente utilizado na construção de aplicativos e o provedor de conteúdo, **Content Provider** que possibilita um aplicativo a acessar ou compartilhar as informações com outros aplicativos e gerenciadores de localizações como GPS e Cell ID, entre outros.

## Libraries

Nesta camada podemos encontrar um conjunto de bibliotecas C/C++, estas que são utilizadas pelo sistema, neste conjunto encontramos a biblioteca C padrão (Libc)

também as bibliotecas referentes a multimídia e visualização de camadas 2D e 3D, funções para navegadores web, funções gráficas, de aceleração de hardware, renderização 3D, fontes em bitmap e vetorizadas, outras funções de acesso a banco de dados SQLite etc.

### Android Runtime

Esta pequena camada de execução é uma instância da máquina virtual Dalvik, criada para cada aplicação executada no Android. Esta Dalvik nada mais é do que uma máquina virtual com melhor desempenho e integração com a nova geração de software e projeção para funcionar com sistemas com baixa frequência de otimização para o consumo mínimo de memória, bateria e CPU.

### Linux Kernel

Atualmente ela utiliza a versão 2.6 do Kernel do Linux para os serviços centrais do sistema, como segurança, memória, processos, pilha de protocolos de rede e modelo de drives. O Kernel também atua como uma camada de abstração entre o hardware e o resto da pilha de software, um dos acessórios interessantes é o **Binder** (IPC) este que é responsável pela obtenção e envio das aplicações requeridas a interface de serviço.

Nesta camada se encontra também um importante sistema próprio de gerenciamento de energia do Kernel a checar periodicamente todos os dispositivos que não estão sendo utilizados por aplicações e os desliga.

### Play Google

A Play Google é uma loja online para estes aplicativos Android, desenvolvida pelo Google, disponível no Link:

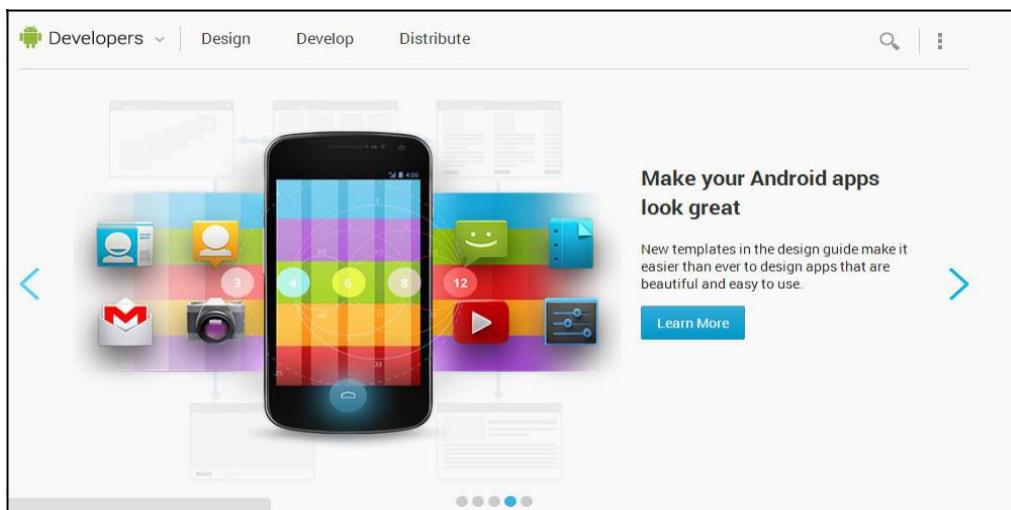
<https://play.google.com/store/apps>



Nesta loja, encontramos tanto aplicativos pagos quanto grátis.

### Android Developers

O site Android Developers <http://developer.android.com> contém todas as informações para oferecer suporte e auxiliar o desenvolver Android.



Neste site você pode encontrar tutoriais, recursos e as últimas novidades do Android.

### **Montando o Ambiente**

Como já explicamos anteriormente, para desenvolver aplicações Android será utilizada a linguagem Java. Qualquer editor pode ser utilizado, inclusive o bloco de notas, compilando na linha de comando do SO, para desenvolver as aplicações neste curso utilizaremos o **Eclipse**, este que é o IDE recomendado pelo Google.

Além do Eclipse conter diversos recursos, ele contém um plugin específico o desenvolvimento de Android que amplia o campo de produtividade no desenvolvimento.

Porém o Google recomenda os seguintes requisitos de software e hardware:

#### *Sistemas operacionais*

- **Windows XP ou sucessores**
- **Mac OS X 10.5.8 ou sucessores**
- **Linux**

#### *Softwares*

- **JDK5 ou JDK6**
- **Eclipse 3.x**
- **Android Developer Tools – Plugin**

#### *Hardware*

- **Espaço livre em disco de 1 GB aproximadamente**
- **Memória mínima de 1 GB**

Para configurar o ambiente siga as instruções.

1º Instalar o SDK do java.

2º Instalar o Eclipse

3º instalar o SDK do Android

4º Instalar a plataforma Android

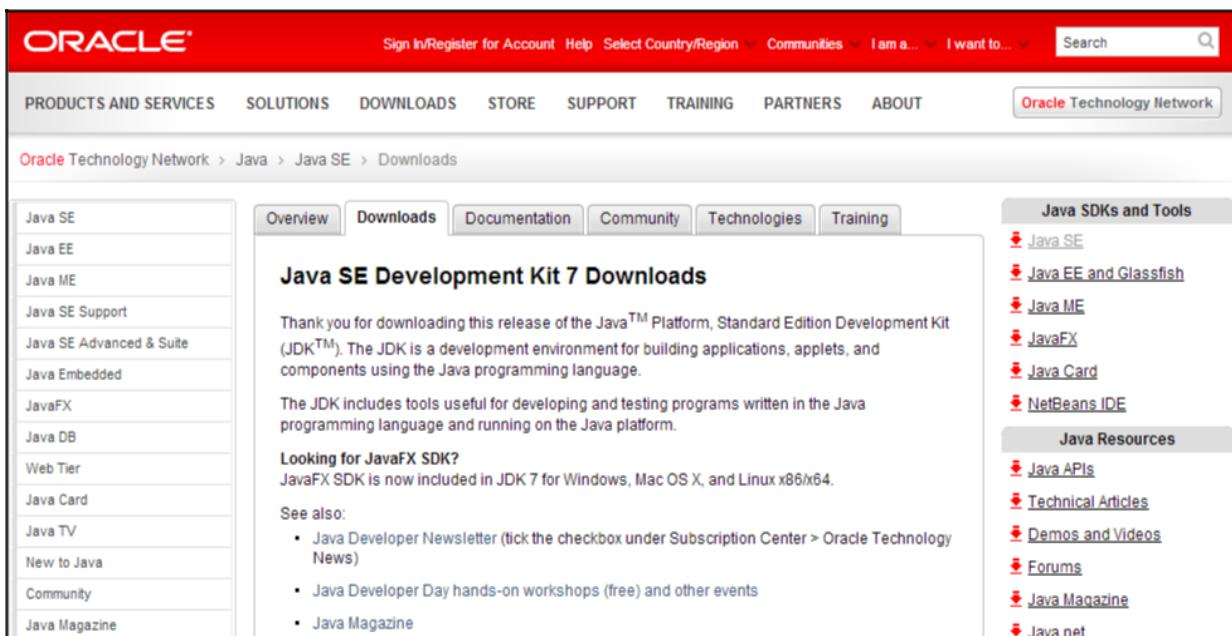
5º Instalar o Plugin Android para Eclipse

A fim de evitar qualquer tipo de conflitos com versões anteriores dos programas supracitados, desinstale estes.

### ***Instalação do SDK do Java***

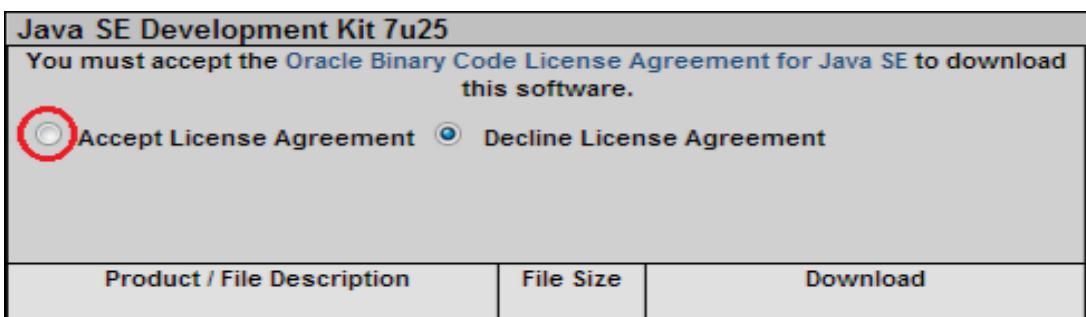
Para efetuar este download do SDK do Java, acesse o endereço:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>



The screenshot shows the Oracle Java Technology Network website. The navigation bar includes links for Sign In/Register for Account, Help, Select Country/Region, Communities, I am a..., I want to..., Search, Oracle Technology Network, PRODUCTS AND SERVICES, SOLUTIONS, DOWNLOADS, STORE, SUPPORT, TRAINING, PARTNERS, and ABOUT. The main content area is titled "Java SE Development Kit 7 Downloads". It features a brief introduction about the JDK, mentions that JavaFX is included in JDK 7, and provides links for Java Developer Newsletter, Java Developer Day workshops, and Java Magazine. On the right side, there are two columns: "Java SDKs and Tools" (links to Java SE, Java EE and Glassfish, Java ME, JavaFX, Java Card, and NetBeans IDE) and "Java Resources" (links to Java APIs, Technical Articles, Demos and Videos, Forums, Java Magazine, and Java.net). The URL in the browser's address bar is <http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>.

Clique em **Accept License Agreement**.



The screenshot shows the Java SE Development Kit 7u25 download page. A message at the top states: "You must accept the Oracle Binary Code License Agreement for Java SE to download this software." Below this, there are two radio buttons: "Accept License Agreement" (which is selected and has a red circle around it) and "Decline License Agreement". At the bottom of the page, there is a table with three columns: "Product / File Description", "File Size", and "Download".

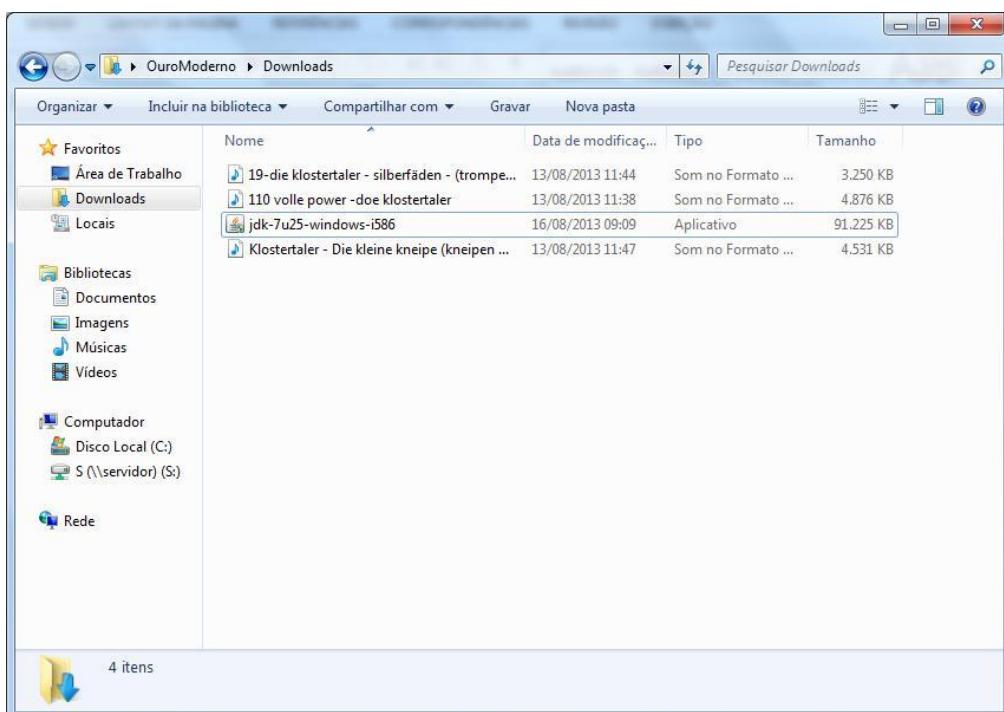
Arraste a barra de rolagem para baixo até encontrar a opção indicada, aqui você terá que escolher a opção referente ao seu sistema operacional. Neste exemplo iremos utilizar a versão para Windows 32 bits.

Java SE Development Kit 7u25		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
Product / File Description	File Size	Download
Linux x86	80.38 MB	<a href="#">jdk-7u25-linux-i586.rpm</a>
Linux x86	93.12 MB	<a href="#">jdk-7u25-linux-i586.tar.gz</a>
Linux x64	81.46 MB	<a href="#">jdk-7u25-linux-x64.rpm</a>
Linux x64	91.85 MB	<a href="#">jdk-7u25-linux-x64.tar.gz</a>
Mac OS X x64	144.43 MB	<a href="#">jdk-7u25-macosx-x64.dmg</a>
Solaris x86 (SVR4 package)	136.02 MB	<a href="#">jdk-7u25-solaris-i586.tar.Z</a>
Solaris x86	92.22 MB	<a href="#">jdk-7u25-solaris-i586.tar.gz</a>
Solaris x64 (SVR4 package)	22.77 MB	<a href="#">jdk-7u25-solaris-x64.tar.Z</a>
Solaris x64	15.09 MB	<a href="#">jdk-7u25-solaris-x64.tar.gz</a>
Solaris SPARC (SVR4 package)	136.16 MB	<a href="#">jdk-7u25-solaris-sparc.tar.Z</a>
Solaris SPARC	95.5 MB	<a href="#">jdk-7u25-solaris-sparc.tar.gz</a>
Solaris SPARC 64-bit (SVR4 package)	23.05 MB	<a href="#">jdk-7u25-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	17.67 MB	<a href="#">jdk-7u25-solaris-sparcv9.tar.gz</a>
Windows x86	89.09 MB	<a href="#">jdk-7u25-windows-i586.exe</a>
Windows x64	90.66 MB	<a href="#">jdk-7u25-windows-x64.exe</a>

Para iniciar o download clique no local indicado.

Java SE Development Kit 7u25		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
Product / File Description	File Size	Download
Linux x86	80.38 MB	<a href="#">jdk-7u25-linux-i586.rpm</a>
Linux x86	93.12 MB	<a href="#">jdk-7u25-linux-i586.tar.gz</a>
Linux x64	81.46 MB	<a href="#">jdk-7u25-linux-x64.rpm</a>
Linux x64	91.85 MB	<a href="#">jdk-7u25-linux-x64.tar.gz</a>
Mac OS X x64	144.43 MB	<a href="#">jdk-7u25-macosx-x64.dmg</a>
Solaris x86 (SVR4 package)	136.02 MB	<a href="#">jdk-7u25-solaris-i586.tar.Z</a>
Solaris x86	92.22 MB	<a href="#">jdk-7u25-solaris-i586.tar.gz</a>
Solaris x64 (SVR4 package)	22.77 MB	<a href="#">jdk-7u25-solaris-x64.tar.Z</a>
Solaris x64	15.09 MB	<a href="#">jdk-7u25-solaris-x64.tar.gz</a>
Solaris SPARC (SVR4 package)	136.16 MB	<a href="#">jdk-7u25-solaris-sparc.tar.Z</a>
Solaris SPARC	95.5 MB	<a href="#">jdk-7u25-solaris-sparc.tar.gz</a>
Solaris SPARC 64-bit (SVR4 package)	23.05 MB	<a href="#">jdk-7u25-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	17.67 MB	<a href="#">jdk-7u25-solaris-sparcv9.tar.gz</a>
Windows x86	89.09 MB	<a href="#">jdk-7u25-windows-i586.exe</a>
Windows x64	90.66 MB	<a href="#">jdk-7u25-windows-x64.exe</a>

Vá até a sua pasta de download e selecione o instaladores do pacote SDK.

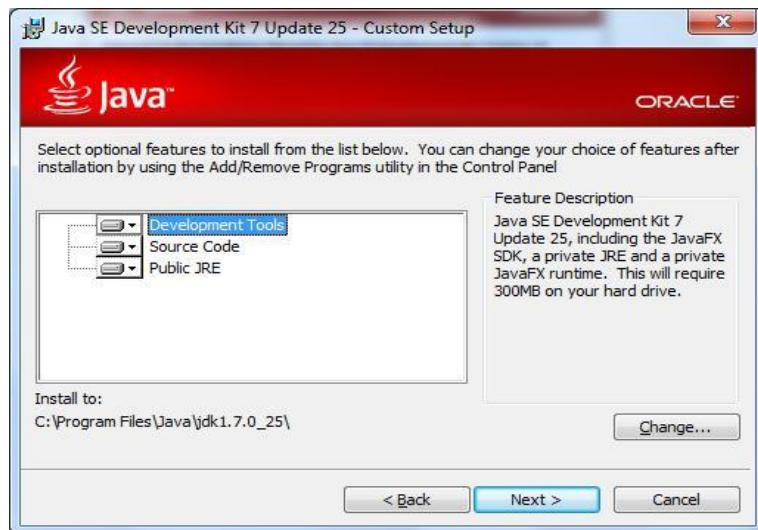


O Windows lhe perguntará se você deseja que este programa faça alterações no seu computador, clique em **Sim**.

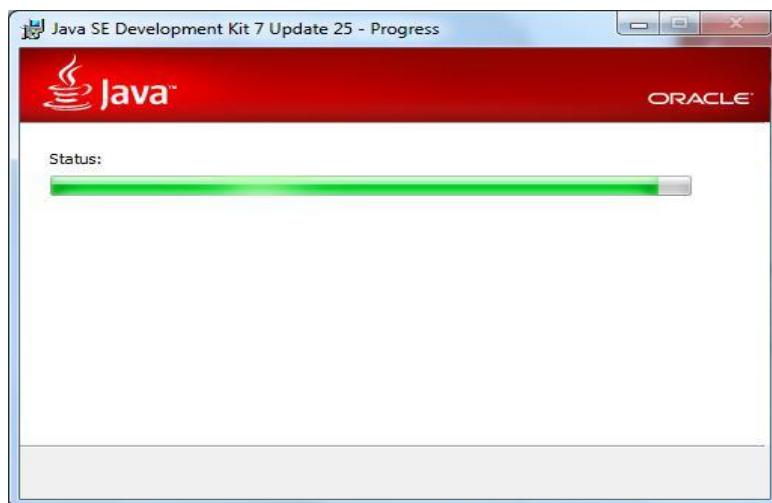
Muito bem, vamos iniciar a instalação, a primeira tela apenas lhe dá as boas-vindas ao programa, clique em **Next**.



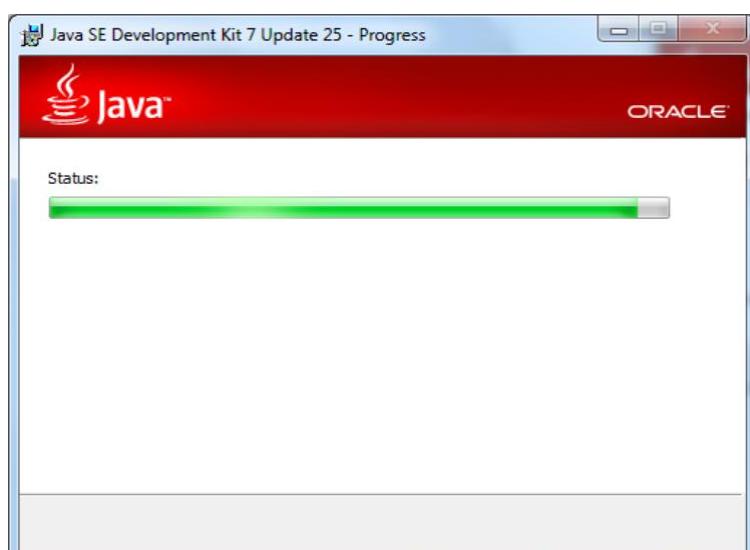
Nesta janela você pode configurar a instalação, mas há necessidades disto, clique novamente em **Next**.



Aguarde alguns minutos até a janela do JRE seja exibida.



Essa é a tela de configuração do JRE. Não modifique nada neste local, apenas clique em **Next**.



Aguarde alguns minutos para a tela de conclusão de instalação ser exibida.



Nesta janela você poderá registrar o produto, mas clique em **Close**.



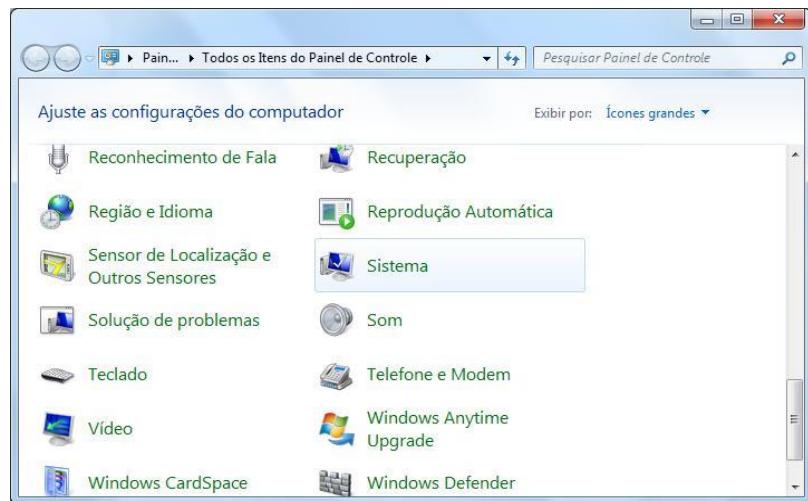
### Configurando o Java

Antes de instalar o SDK do Android será necessário realizar algumas configurações de ambiente do Windows.

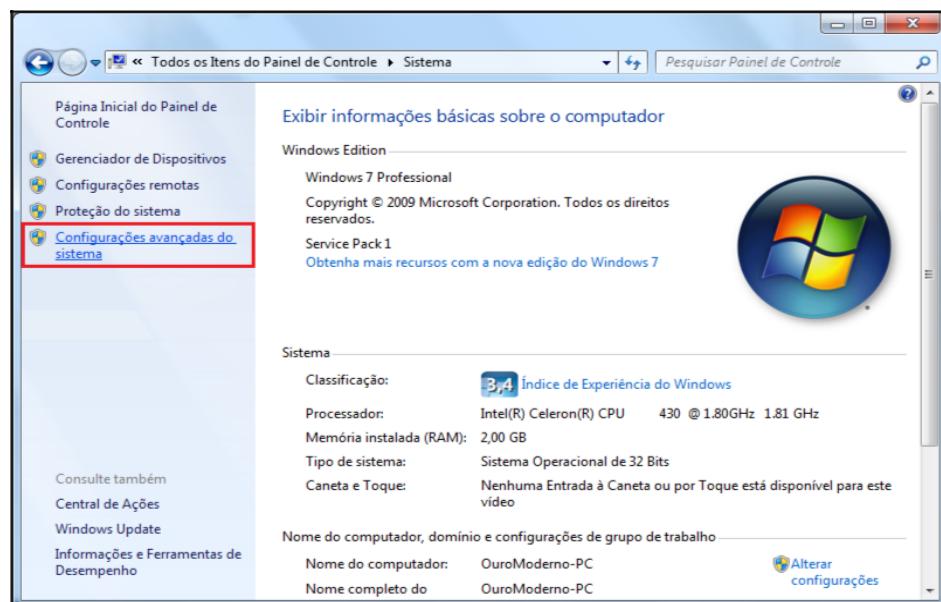
Desenvolvemos este curso para o sistema Windows 7, mas se houver problemas com outra versão do Windows ou outro sistema operacional, basta adaptar as instruções deste curso para o seu sistema.

Clique no menu **Iniciar** e selecione a opção **Painel de Controle**.

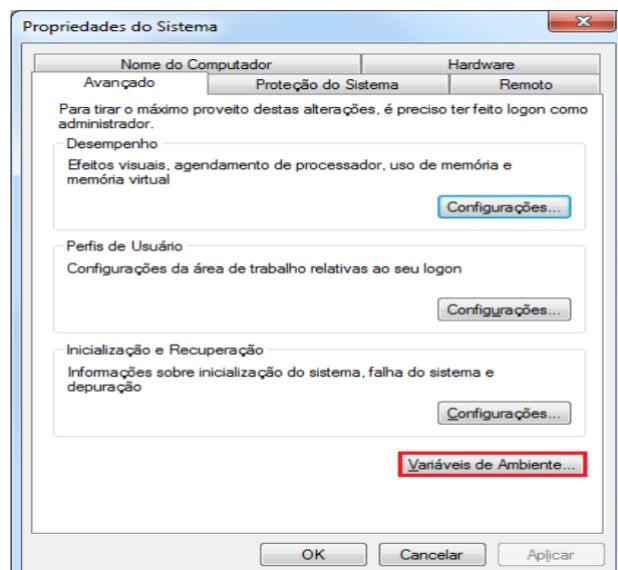
Clique em **Sistema** ou **Sistema e segurança**, depende de como seu computador está configurado.



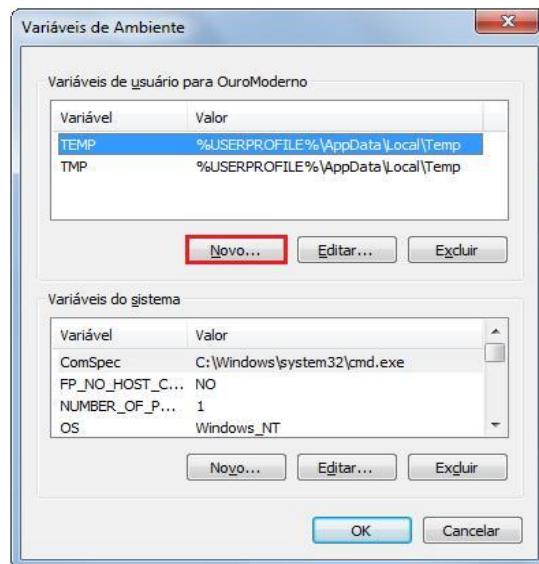
Clique na opção **Configurações avançadas do sistema**.



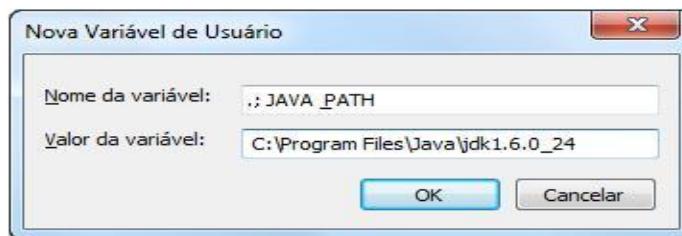
Nesta caixa de diálogo clique em **Variáveis de Ambiente**.



Agora iremos adicionar três variáveis de Ambiente.  
Clique em **Novo**.

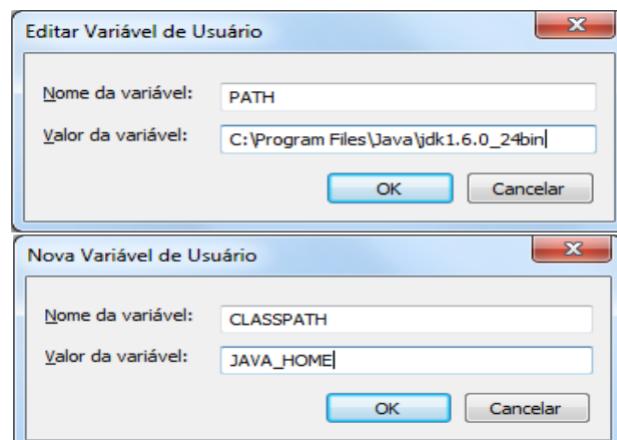


Nas caixas de diálogos vazias que irão aparecer, preencha com as ações das variáveis representadas na imagem abaixo.

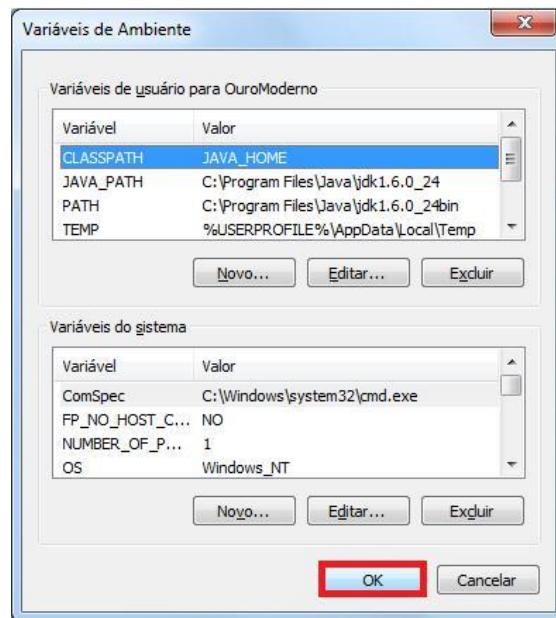


Clique em OK.

Da mesma forma que está variável, crie outras conforme as imagens abaixo, não esquecendo de clicar em Ok.



Clique em OK para salvar as configurações.



## Android SDK

Para baixar o Android SDK acesse o site:

<http://developer.android.com/sdk/index.html>.

Caso a sua plataforma for Windows clique no local indicado.

Mas se você estiver trabalhando com outra plataforma clique no local indicado.

Developers      Design      Develop      Distribute

Training      API Guides      Reference      Tools      Google Services

### Get the Android SDK

**Developer Tools**

- Download**
  - Setting Up the ADT Bundle
  - Setting Up an Existing IDE
  - Android Studio
  - Exploring the SDK
  - Download the NDK
- Workflow
- Support Library
- Tools Help
- Rewards
- Samples
- ADK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in ADT (Android Developer Tools) to streamline your Android app development.

With a single download, the ADT Bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

**Android Studio Early Access Preview**

A new Android development environment called Android Studio, based on IntelliJ IDEA, is now available as an early access preview. For more information, see [Getting Started with Android Studio](#).

If you prefer to use an existing version of Eclipse or another IDE, you can instead take a more customized approach to installing the Android SDK. See the following instructions:

- ▼ USE AN EXISTING IDE
- ▼ SYSTEM REQUIREMENTS
- ▼ DOWNLOAD FOR OTHER PLATFORMS

**Download the SDK**  
ADT Bundle for Windows

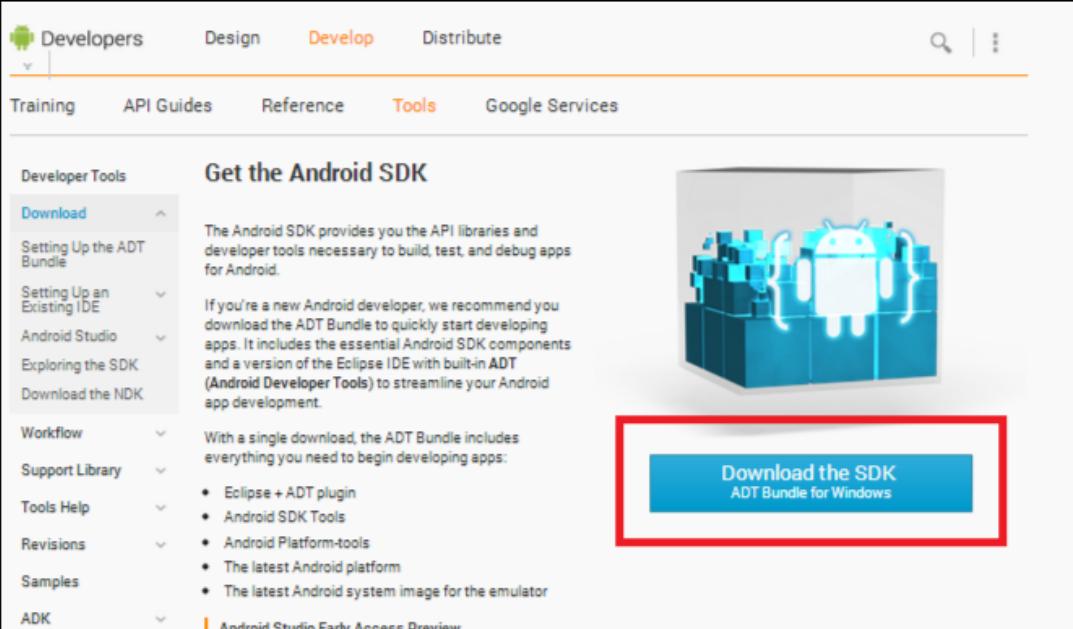


Basta selecionar a sua plataforma.

^ DOWNLOAD FOR OTHER PLATFORMS			
ADT Bundle			
Platform	Package	Size	MD5 Checksum
Windows 32-bit	adt-bundle-windows-x86-20130729.zip	463931746 bytes	51faf4e5fdf9c5b4a176179a99ce3511
Windows 64-bit	adt-bundle-windows-x86_64-20130729.zip	464064756 bytes	e8f05c1fddb8e609e880de23113c7426
Mac OS X 64-bit	adt-bundle-mac-x86_64-20130729.zip	428792424 bytes	6c42b9966abcf8a75c0ee83d0d95882
Linux 32-bit	adt-bundle-linux-x86-20130729.zip	457716139 bytes	b3686d10dc1cbceba1074404d4386283
Linux 64-bit	adt-bundle-linux-x86_64-20130729.zip	458006784 bytes	1fabcc3f772ba8b2fc194d6e0449da17
SDK Tools Only			
Platform	Package	Size	MD5 Checksum
Windows 32 & 64-bit	android-sdk_r22.0.5-windows.zip	113510621 bytes	30695dff041e0d7cf9ff948ab0c48920
	installer_r22.0.5-windows.exe (Recommended)	93505782 bytes	940849be19ac6151e3e35c8706c81d86
Mac OS X 32 & 64-bit	android-sdk_r22.0.5-macosx.zip	77225724 bytes	94f3obe896c332b94ee0408ae610a4b8
Linux 32 & 64-bit	android-sdk_r22.0.5-linux.tgz	105641005 bytes	8201b10c21510f082c54f58a9bb082c8

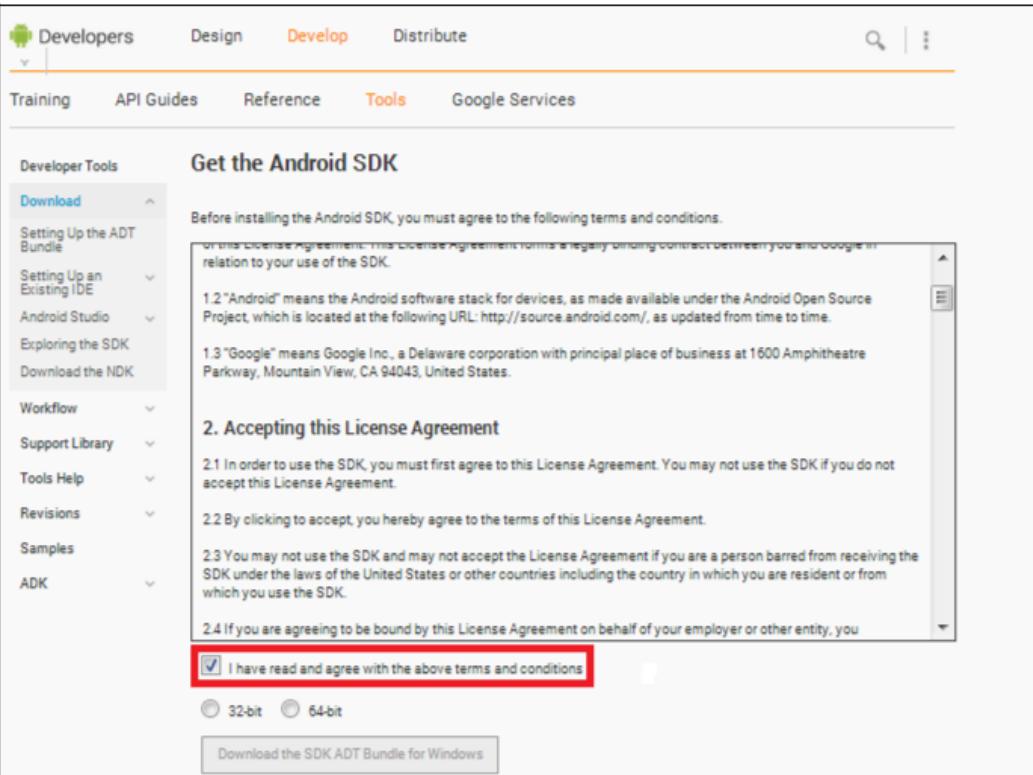
Mas neste curso iremos trabalhar com a plataforma do Windows.

Clicaremos no link selecionado.



The screenshot shows the 'Get the Android SDK' page on the official Android Developers website. The left sidebar has a 'Developer Tools' section with a 'Download' heading expanded, showing options like 'Setting Up the ADT Bundle', 'Android Studio', etc. The main content area is titled 'Get the Android SDK' and contains text about the ADT Bundle and a large image of an Android robot. A red box highlights the 'Download the SDK ADT Bundle for Windows' button.

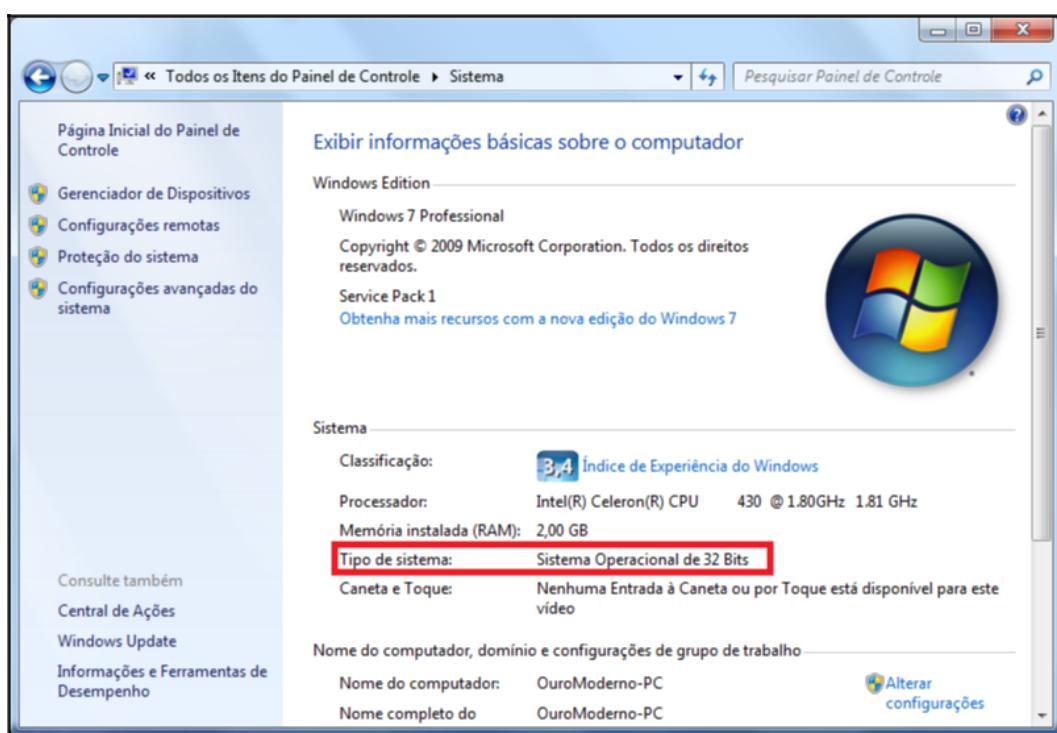
Será necessário você clicar no local indicado provando que você leu e aceita os termos indicados.



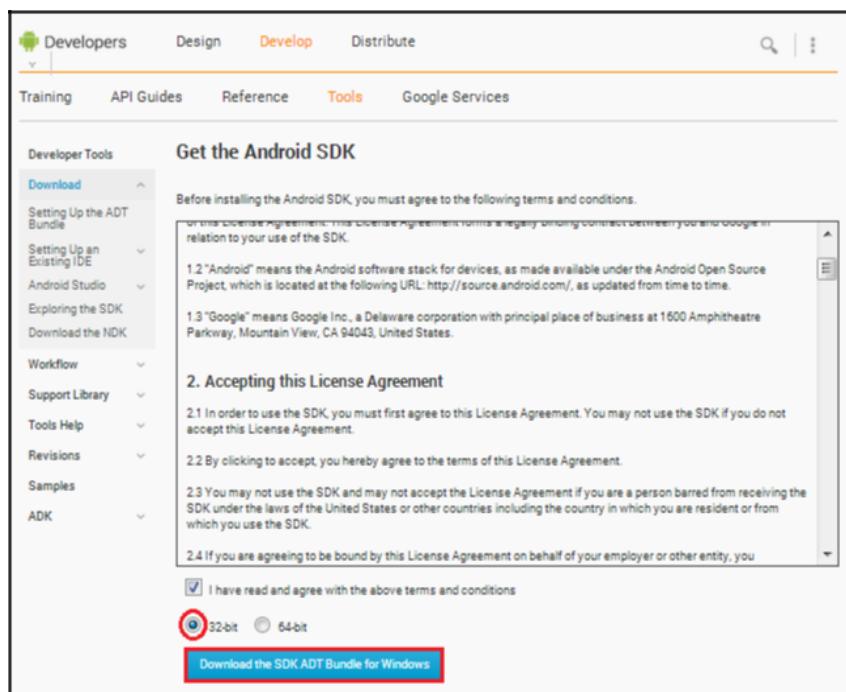
The screenshot shows the same 'Get the Android SDK' page, but now the 'I have read and agree with the above terms and conditions' checkbox is checked and highlighted with a red box. The rest of the page content is identical to the first screenshot.

Agora volte a janela **Sistema**.

Verifique se o seu Sistema funciona com 32 ou 64 bits.



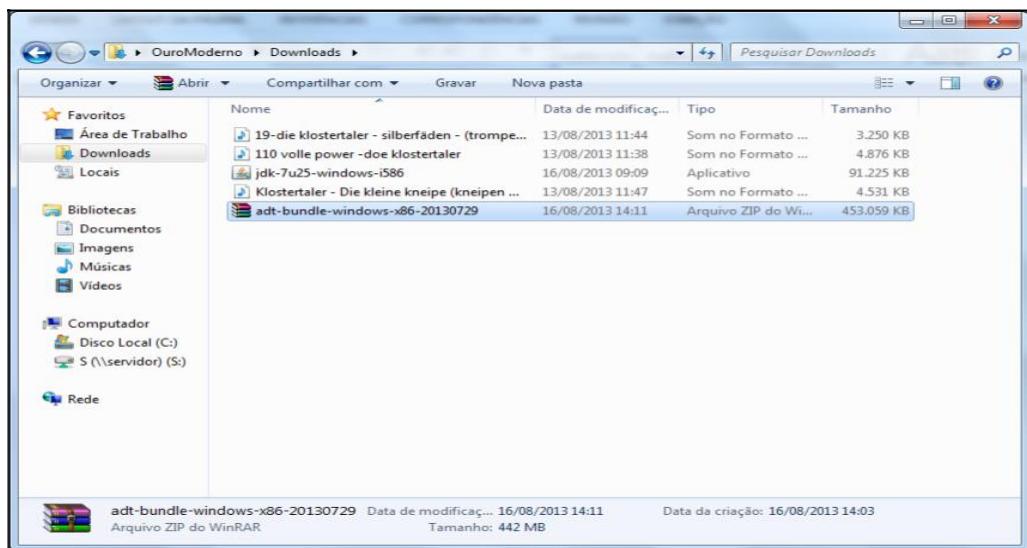
Veja que neste exemplo o sistema funciona com 32 bits, por isso iremos clicar nesta opção e logo após clique em **Download the SDK ADT Bundle for Windows**.



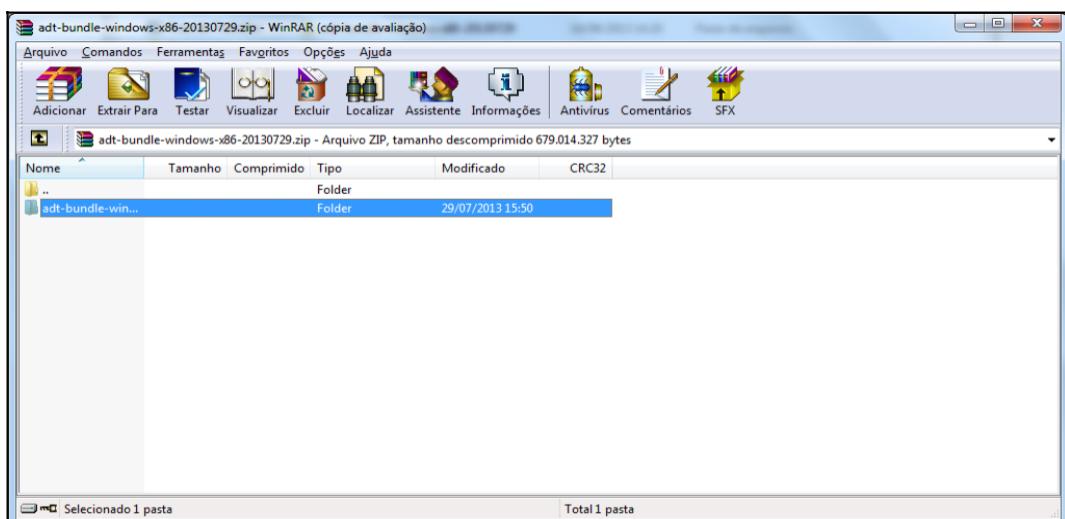
Aguarde alguns minutos até o download ser completado.

Vá até a sua pasta de download.

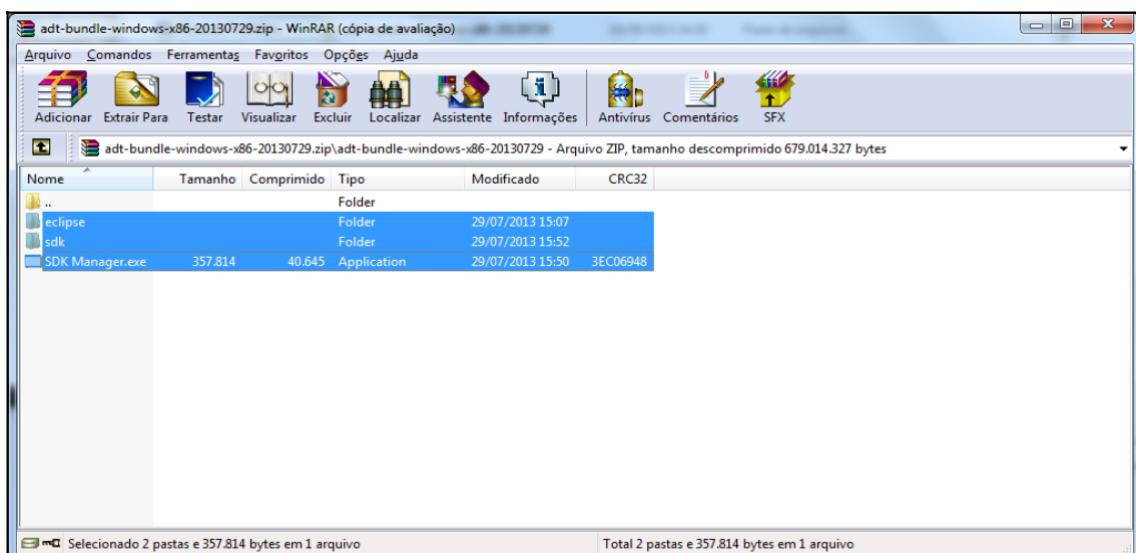
Abra o arquivo indicado.



Abra a pasta indicada.

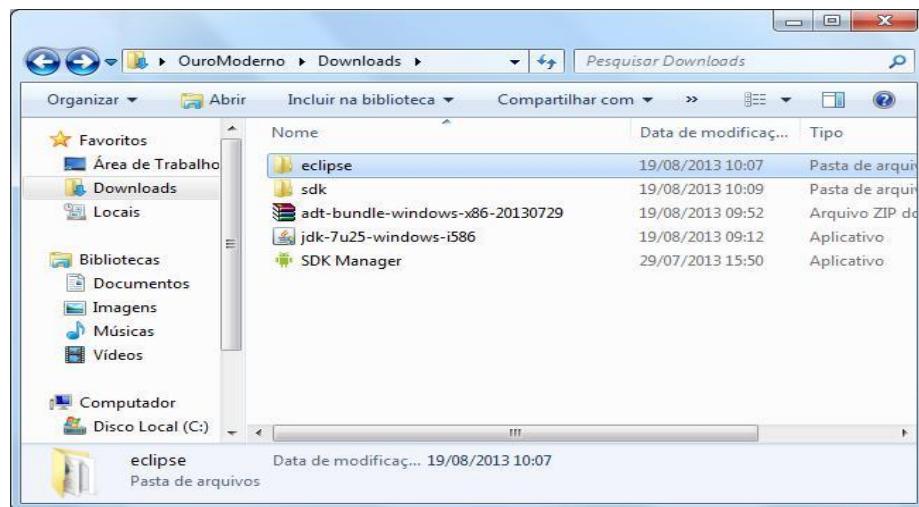


Copie as três pastas indicadas para a pasta download.

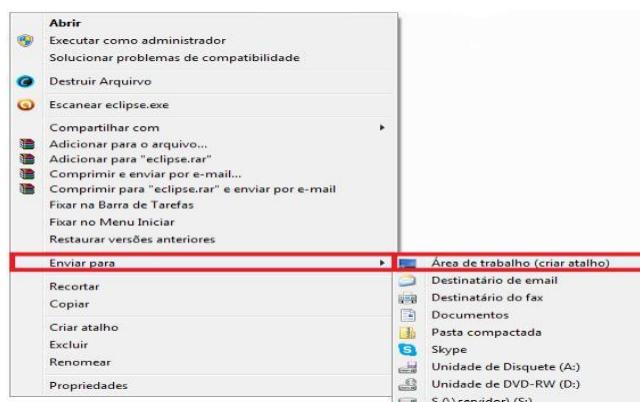


Neste local as opções cortar, recortar e colar não são exibidas ao clicar com o botão direito do mouse, por isso o conveniente é usar os recursos CTRL + C = copiar, CTRL + X = recortar e CTRL + V = colar.

### Abra a pasta eclipse.



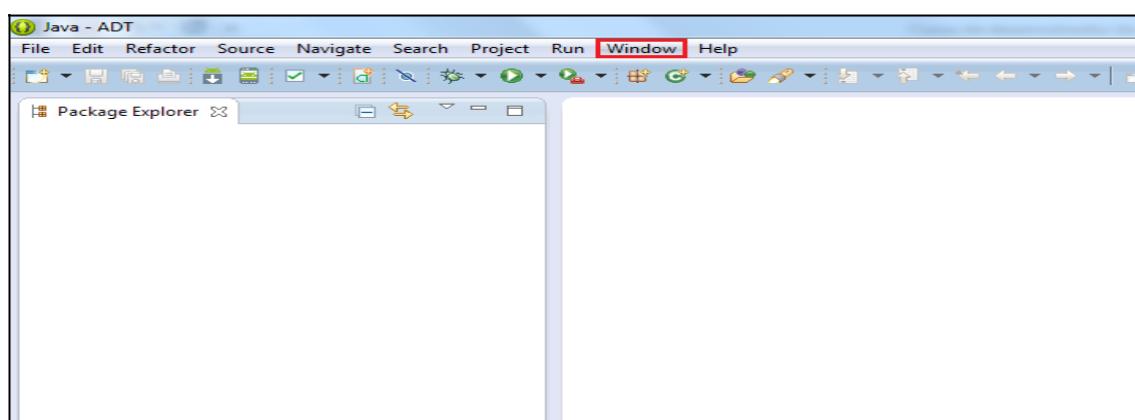
Crie um atalho para o Eclipse na área de trabalho.



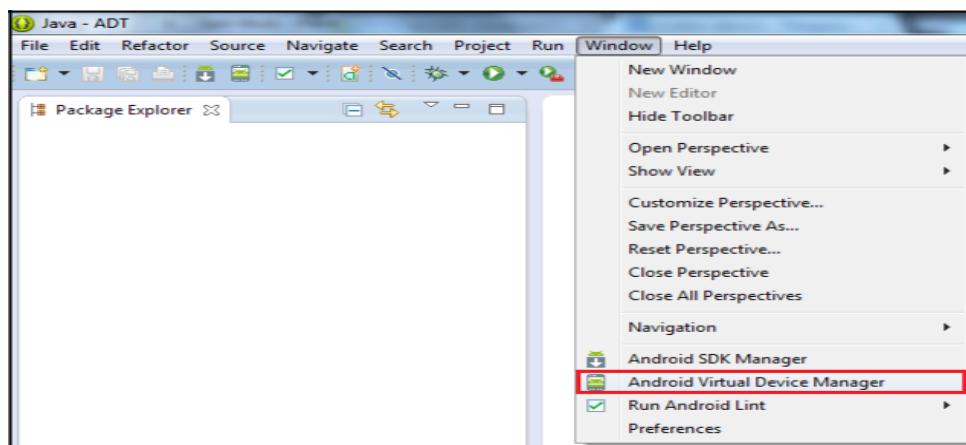
Abra o Eclipse.

Muito bem, está é a página inicial do Eclipse, vamos iniciar a criação do nosso emulador.

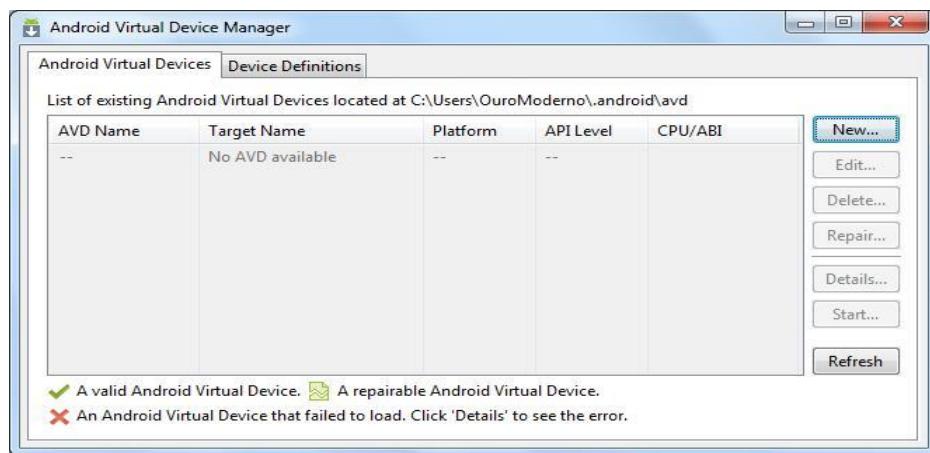
Clique no guia **Window**.



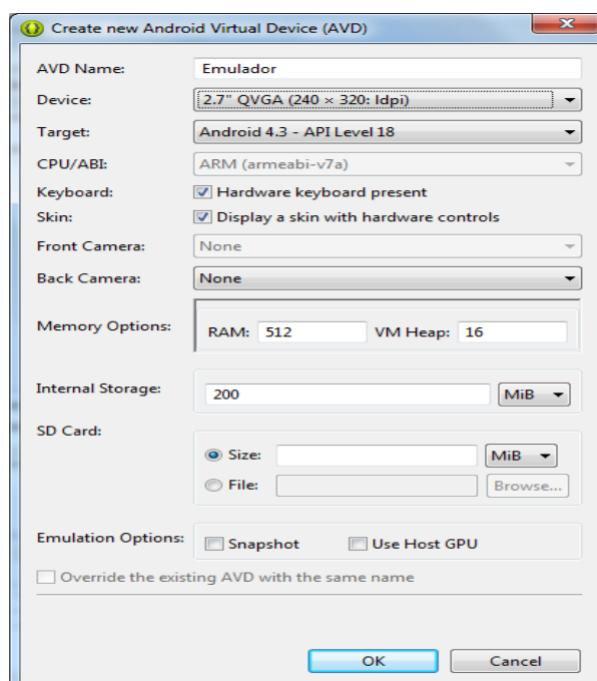
Clique na opção **Android Virtual Device Manager**.



Vamos criar um novo emulador, para iniciar isso clique em **New**.

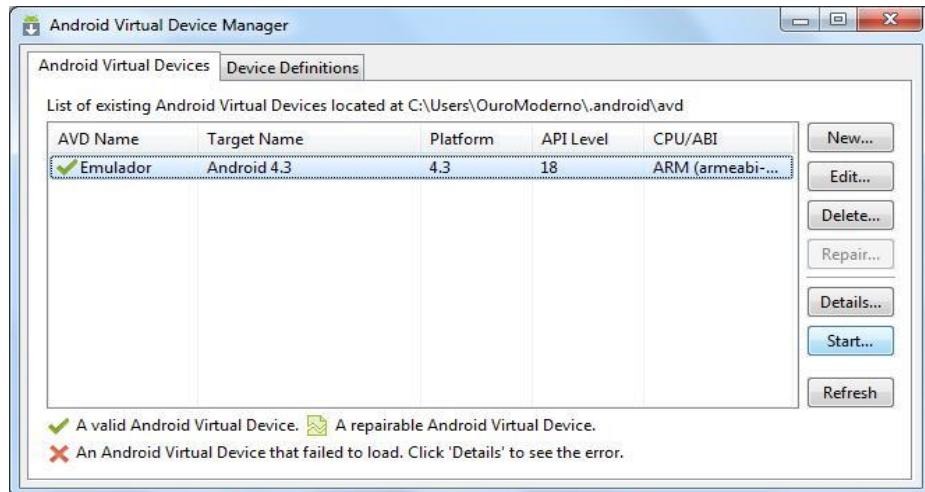


Nos campos indicados preencha com as respectivas informações.

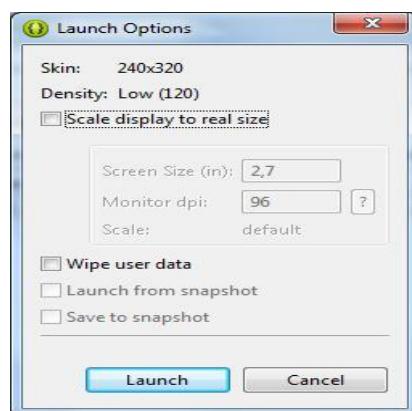


Clique em **OK**, depois selecione o projeto de emulador.

Clique em **Start**.



Após isso clique em **Launch**.



Espere alguns instantes até o emulador iniciar.



Muito bem, chegamos ao fim desta primeira aula do curso de Desenvolvedor de Android, não se esqueça de fazer os exercícios da apostila.

***Finalizamos nossa primeira aula. Por favor, não deixe de fazer nenhum exercício para que seu APP possa sair conforme o planejamento.***

### ***EXERCÍCIO DE FIXAÇÃO***

- 1)** Baixe e instale os programas necessários para desenvolver aplicativos Android.
- 2)** Após fazer todas os passos necessários, ensinado na aula e apostila, abra o Eclipse.
- 3)** Crie um novo *emulador*, e execute-o.
- 4)** Experimente diferentes versões do Android e resoluções de tela no emulador.



## ANOTAÇÕES



## ANOTAÇÕES



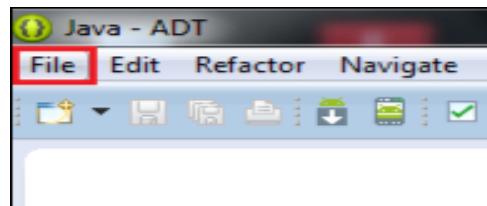
## ANOTAÇÕES



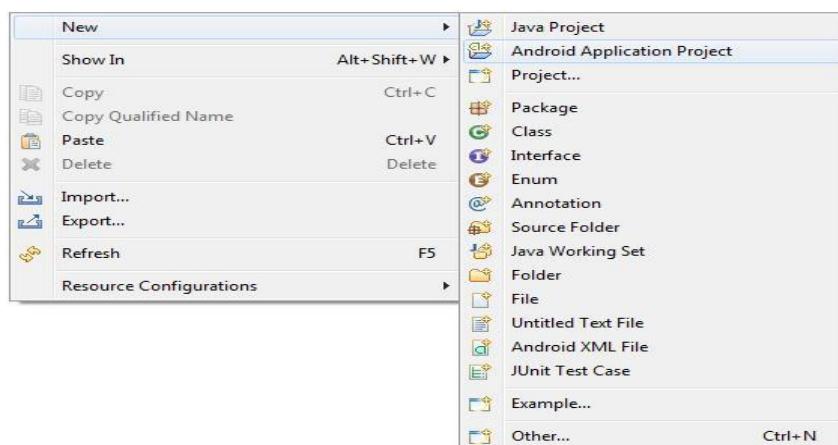
## AULA 2

### Configurando o Eclipse

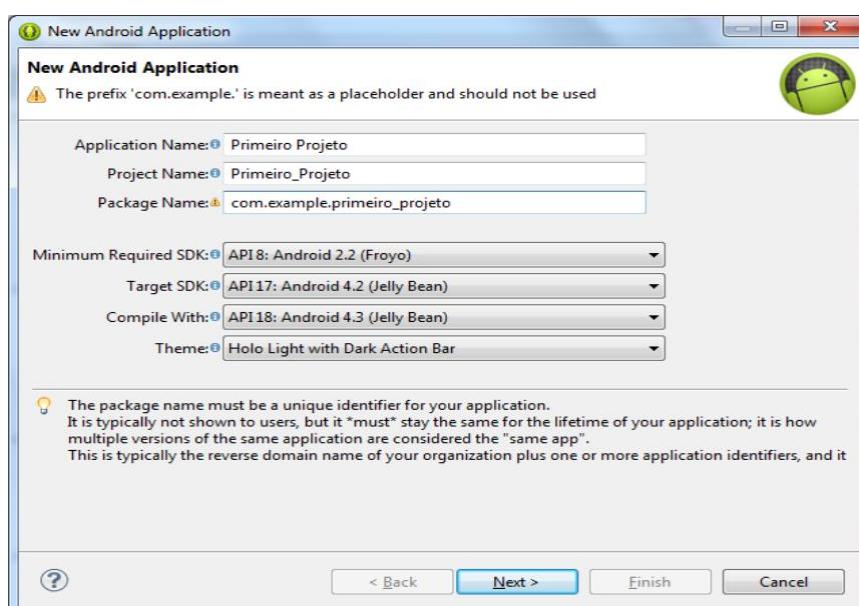
Olá seja bem-vindo a segunda aula do curso de desenvolvedor de Android.  
 Abra a janela do Eclipse.  
 Clique com o botão direito do mouse no local indicado.



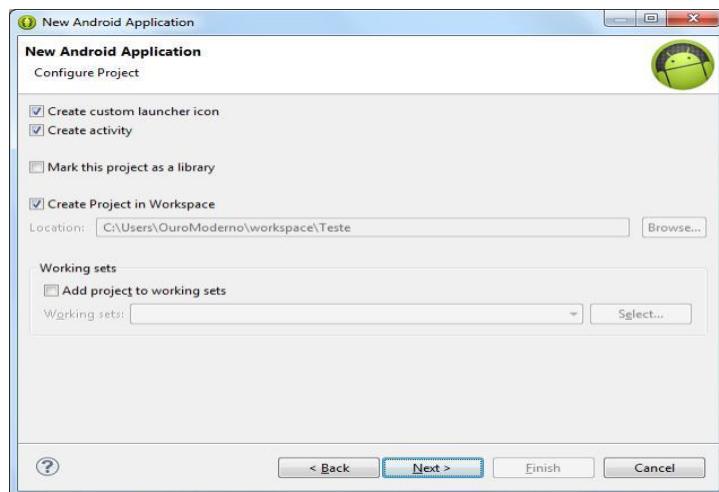
Clique em **New** e depois em **Android Application Project**.



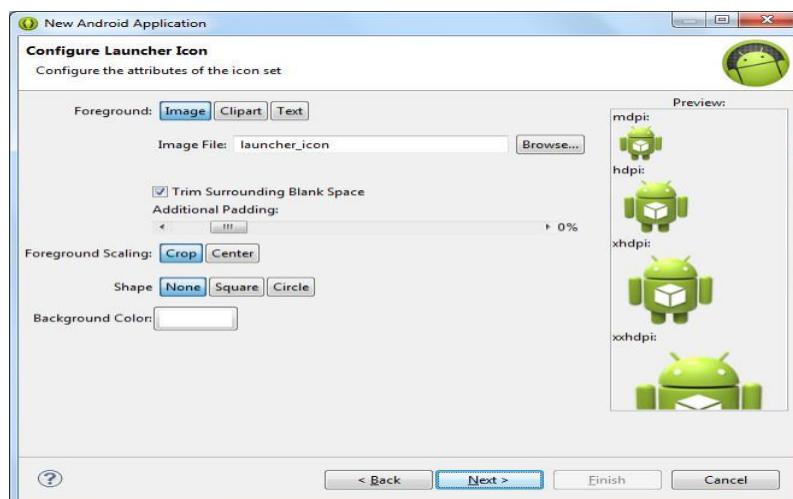
Nas caixas de diálogo digite os textos referentes a imagem abaixo.



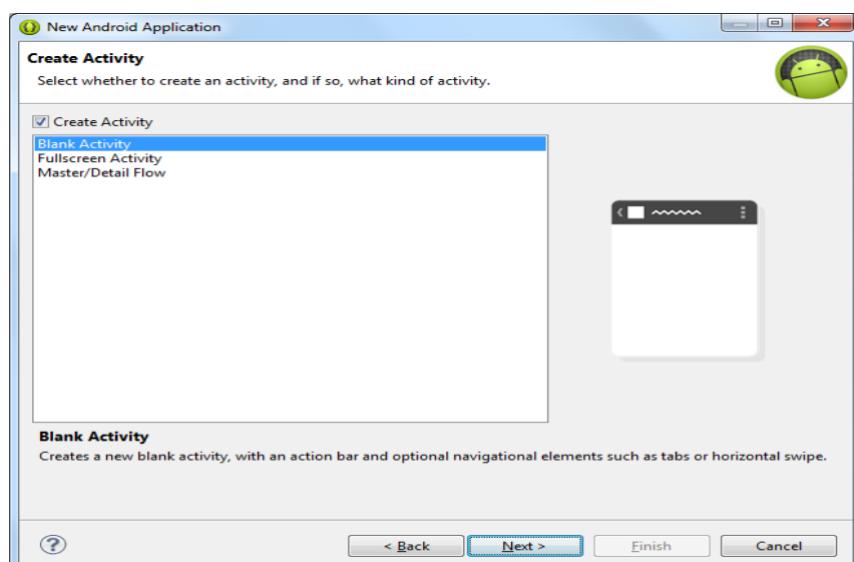
Clique em **Next**.  
Clique novamente em **Next**.



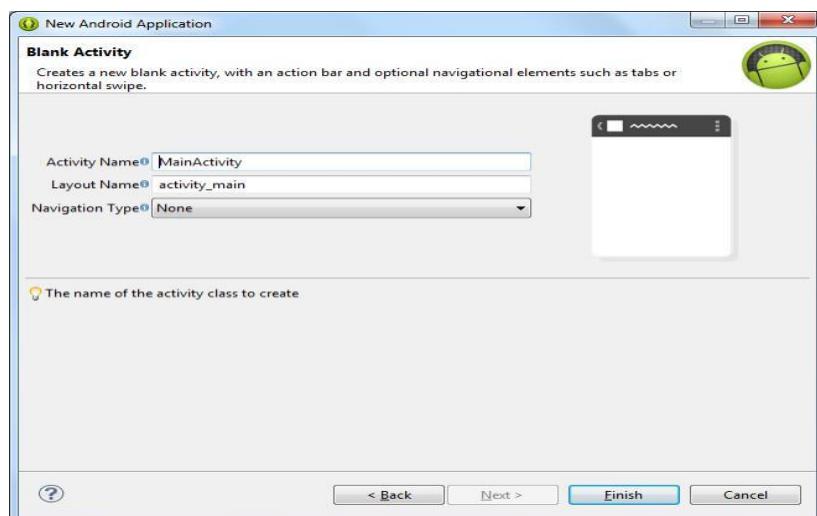
Clique mais uma vez em **Next**.



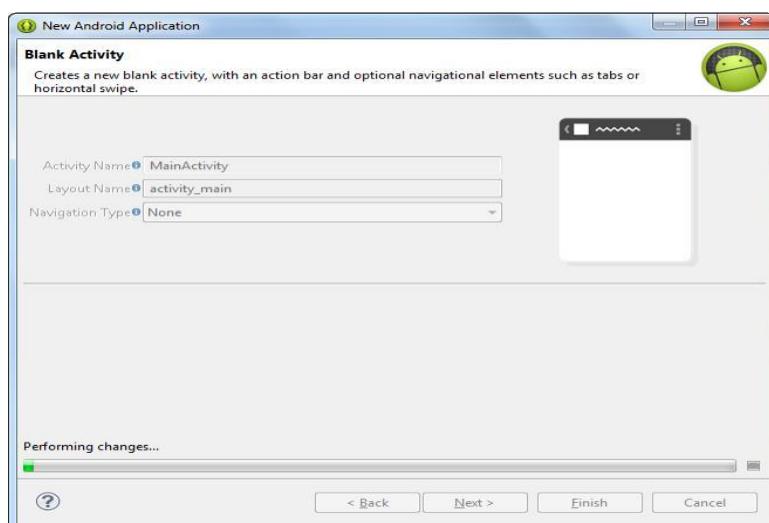
Clique novamente em **Next**.



Clique em **Finish**.

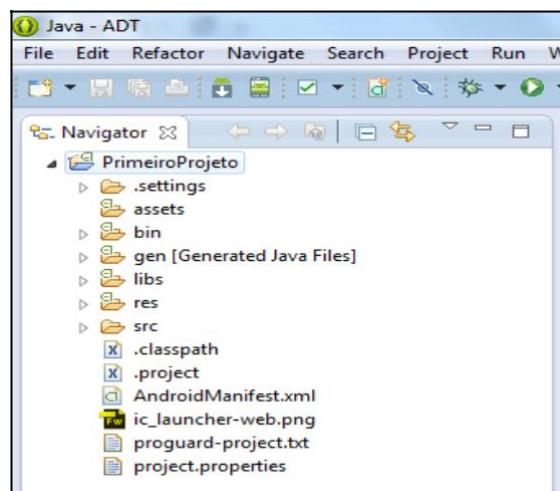


Aguarde alguns instantes até o novo projeto ser concluído.



### **Entendendo o projeto Android**

Observe que o projeto já foi criado com uma série de pastas e arquivos.



Nestas pastas você pode encontrar várias funções por exemplo.

**Na pasta `src`** ficam contidas as classes em Java.1

**Na pasta gen(Gereted Java Files)** Nela estão as classes que contém as constantes referentes aos recursos de aplicação. Ela também é uma classe criada pelo template\* e não deve ser alterada

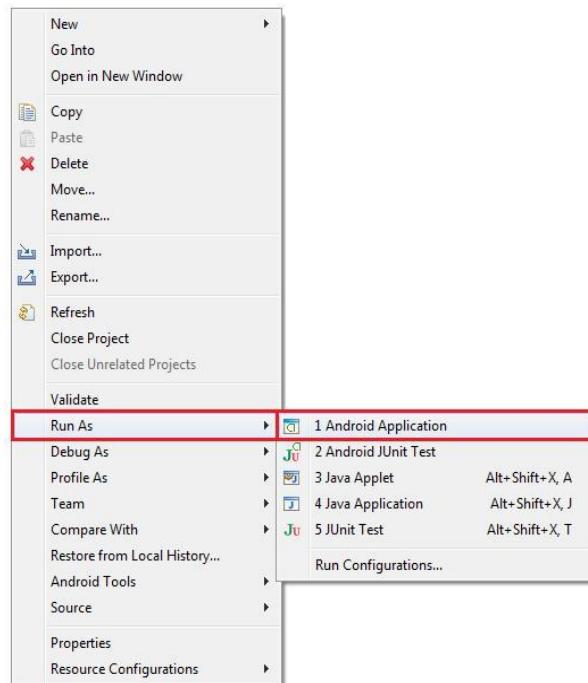
**Na pasta `assets`** podem ser adicionados os recursos extras do projeto. Eles não são gerenciados automaticamente pelo plugin Android, sendo necessário fornecer um nome e um caminho válidos para todos os recursos que forem adicionados.

Os recursos que são adicionados na pasta `res` serão automaticamente gerenciados pelo plugin.

\*Template é um documento sem conteúdo, com apenas a apresentação visual.

## Executando o arquivo

Clique com o botão direito do mouse sobre o título do projeto no caso **Primeiro Projeto**, selecionando as opções indicadas da imagem abaixo.



O emulador poderá demorar alguns minutos até abrir, ele funciona como se fosse um celular dentro do seu computador.

O emulador irá abrir com está janela.



Ao iniciar o emulador este estará bloqueado.



Para desbloquear clique mantenha pressionado o mouse no ícone de cadeado e arraste o ícone até o local indicado.

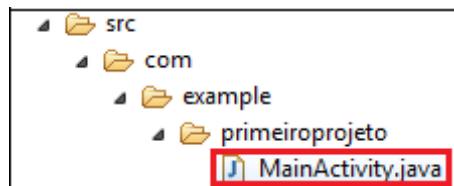


A aplicação está sendo exibida.

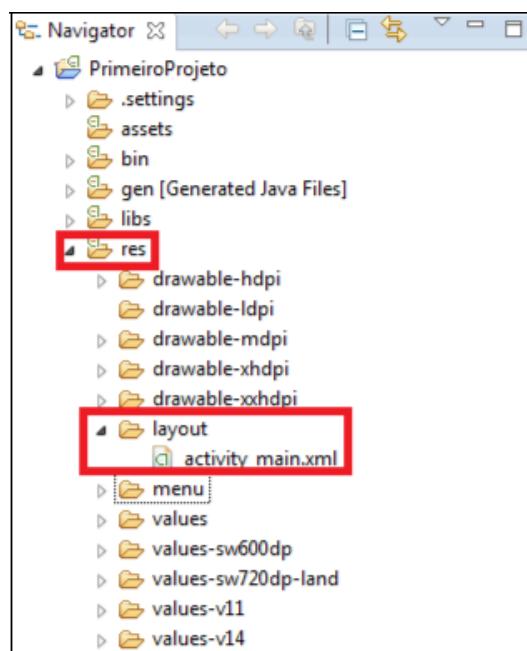


Mesmo que você não tenha programado nada, a mensagem **Hello World!** está sendo exibida. Agora vamos entender como esse projeto foi executado.

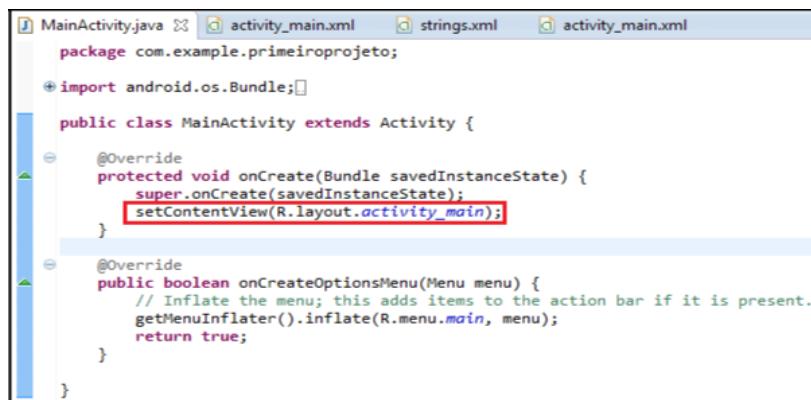
Va até a pasta **MainActivity.java**, localizada dentro da pasta **src**.



A pasta **layout** está dentro da pasta **res**.



Repare que a área indicada mostra que dentro da pasta **Layout** está um arquivo chamado **activity\_main**.



```

MainActivity.java
package com.example.primeiroprojeto;

import android.os.Bundle;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}

```

Dentro desta pasta você encontra uma informação de que a mensagem **Hello World!** Está dentro da pasta **string**.



```

activity_main.xml
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

```

Aqui dentro está a referência a frase que já vem instalada no Aplicativo.



```

strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Primeiro Projeto</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
</resources>

```

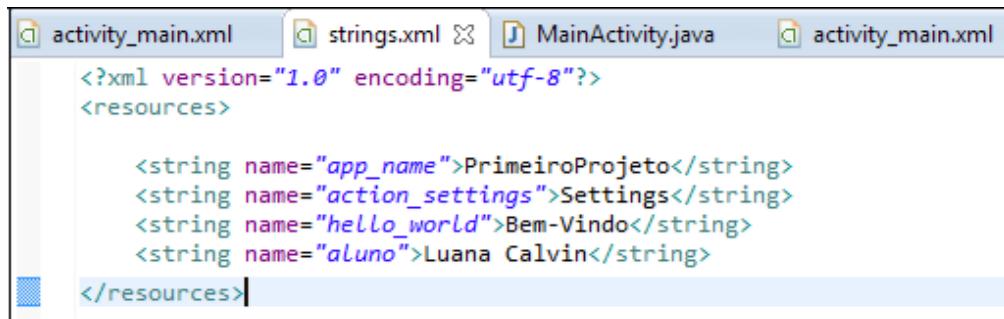
### Adicionando recursos no xml

Para adicionar textos e frases no seu APP, você deve criar novas recurso de strings, para adicionar um novo recurso, basta inserir uma linha entre as tags.

Mas você deve tomar cuidado pois o arquivo xml diferencia as maiúsculas das minúsculas e as tags devem estar corretamente formadas.

As tags são modos de abrir e fechar programações, cada vez que uma programação está pronta ela deve ser fechada com um tag, que podem ser de diversas formas, depende da linguagem que está sendo utilizada.

Ainda em strings, adicione uma linha conforme a imagem baixo:



```

<?xml version="1.0" encoding="utf-8"?>
<resources>

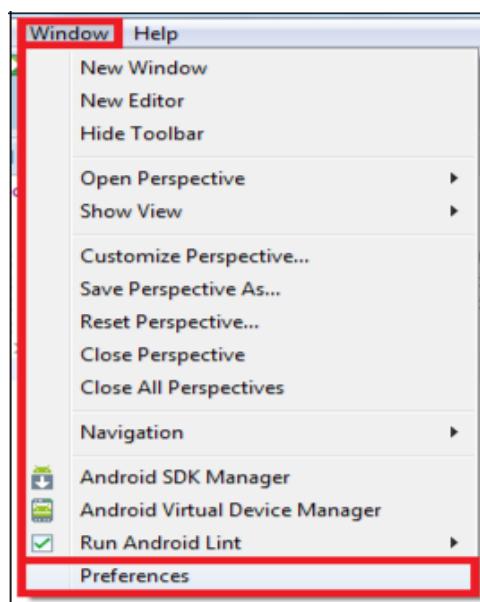
    <string name="app_name">PrimeiroProjeto</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Bem-Vindo</string>
    <string name="aluno">Luana Calvin</string>

</resources>

```

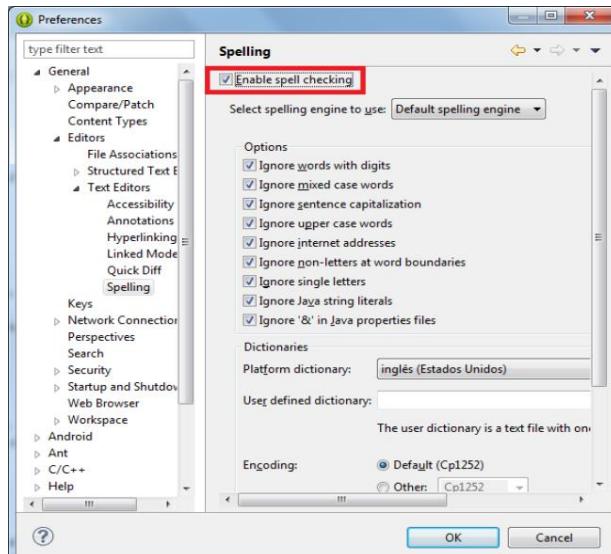
Dê um **CTRL + S** para salvar.

Abra a guia casa **Window** e selecione a opção **Preferences**.



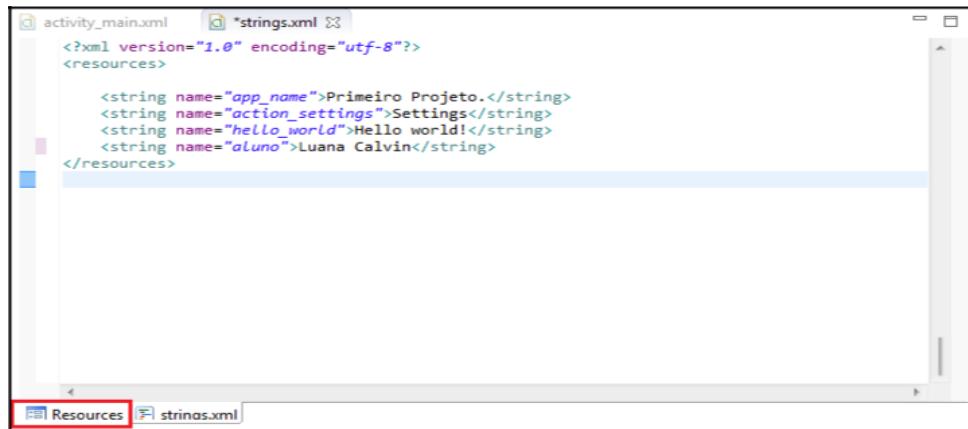
Na janela General abra a opção **Editors> Text Editors** e de um duplo clique na opção **Spilling**.

Na janela **Spilling** desmarque a opção **Enable spell checking** e clique em **OK**.

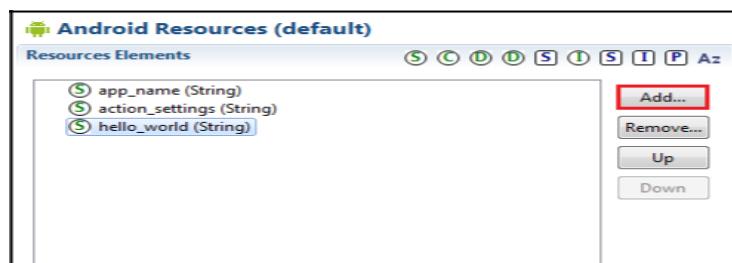


## Adicionando recursos no modo design

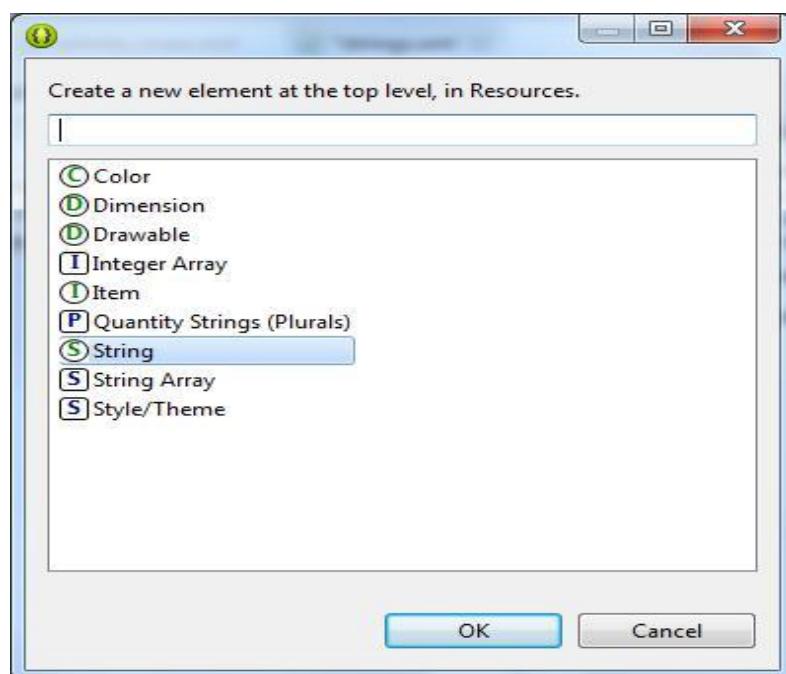
Clique em **Resources**.



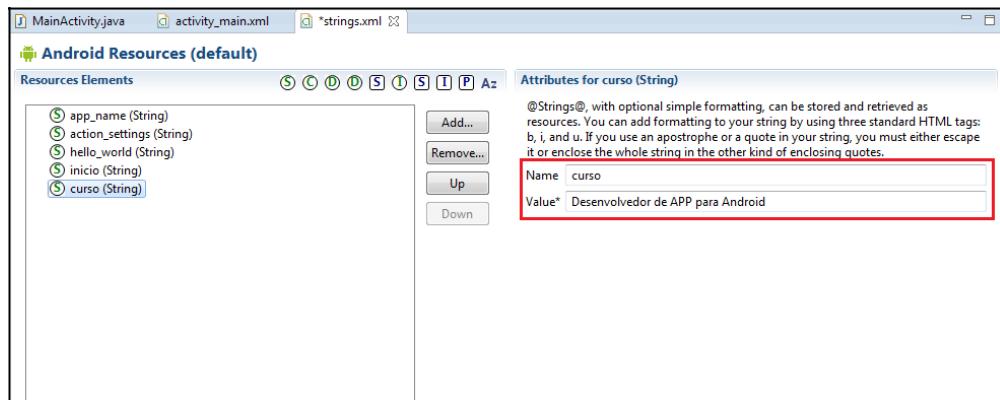
Clique em **Add**.



Selecione a opção **String** e clique em **OK**.

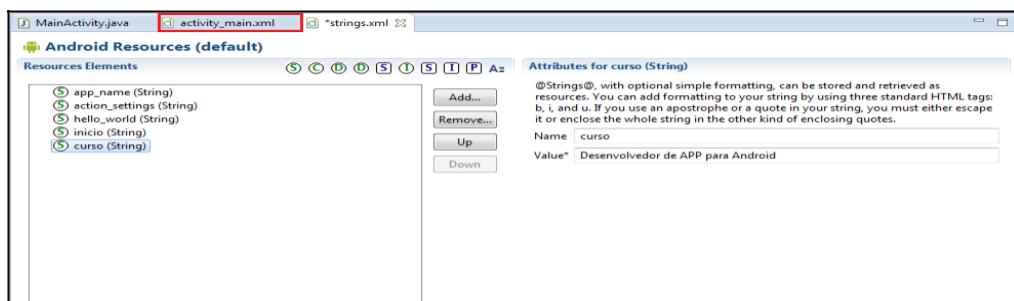


Agora basta definir o **name** que será o nome da string e o **value**, que será o texto que nela deverá estar contido.



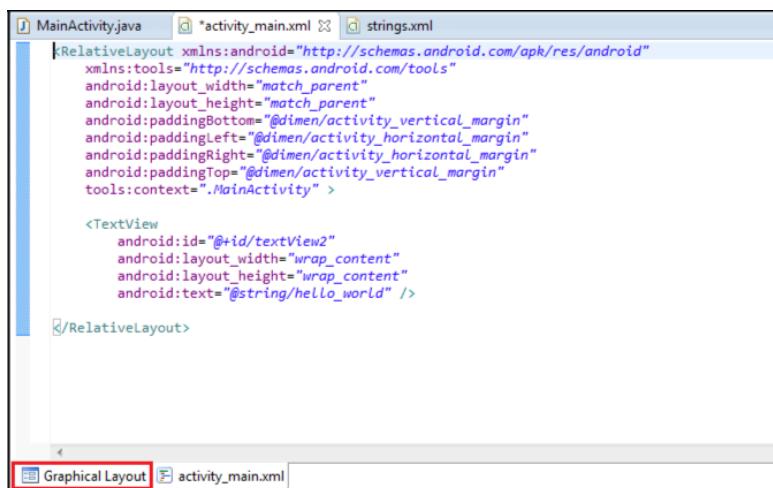
De um **CTRL + S**.

Agora vá até **activy\_main**.



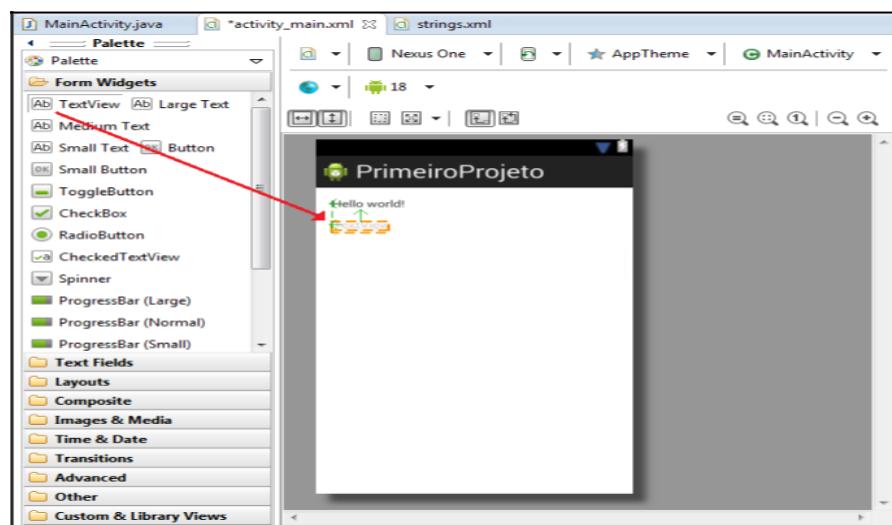
E clique em **CTRL + S**.

Clique em **Graphical Layout**.

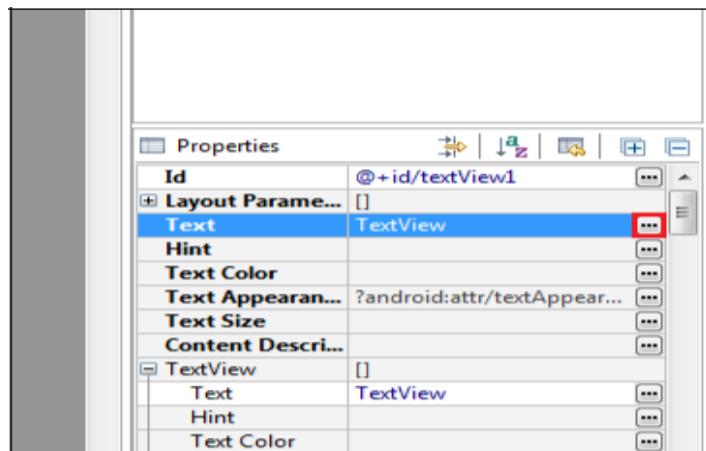


**Adicionando Widgets à aplicação**

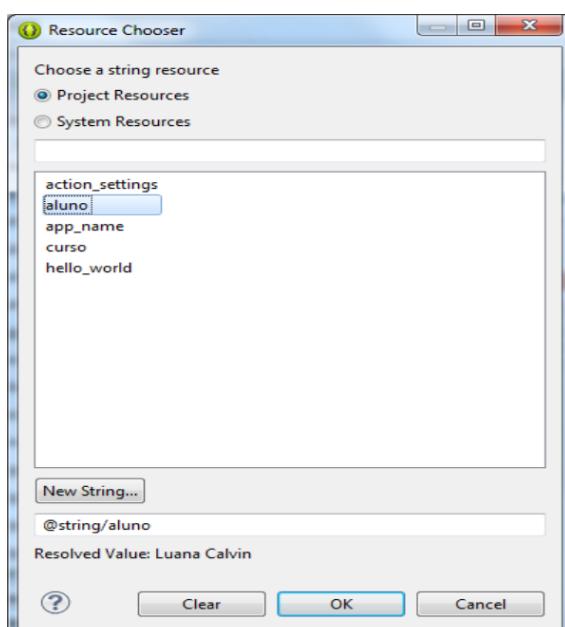
Clique em **TextWiev** e arraste até o local indicado.



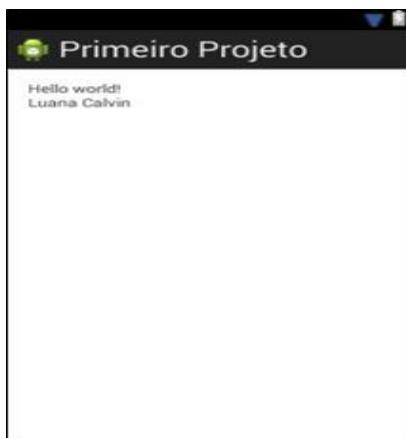
Agora vamos localizar um texto para inserir neste local. Clique no local indicado.



Selecione **aluno** e clique em **OK**.



Veja que o texto que está contido na string **aluno** foi inserido na caixa de diálogo.



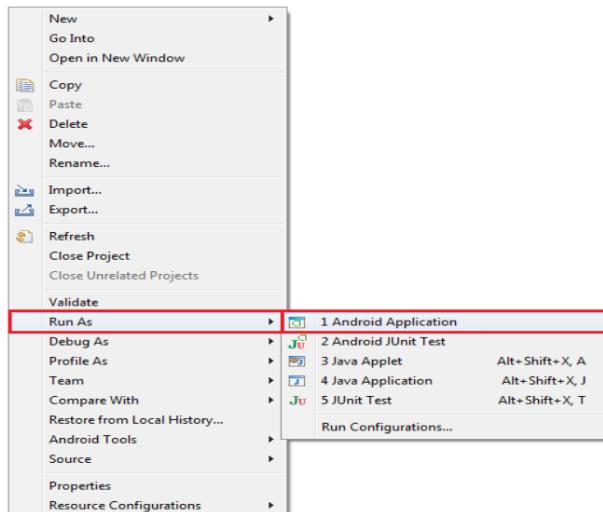
Agora faça a mesma coisa com a string **curso** e de um **OK**.



Pressione **CTRL + S**

### Executando o projeto android

Clique com o botão direito do mouse sobre o título do projeto selecionando as opções indicadas da imagem abaixo.



O emulador poderá demorar alguns minutos até abrir.

Veja as três strings que projetamos.



Muitos bem, chegamos ao final desta segunda aula, não se esqueça de fazer os exercícios desta apostila.

### ***Exercício de Fixação***

- 1)** Abra o Eclipse e crie um novo projeto.
- 2)** Altere o nome do programa para “Primeiro Projeto”.
- 3)** Configure o futuro aplicativo para exibir: “Primeiro projeto do curso: “. E embaixo, em outro *TextView* escreva suas expectativas sobre o curso.
- 4)** Rode o programa e veja como ficou.
- 5)** *Opção adicional:* para otimizar seu Eclipse, você pode clicar com o botão direito sobre seus projetos finalizados e selecionar a opção “Close Project”.

*Obs: Caso ocorra algum problema, você pode acessar o site*

<http://stackoverflow.com/questions/tagged/android> e pesquisar soluções.

## **ANOTAÇÕES**



## **ANOTAÇÕES**



## ANOTAÇÕES



## ANOTAÇÕES

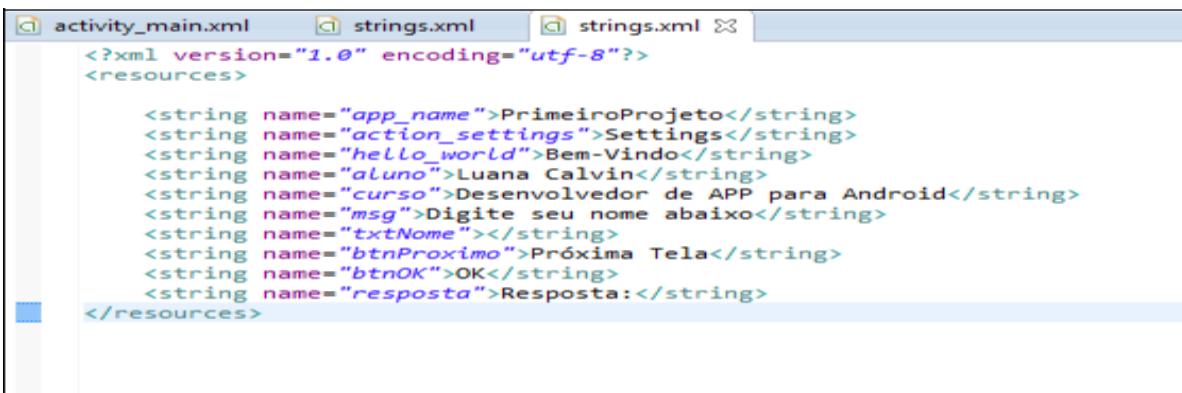


### Trabalhando com eventos

Olá seja bem-vindo a terceira aula do curso de **Desenvolvedor de Android**, nesta aula você irá aprender a trabalhar com eventos para Android, eventos são programação feitas, como por exemplo. Avisar que alguém está realizando uma chamada para o seu aparelho, informar a você que o seu aparelho está com a bateria em nível baixo, etc.

**Muito bem, vamos à prática, abra a pasta strings.xml.**

Abra o seu projeto no eclipse e adicione os recursos **msg**, **txtname**, **btnProximo**, **btnOK** e **resposta**, conforme a imagem abaixo.



```

<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">PrimeiroProjeto</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Bem-Vindo</string>
    <string name="aluno">Luana Calvin</string>
    <string name="curso">Desenvolvedor de APP para Android</string>
    <string name="msg">Digite seu nome abaixo</string>
    <string name="txtName"></string>
    <string name="btnProximo">Próxima Tela</string>
    <string name="btnOK">OK</string>
    <string name="resposta">Resposta:</string>

</resources>

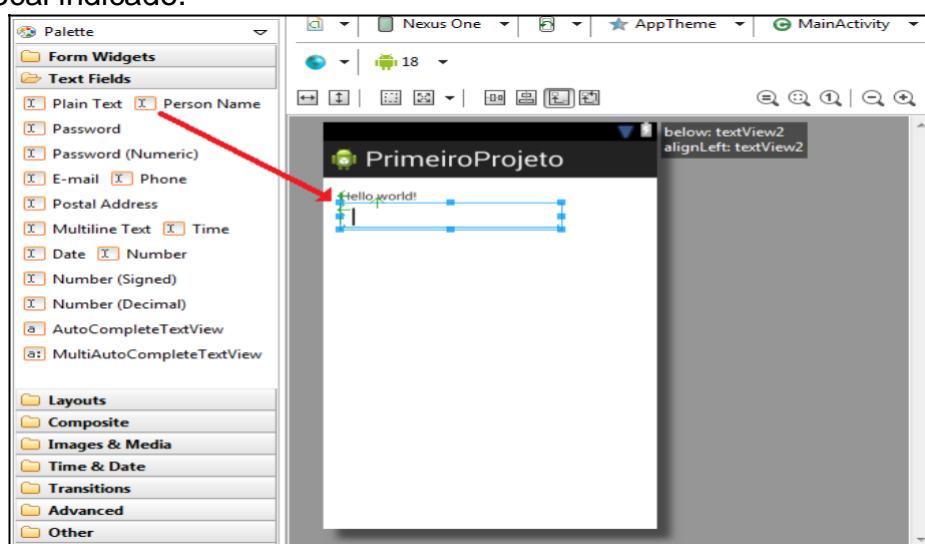
```

Salve as modificações dando um **CTRL + S**.

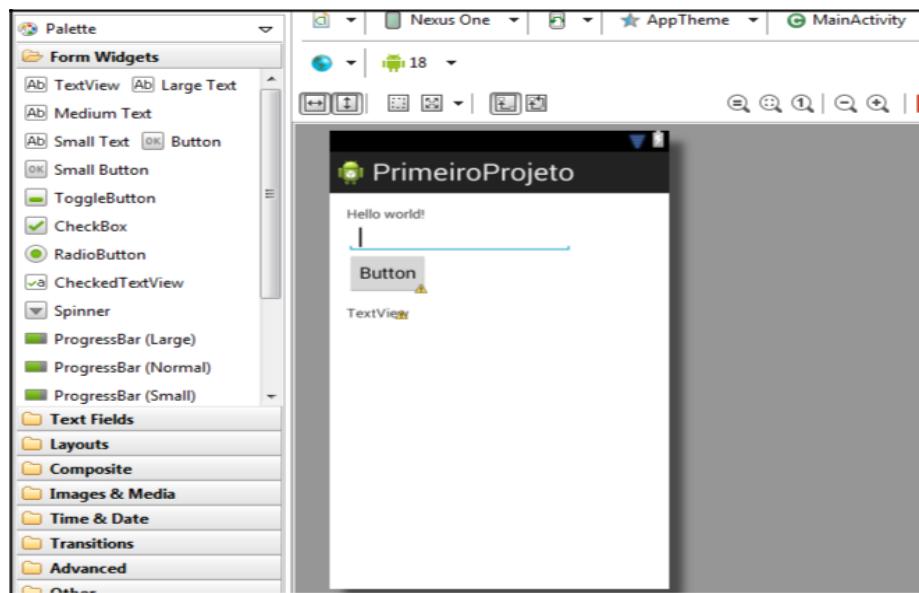
Abra a pasta **Ctivity\_main.xml**.

Remova os dois últimos **TextViews**.

Abra a seção de Palettes **Text Fields** e arraste a opção **Person Name** até o local indicado.



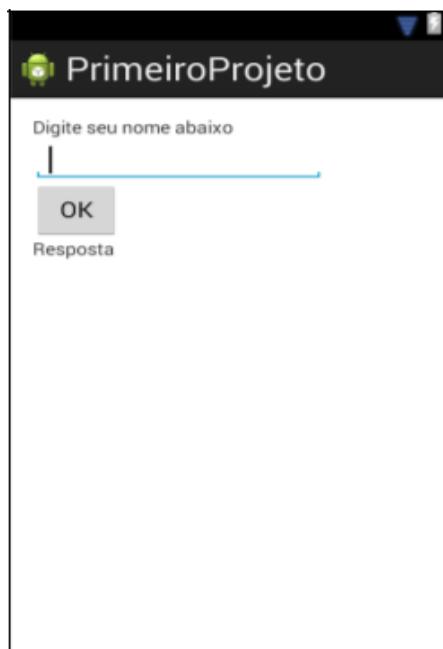
Adicione outros dois Widgets, **Button** e **TextView**.



Altere o texto do Primeiro **TextView** para o recurso **msg**.

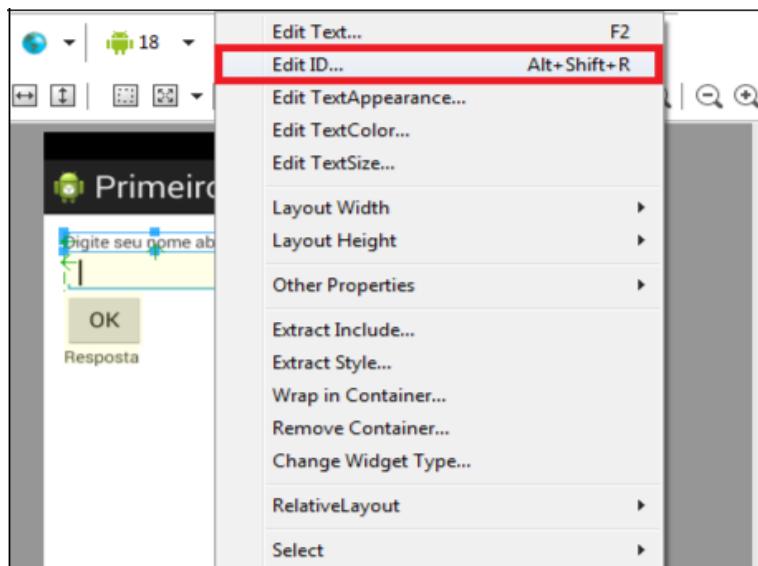
Vincule o recurso **txtNome** ao Widget **EditText**. O recurso **btnOK** ao Widget **Button**, e o recurso **resposta** ao outro **TextView**.

Veja como a tela ficará.

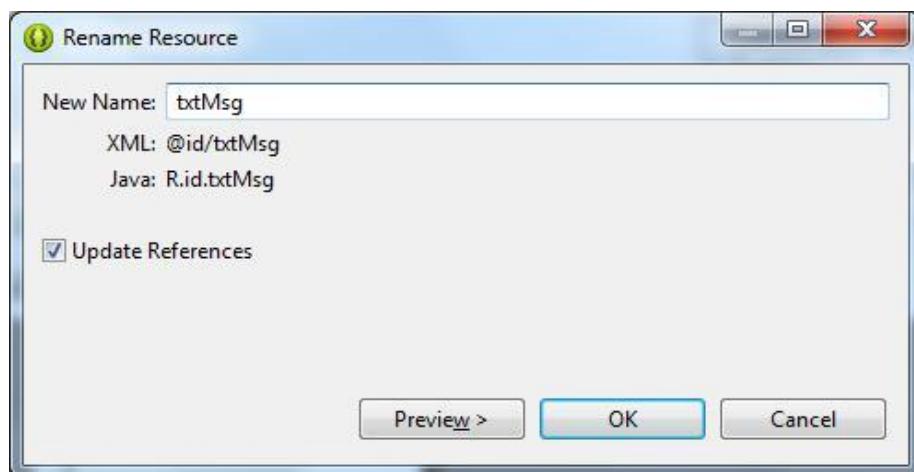


Agora os Widgets serão referenciados no código, para isso é necessário definir os seus IDs.

Clique com o botão direito no primeiro Widgets e selecione **Edit ID...**



Neste campo preencha com o nome **txtMsg**.



Repita o mesmo processo com o **Person name** preenchendo o texto **etxtNome**, agora preencha os nomes **btnOk** para o Button e **txtResposta** para o TexView resposta.

Salve as alterações.

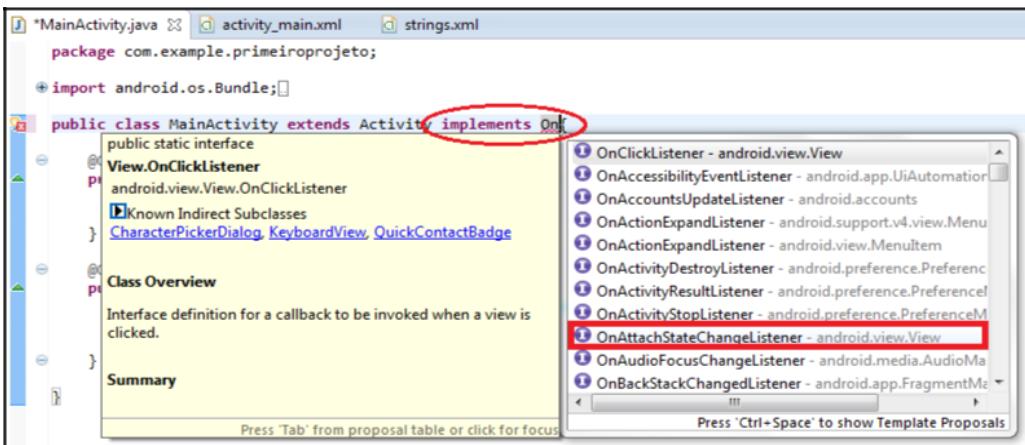
### **Adicionando as alterações**

Para adicionar um evento à classe é necessário implementar uma ou mais interfaces, de acordo com o evento desejado.

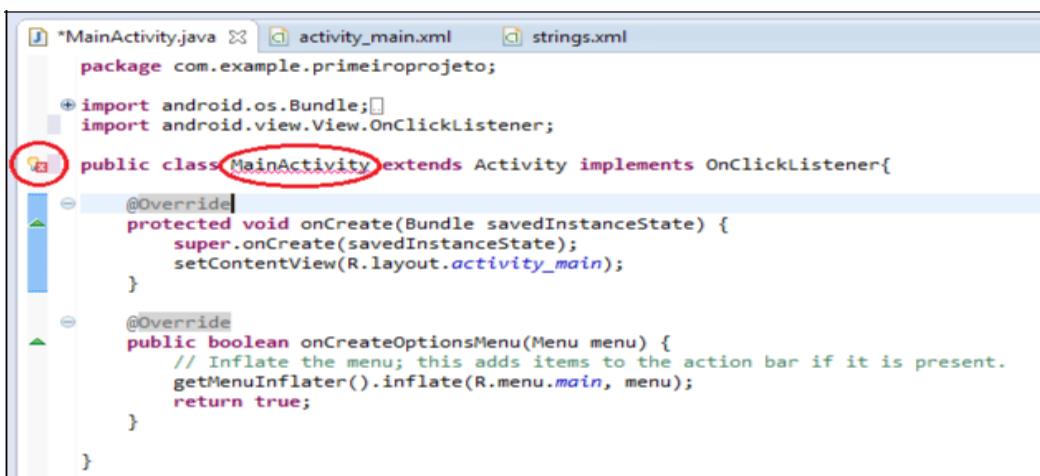
Para implementar o evento “clique no botão”, utilize a interface **OnclickLister** do pacote **android.view.View**.

Clique em **MainActivity.java**.

No código da classe digite **Implements On** e pressione **CTRL + Espaço**. Selecione a opção **OnClicklister** do pacote **android.view.View**.

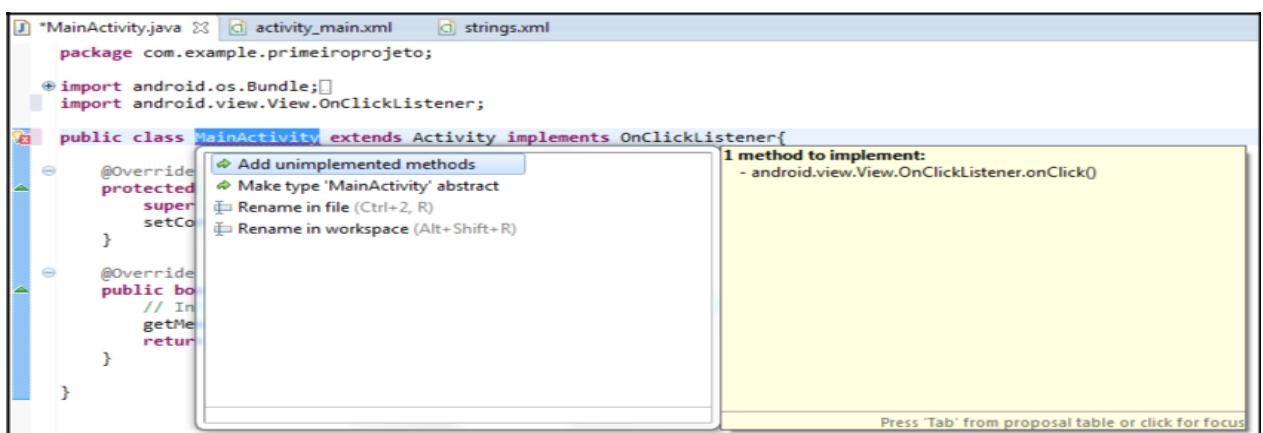


Note que está aparecendo um erro na classe, representado por uma linha vermelha abaixo da palavra Main Activity.java, para identifica-lo clique no ícone de lâmpada com o X vermelho na frente da linha.

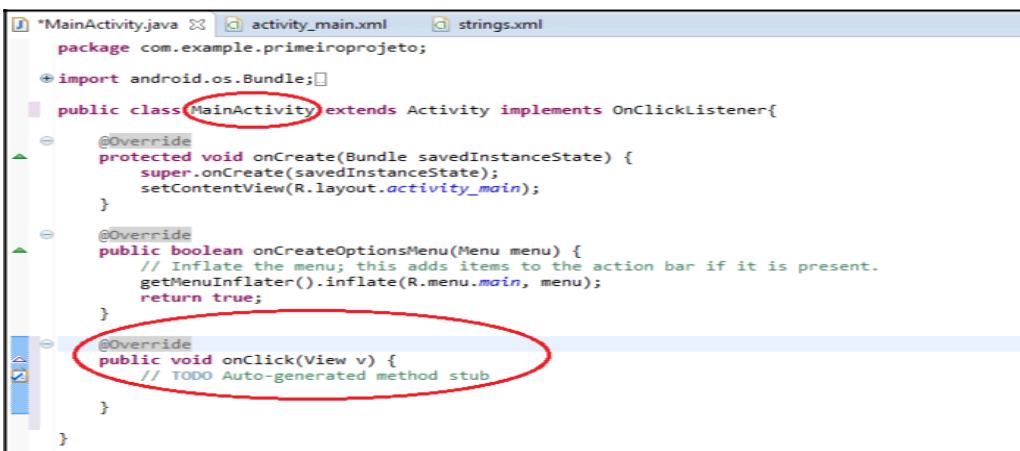


Observe que pela mensagem de erro, está faltando a implementação do método **onClick()**. Como a estruturação da mesma forma que a documentação do Java, o Eclipse já fornece possíveis soluções para o erro.

Selecione a primeira opção sugerida pelo Eclipse (**Add unimplemented methods**).



Veja que as alterações foram efetuadas, e uma nova tag surgiu.



```

*MainActivity.java  activity_main.xml  strings.xml
package com.example.primeiroprojeto;

import android.os.Bundle;

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
    }
}

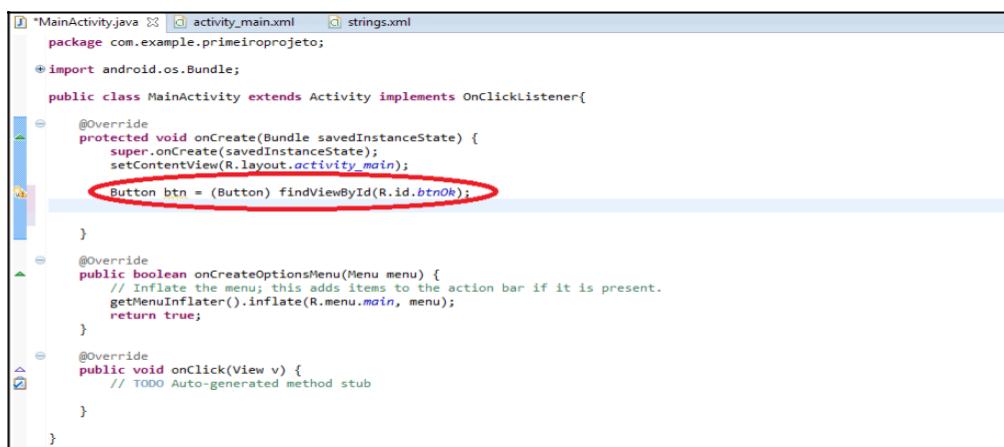
```

## Associando o botão ao evento

Agora abra a pasta **MainActivity.java**.

Digite o código indicado na imagem.

**Button btn = (Button) findViewById (R.id.btnOk);**



```

*MainActivity.java  activity_main.xml  strings.xml
package com.example.primeiroprojeto;

import android.os.Bundle;

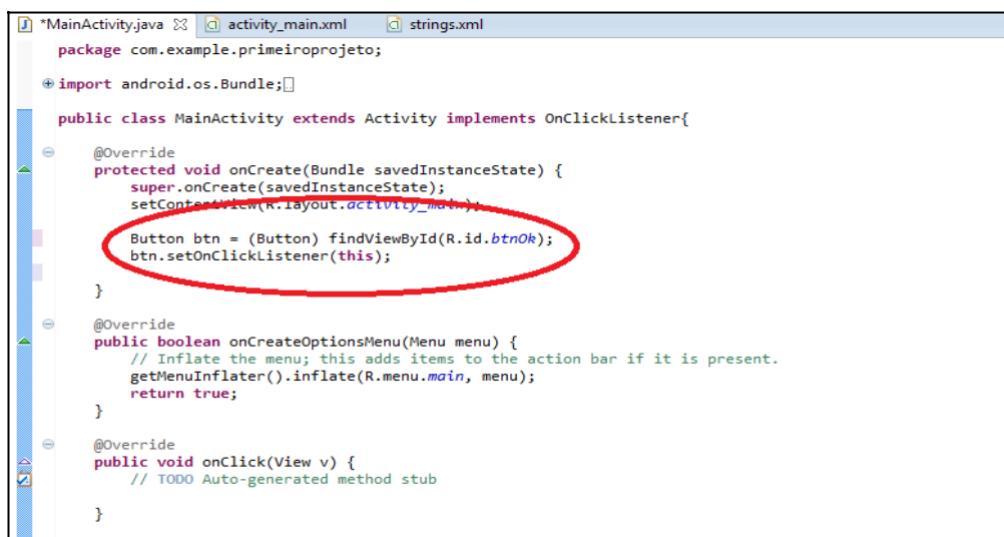
public class MainActivity extends Activity implements OnClickListener{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn = (Button) findViewById(R.id.btnOk);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
    }
}

```

Obs.: insira o código exatamente no mesmo local conforme a imagem. Agora insira o outro código abaixo do que acabamos de digitar. **Btn.setOnClickListener(this);**



```

*MainActivity.java  activity_main.xml  strings.xml
package com.example.primeiroprojeto;

import android.os.Bundle;

public class MainActivity extends Activity implements OnClickListener{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn = (Button) findViewById(R.id.btnOk);
        btn.setOnClickListener(this);
    }

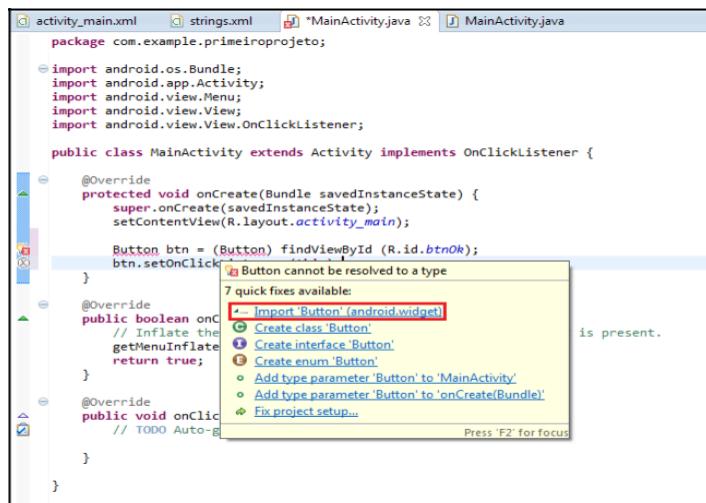
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
    }
}

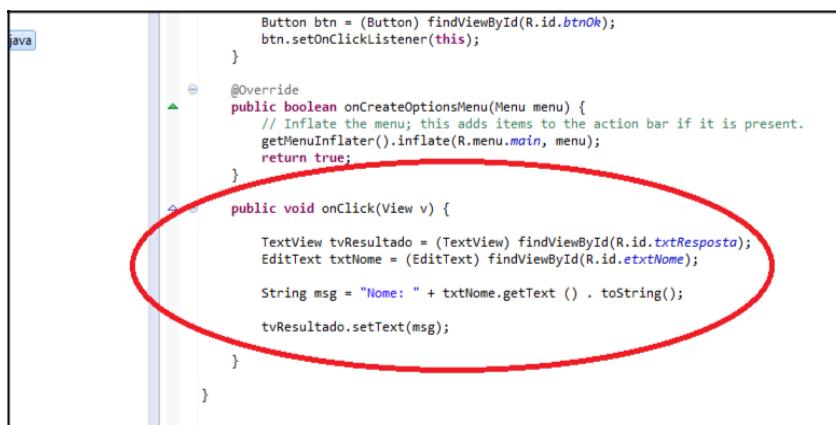
```

Repare que a palavra Button ficou com um sublinhado vermelho indicando um possível erro.

Para concertar isso, basta levar o cursor até o local indicado que uma janela irá surgir, nesta você deve selecionar a opção indicada.

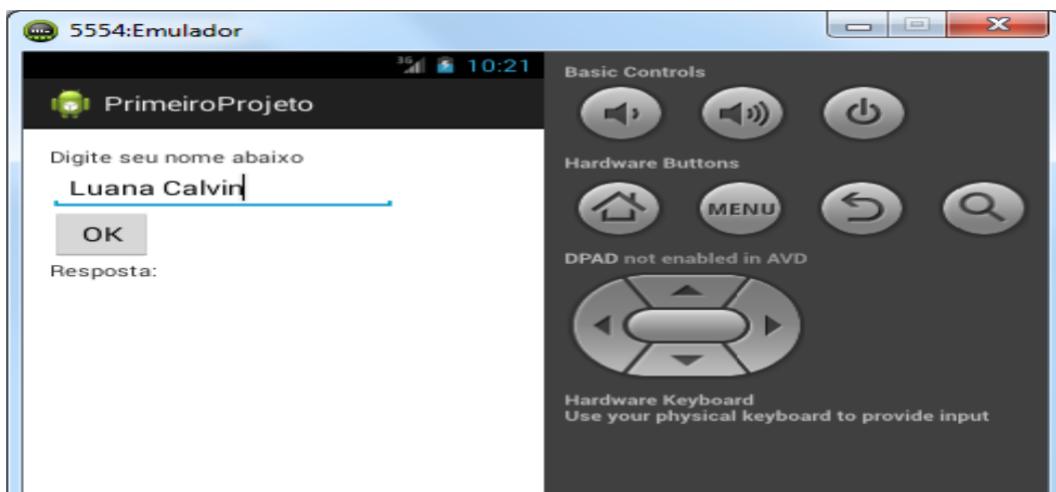


Agora no local indicado insira o código referente a imagem abaixo.



Execute o projeto, isso pode demorar alguns instantes.

No local indicado digite o nome do aluno (a), no caso **Luana Calvin**.



Clique em **OK**.

Veja que o que você digitou no campo acima, apareceu no espaço resposta logo abaixo.



Salve o projeto e feche-o.

Muito bem, chegamos ao final desta aula, não se esqueça de fazer os exercícios desta apostila.

### **EXERCÍCIO DE FIXAÇÃO**

- 1)** Abra o Eclipse e crie um novo projeto.
- 2)** Altere o nome do programa para “Segundo Projeto”.
- 3)** Crie dois campos de texto para informar um nome e uma idade.
- 4)** Abaixo dos campos, crie um botão de confirmação que, quando clicado, exibirá os dados informados em dois TextViews encontrados abaixo do botão.

## ANOTAÇÕES



## ANOTAÇÕES





## **ANOTAÇÕES**

## ANOTAÇÕES



## AULA 4

### Criando uma nova tela

Olá seja bem-vindo a quarta aula do curso de **Desenvolvedor de Aplicativo para Android**, nesta aula você aprenderá a criar novas activities, isto é, novas telas para o aplicativo, estas que são acionadas por botões.

Para criar uma nova activity você deverá seguir os seguintes passos.

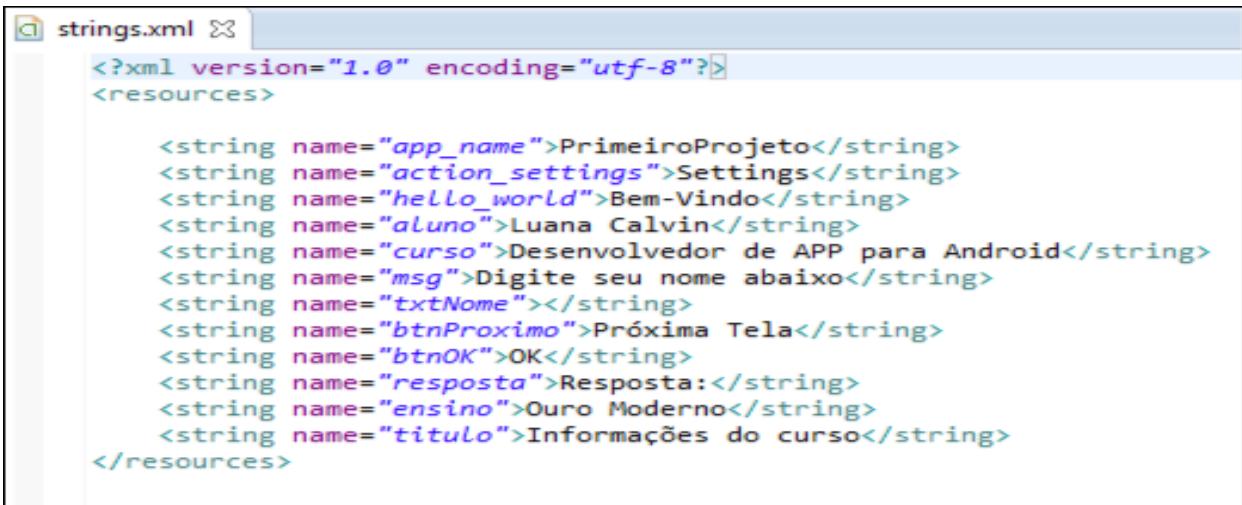
- 1 – Criar os recursos de texto da activity no arquivo **Strings.xml**.
- 2 – Criar um novo arquivo de layout na pasta **res/layout**.
- 3 – Criar uma nova classe Java estendendo a classe Activity.
- 4 – Criar um novo código assim chamando a nova Activity.

Obs.: esses passos devem ser feitos exatamente como estão indicados, caso contrário a activity não será reconhecida.

Abra o Eclipse.

#### 1 – Criando recursos

Adicione no arquivo **strings.xml** os recursos **título** (sem acento) e **ensino**, conforme indicado na seguinte imagem.

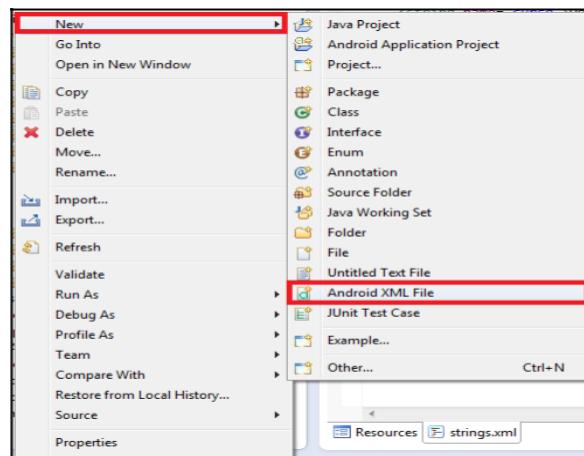


```
<?xml version="1.0" encoding="utf-8"?>
<resources>

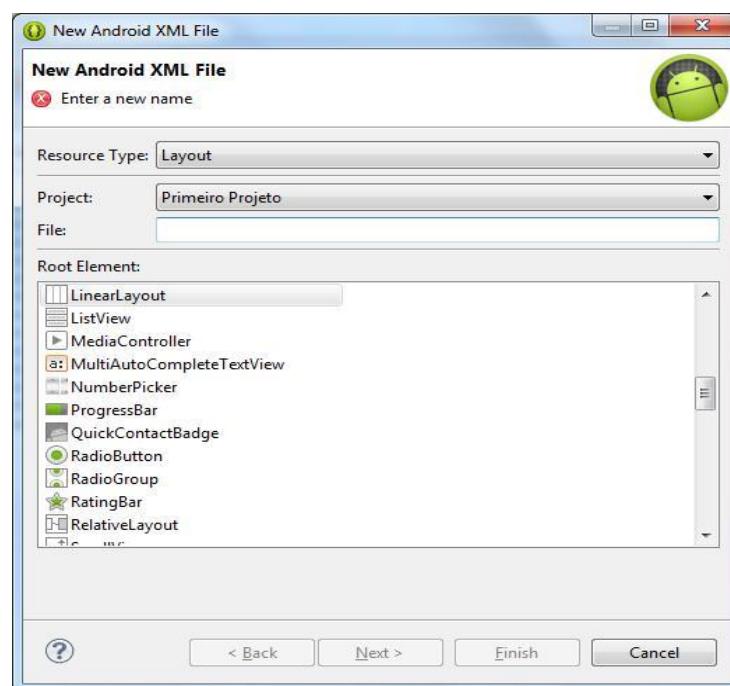
    <string name="app_name">PrimeiroProjeto</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Bem-Vindo</string>
    <string name="aluno">Luana Calvin</string>
    <string name="curso">Desenvolvedor de APP para Android</string>
    <string name="msg">Digite seu nome abaixo</string>
    <string name="txtNome"></string>
    <string name="btnProximo">Próxima Tela</string>
    <string name="btnOK">OK</string>
    <string name="resposta">Resposta:</string>
    <string name="ensino">Ouro Moderno</string>
    <string name="titulo">Informações do curso</string>
</resources>
```

#### 2 – Criando um novo arquivo de layout

Abra a pasta **res** e em **layout** clique no botão direito do mouse e selecione a opção **New > Android** conforme a próxima imagem.



A janela **New Android XML File** será exibida.



No campo chamado de **File** digite o nome **main2** e clique em **Finish**.

Adicione três **textViews** à tela.



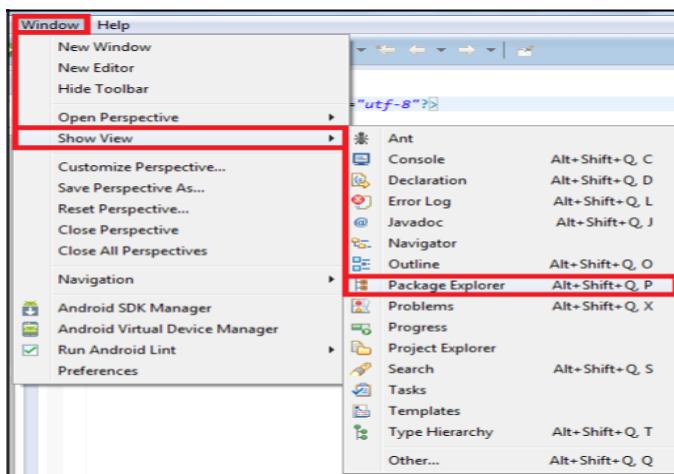
Associe os recursos **título**, **curso** e **ensino** conforme a imagem.  
Salve dando um **CTRL + S**.

### 3 – Criando uma Classe Java

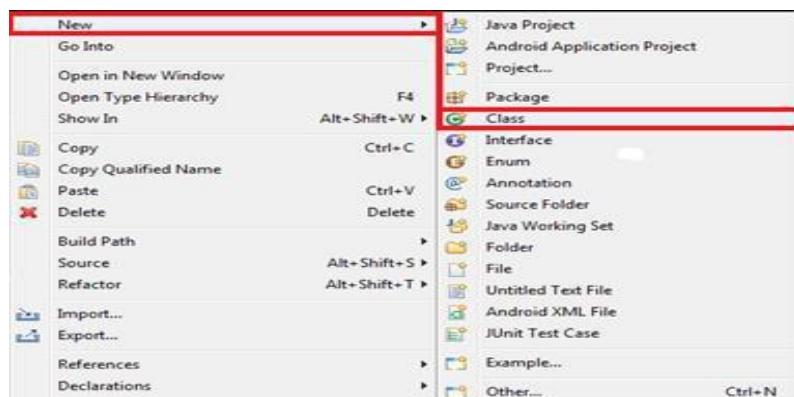
Como explicamos anteriormente para criar uma tela no Android é necessário ter uma classe Java que estende a classe Activity.

Mas primeiro devemos alterar o modo de visualização.

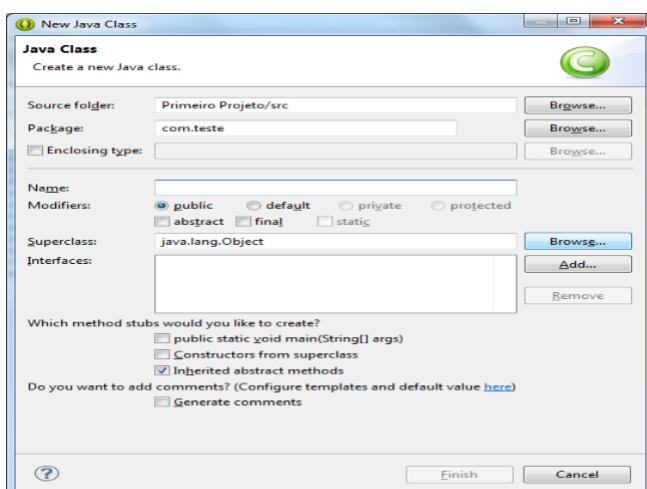
Clique em **Window>Show View>Package Explorer**.



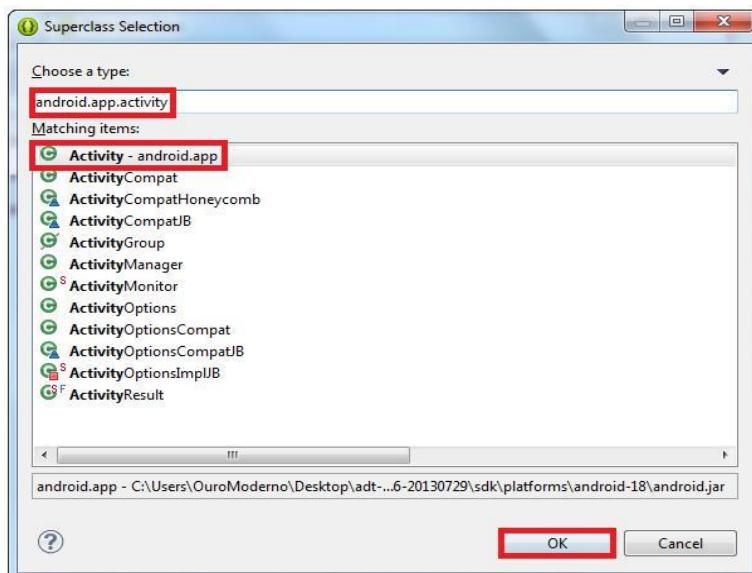
Sobre o pacote **com.teste** clique com o botão direito e selecione a opção **New>Class**.



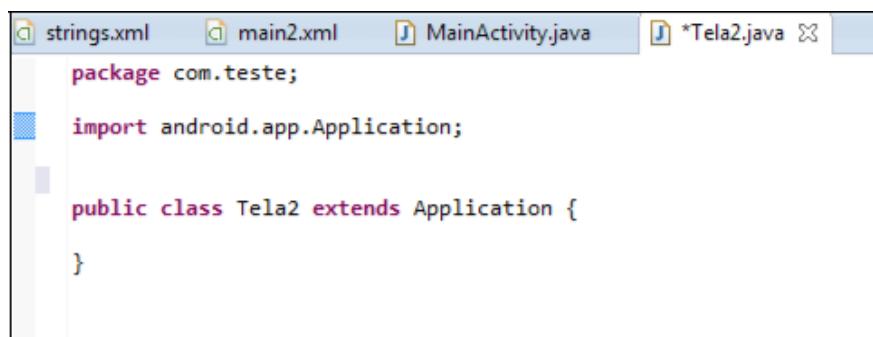
Vamos escolher a classe que irá estender a classe Activity, cliquem no botão **Browse** na opção Superclass.



No campo indicado digite **Android.app**, selecione a opção **Activity - Android.app** e clique em **OK**.



Em **Name** preencha **Tela2** e clique em **Finish**.  
Veja que surgiu um novo arquivo chamado **Tela2**.



```

package com.teste;

import android.app.Application;

public class Tela2 extends Application {
}

```

Agora adicione os códigos conforme a imagem abaixo e salve.



```

package com.teste;

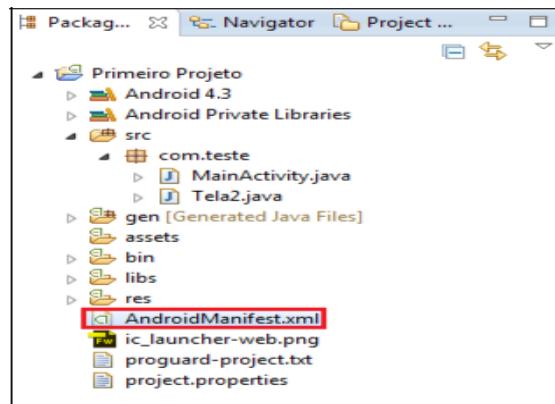
import android.app.Activity;
import android.os.Bundle;

public class Tela2 extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main2);
    }
}

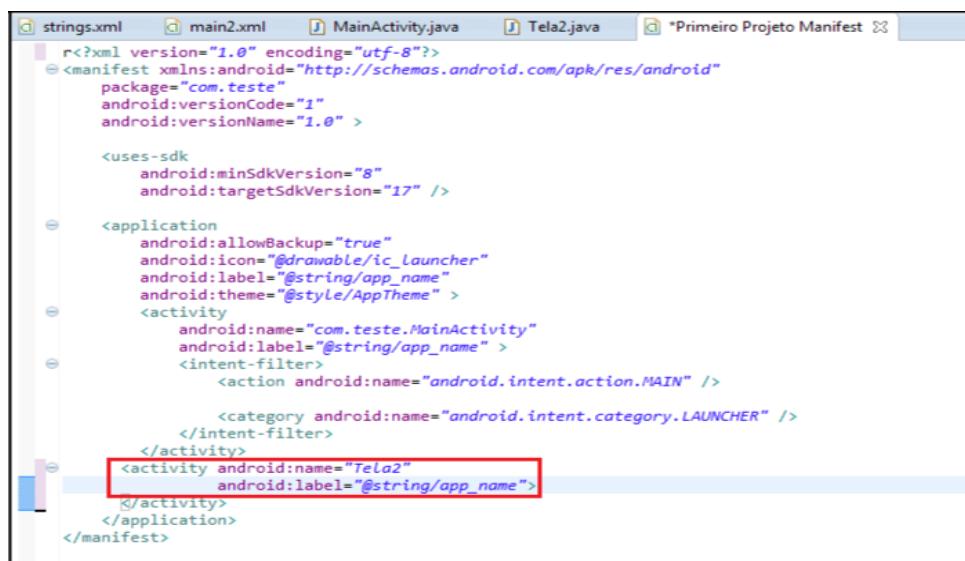
```

#### 4 – Exibindo uma nova tela

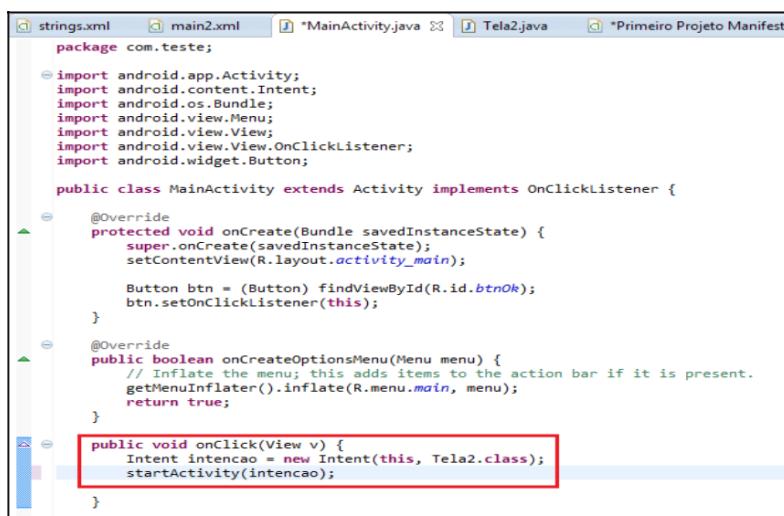
Para exibir a nova tela primeiramente é necessário mapear o activity no arquivo **AndroidManifest.xml**, abra este.



Utilizando a aba **AndroidManifest.xml** insira o código, conforme a imagem abaixo.



Agora altere o código do evento **onClick** para abrir a segunda tela, no activity **MainActivity.java**, conforme abaixo.



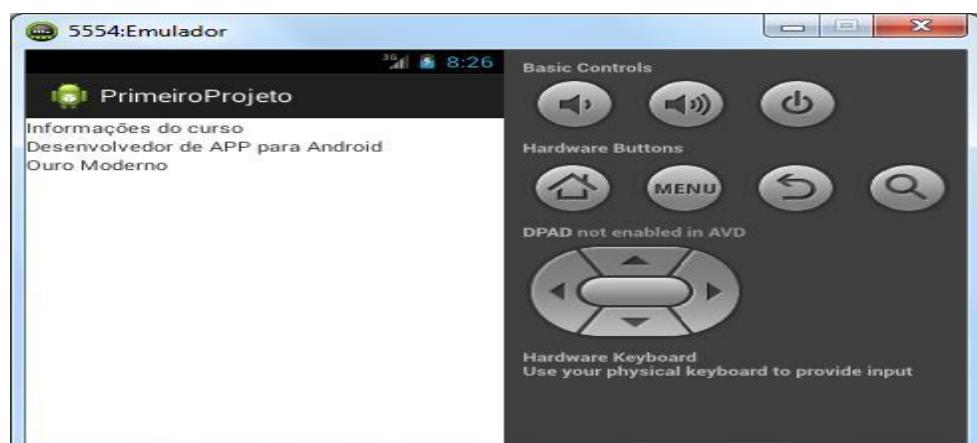
## Executando o projeto visualizando a nova tela

Execute o projeto.



Clique em **OK**.

Veja que o emulador nos levou a um próxima tela.



### Voltando para a tela anterior

Para retornar a tela anterior será necessário adicionar um botão na tela **main2.xml** e programa-lo para retornar.

Crie uma nova string referente ao botão.



```

<?xml version="1.0" encoding="utf-8"?>
<resources>

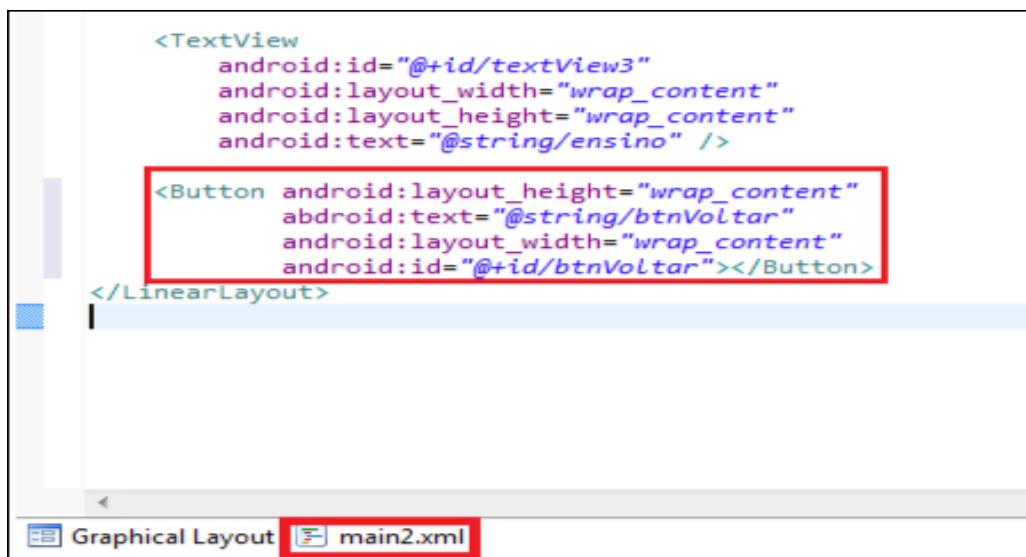
    <string name="app_name">PrimeiroProjeto</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Bem-Vindo</string>
    <string name="aluno">Luana Calvin</string>
    <string name="curso">Desenvolvedor de APP para Android</string>
    <string name="msg">Digite seu nome abaixo</string>
    <string name="txtNome"></string>
    <string name="btnProximo">Próxima Tela</string>
    <string name="btnOK">OK</string>
    <string name="resposta">Resposta:</string>
    <string name="ensino">Ouro Moderno</string>
    <string name="titulo">Informações do curso</string>
    <string name="btnVoltar">Voltar</string>
</resources>

```

Vá até o arquivo **main2.xml** clique em **Graphical Layout** e insira um botão até o local indicado.

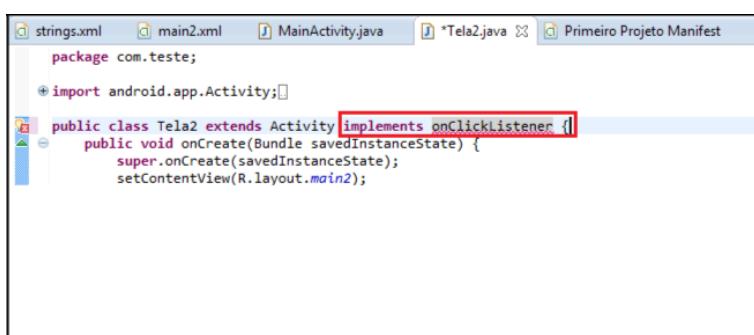


Clicando em **main2.xml** e insira o código no local indicado.



De um **CTRL + S**.

Selecione o arquivo **Tela2** e insira o código **Implements onClickListener** no local indicado.



Selecione a opção indicada.



```

package com.teste;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class Tela2 extends Activity implements OnItemClickListener {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main2);
        Button btnVoltar = (Button) findViewById(R.id.btnVoltar);
        btnVoltar.setOnClickListener(this);
    }

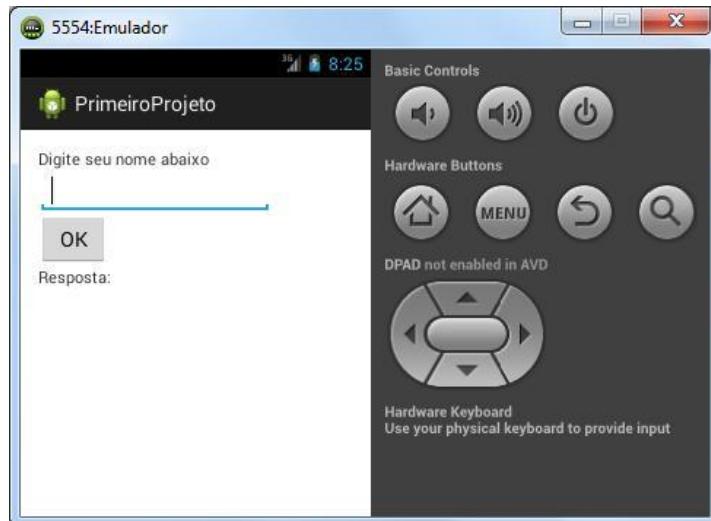
    public void onClick(View v) {
        finish();
    }
}

```

Faça o mesmo com as outras palavras que possam surgir com esta linha vermelha.

### Executando a nova tela

Execute o projeto conforme aprendemos anteriormente, clique em **OK**.



Agora clique em **Voltar**.



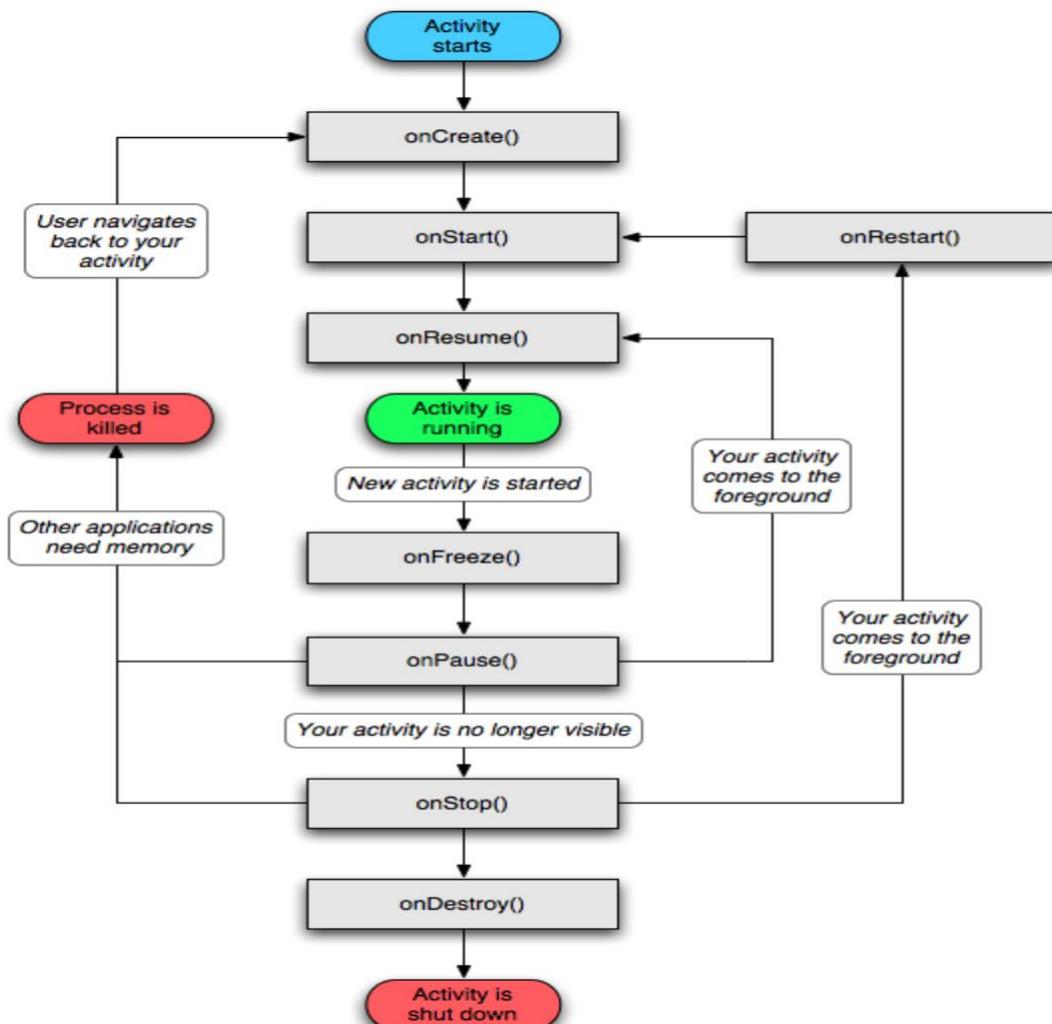
Note que o emulador nos enviou novamente para a primeira tela.



## Ciclo de vida das Activities

As activities são armazenadas na memória como se fossem uma pilha (stack) de telas, uma em cima da outra, estando visível para o usuário a que está por cima de todas.

Quando uma activity é criada ela é apresentada no topo da pilha, a criada anteriormente ficará logo abaixo.



**oneCreate()**: Método chamado quando a atividade é criada, nele são criadas as views e obtidos os dados que irão carregar as listas da tela.

**onStart()**: Esse método é utilizado quando a atividade está no topo da pilha e o usuário já pode interagir com a tela.

**onResume()**: Ele é chamado quando a atividade está no topo da pilha e o usuário já pode interagir com a tela.

**onPause()**: Chamado quando a atividade está começando a ser substituída ou a aplicação está começando a se encerrar. Geralmente é utilizado para gravar as informações que ainda não foram salvas.

**onRestart()**: Esse método ocorre quando a atividade é pausada e chamada novamente.

**onStop()**: Esse método ocorre quando a atividade não estiver sendo executada e saiu do topo da pilha de atividades.

**onDestroy()**: Chamado quando a atividade é finalizada; pode ser por código ou quando o sistema precisa liberar recursos.

## Entendendo o ciclo de vida

Assim que uma atividade é iniciada, é colocada no topo da pilha de atividades e se torna uma atividade em execução, caso exista alguma atividade anterior, ela ficará em uma posição logo abaixo na pilha e só passará para o estado de execução quando a atividade acima for finalizada.

Uma atividade pode se encontrar em quatro estados, são eles:

**Executando** – A atividade que está ativa na tela do dispositivo.

**Parada** – Uma atividade que perdeu o foco para outra, mas que mantém todas as informações de estado, porém sem interagir com o usuário. Pode ser finalizada se o sistema operacional necessitar de memória (recursos).

**Interrompida** – Caso a atividade não esteja sendo utilizada, mas que mantém suas informações de estado, porém é finalizada para a liberação da memória, perdendo suas informações.

**Finalizada** – Quando a atividade é finalizada pelo sistema operacional ou por uma implementação do desenvolvedor.

Tanto no status de parada como no de interrompida, o sistema pode finalizar a atividade, caso os recursos estejam baixos, sem que sejam chamados os métodos `onDestroy` ou `onStop`, como demonstra o diagrama acima. Podem ocorrer três ciclos com uma atividade.

**Ciclo completo** – Inicia-se com o método `onCreate()` e a atividade realiza toda a configuração, passando de um estado para o outro e terminando no método `onDestroy` ou `onStop`, como demonstra o diagrama acima.

**Ciclo da vida visível** – Inicia-se no método `onCreate()` e termina no método `onStop()`. É o ciclo onde a atividade está disponível e visível para o usuário, mesmo que ele não esteja interagindo com a tela. Quando o usuário não visualizar mais a atividade, o método `onStop()` é chamado, ou então, para tornar a atividade visível, chama-se o método `onStart()`.

**Primeiro ciclo da atividade** – Ocorre entre os métodos `onResume()` e `onPause()`. Este é o período em que a atividade está visível para o usuário, no topo da pilha de atividades. O código utilizado nesse período não pode ser pesado, pela iminente possibilidade de troca dos estados `onPause()` e `onResume()`.

## Passando dados entre as telas

A passagem de informações de uma activity para outra é bem simples, basta na primeira activity utilizar uma distância de classes bundles (basicamente uma coleção de pares chaves/valores), adicionar algumas chaves na coleção e, por último, adicionar a colação para a **Intent** através do método **putExtras**.

Abra a pasta **MainActivity.java** e insira os códigos abaixo no local indicado.

```
public void onClick(View v) {
    Intent intencao = new Intent(this, Tela2.class);
    EditText txtNome = (EditText) findViewById(R.idetxtNome);

    String nome = txtNome.getText().toString();

    Bundle paramentros = new Bundle();
    paramentros.putString("nome", nome);

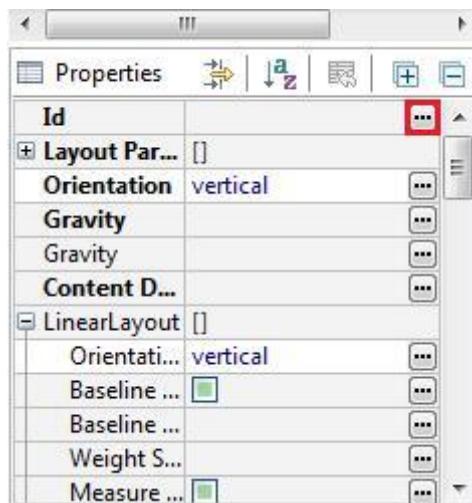
    intencao.putExtras(paramentros);

    startActivity(intencao);
}
```

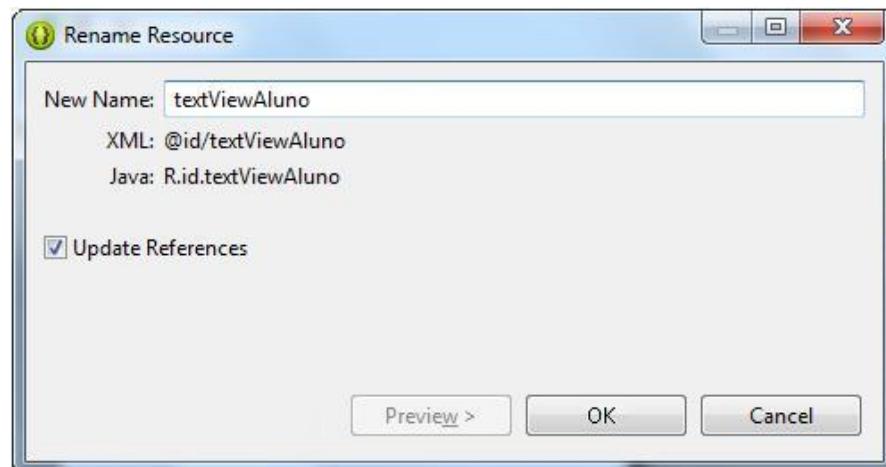
Abra a pasta **main2** e insira um **textView** entre o **textView** referente a **String ensino** e o botão **Voltar**.



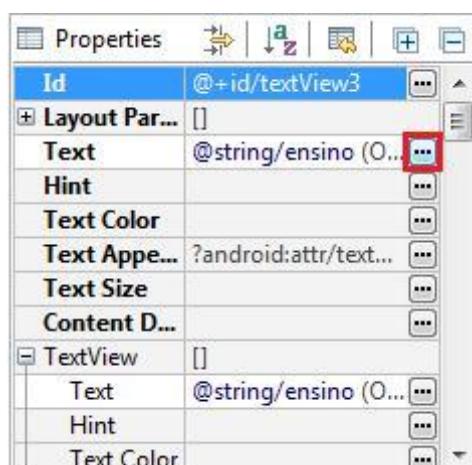
Clique no local indicado, vamos definir um nome para este textWiew.



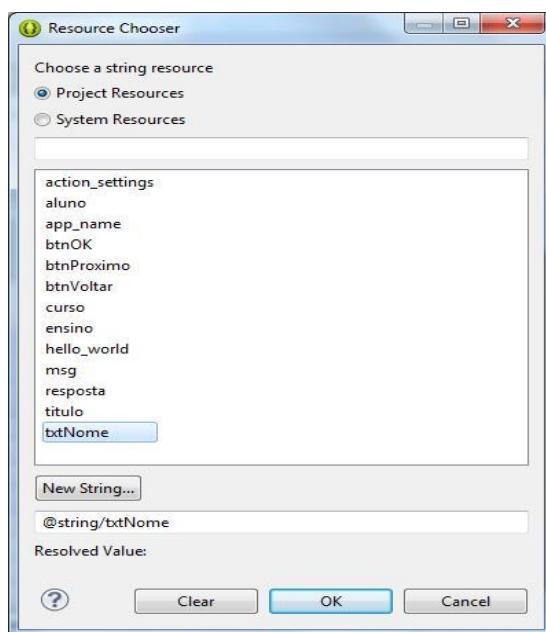
Defina o nome **textViewAluno** e clique em **OK**.



Agora clique no local indicado para referenciar um string a este textView.  
Clique no local indicado.



Selecione a opção txtNome, de um duplo clique nesta ou clique em OK.



Agora abra a pasta **Tela2** e adicione os códigos nesta, conforme a imagem baixo.

```
+ import android.app.Activity;■
public class Tela2 extends Activity implements OnClickListener {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main2);

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

        String nome = args.getString("nome");

        TextView viewNome = (TextView) findViewById(R.id.textViewAluno);
        viewNome.setText("Aluno:" + nome);

        Button btnVoltar = (Button) findViewById(R.id.btnVoltar);
        btnVoltar.setOnClickListener(this);
    }
}
```

Execute o arquivo.

No local reservado para inserção de texto, digite o seu nome e pressione OK.



Veja que o emulador está exibindo na segunda tela, o texto que você acabará de inserir na primeira.



Muito bem, chegamos ao fim desta quarta aula, não se esqueça de fazer os exercícios desta apostila.

### ***EXERCÍCIO DE FIXAÇÃO***

- 1)** Abra o Eclipse e crie um novo projeto.
- 2)** Altere o nome do programa para “Terceiro Projeto”.
- 3)** Crie uma activity para receber os dados enviados pela primeira activity.
- 4)** A primeira activity deve conter um botão de confirmar para ir para a próxima, enquanto a segunda deve possuir um botão de confirmação para voltar para a primeira.

Abaixo, crie um botão de confirmação que, quando clicado, exibirá os dados informados em dois *TextViews* encontrados abaixo do botão.

## NOTAÇÕES



NOTAÇÕES



## NOTAÇÕES



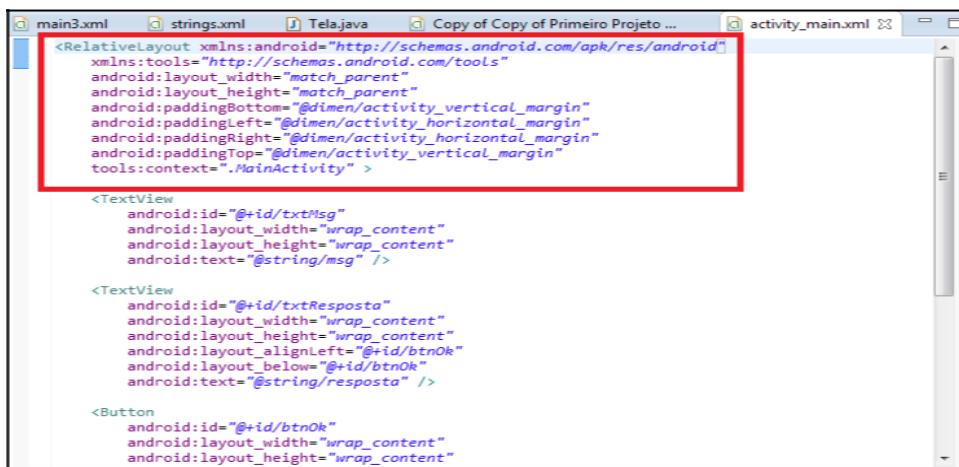
## NOTAÇÕES



### Layouts

Olá seja bem-vindo a quinta aula do curso de desenvolvedor de Aplicativos para Android, nesta aula você irá aprender a trabalhar com layouts de elementos dentro das activity, como por exemplo centralizar, alinhar a esquerda, direita, organizar na vertical ou horizontal, enfim com as várias posições em que os elementos podem se encontrar.

Até agora foi usado o layout padrão para a construção da estrutura da activity, para verificar basta abrir o código-fonte de um arquivo de layout (**activity\_main.xml** ou **main2.xml**).



```

<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

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

    <TextView
        android:id="@+id/txtResposta"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id	btnOk"
        android:layout_below="@+id	btnOk"
        android:text="@string/resposta" />

    <Button
        android:id="@+id	btnOk"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

```

O **LinearLayout** posiciona os objetos na sequência, um após o outro, na horizontal ou vertical, dependendo do atributo **orientation**.

O Android disponibiliza uma série de opções de layouts, sendo os principais.

- **LinearLayout**
- **TableLayout**
- **RelativeLayout**
- **AbsoluteLayout**
- **FrameLayout**

A primeira opção é o layout padrão, usado ao longo do curso e demonstrado no exemplo acima. Os demais serão demonstrados.

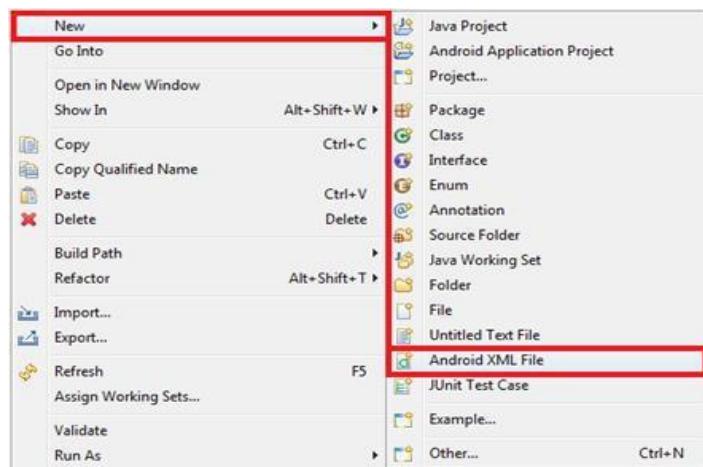
#### **TableLayout**

Ele permite trabalhar com linhas como se fosse um html porém bem mais limitada e as colunas são criadas automaticamente quando adicionado um widget.

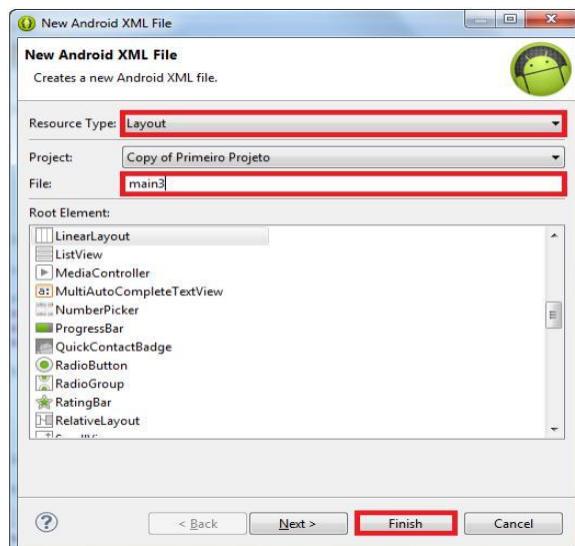
Crie um novo arquivo de layout chamado **main3**. Primeiro selecione a opção layout conforme mostra a próxima imagem.

Clique com o botão direito do mouse em **Layout**.

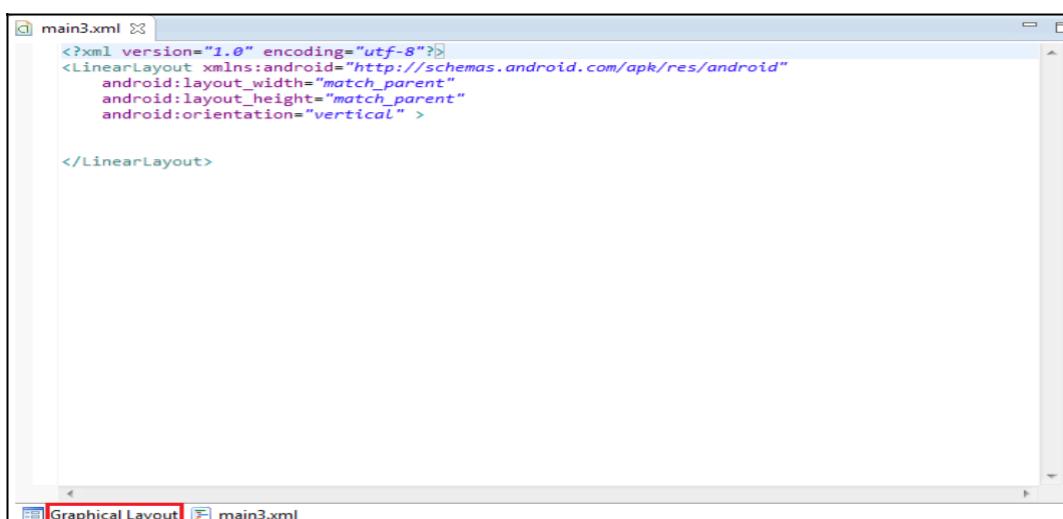
Clique em **New>Android XML File**.



Selecione a opção **Layout** em **Resource Type**, digite **main3** em **File** e clique em **Finish**.

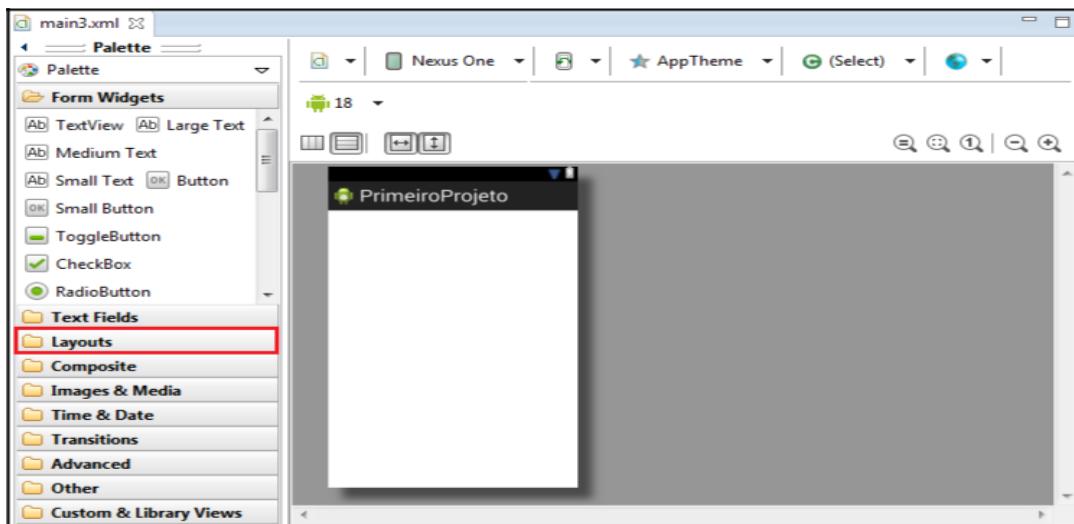


Mude para o modo gráfico e clique na aba **Graphical Layout**.



## Adicionando linhas ao Layout

Para adicionar linhas ao layout selecione o objeto **TableRow**, presente na pasta **Layout** do **Palette**.



Siga os passos.

- 1- Adicione um **TableRow**.
- 2- Insira um **TextView**.
- 3- Abra a pasta **Text Fields**.
- 4- Adicione um **Person Name**.
- 5- Adicione mais um **TableRow**.
- 6- Adicione um **Button**.

Crie uma nova String igual a referida logo abaixo.

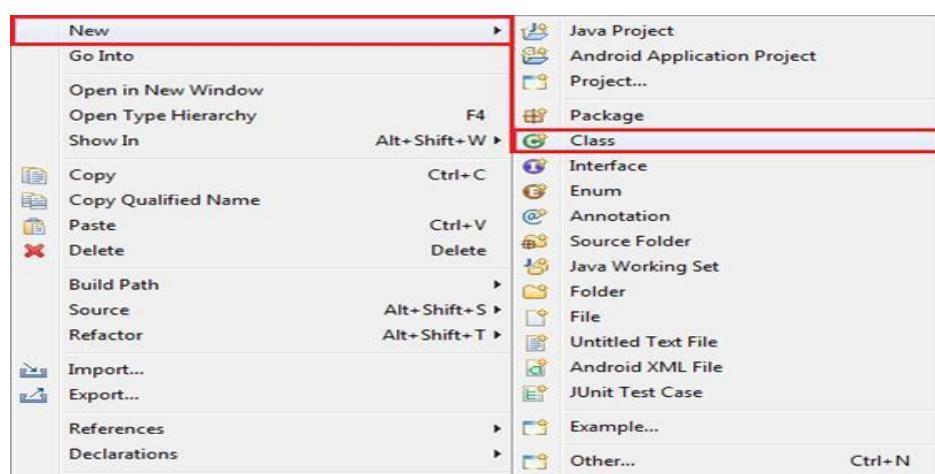
```
<string name="nome">Nome:</string>
```

Referencie os Widgets recentemente criados as string indicadas.

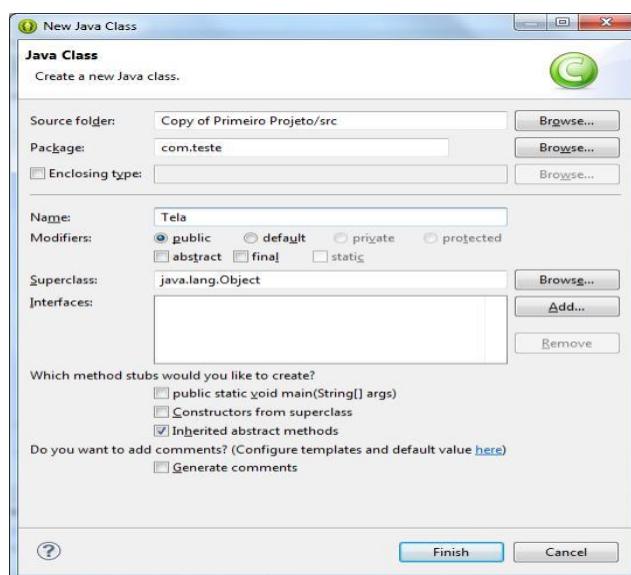
**TextWiew – nome**  
**Person name – txtname**  
**Button – btnOk**

## Criando a Activity

Crie uma classe java que irá estender a classe Activity do Android, conforme mostra o código a seguir.



Aplique o nome **Tela** para esta nova classe e clique em **Finish**.



Complete o arquivo com o seguinte código.

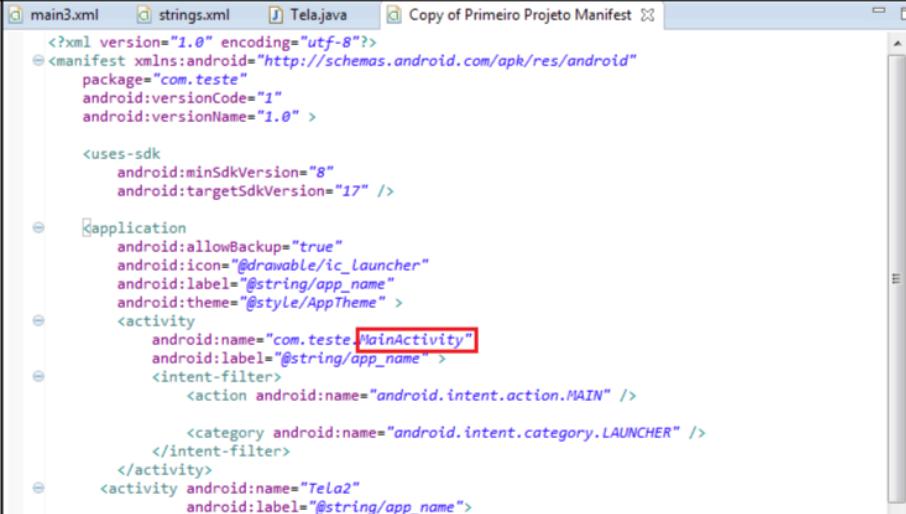
```

package com.teste;
import android.app.Activity;
import android.os.Bundle;

public class Tela extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main3);
    }
}

```

Para definir esta activity como tela inicial da aplicação, é preciso modificar a linha, alterando a palavra **MainActivity** por **Tela** na pasta **AndroidManifest.xml**.



```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.teste"
    android:versionCode="1"
    android:versionName="1.0" >

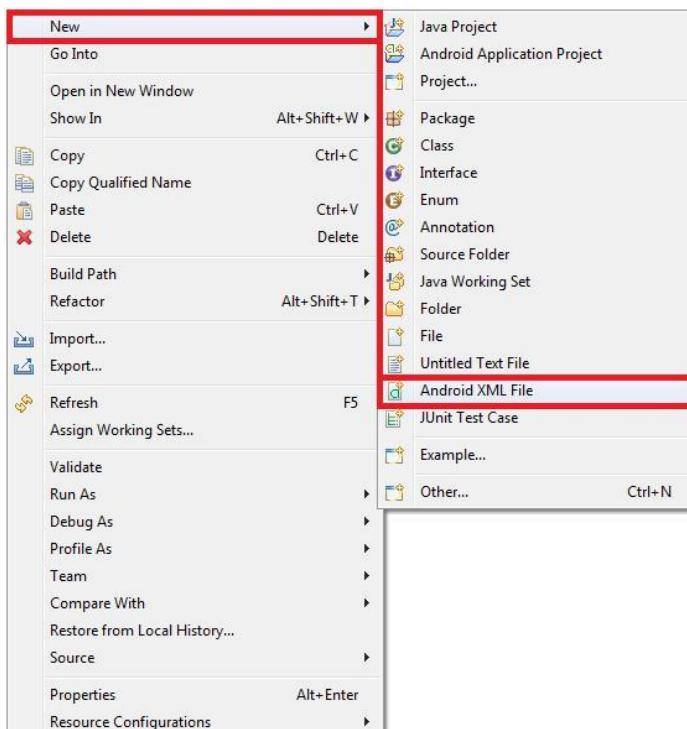
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.teste.MainActivity"
            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="Tela2"
            android:label="@string/app_name">
    
```

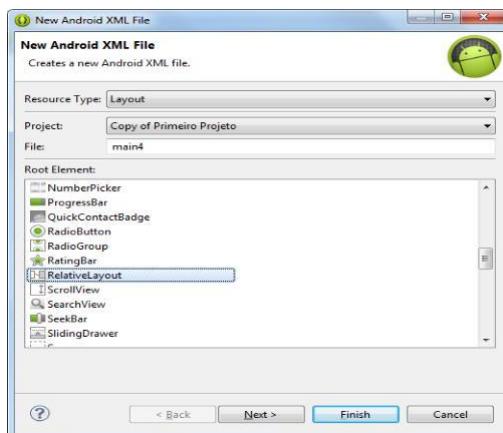
## Estilos de layout

Quando criada a Activity, está pode ser configurada em diversos tipos de layouts, que podem facilitar na organização da posição dos widgets dentro das telas. Vamos conhecer alguns tipos de layout disponíveis no eclipse.

Primeiramente vamos criar um novo arquivo em **xml** conforme aprendemos anteriormente.



Aplique o nome **main4** para a tela, selecione a opção **RelativeLayout** e clique e **Finish**.



Clique em **Graphical Layout**.



Adicione os mesmos Widgets e referenciae estes com as mesmas strings do exemplo anterior.



Neste estilo os widget podem ser inseridos exatamente onde você desejar, o tamanho dos widgets também pode ser configurado através dos códigos.

```

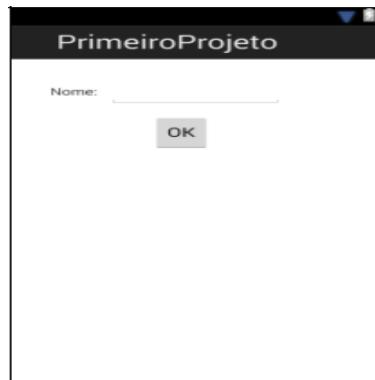
<Button
    android:id="@+id/button1"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_below="@+id/editText1"
    android:layout_marginLeft="38dp"
    android:layout_marginTop="14dp"
    android:layout_toRightOf="@+id/textView1"
    android:text="@string/btnOK" />

<EditText
    android:id="@+id/editText1"
    android:layout_width="150dp"
    android:layout_height="40dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="26dp"
    android:ems="10"
    android:inputType="textPersonName"
    android:text="@string/txtNome" />

<TextView
    android:id="@+id/textView1"
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:layout_alignBaseline="@+id/editText1"
    android:layout_alignBottom="@+id/editText1"
    android:layout_toLeftOf="@+id/editText1"
    android:text="@string/nome" />

```

Repare que o formato dos Widgets foram alterados.



Agora crie um novo arquivo em **xml** chamado **main5** e aplique o estila **AbsoluteLayout**.

Nas variáveis X e Y aplique os seguintes valores de códigos.

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="100dp"
    android:layout_y="100dp"
    android:ems="10"
    android:inputType="textPersonName"
    android:text="@string/txtNome" >

    <requestFocus />
</EditText>

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="110dp"
    android:layout_y="140dp"
    android:text="@string/btnOK" />

<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="49dp"
    android:layout_y="115dp"
    android:text="@string/nome" />
```

Veja como os Widgets ficarão posicionados.



Você pode aplicar vários estilos de Layout, faça testes com os outros exemplos.

Muito bem, chegamos ao fim desta aula, não se esqueça de fazer os exercícios desta apostila.

### **EXERCÍCIO DE FIXAÇÃO**

O objetivo da aula é familiarizar você com os diferentes layouts que o Android nos proporciona. Para você escolher algum de sua preferência, você deve primeiro experimentar os diversos modelos disponíveis.

- 1)** Abra o Eclipse e crie um novo projeto.
- 2)** Altere o nome do programa para “Quarto Projeto”.
- 3)** Com esse projeto recém criado como base, coloque alguns widgets pela tela.
- 4)** Crie uma activity para cada tipo de layout.
- 5)** Utilize todos os recursos de cada layout.

## NOTAÇÕES





## NOTAÇÕES



## NOTAÇÕES

## NOTAÇÕES



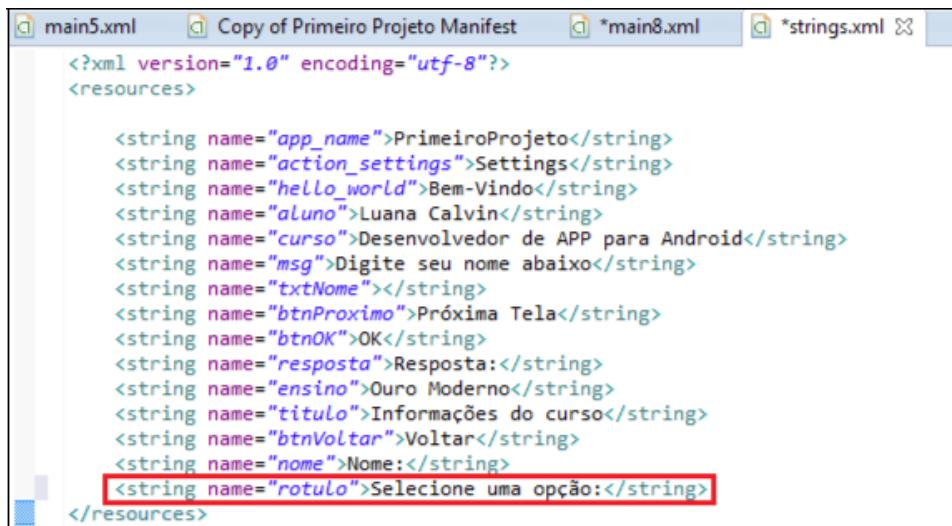
## NOTAÇÕES

### **Usando Widgets para a seleção.**

#### **ListView**

Para exibir listas no Android utiliza-se o widget **ListView** em um arquivo de layout e utilizar o método **setAdapter** para carregar os itens da lista, sendo que os itens podem ser criados em um arquivo de recursos – ou via código.

Crie as strings referente a imagem abaixo.



```

main5.xml Copy of Primeiro Projeto Manifest *main8.xml *strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">PrimeiroProjeto</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Bem-Vindo</string>
    <string name="aluno">Luana Calvin</string>
    <string name="curso">Desenvolvedor de APP para Android</string>
    <string name="msg">Digite seu nome abaixo</string>
    <string name="txtNome"></string>
    <string name="btnProximo">Próxima Tela</string>
    <string name="btnOK">OK</string>
    <string name="resposta">Resposta:</string>
    <string name="ensino">Ouro Moderno</string>
    <string name="titulo">Informações do curso</string>
    <string name="btnVoltar">Voltar</string>
    <string name="nome">Nome:</string>
    <string name="rotulo">Selecione uma opção:</string>
</resources>

```

Faça este exemplo: crie um novo arquivo de layout chamado **main8** e adicione dois **TextViews** e um **ListView** entre estes, e refcrcie de acordo com a imagem abaixo.

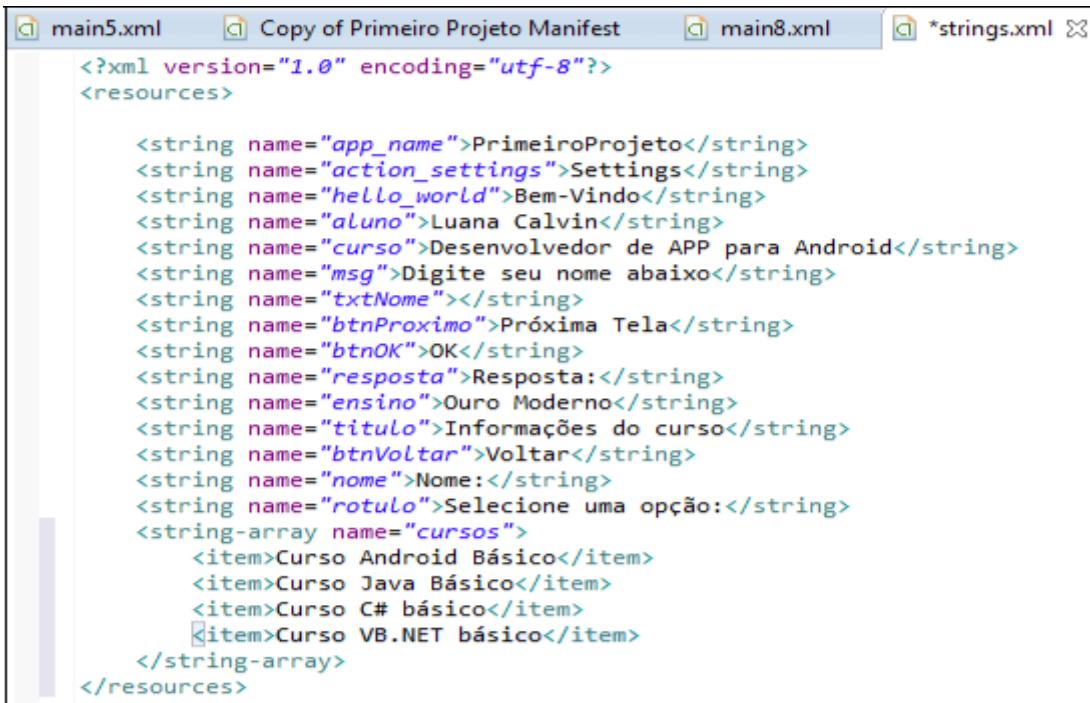


Mais tarde aprenderemos a aplicar informações a este ListView.

#### **Exemplos de ListView**

A informação pode ser carregada usando os dados do arquivo de recursos; nesse caso, adicione o arquivo de recursos **string-array** e nele adicione alguns itens, conforme o recurso abaixo.

Adicione os códigos no arquivo string.



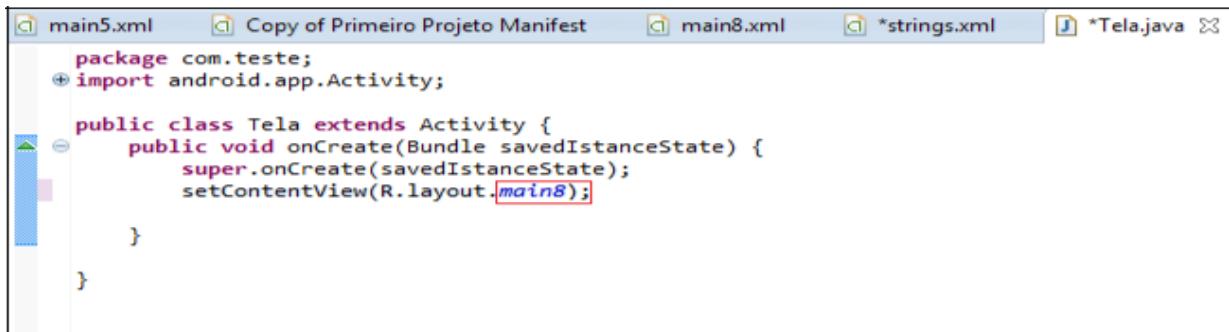
```

<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">PrimeiroProjeto</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Bem-Vindo</string>
    <string name="aluno">Luana Calvin</string>
    <string name="curso">Desenvolvedor de APP para Android</string>
    <string name="msg">Digite seu nome abaixo</string>
    <string name="txtNome"></string>
    <string name="btnProximo">Próxima Tela</string>
    <string name="btnOK">OK</string>
    <string name="resposta">Resposta:</string>
    <string name="ensino">Ouro Moderno</string>
    <string name="titulo">Informações do curso</string>
    <string name="btnVoltar">Voltar</string>
    <string name="nome">Nome:</string>
    <string name="rotulo">Selecione uma opção:</string>
    <string-array name="cursos">
        <item>Curso Android Básico</item>
        <item>Curso Java Básico</item>
        <item>Curso C# básico</item>
        <item>Curso VB.NET básico</item>
    </string-array>
</resources>

```

Agora altere o código do arquivo **Tela**.



```

package com.teste;
import android.app.Activity;

public class Tela extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main8);
    }
}

```

Na pasta **src** exclua todos os arquivos exceto o arquivo **Tela**.  
 Renomeie este arquivo para o nome **MainActivity**.  
 Agora insira os códigos referentes a imagem abaixo.



```

package com.teste;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MainActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main8);

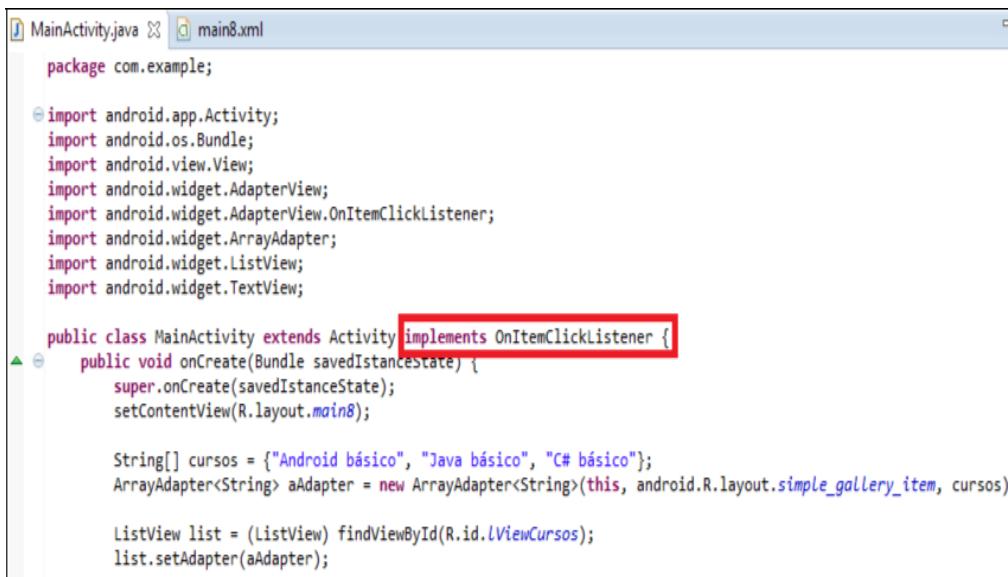
        String[] cursos = getResources().getStringArray(R.array.cursos);
        ArrayAdapter<String> aAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_gallery_item, cursos);

        ListView list = (ListView) findViewById(R.id.listView);
        list.setAdapter(aAdapter);
    }
}

```

Para identificar o item selecionado, basta implementar na activity a interface **OnItemClickListener** e o método **onItemClick**, e associar o evento **ItemClick** ao widget **ListWiew**, conforme o código da imagem abaixo.

Implemente o projeto inserindo os códigos no local indicado.



```

>MainActivity.java
```

```

package com.example;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

public class MainActivity extends Activity implements OnItemClickListener {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main8);

        String[] cursos = {"Android básico", "Java básico", "C# básico"};
        ArrayAdapter<String> aAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_gallery_item, cursos);

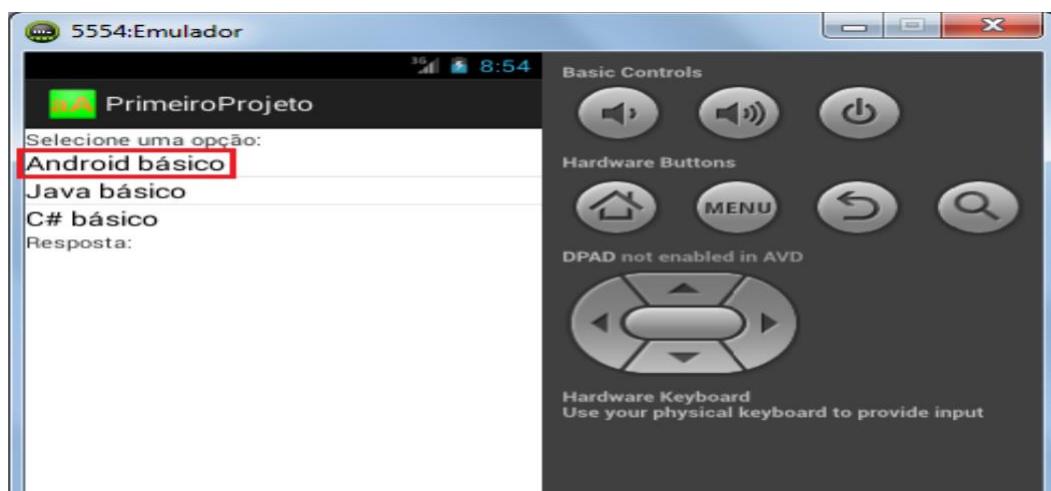
        ListView list = (ListView) findViewById(R.id.listViewCursos);
        list.setAdapter(aAdapter);
    }
}

```

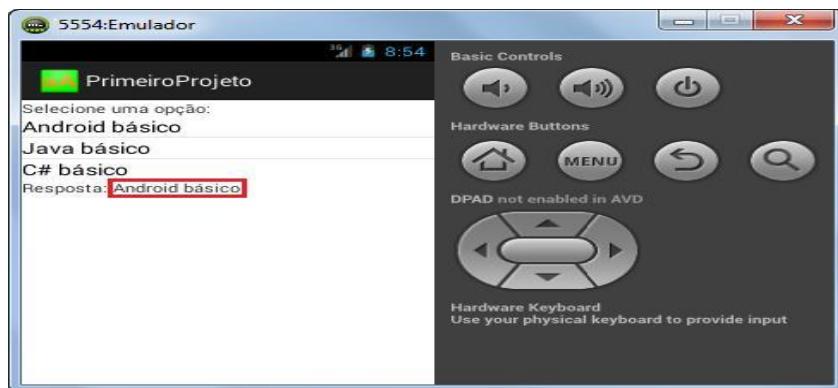
Execute o projeto.



Clique no local indicado.

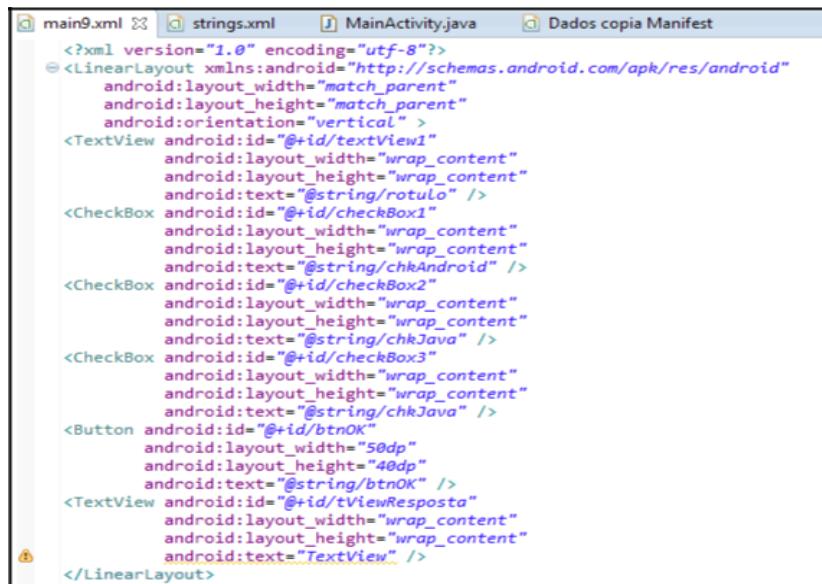


Repare que em **Resposta:** aparecerá o texto referente a opção selecionada.



### CheckBox

O **CheckBox** é um widget que permite ao usuário selecionar quantas opções desejar. Faça este exemplo com o nome **main9**, crie o arquivo de layout baixo.

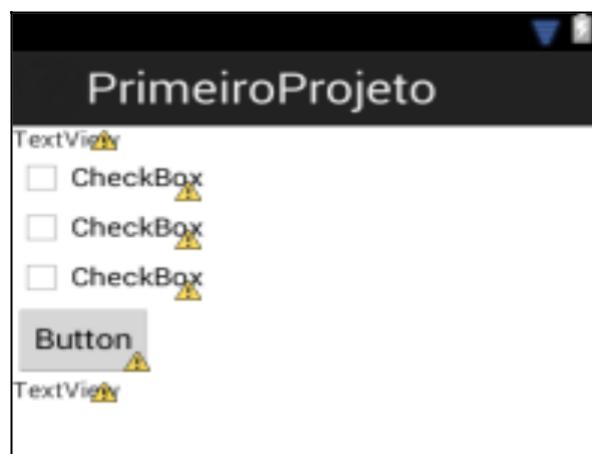


```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/rotulo" />
    <CheckBox android:id="@+id/checkBox1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/chkAndroid" />
    <CheckBox android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/chkJava" />
    <CheckBox android:id="@+id/checkBox3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/chkJava" />
    <Button android:id="@+id/btnOK"
        android:layout_width="50dp"
        android:layout_height="40dp"
        android:text="@string/btnOK" />
    <TextView android:id="@+id/tViewResposta"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView" />
</LinearLayout>

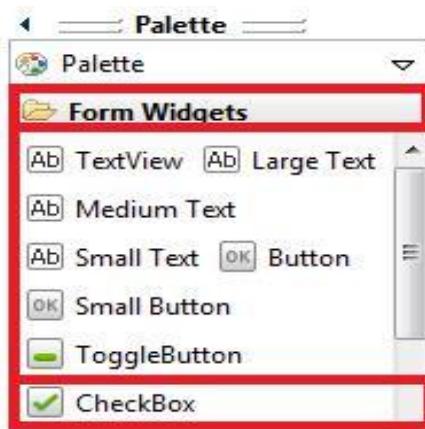
```

Adicione os Widgets conforme a imagem abaixo.

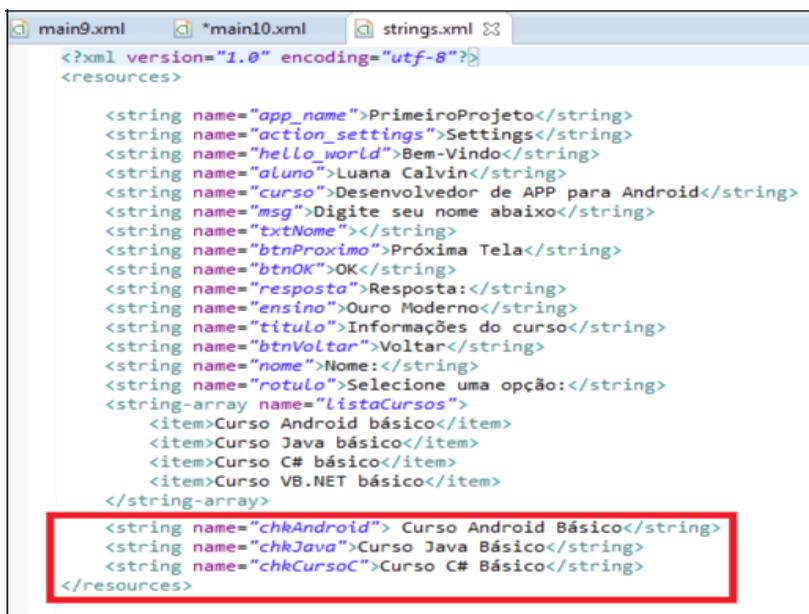


Primeiramente adicione um **TextView**.

Após insira três **CheckBox**, os CheckBox podem ser encontrados no palete **Form Widget**.



Também adicione um **Botão** e um **TextView** logo abaixo.  
Cria três novas strings.



```

<resources>
    <string name="app_name">PrimeiroProjeto</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Bem-Vindo</string>
    <string name="aluno">Luana Calvin</string>
    <string name="curso">Desenvolvedor de APP para Android</string>
    <string name="msg">Digite seu nome abaixo</string>
    <string name="txtNome"></string>
    <string name="btnProximo">Próxima Tela</string>
    <string name="btnOK">OK</string>
    <string name="resposta">Resposta:</string>
    <string name="ensino">Ouro Moderno</string>
    <string name="titulo">Informações do curso</string>
    <string name="btnVoltar">Voltar</string>
    <string name="nome">Nome:</string>
    <string name="rotulo">Selecione uma opção:</string>
    <string-array name="listaCursos">
        <item>Curso Android básico</item>
        <item>Curso Java básico</item>
        <item>Curso C# básico</item>
        <item>Curso VB.NET básico</item>
    </string-array>
    <string name="chkAndroid"> Curso Android Básico</string>
    <string name="chkJava">Curso Java Básico</string>
    <string name="chkCursoC">Curso C# Básico</string>
</resources>

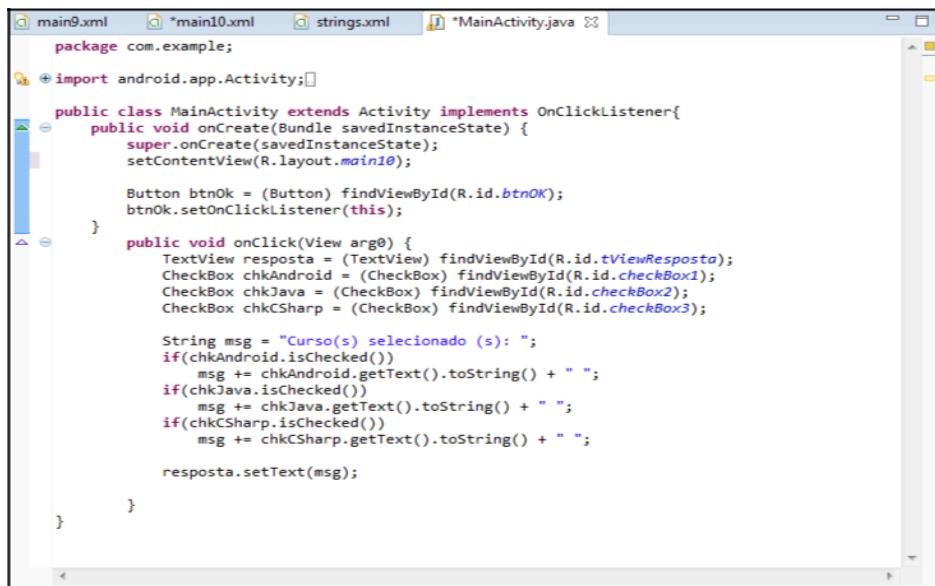
```

Para evitar conflitos com outros projetos, altere o nome da string **btnOk** para **btnOK**.

Referencie os Widgets as strings, conforme a imagem abaixo.



Insira os códigos no arquivo **MainActivity**.



```

package com.example;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.TextView;

public class MainActivity extends Activity implements OnClickListener {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main10);

        Button btnOK = (Button) findViewById(R.id.btnOK);
        btnOK.setOnClickListener(this);
    }

    public void onClick(View arg0) {
        TextView resposta = (TextView) findViewById(R.id.tViewResposta);
        CheckBox chkAndroid = (CheckBox) findViewById(R.id.checkBox1);
        CheckBox chkJava = (CheckBox) findViewById(R.id.checkBox2);
        CheckBox chkSharp = (CheckBox) findViewById(R.id.checkBox3);

        String msg = "Curso(s) selecionado (s): ";
        if(chkAndroid.isChecked())
            msg += chkAndroid.getText().toString() + " ";
        if(chkJava.isChecked())
            msg += chkJava.getText().toString() + " ";
        if(chkSharp.isChecked())
            msg += chkSharp.getText().toString() + " ";

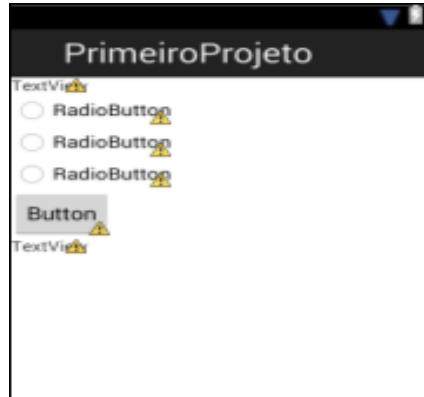
        resposta.setText(msg);
    }
}

```

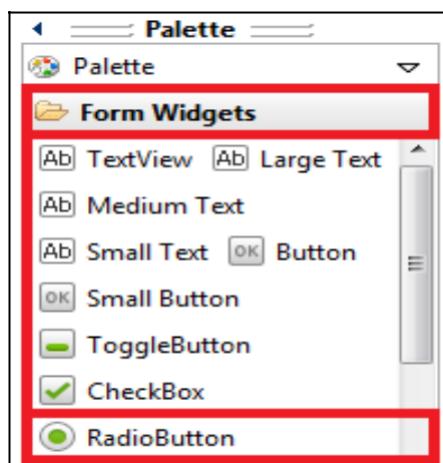
## RadioButton

O **RadioButton**, assim como o **CheckBox** também é um widget que apresenta opções, mas permite ao usuário selecionar apenas uma opção dentro do grupo.

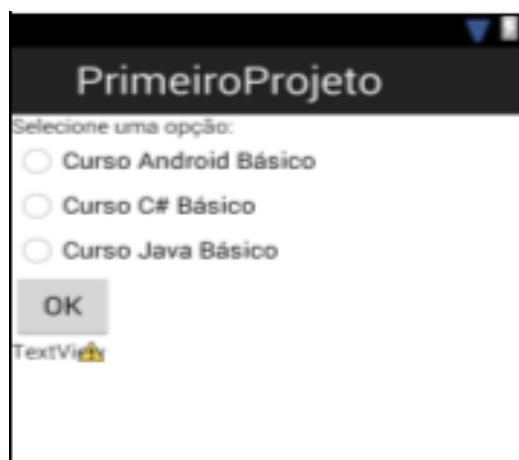
Crie um arquivo **xml** chamado **main10** e aplique os widget conforme a imagem abaixo.



Para inserir um **RadioButton** basta clicar no paleta **Form Widgets** e arrastar a opção **RadioButton**.



Referencie os widget com as strings conforme a imagem abaixo.



O último TextView deverá ser referenciado com a string **txtNome**.  
Aplique os seguintes nomes aos widgets.

**Selecionar uma opção = tViewSelecionar**  
**Curso Android Básico = rdAndroidBasico**  
**Curso Java Básico = rdJavaBasico**  
**Curso C# Básico = rdCSharpBasico**  
**OK = btnOK**  
**TextView = tViewResposta**

Neste exemplo criamos um projeto do zero, por isso não teremos as strings utilizadas nas aulas anteriores.

```

<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Radio</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="txtNome"></string>
    <string name="btnOK">OK</string>
    <string name="rotulo">Selecionar uma opção:</string>

    <string-array name="listaCursos">
        <item>Curso Android básico</item>
        <item>Curso Java básico</item>
        <item>Curso C# básico</item>
        <item>Curso VB.NET básico</item>
    </string-array>
    <string name="chkAndroid"> Curso Android Básico</string>
    <string name="chkJava">Curso Java Básico</string>
    <string name="chkCursoC">Curso C# Básico</string>
</resources>

```

Crie uma nova activity denominada **Radios.java**.



```

main10.xml Radios.java strings.xml
package com.radio;

import android.app.Activity;
public class Radios extends Activity implements OnClickListener {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main10);

        Button btnOK = (Button) findViewById(R.id.btnOK);
        btnOK.setOnClickListener(this);
    }

    public void onClick(View v) {
        TextView resposta = (TextView) findViewById(R.id.tViewResposta);
        RadioButton rdAndroid = (RadioButton) findViewById(R.id.rdAndroidBasico);
        RadioButton rdJava = (RadioButton) findViewById(R.id.rdJavaBasico);
        RadioButton rdSharp = (RadioButton) findViewById(R.id.rdcSharpBasico);

        String msg = "Curso selecionado: ";
        if(rdAndroid.isChecked())
            msg += rdAndroid.getText().toString() + " ";
        if(rdJava.isChecked())
            msg += rdJava.getText().toString() + " ";
        if(rdSharp.isChecked())
            msg += rdSharp.getText().toString() + " ";

        resposta.setText(msg);
    }
}

```

Abra o arquivo chamado **AndroidManifest.xml** e altere o código para Radio no local indicado.



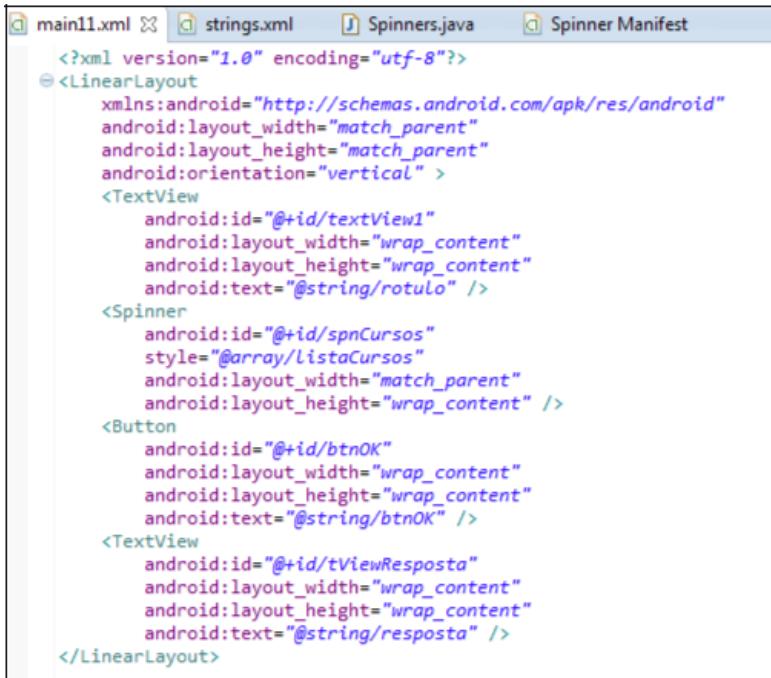
Com isto o emulador abrirá diretamente para a activity **Radios.java**. Execute o projeto.



## Spinner

Spiner é um widget que permite ao usuário selecionar apenas na opção em uma caixa suspensa como um ComboBox.

Crie este novo arquivo de layout com o nome main11 conforme a imagem abaixo.



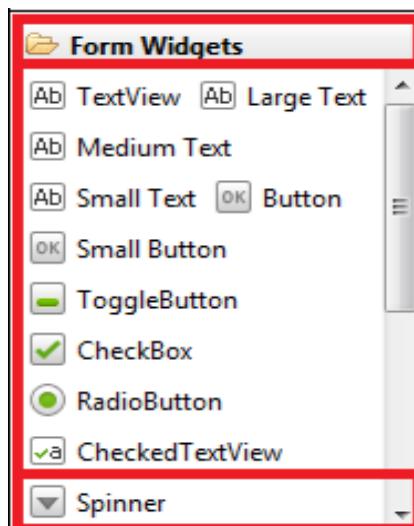
```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/rotulo" />
    <Spinner
        android:id="@+id/spnCursos"
        style="@array/listaCursos"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/btnOK"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btnOK" />
    <TextView
        android:id="@+id/tViewResposta"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/resposta" />
</LinearLayout>

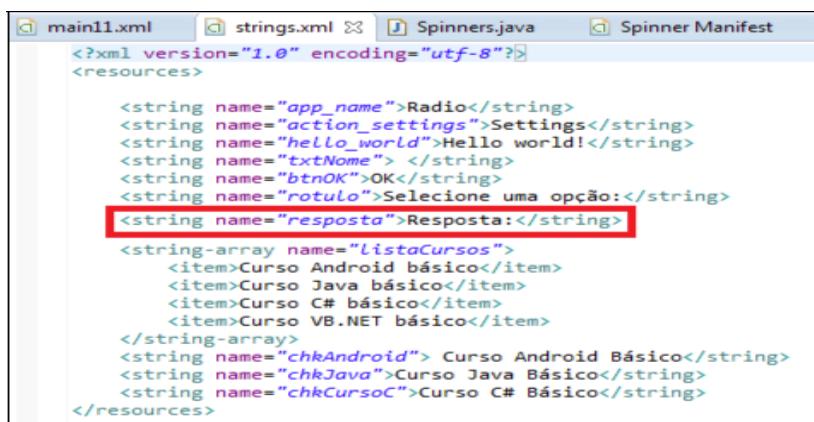
```

No modo gráfico você pode observar os Widgets que deveram ser instalados, adicione um TextView, Spinner, Button e um TextView.

Para adicionar um **Spinner** basta clicar no palete **Form Widget** e arrastar a opção **Spinner**.



Crie uma nova string chamada **resposta**.



```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Radio</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="txtName"></string>
    <string name="btnOK">OK</string>
    <string name="rotulo">Selecione uma opção:</string>
    <string name="resposta">Resposta:</string>
    <string-array name="ListaCursos">
        <item>Curso Android básico</item>
        <item>Curso Java básico</item>
        <item>Curso C# básico</item>
        <item>Curso VB.NET básico</item>
    </string-array>
    <string name="chkAndroid"> Curso Android Básico</string>
    <string name="chkJava">Curso Java Básico</string>
    <string name="chkCursoC">Curso C# Básico</string>
</resources>

```

Associe os Widget com as seguintes strings.

**TextViev = rotulo**

**Button = btnOK**

**TextViev = resposta**

Obs.: O Widget Spinner não será relacionado com nenhuma string nesta etapa, esta relação se dar através da inserção de códigos.

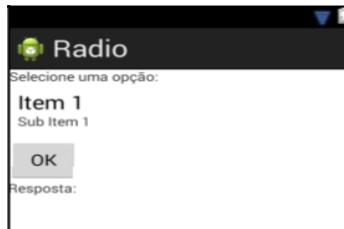
Os nomes dos widgets deverão aplicados conforme a imagem abaixo.

**TextViev = textView1**

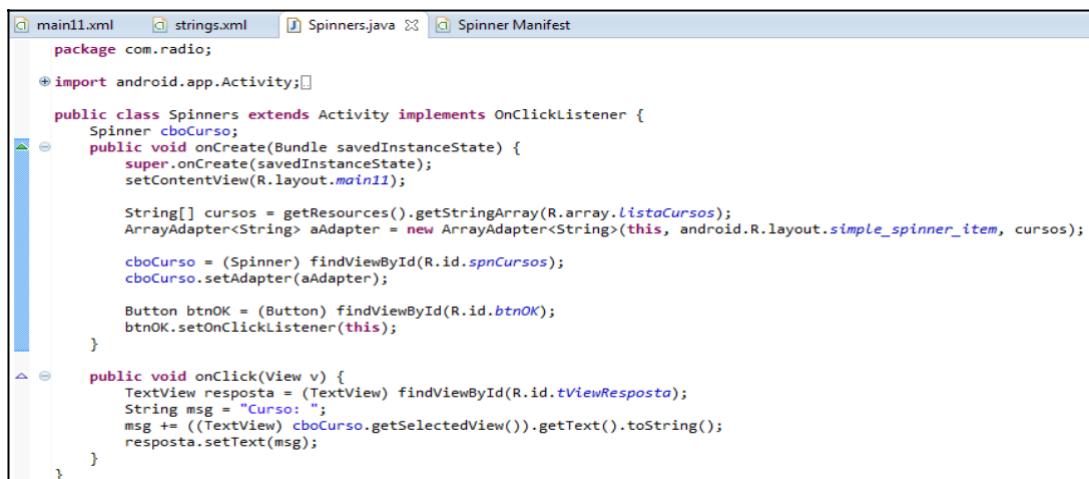
**Spinner = spnCursos**

**Button = btnOK**

**TextViev = tViewResposta**



Crie uma classe em Java com o nome **Spinners.java** e insira os códigos de acordo com a imagem abaixo.



```

package com.radio;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class Spinners extends Activity implements OnClickListener {
    Spinner cboCurso;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main1);
        String[] cursos = getResources().getStringArray(R.array.ListaCursos);
        ArrayAdapter<String> aAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, cursos);
        cboCurso = (Spinner) findViewById(R.id.spnCursos);
        cboCurso.setAdapter(aAdapter);
        Button btnOK = (Button) findViewById(R.id.btnOK);
        btnOK.setOnClickListener(this);
    }
    public void onClick(View v) {
        TextView resposta = (TextView) findViewById(R.id.tViewResposta);
        String msg = "Curso: ";
        msg += ((TextView) cboCurso.getSelectedView()).getText().toString();
        resposta.setText(msg);
    }
}

```

No arquivo **AndroidManifest** altere o código conforme a imagem abaixo.



```

<manifest version="1.0" encoding="utf-8">
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher" // Red box here
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.radio.Spinners"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

Execute o projeto.



Clique no local indicado.



Selecione a opção **Curso C# básico** e clique em **OK**.



Pronto, selecionamos a opção requerida.

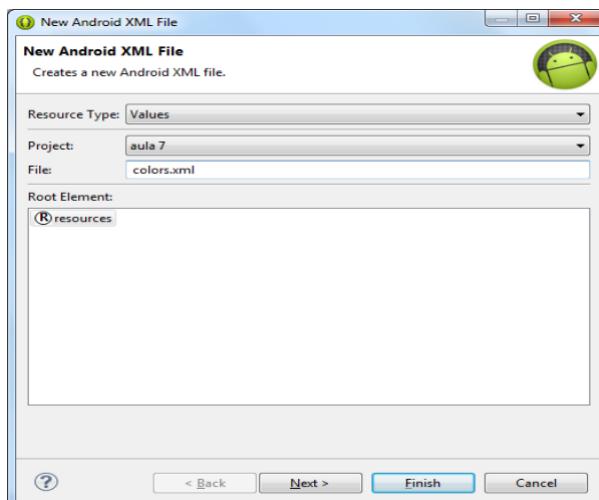


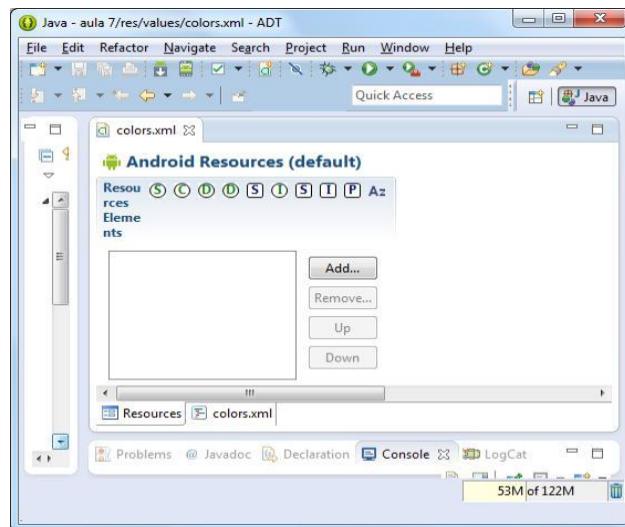
## Recurso Color

Existem várias formas de aplicar修改as cores do projeto, sendo a forma mais simples criar um arquivo xml dentro da pasta **res/values**, contendo o nome da cor e o valor em **RGB**.

Obs.: Esse arquivo deve obrigatoriamente ser denominado por **colors.xml**. Abra o arquivo da sua **Terceira aula**.

Para criar o arquivo, clique com o botão direito do mouse sobre a pasta **res/values** e selecione a opção **New>Android XML File**; a caixa de diálogo abaixo será exibida, aplique com o nome **colors.xml** e clique em **Finish**.





### Adicionando um recurso de cor

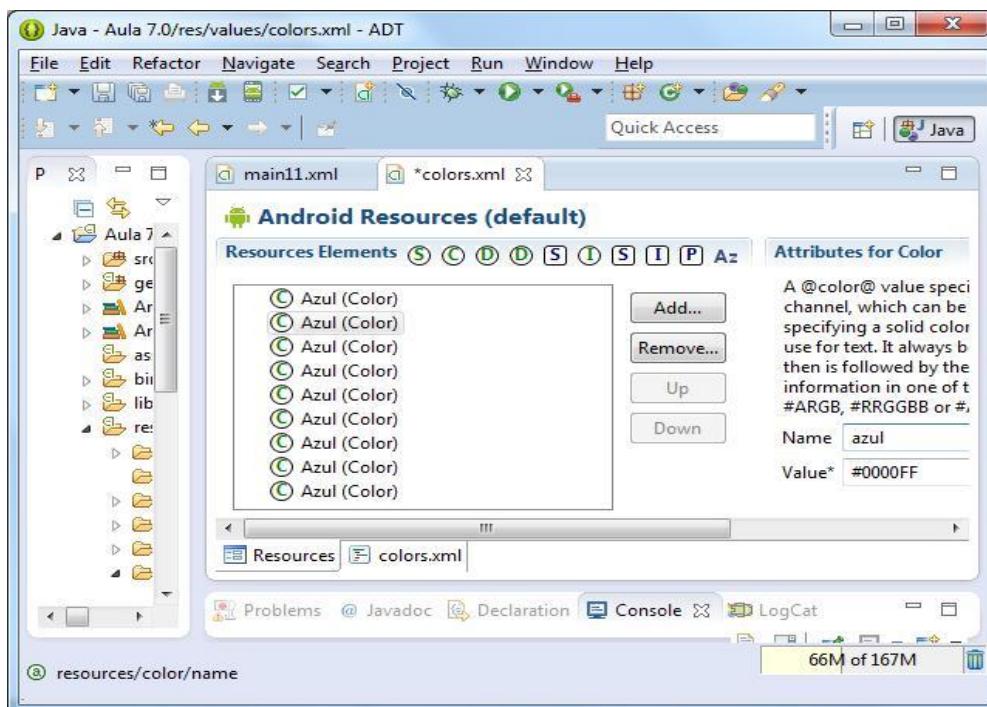
Para adicionar um recurso de cor, clique no botão **Add...**. A seguinte caixa de diálogo será exibida.

Selecione **Color** e clique em **OK**.

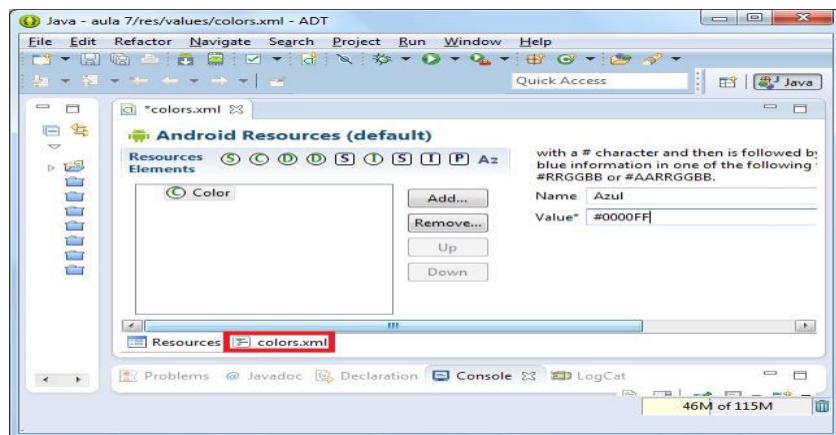
Em **Name** preencha o nome da cor; em **Value** o valor da cor em hexadecimal seguindo um dos padrões abaixo.

- **#RGB**
- **#ARGB**
- **#RRGGBB**
- **#AARRGGBB**

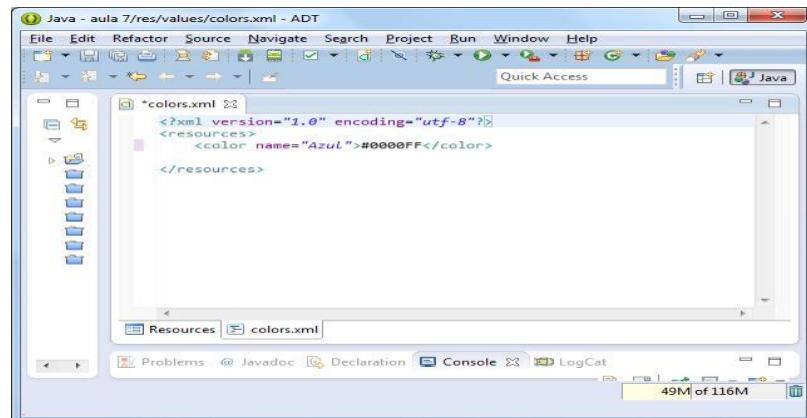
Neste exemplo digite **azul** em **Name** e **#0000FF** em **Value**, conforme a imagem abaixo.



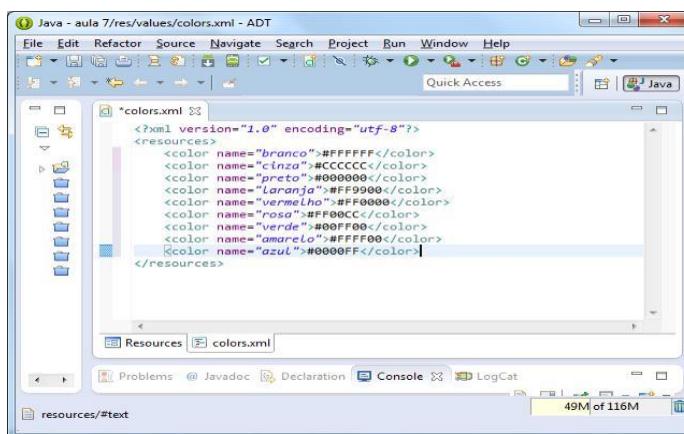
Mude para o código-fonte clicando na aba **colors.xml**.



Note que a cor foi criada em uma tag **color** onde o atributo **name** possui o nome da cor, e entre as tags ficando localizado o valor hexadecimal.



Preencha algumas cores no arquivo pelo método visual ou diretamente no código-fonte.



Agora vamos entender como é possível alterar a cor do texto do widget utilizando o atributo **textColor**, da seguinte forma.

`android:textColor="@color/vermelho"`

Para alterar a cor de fundo utilize o atributo **background**.

`android:background="@color/branco"`

Aplique a cor vermelha como cor de texto em casa tag de códigos dos Widgets.

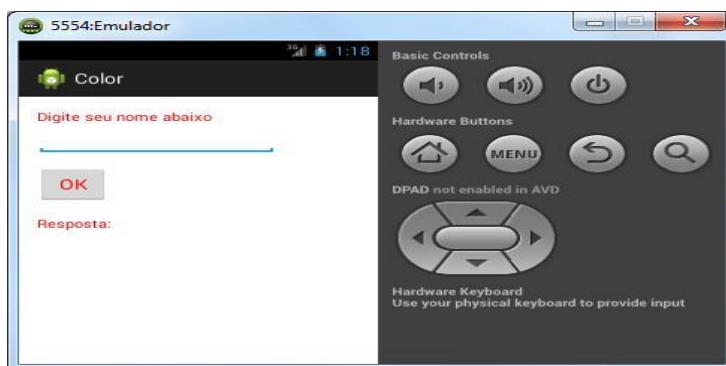


```

<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/txtMsg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/msg"
        android:textColor="@color/vermelho" />
    <EditText
        android:id="@+id/etxtNome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/txtMsg"
        android:layout_below="@+id/txtMsg"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="@string/txtName"
        android:textColor="@color/vermelho" >
        <requestFocus />
    </EditText>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="OK" />
</RelativeLayout>

```

Execute o projeto.



Muito bem, estamos no fim desta sexta aula do Curso de Desenvolvedor para Aplicativos de Android, não se esqueça de fazer os exercícios desta apostila.

### **EXERCÍCIO DE FIXAÇÃO**

- 1) Abra o Eclipse e crie um novo projeto.
- 2) Altere o nome do programa para “Quinto Projeto”.
- 3) Insira um ListView e um TextView para resposta.
- 4) Preencha o ListView com uma lista de carros.
- 5) Faça com que o carro clicado seja exibido no TextView.
- 6) Crie outras outra tela com a mesma funcionalidade, porém utilizando CheckBox e um botão de Ok para exibir o resultado.
- 7) Repita o processo anterior utilizando o RadioButton.

## NOTAÇÕES



## NOTAÇÕES



## NOTAÇÕES



## NOTAÇÕES



### Tipos de Recursos

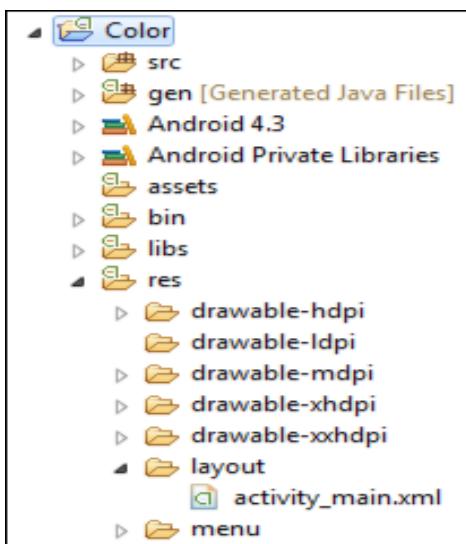
Todos os recursos do projeto ficam organizados dentro da pasta **res**; eles são mapeados para binário e acessados através de classe de recurso (a classe **R** da pasta **gen**). Para evitar qualquer problema de compilação o Android define onde cada tipo de recurso deve ser incluído, como os layouts na pasta **res/layout** e as strings na pasta **res/values**, dentro de um arquivo **strings.xml**.

Veja agora mais alguns tipos de recursos que podem ser adicionados ao projeto.

#### Recursos Gráficos

Os recursos gráficos estão na pasta **res/drawable** e podem ser imagens (png, jpg, ou gif) ou até um xml que define ou faz a uma referência a uma imagem. Neste curso básico serão demonstradas apenas as imagens.

Veja que dentro da pasta **res** existem cinco pastas **drawable**.



Obs.: Como será explicado à frente, o arquivo deve possuir o mesmo nome nas cinco pastas ou ocorrerá um erro de compilação.

Para ajudar a organizar os ícones da aplicação, a documentação do Android sugere a seguinte nomenclatura para a nomeação dos ícones.

tipo de ícone	Prefixo	Exemplo
Ícones em geral	ic	ic_star.png
Launcher	ic_launcher	ic_launcher_calendar.png
Menu	ic_menu	ic_menu_archive.png
Status bar	ic_stat_notify	ic_stat_notify_msg.png
Tab	ic_tab	ic_tab_recent.png
Dialog	ic_dialog	ic_dialog_info.png

```

ic_menu/...
    ldpi/...
        _pre_production/...
            working_file.psd
            ic_menu_next.png
    mdpi/...
        _pre_production/...
            ic_menu_next.png
    hdpi/...
        _pre_production/...
            working_file.psd
            ic_menu_next.png

```

Dessa forma você pode determinar facilmente onde cada arquivo será adicionado ao projeto. Para efeito de comparação, os códigos citados ficaram organizados da seguinte forma na aplicação.

```

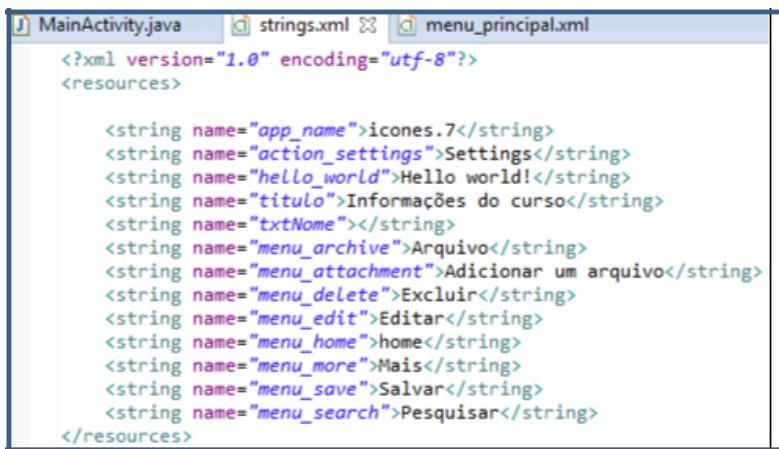
res/...
    drawable-ldpi/...
        ic_menu_next.png
    drawable-mdpi/...
        ic_menu_next.png
    drawable-hdpi/...
        ic_menu_next.png

```

## Menus e submenus

Primeiramente crie um novo projeto chamado **ícones aula 7**.

Para criar o menu do próximo exemplo, é necessário criar as strings representadas na imagem abaixo.



```

<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">ícones.7</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="titulo">Informações do curso</string>
    <string name="txtNome"></string>
    <string name="menu_archive">Arquivo</string>
    <string name="menu_attachment">Adicionar um arquivo</string>
    <string name="menu_delete">Excluir</string>
    <string name="menu_edit">Editar</string>
    <string name="menu_home">home</string>
    <string name="menu_more">Mais</string>
    <string name="menu_save">Salvar</string>
    <string name="menu_search">Pesquisar</string>
</resources>

```

## Recursos de menu

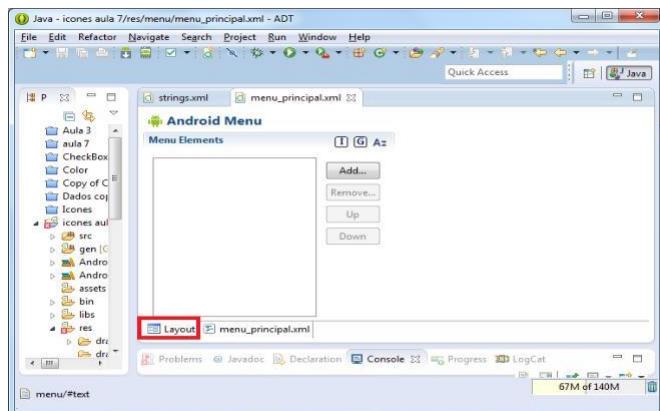
O conteúdo dos menus no Android fica organizado em uma arquivo xml dentro da pasta **res/menu**, se esta pasta não estiver disponível para o seu projeto, você deve clicar com o botão direito do mouse sobre a pasta **res** e selecionar a opção **New> Folder**.

Na caixa de diálogo que irá abrir você deverá preencher o nome **menu** no campo **Folder name**, conforme a imagem abaixo e clicar em **Finish**.

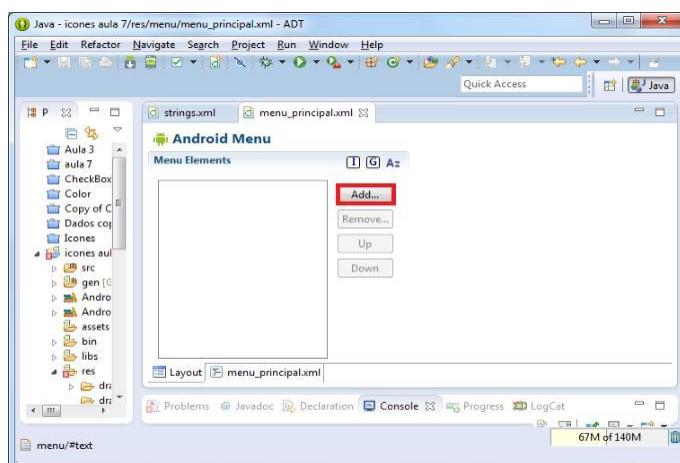
Vamos criar uma novo arquivo de menu xml, clique no botão direito do mouse sobre a pasta menu e selecione a opção **New> Android XML File**.

Preencha o campo **File** com o nome **menu\_principal.xml** conforme a imagem abaixo, e clique em **Finish**.

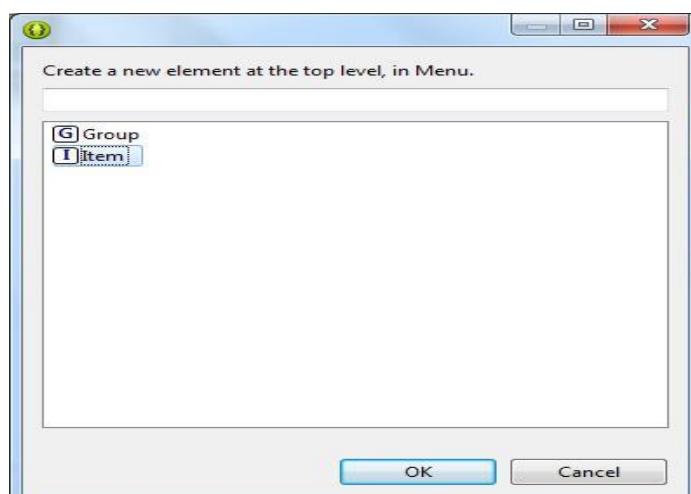
Na janela de diálogo que irá surgir clique no local indicado.



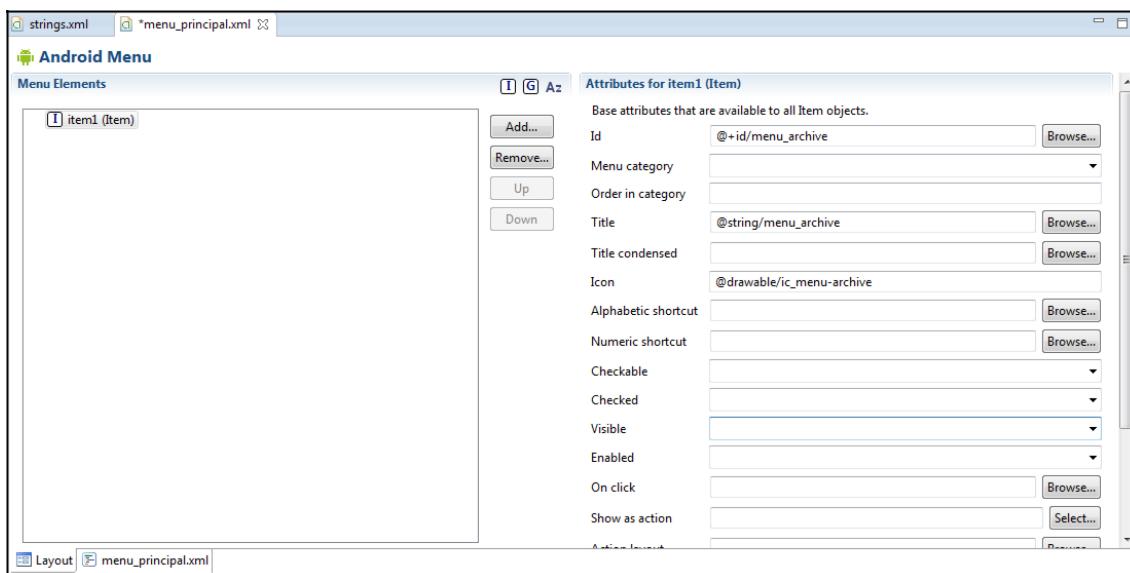
Clique em **Add...**



Você também pode adicionar um **item** ou um **Group**. O elemento **item** representa cada item de menu e o elemento **Group**, que é opcional e invisível, permite agrupar itens de menu para a configuração de propriedades comuns. Selecione o elemento **item** e clique em **OK**.

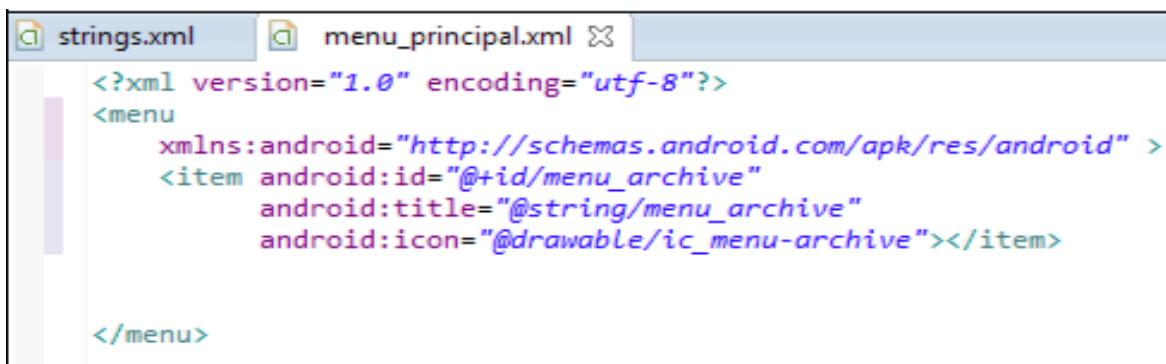


Note que há várias opções de atributos para um item do menu, mas neste caso só altere os atributos **id**, **title**, **icon**.



## Preparando o exemplo

Mude para o seguinte código de fonte.



```
<?xml version="1.0" encoding="utf-8"?>
<menu>
    <item android:id="@+id/menu_archive"
          android:title="@string/menu_archive"
          android:icon="@drawable/ic_menu-archive"></item>

</menu>
```

Repare os 3 atributos no item do menu:

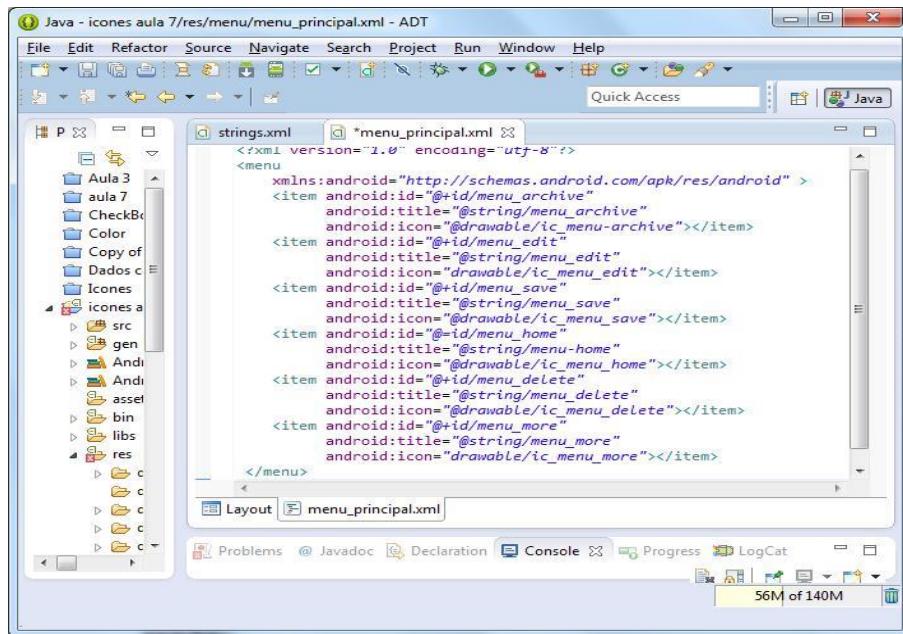
```
<item android:id="@+id/menu_archive"
      android:title="@string/menu_archive"
      android:icon="@drawable/ic_menu-archive"></item>
```

Possuem a função.

- **Android:id** Identificar o item do menu via programação. Esse valor deve ser único. Utiliza-se **menu\_archive** para coerente com os recursos de texto e imagens.
- **Android:icon** Configurar o ícone que será apresentado no menu, utiliza-se **@drawable/ic\_menu\_archive** para obter o ícone a partir dos recursos de imagem, ou seja, as imagens nas pastas drawable.

- **Android:title** Configurar o texto que será apresentado no item, utiliza-se **@string/menu\_archive** para obter o texto a partir do arquivo de recursos.

Agora adicione os demais itens no arquivo.



## Carregando o menu

Para utilizar o menu crie uma novo arquivo com dois **TextView**, conforme indicado no código abaixo.

Repare que referenciamos os **TextView** com as Strings **Titulo** e **txtNome** consequentemente.

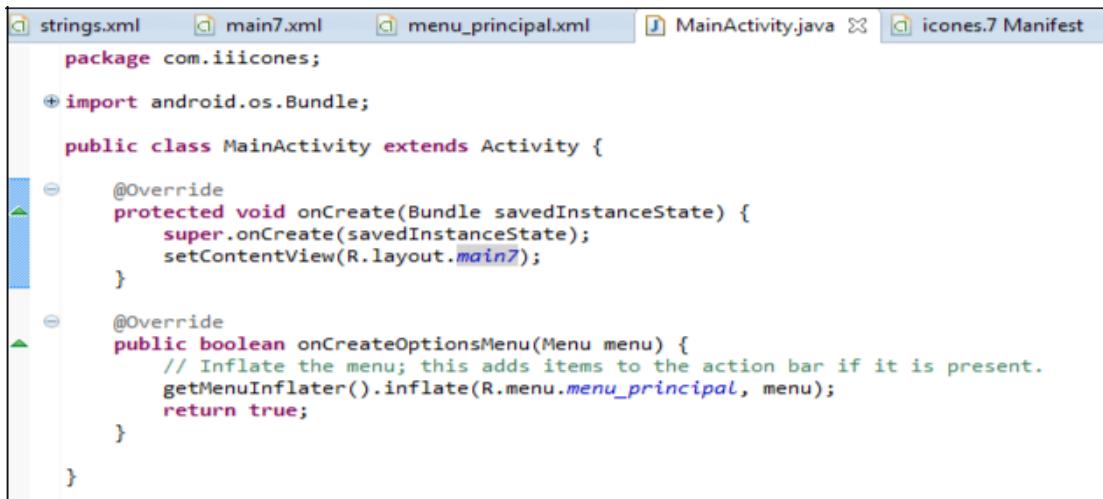


```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/tViewTitulo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/titulo" />
    <TextView
        android:id="@+id/tViewStatus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/txtNome" />
</LinearLayout>

```

Em **MainActivity** insira os códigos referentes a imagem abaixo.



```

strings.xml main7.xml menu_principal.xml MainActivity.java icones.7 Manifest
package com.iiicones;

import android.os.Bundle;

public class MainActivity extends Activity {

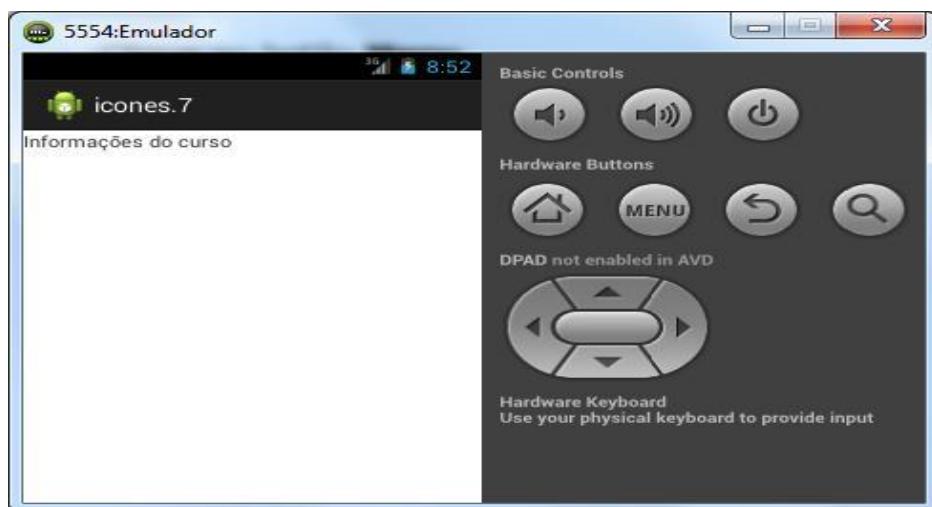
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main7);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_principal, menu);
        return true;
    }

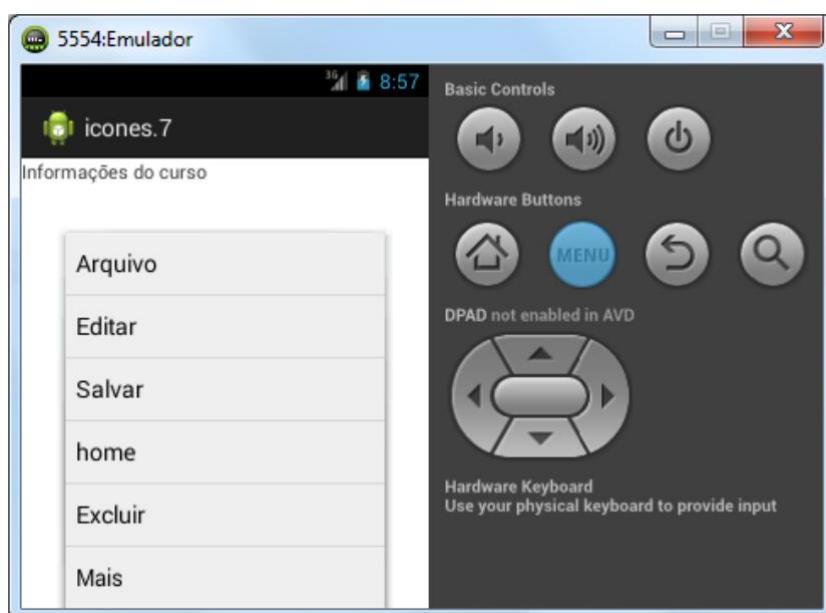
}

```

## Execute o projeto

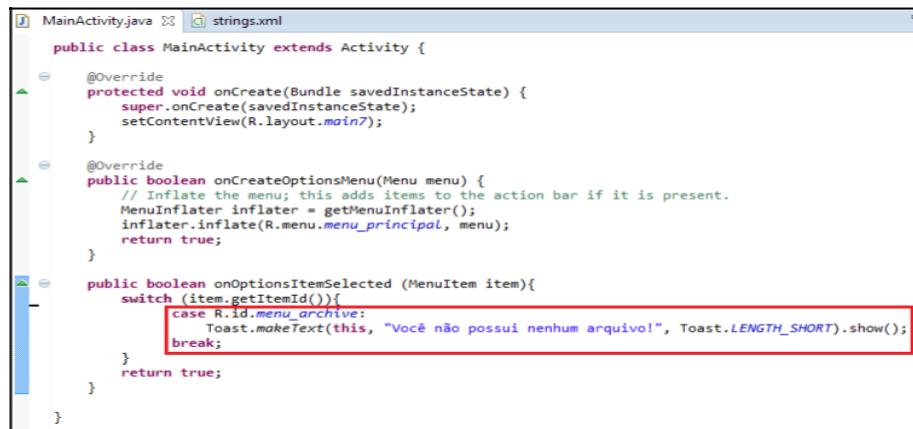


Clique no botão Menu.



Veja os menus que estão sendo exibidos, vamos agora criar os **submenus**.

Aplique o código referente a imagem abaixo no arquivo **MainActivity**



```

MainActivity.java
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main7);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_principal, menu);
        return true;
    }
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_arquivo:
                Toast.makeText(this, "Você não possui nenhum arquivo!", Toast.LENGTH_SHORT).show();
                break;
        }
        return true;
    }
}

```

Veja que no código inserido acima, na primeira linha chamamos uma referência ao atributo **menu\_archive**, que quando acionado deverá apresentar a mensagem que está referenciada na segunda linha “**Você não possui nenhum arquivo!**”

Clique em Arquivo.



Veja que a mensagem inserida anteriormente no código está sendo exibida temporariamente.



## Submenus

Um **submenu** nada mais é do que um menu que se abre ao ser clicado em algum dos itens. Sua utilização é recomendada quando a aplicação possui muitas funcionalidades que podem ser agrupadas como submenus de um menu principal.

Para adicionar um **submenu**, basta criar o elemento menu **dentro** de um elemento item no arquivo de menu.

```

<item android:id="@+id/menu_delete"
      android:title="@string/menu_delete"
      android:icon="@drawable/ic_menu_delete"></item>
<item android:id="@+id/menu_more"
      android:title="@string/menu_more"
      android:icon="@drawable/ic_menu_more"></item>
</menu>
<item android:id="@+id/menu_search"
      android:title="@string/menu_search"
      <item android:id="@+id/menu_attachment"
            android:title="@string/menu_attachment"
            </item>
      </item>
</menu>

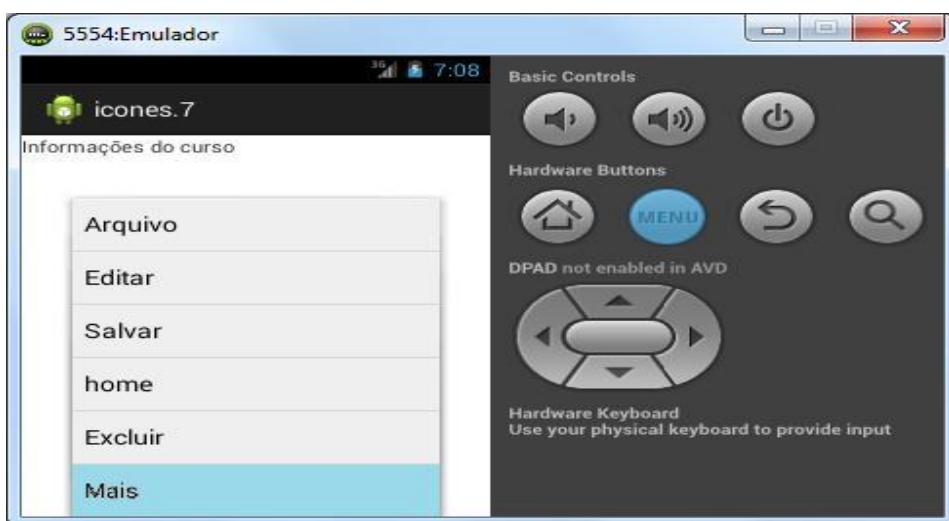
```

Observe que o submenu foi adicionado dentro da opção Mais (menu\_more).

Não é preciso configurar mais nada no activity, basta executar a aplicação e clicar na opção Mais para exibir o submenu.

Execute o projeto.

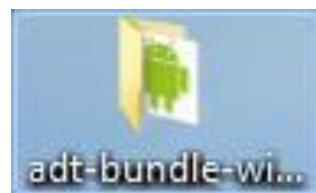
Clique em **Menu> Mais**.



## Ícones

No momento você não precisa se preocupar com a criação de ícones porque todos os que serão utilizados nos próximos exemplos são disponibilizados pelo SDK.

Vá até a pasta adt-bundle que está instalada no seu computador.



Após siga o caminho indicado.

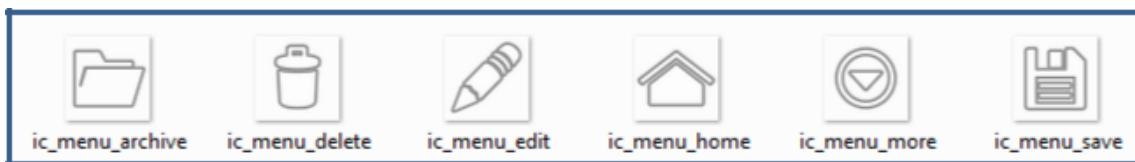
### SDK> Platforms> Android-18> Data> Res

Sinalizamos as pastas das quais vamos extrair os ícones.

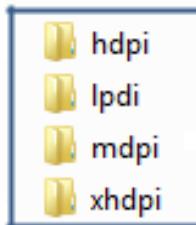
Primeiro vamos abrir a pasta **drawable-hdpi**.

Nome	Data de modificaç...	Tipo	Tamanho
anim-sw720dp	10/07/2013 19:05	Pasta de arquivos	
color	10/07/2013 19:05	Pasta de arquivos	
drawable	10/07/2013 19:05	Pasta de arquivos	
drawable-en-hdpi	10/07/2013 19:05	Pasta de arquivos	
drawable-en-ldpi	10/07/2013 19:05	Pasta de arquivos	
drawable-en-mdpi	10/07/2013 19:05	Pasta de arquivos	
<b>drawable-hdpi</b>	<b>10/07/2013 19:05</b>	<b>Pasta de arquivos</b>	
drawable-land-hdpi	10/07/2013 19:05	Pasta de arquivos	
drawable-land-ldpi	10/07/2013 19:05	Pasta de arquivos	
drawable-land-mdpi	10/07/2013 19:05	Pasta de arquivos	
drawable-land-xhdpi	10/07/2013 19:05	Pasta de arquivos	
<b>drawable-ldpi</b>	<b>16/10/2013 14:06</b>	<b>Pasta de arquivos</b>	
drawable-ldrtl-hdpi	10/07/2013 19:05	Pasta de arquivos	
drawable-ldrtl-ldpi	10/07/2013 19:05	Pasta de arquivos	
drawable-ldrtl-mdpi	10/07/2013 19:05	Pasta de arquivos	
drawable-ldrtl-xhdpi	10/07/2013 19:05	Pasta de arquivos	
<b>drawable-mdpi</b>	<b>10/07/2013 19:05</b>	<b>Pasta de arquivos</b>	
drawable-nodpi	10/07/2013 19:05	Pasta de arquivos	
drawable-sw600dp-hdpi	10/07/2013 19:05	Pasta de arquivos	
drawable-sw600dp-mdpi	10/07/2013 19:05	Pasta de arquivos	
drawable-sw600dp-nodpi	10/07/2013 19:05	Pasta de arquivos	
drawable-sw600dp-xhdpi	10/07/2013 19:05	Pasta de arquivos	
drawable-sw720dp-nodpi	10/07/2013 19:05	Pasta de arquivos	
<b>drawable-xhdpi</b>	<b>10/07/2013 19:05</b>	<b>Pasta de arquivos</b>	
interpolator	10/07/2013 19:05	Pasta de arquivos	
layout	10/07/2013 19:05	Pasta de arquivos	

Note que está disponível uma grande quantidade de ícones nas diversas resoluções, mas selecione apenas estes representados na imagem abaixo.



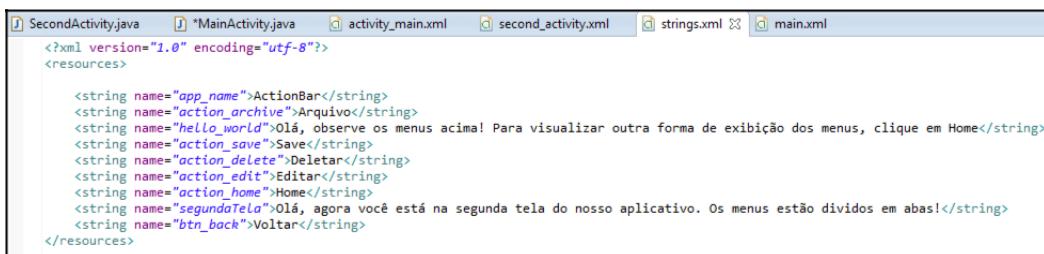
Agora vá até o **workspace** e abra o seu projeto, dentro da pasta **res** crie uma pasta chamada **ícones**, dentro desta crie uma pasta chama **ic\_menu** então dentro desta pasta crie mais outras pastas conforme a imagem abaixo.



Agora cole os ícones pré-selecionados na pasta referente, ou seja **hdpi**, repita o mesmo com as outras pastas **ldpi**, **mdpi**, **xhdpi**.

Crie um projeto novo com o nome **ActionBar**.

Crie as seguintes **Strings**.

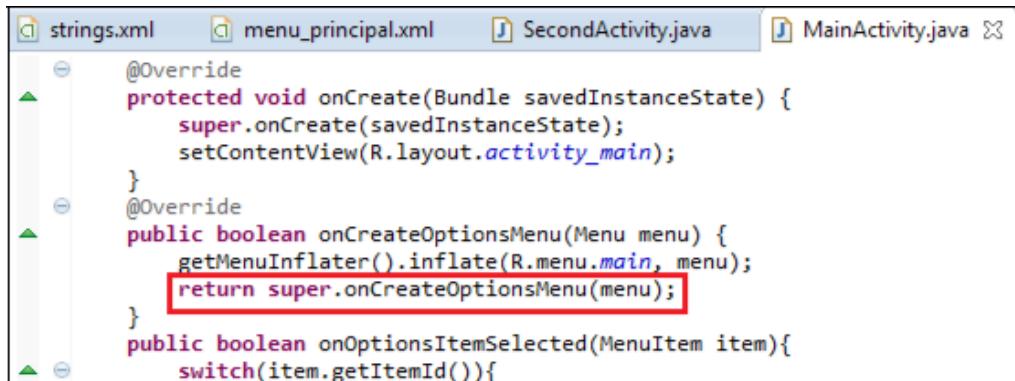


```

SecondActivity.java *MainActivity.java activity_main.xml second_activity.xml strings.xml main.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">ActionBar</string>
    <string name="action_archive">Arquivo</string>
    <string name="hello_world">Olá, observe os menus acima! Para visualizar outra forma de exibição dos menus, clique em Home</string>
    <string name="action_save">Save</string>
    <string name="action_delete">Deletar</string>
    <string name="action_edit">Editar</string>
    <string name="action_home">Home</string>
    <string name="segundatela">Olá, agora você está na segunda tela do nosso aplicativo. Os menus estão divididos em abas!</string>
    <string name="btn_back">Voltar</string>
</resources>

```

Em **MainActivity.java** aplique o código de **return** de acordo com a imagem abaixo.

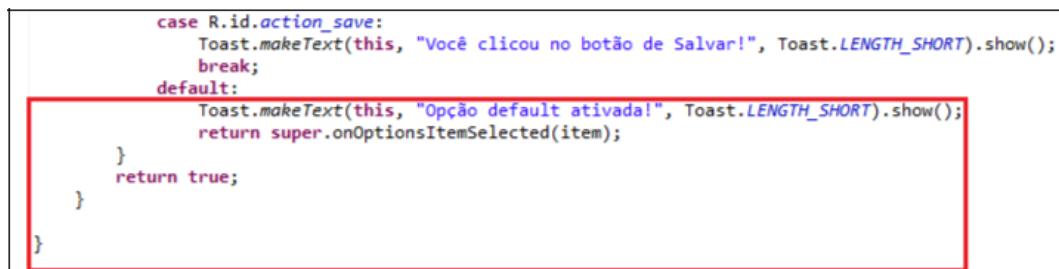


```

strings.xml menu_principal.xml SecondActivity.java MainActivity.java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    return super.onCreateOptionsMenu(menu);
}
public boolean onOptionsItemSelected(MenuItem item){
    switch(item.getItemId()){

```

A ação default ocorre quando nenhum dos outros requisitos foi encontrado, e/ou quando o break não existe. Na linha de baixo insira o código de return.



```

        case R.id.action_save:
            Toast.makeText(this, "Você clicou no botão de Salvar!", Toast.LENGTH_SHORT).show();
            break;
        default:
            Toast.makeText(this, "Oção default ativada!", Toast.LENGTH_SHORT).show();
            return super.onOptionsItemSelected(item);
    }
    return true;
}

```

Dentro da pasta src crie um arquivo na classe Java como o nome **SecondActivity** e aplique os código das imagens seguir.

```

>MainActivity.java   strings.xml   menu_principal.xml   SecondActivity.java
package com.example.com.filipe.actionbar;

import android.app.ActionBar;

public class SecondActivity extends Activity implements TabListener {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.second_activity);

        ActionBar bar = getActionBar();
        bar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
        Tab tabDel = bar.newTab();
        tabDel.setText("Excluir");
        tabDel.setIcon(R.drawable.ic_menu_delete);
        tabDel.setTabListener(this);
        bar.addTab(tabDel);
        Tab tabSalvar = bar.newTab();
        tabSalvar.setText("Salvar");
        tabSalvar.setIcon(R.drawable.ic_menu_save);
        tabSalvar.setTabListener(this);
        bar.addTab(tabSalvar);

        Tab tabEditar = bar.newTab();
        tabEditar.setText("Editar");
        tabEditar.setIcon(R.drawable.ic_menu_edit);
        tabEditar.setTabListener(this);
        bar.addTab(tabEditar);

        Button btnClose = (Button) findViewById(R.id.btnClose);
        btnClose.setOnClickListener(new View.OnClickListener() {
            public void onClick(View arg0) {
                finish();
            }
        });
    }

    @Override
    public void onTabReselected(Tab tab, FragmentTransaction ft) {
        Toast.makeText(this, "Você selecionou a aba que já estava selecionada!", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onTabSelected(Tab tab, FragmentTransaction ft) {
        Toast.makeText(this, "Você selecionou a aba: " + tab.getText() + "!", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onTabUnselected(Tab tab, FragmentTransaction ft) {

    }
}

```

Dentro da pasta menu, crie o arquivo chamado **main**, e aplique o código referente a imagem abaixo.

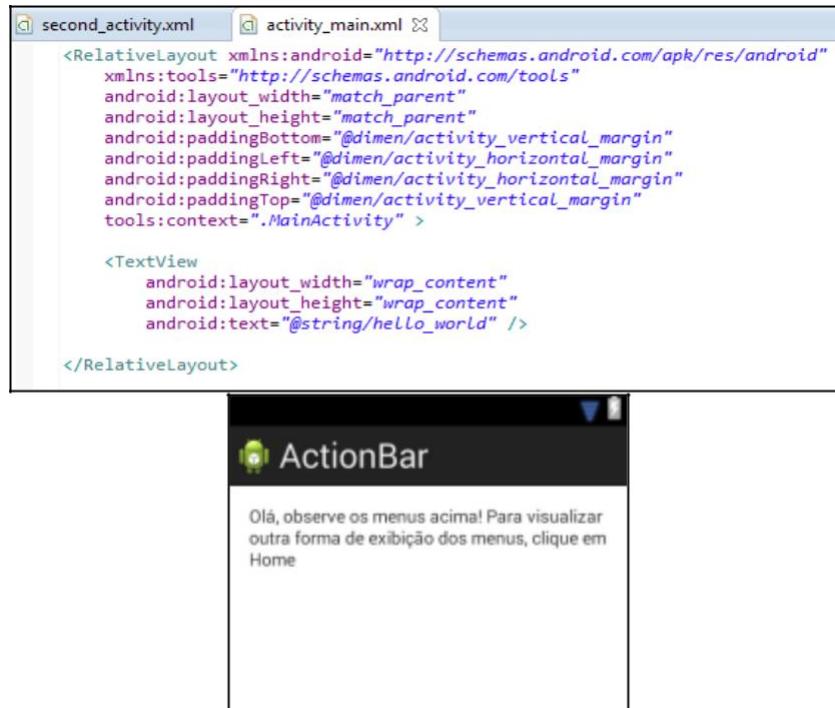
```

>MainActivity.java   SecondActivity.java   activity_main.xml   second_
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/action_archive"
          android:icon="@drawable/ic_menu_archive"
          android:menuCategory="system"
          android:showAsAction="never"
          android:title="@string/action_archive"/>
    <item android:id="@+id/action_save"
          android:icon="@drawable/ic_menu_save"
          android:menuCategory="system"
          android:orderInCategory="200"
          android:showAsAction="ifRoom"
          android:title="@string/action_save"/>
    <item android:id="@+id/action_delete"
          android:icon="@drawable/ic_menu_delete"
          android:menuCategory="system"
          android:orderInCategory="200"
          android:showAsAction="ifRoom"
          android:title="@string/action_delete"/>
    <item android:id="@+id/action_edit"
          android:icon="@drawable/ic_menu_edit"
          android:menuCategory="system"
          android:orderInCategory="200"
          android:showAsAction="ifRoom"
          android:title="@string/action_edit"/>
    <item android:id="@+id/action_home"
          android:icon="@drawable/ic_menu_home"
          android:showAsAction="always/withText"
          android:menuCategory="system"
          android:orderInCategory="100"
          android:title="@string/action_home"/>
</menu>

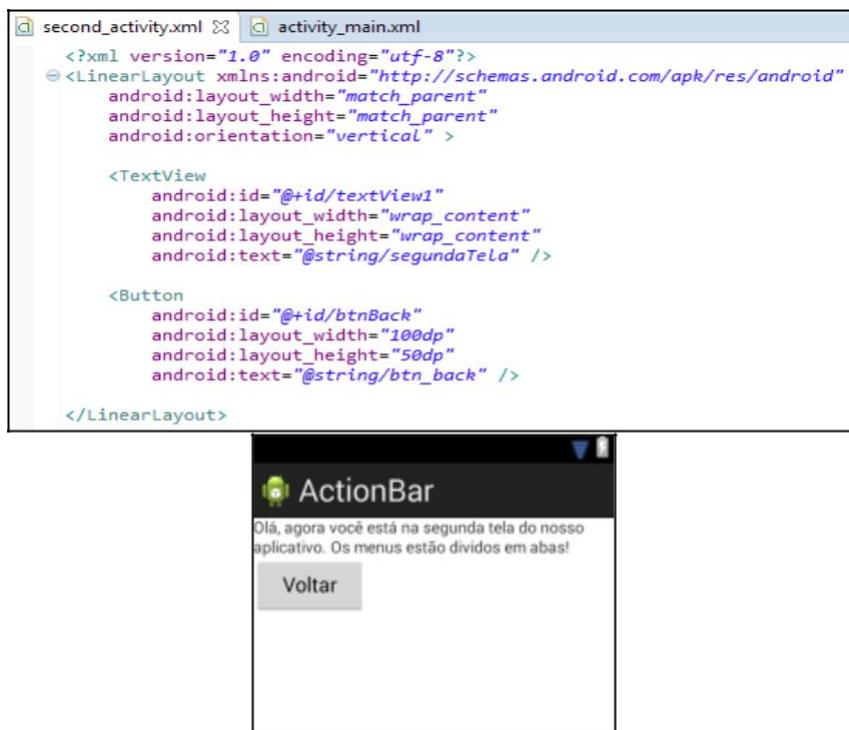
```

Na pasta layouts crie os arquivos **activity\_main** e **Second\_activity**.

- **activity\_main**



- **Second\_activity**



Aplique ao arquivo **AndroidManifest** os código referentes a imagem abaixo.

```

com.Filipe.actionbar Manifest

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.com.actionbar"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.com.actionbar.MainActivity"
            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=".SecondActivity"></activity>
    </application>

</manifest>

```

Execute o projeto.



Experimente as opções do Menu.  
Clique em Home.



Observe as opções na barra superior do emulador.



Para ver mais opções basta clicar e arrastar.



Clique em voltar para retornar a tela inicial, veja todas as opções deste projeto.

Muito bem chegamos ao final desta aula, não se esqueça de fazer os exercícios desta apostila.

### ***EXERCÍCIO DE FIXAÇÃO***

- 1)** Abra o Eclipse e crie um novo projeto.
- 2)** Altere o nome do programa para “Sexto Projeto”.
- 3)** Adicione os itens Arquivo, Exibir, Editar, Salvar e Deletar ao menu do aplicativo.
- 4)** Crie um submenu com 3 itens do menu principal.
- 5)** Crie uma ActionBar com pelo menos 4 opções e seus respectivos ícones



NOTAÇÕES





## NOTAÇÕES



123

Trabalhando com Intent

### O que é Intent?

Intent é uma estrutura de dados passiva que comporta uma descrição abstrata da operação a ser realizada. O Android utiliza os Intents para enviar mensagens de uma aplicação (activities, services e broadcast receivers).

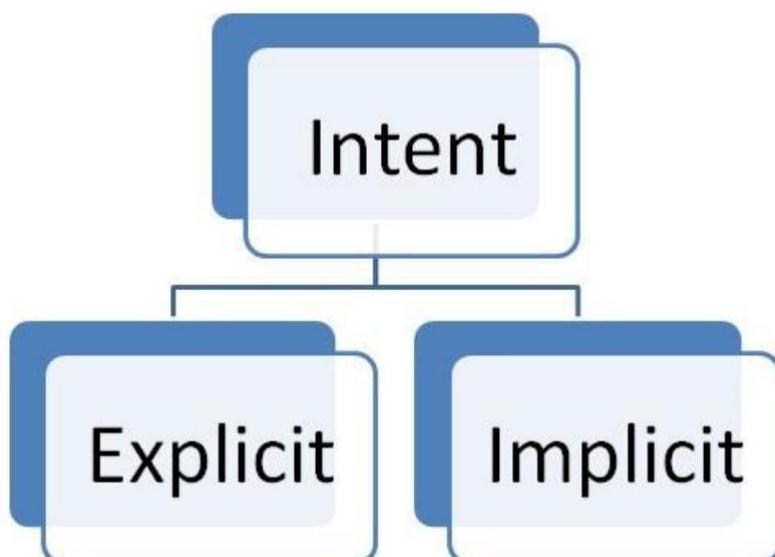
Por exemplo, é possível abrir uma Activity através do Intent com o código abaixo, que é bastante intuitivo.

```
public void onClick(View v) {  
    Intent intencao = new Intent(this, Tela2.class);  
    startActivity(intencao);  
}
```

Já para abrir outra aplicação como, por exemplo, o discador, utilizamos o código desta maneira:

```
public void onClick(View v) {  
    Intent intencao = new Intent(Intent.ACTION_DIAL,Uri.parse("tel:05501126267282"));  
    startActivity(intencao);  
}
```

Existem dois tipos de Intents, as implícitas e as explícitas.



Intents implícitas são quando a classe-alvo não é nomeada, como no exemplo que demos ao discador.

```
Intent intencao = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:05501126267282));
```

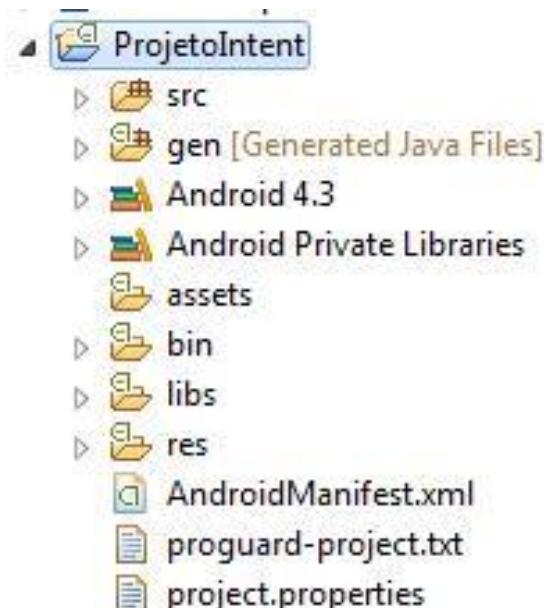
Já os explícitos informam exatamente a classe que se quer abrir.

```
Intent intencao = new Intent(this, Tela2.class);
```

### Intent Filter

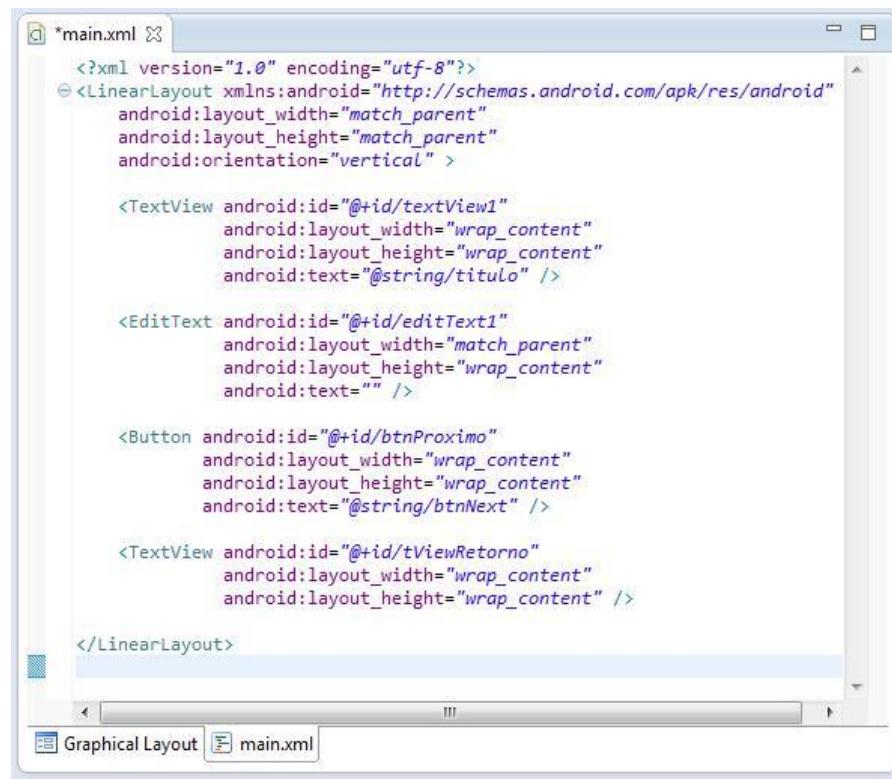
Pois bem, agora vamos abordar o Intent Filter e dar início aos exemplos.

Criamos um projeto novo para iniciar os exemplos:



O Intent Filter é declarado no AndroidManifest, mas primeiro vamos criar dois layouts e suas respectivas activities.

No primeiro layout criamos o básico. Um campo para digitar o nome e um botão para dar retorno ao TextView.



```

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

    <TextView android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/titulo" />

    <EditText android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="" />

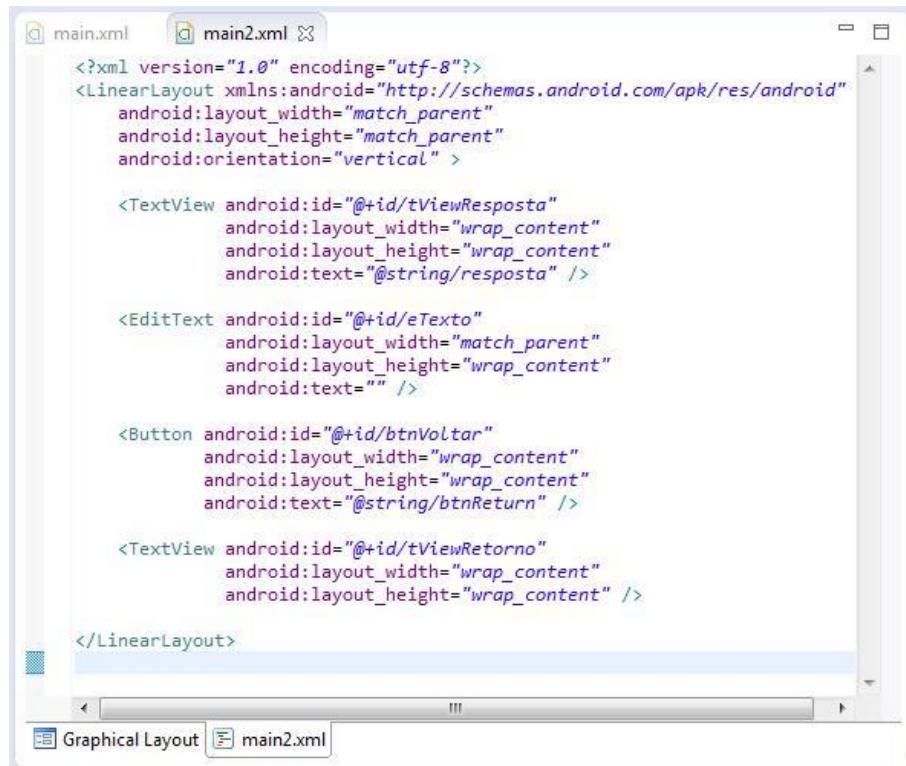
    <Button android:id="@+id/btnProximo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btnNext" />

    <TextView android:id="@+id/tViewRetorno"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>

```

No segundo Layout, criamos os mesmos widgets, porém o primeiro TextView vai receber informações do primeiro Layout e o botão é de voltar.



```

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

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

    <EditText android:id="@+id/eTexto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="" />

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

    <TextView android:id="@+id/tViewRetorno"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>

```

Agora criamos o segundo Layout, da forma que já conhece.

```

package com.example.projetointent;

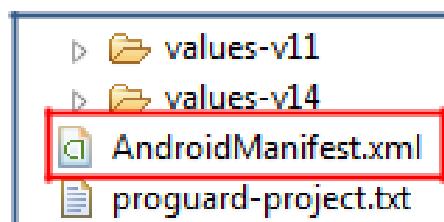
import android.app.Activity;
import android.os.Bundle;

public class SubActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main2);
    }
}

```

Agora vem a novidade, pois vamos mapear a activity no AndroidManifest utilizando o Intent Filter.

Clique duas vezes para abrir o AndroidManifest:



Repare que a activity e o Intent Filter do primeiro Layout já está presente.

```

<activity
    android:name="com.example.projetointent.MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER"
    </intent-filter>
</activity>

```

Veja a opção action criada:

```

<activity android:name=".SubActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.SUBACTIVITY"
            <category android:name="android.intent.category.DEFAULT"
        </intent-filter>
</activity>

```

Nesta opção foi criada a ação que será utilizada para referenciar o activity quando for chamado no código da activity principal.

A classe principal fica desta forma:

```

public class MainActivity extends Activity implements OnClickListener {

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

        Button btnProximo = (Button) findViewById(R.id.btnProximo);
        btnProximo.setOnClickListener(this);
    }

    public void onClick(View v) {
        Intent intencao = new Intent("com.example.projetointent.action.SUBACTIVITY");
        startActivity(intencao);
    }
}

```

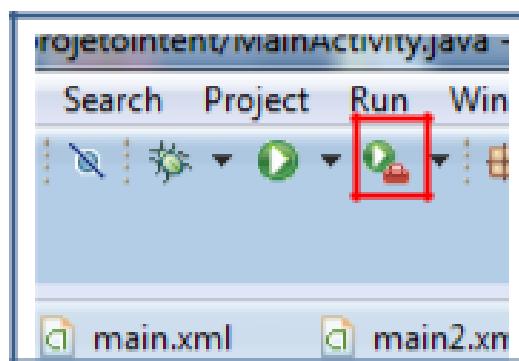
Repare que desta vez não foi preciso informar o nome da classe, apenas o nome da ação definida no AndroidManifest.xml.

```

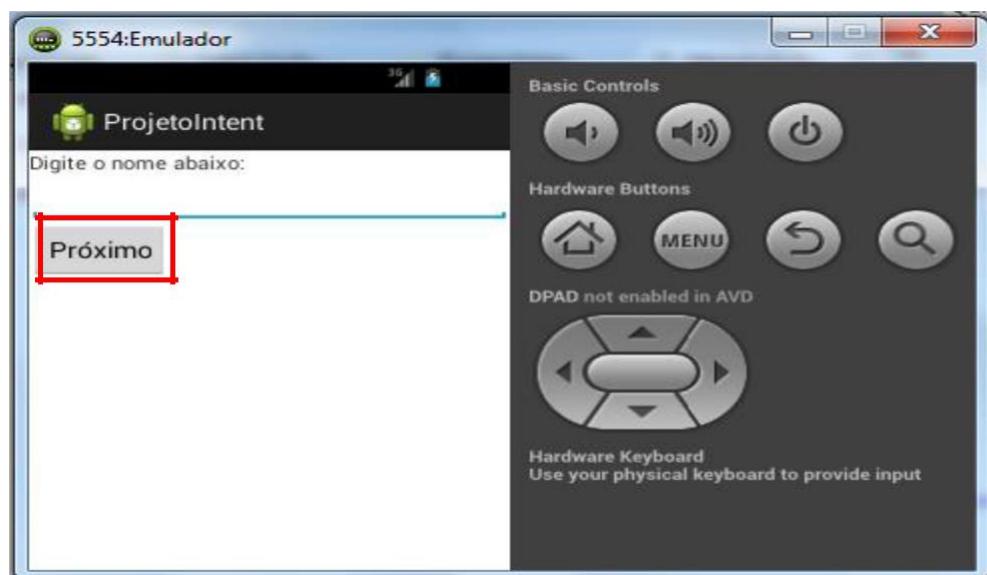
Intent intencao = new Intent("com.example.projetointent.action.SUBACTIVITY");
startActivity(intencao);

```

Clique para executar a ação:



Clique em Próximo, não é necessário digitar o nome ainda.



Veja que avançamos para a segunda tela, tudo através do Intent.



Vamos ver como podemos enviar dados de uma activity para outra através do Intent. Assim podemos trazer para a segunda tela o nome digitado.

Feche o emulador.



Já foi explicado que, para enviar dados para outra activity é preciso criar um Bundle, adicionar os dados no padrão chave/valor e adicionar o Bundle Intent.

Podemos fazer isto da seguinte forma:

```
public void onClick(View v) {
    Intent intencao = new Intent("com.example.projetointent.action.SU
    EditText txtNome = (EditText) findViewById(R.id.textView1);

    String nome = txtNome.getText().toString();

    Bundle parametros = new Bundle ();
    parametros.putString("nome", nome);
    intencao.putExtras(parametros);
    startActivity(intencao);
}
```

Mas também é possível adicionar os dados diretamente no Intent com o método putExtra, tendo menos trabalho.

```

public void onClick(View v) {
    Intent intencao = new Intent("com.example.projetointent.action.SU
    EditText txtNome = (EditText) findViewById(R.id.textView1);

    String nome = txtNome.getText().toString();

    intencao.putExtra("nome", nome);

    startActivityForResult(intencao);
}

```

Já na activity que está recebendo os dados, criamos o código desta maneira:

```

public class SubActivity extends Activity implements OnClickListener {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main2);

        Intent intencao = getIntent();

        Bundle extras = intencao.getExtras();
        if (extras == null) return;

        String nome = extras.getString("nome");
        if (nome != null){
            TextView tTitulo = (TextView) findViewById(R.id.tViewResposta)
            tTitulo.setText(nome + " digite algo abaixo:");
        }

        Button btnVoltar = (Button) findViewById(R.id.btnVoltar);
        btnVoltar.setOnClickListener(this);
    }

    public void onClick(View v){
        finish();
    }
}

```

Vamos comentar o código para você saber exatamente o que está sendo feito.

É preciso pegar a Intent atual:

```
Intent intencao = getIntent();
```

Depois, com ela, pegam-se os extras.

```
Bundle extras = intencao.getExtras();
```

Com o activity pode ser chamado por qualquer aplicação que informe a sua ação, deve-se verificar se o Bundle é null, ou seja: Se não possui nenhum valor.

```
if (extras == null) return;
```

Logo em seguida pegamos o valor da chave nome.

```
String nome = extras.getString("nome");
```

Depois verificamos se a chave possui algum valor. Se possuir, adicionamos o valor ao TextView.

```
if (nome != null){
    TextView tTitulo = (TextView) findViewById(R.id.tViewResposta);
    tTitulo.setText(nome + " digite algo abaixo:");
}
```

Em seguida, o botão foi vinculado ao evento Click.

```
Button btnVoltar = (Button) findViewById(R.id.btnVoltar);
btnVoltar.setOnClickListener(this);
```

Por fim, no método onClick, colocamos o método finish() para voltar à activity anterior.

```
public void onClick(View v){
    finish();
}
```

Clique em executar novamente.



Em nome, vamos digitar Gustavo e depois clique em Próximo.



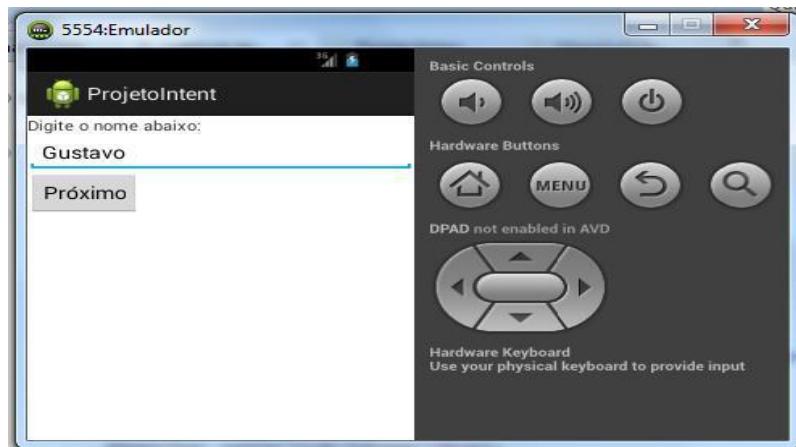
Pronto, veja que o nome apareceu na tela:



Voltar

Clique no botão Voltar para testar.

Pronto, veja que voltamos para a activity inicial.



Feche seu emulador.

### Realizando uma chamada

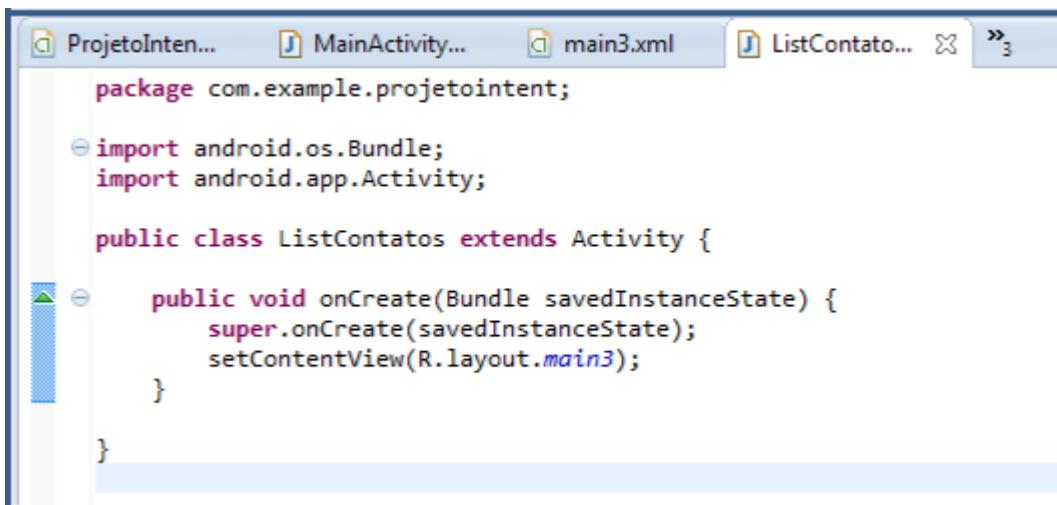
Vamos usar um exemplo de como realizar uma chamada utilizando um contato de celular.

Primeiro criamos um novo arquivo de layout. O layout deve conter os itens abaixo:

```
<TextView  
    android:id="@+id/textView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/titulo2" />  
  
<ListView  
    android:id="@+id/ListViewContatos"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" >  
</ListView>
```

Note que foi utilizado o ListView para listar os contatos.

Para chamar o arquivo de layout e carregar os contatos, criamos a activity ListContatos.



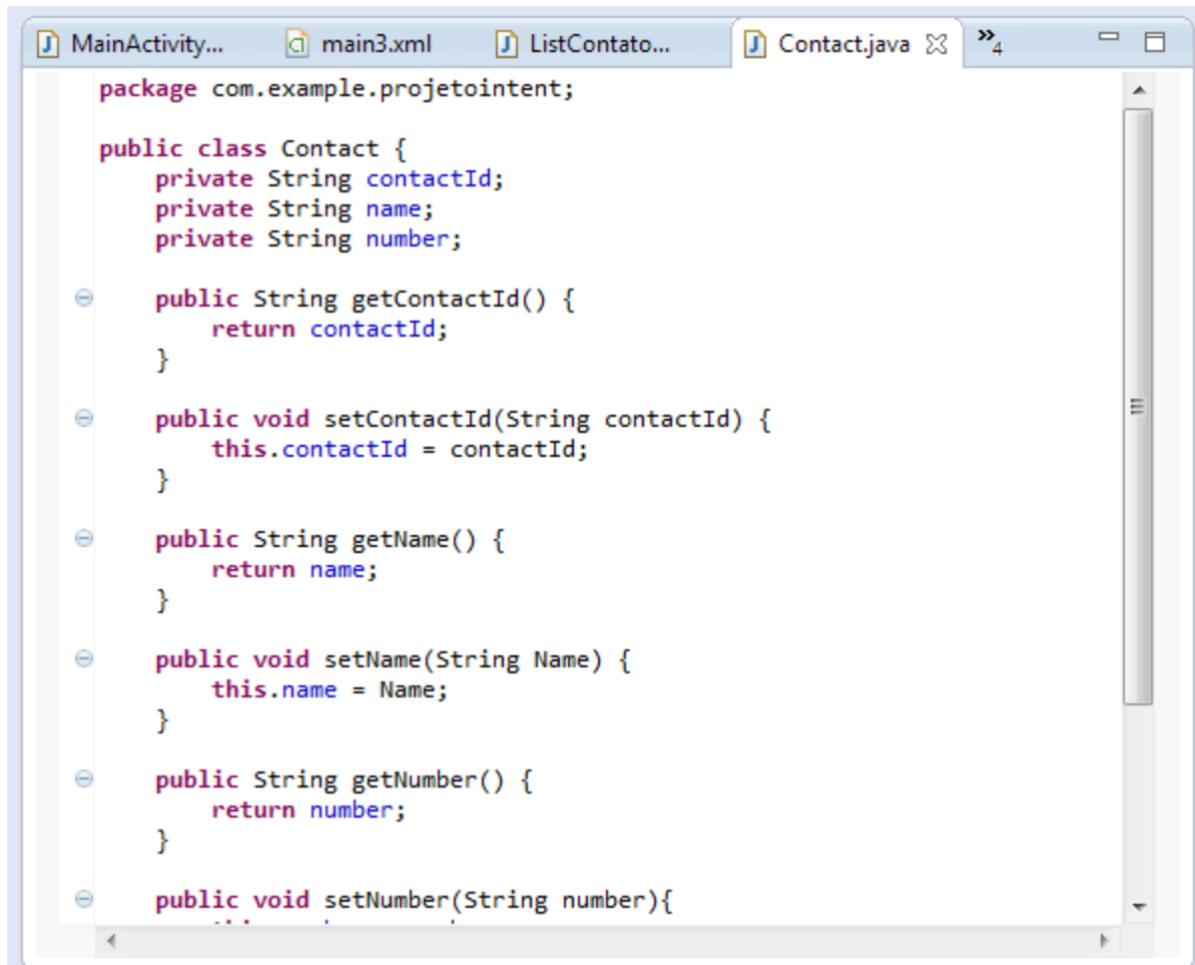
```
ProjetoIntent... MainActivity... main3.xml ListContato... > 3
package com.example.projetointent;

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

public class ListContatos extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main3);
    }
}
```

Antes de criar o contato, também criamos a classe Contact.java.



```
MainActivity... main3.xml ListContato... Contact.java > 4
package com.example.projetointent;

public class Contact {
    private String contactId;
    private String name;
    private String number;

    public String getContactId() {
        return contactId;
    }

    public void setContactId(String contactId) {
        this.contactId = contactId;
    }

    public String getName() {
        return name;
    }

    public void setName(String Name) {
        this.name = Name;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number){
        ...
    }
}
```

Utiliza esta classe para organizar os contatos. Ela não é a melhor forma porque um contato pode ter mais de um número, mas como o exemplo é simples você não precisa se preocupar com detalhes.

```
public class Contact {  
    private String contactId;  
    private String name;  
    private String number;  
  
    public String getContactId() {  
        return contactId;  
    }  
  
    public void setContactId(String contactId) {  
        this.contactId = contactId;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String Name) {  
        this.name = Name;  
    }  
  
    public String getNumber() {  
        return number;  
    }  
  
    public void setNumber(String number){  
        this.number = number;  
    }  
  
    public String toString() {  
        return name + ":" + number;  
    }  
}
```

Observe o método `toString()`, da classe objeto, foi sobescrito. Desta forma, é possível controlar a exibição da classe `Contact` do widget `ListView`.

```
public String toString() {  
    return name + ":" + number;  
}
```

Voltando a activity principal, criamos o método `getContats()` para carregar os contatos.

```

private List<Contact> getContacts(){
    List<Contact> contacts = new ArrayList<Contact>();
    try{
        Cursor contact = getContentResolver().query(
            ContactsContract.Contacts.CONTENT_URI, null, null, null,
            null);
        while(contact.moveToNext()) {
            int indexContactID = contact.getColumnIndex(ContactsContract.Contacts._ID);
            String ContactID = contact.getString(indexContactID);
            int indexDisplayName = contact.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME);
            String name = contact.getString(indexDisplayName);
            int indexHasPhone = contact.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER);
            String hasPhone = contact.getString(indexHasPhone);
        }
    }
}

```

O Código inteiro está abaixo, mas não se assuste, ele é mais simples do que parece. Vamos explicar.

```

private List<Contact> getContacts(){
    List<Contact> contacts = new ArrayList<Contact>();
    try{
        Cursor contact = getContentResolver().query(
            ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
        while(contact.moveToNext()) {
            int indexContactID = contact.getColumnIndex(ContactsContract.Contacts._ID);
            String ContactID = contact.getString(indexContactID);
            int indexDisplayName = contact.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME);
            String name = contact.getString(indexDisplayName);
            int indexHasPhone = contact.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER);
            String hasPhone = contact.getString(indexHasPhone);
            if(Integer.parseInt(hasPhone) == 1){
                Cursor phoneCursor = getContentResolver().query(
                    ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
                    ContactsContract.CommonDataKinds.Phone.CONTACT_ID+"='"+ContactID+"'", null, null);
                while(phoneCursor.moveToNext()){
                    int indexNumber = phoneCursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER);
                    String number = phoneCursor.getString(indexNumber);
                    Contact oContact = new Contact();
                    oContact.setContactId(ContactID);
                    oContact.setName(name);
                    oContact.setNumber(number);
                    contacts.add(oContact);
                }
                phoneCursor.close();
            }else{
                Contact oContact = new Contact();
                oContact.setContactId(ContactID);
                oContact.setName(name);
                contacts.add(oContact);
            }
        }
        contact.close();
    }catch(Exception e){
        e.printStackTrace();
    }
    return contacts;
}

```

Primeiro foi criado uma List de contatos, que contém todos os contatos.

```
List<Contact> contacts = new ArrayList<Contact>();
```

Depois, para percorrer os contatos do celular, foi criado um cursor.

```

Cursor contact = getContentResolver().query(
    ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

```

Para acessar os elementos do cursor, criamos um laço `while` e declaramos um inteiro para armazenar o índice na coluna ID do cursor.

```
while(contact.moveToNext()) {
    int indexContactID = contact.getColumnIndex(ContactsContract.Contacts._ID);
```

Utilizamos uma String para pegar o ID do contato.

```
String ContactID = contact.getString(indexContactID);
```

O mesmo procedimento foi repetido para pegar o Nome e o HAS\_PHONE\_NUMBER

```
int indexDisplayName = contact.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME);
String name = contact.getString(indexDisplayName);
int indexHasPhone = contact.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER);
String hasPhone = contact.getString(indexHasPhone);
```

O HAS\_PHONE\_NUMBER informa se o contato possui algum telefone. Se ele possuir o HAS\_PHONE\_NUMBER, terá valor 1.

```
if(Integer.parseInt(hasPhone) == 1){
```

Como um contato pode ter mais de um telefone, foi criado mais um cursor para percorrer os telefones de contato.

```
if(Integer.parseInt(hasPhone) == 1){
    Cursor phoneCursor = getContentResolver().query(
        ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
        ContactsContract.CommonDataKinds.Phone.CONTACT_ID+"='"+ContactID+"'", null, null);
```

O laço while foi utilizado novamente para percorrer o cursor.

```
while(phoneCursor.moveToNext()){
```

Também foi pego o índice da coluna Number e o número do contato.

```
int indexNumber = phoneCursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER);
String number = phoneCursor.getString(indexNumber);
```

Em seguida, foi criada uma instância da classe Contact.

```
Contact oContact = new Contact();
```

O id, nome e o número do contato no objeto da classe foram setados e o objeto adicionado na lista.

```
oContact.setContactId(ContactID);
oContact.setName(name);
oContact.setNumber(number);
contacts.add(oContact);
```

Fora do laço while do número do telefone, o cursor foi fechado.

```
phoneCursor.close();
```

Se o contato não possuir é criada uma nova instância. O id e nome do contato são setados ao objeto e adicionamos na lista.

```
}else{
    Contact oContact = new Contact();
    oContact.setContactId(ContactID);
    oContact.setName(name);
    contacts.add(oContact);
}
```

Por fim, fechamos o cursor de contatos e retomamos a lista.

```
}
contact.close();
}catch(Exception e){
    e.printStackTrace();
}
return contacts;
```

Agora, para carregar a lista, alteramos o método onCreate da maneira em destaque. Leia com atenção.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main3);

    List<Contact> contatos = getContacts();

    ArrayAdapter<Contact> oAdapter =
        new ArrayAdapter<Contact>(this, android.R.layout.simple_gallery_item, contatos);

    ListView list = (ListView) findViewById(R.id.listViewContatos);
    list.setAdapter(oAdapter);
}
```

Antes de executar precisamos alterar duas linhas no AndroidManifest.

Para ler os contatos foi preciso adicionar a permissão. Também adicionamos a activity inicial, da maneira que você já conhece.

Veja como fica o código:

```

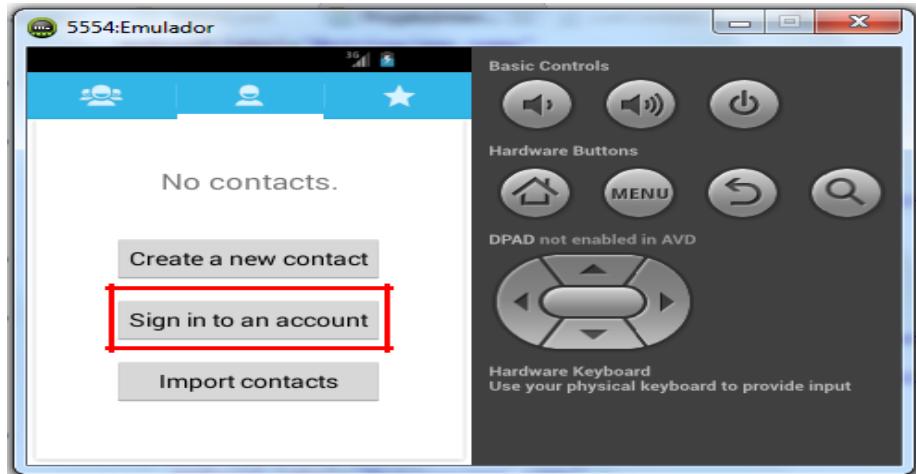
<activity
    android:name=".ListContatos"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name="com.example.projetointent.SubActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="com.example.projetointent.action.SU
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
</activity>
<activity
    android:name=".MainActivity"
    android:label="@string/app_name" >
</activity>
</application>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
</manifest>
```

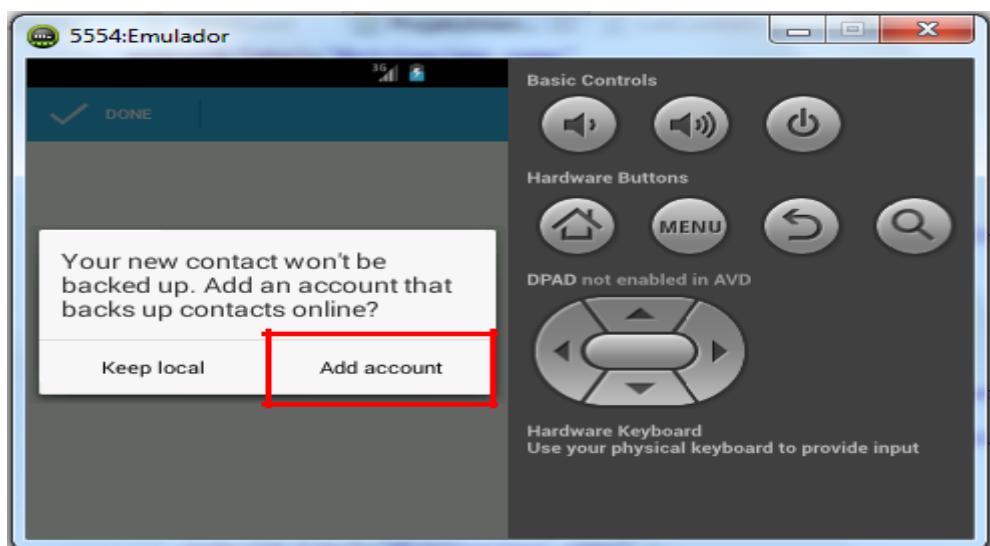
Finalmente podemos testar, clique para abrir o emulador novamente. Para adicionar um contato, você clica no ícone “Contatos”.



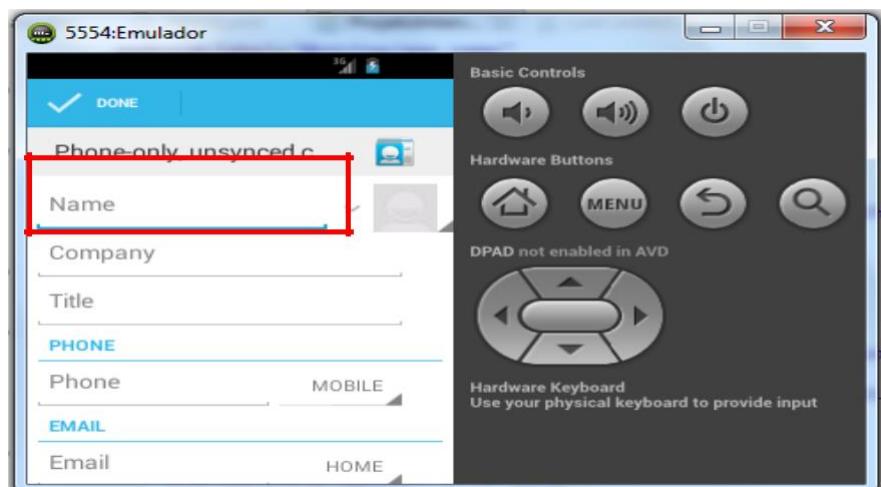
Agora, clicamos na opção de criar um novo contato.



Teremos então a opção de criar um contato local ou adicionar os contatos de uma conta de email. Selecione para criar um contato local.



Clique para inserir um nome.



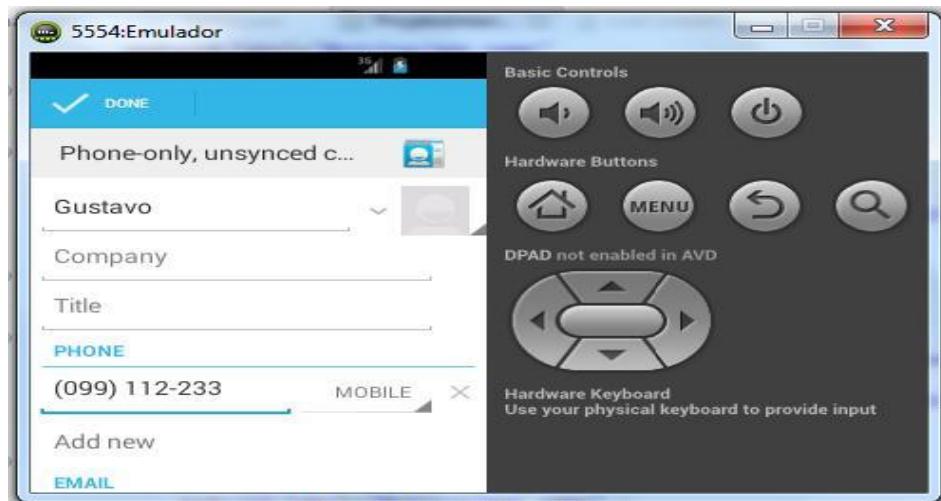
Como nome, digitamos “Gustavo” sem aspas.



Então, clicamos na opção para inserir um número de telefone.



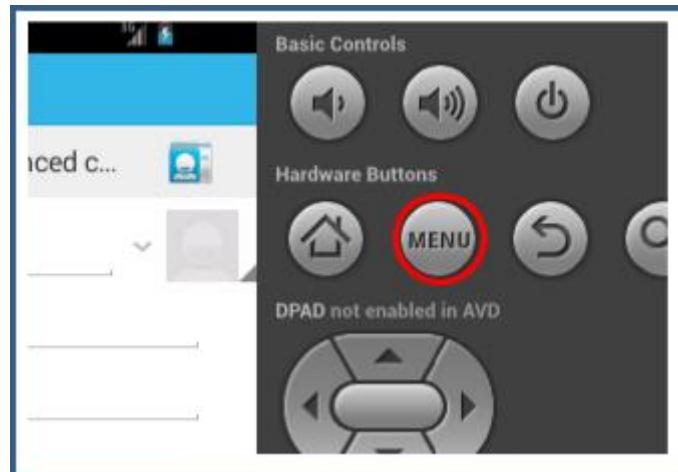
Digite “099112233” como número simbólico. Sem as aspas também.



Clique em “Done” para adicionar o contato.



Pronto. Agora clique na casa para voltar à área de trabalho.



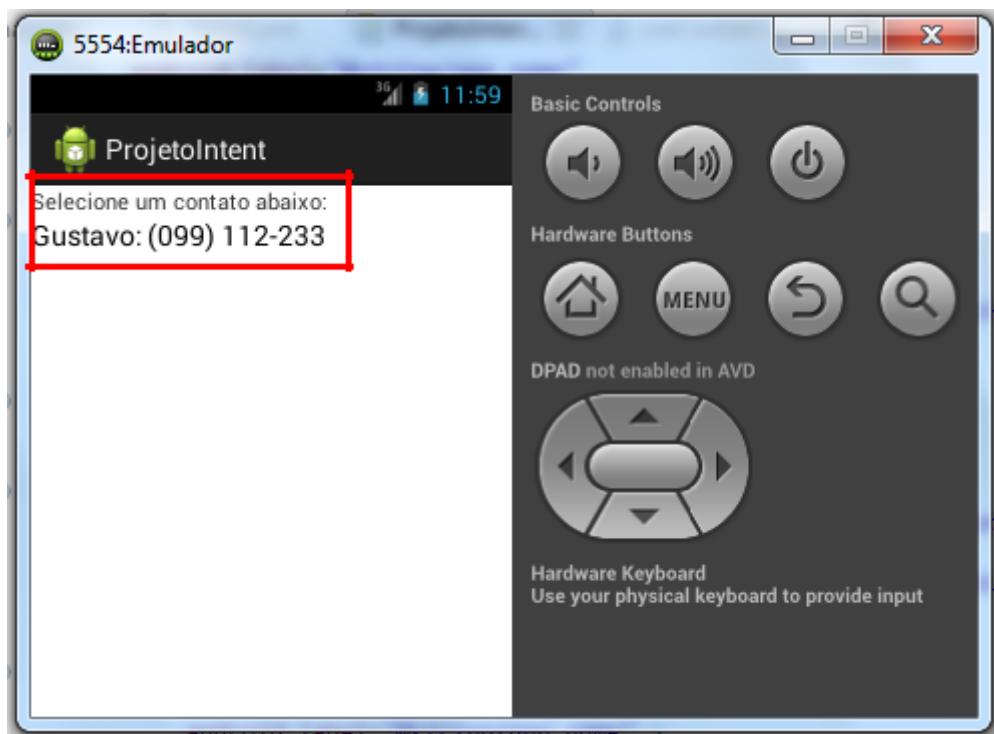
Clique para abrir os aplicativos:



Clique e abra o nosso aplicativo.



Veja que o Intent carregou o contato para o botão.



Desta forma você tem tudo pronto, basta você implementar o ACTION\_CALL.

Muitos bem, chegamos ao final desta segunda aula, não se esqueça de fazer os exercícios desta apostila.

### **EXERCÍCIO DE FIXAÇÃO**

- 1)** Abra o Eclipse e crie um novo projeto.
- 2)** Altere o nome do programa para “Setimo Projeto”.
- 3)** Adicione um EditText e um botão de avançar à tela.
- 4)** Crie outra activity com um TextView e um botão de voltar
- 5)** Utilize o Bundle e o Intent para levar a informação do EditText para o TextView da próxima tela.
- 6)** Crie outra activity que exiba os contatos do telefone através do Intent.





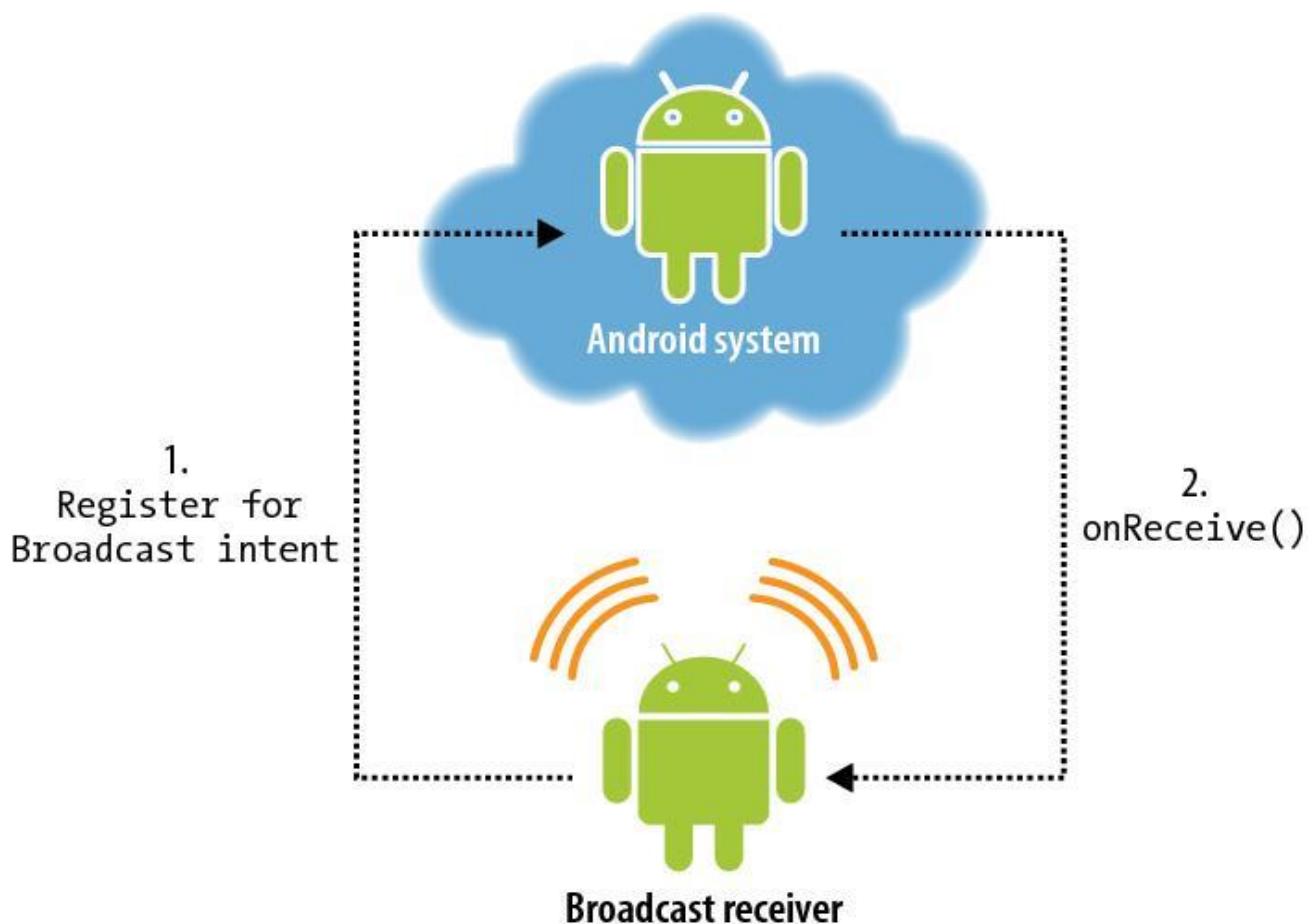


## NOTAÇÕES



**BroadcastReceiver**

Nesta última aula, vamos entender o que é BroadcastReceiver.



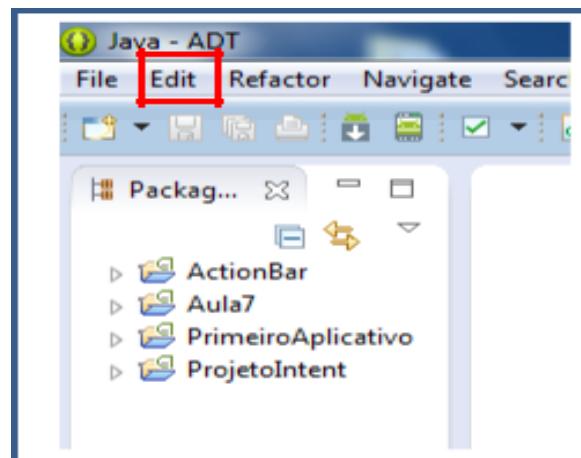
Um aparelho de telefone, durante seu funcionamento, realiza “eventos” como receber e realizar uma chamada, carregar a bateria, enfim, o tempo todo realiza uma operação.

No Android, para informar as aplicações sobre quais eventos estão sendo executados, ele utiliza o BroadcastReceiver.

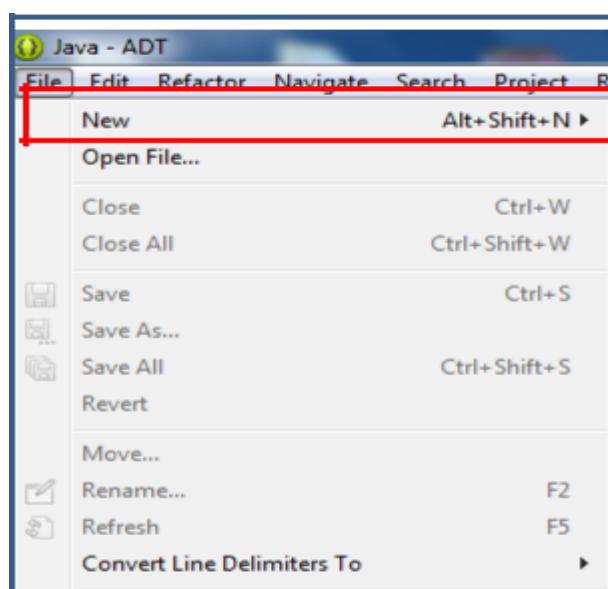
O BroadcastReceiver também pode ser utilizado para que a aplicação seja executada assim que um evento for executado.

Vamos criar um exemplo. Precisamos criar uma classe que estende a classe BroadcastReceiver e registrar no AndroidManifest. Assim, toda vez que o evento “escutado” for executado o Android irá chamar a classe e executar o método onReceive().

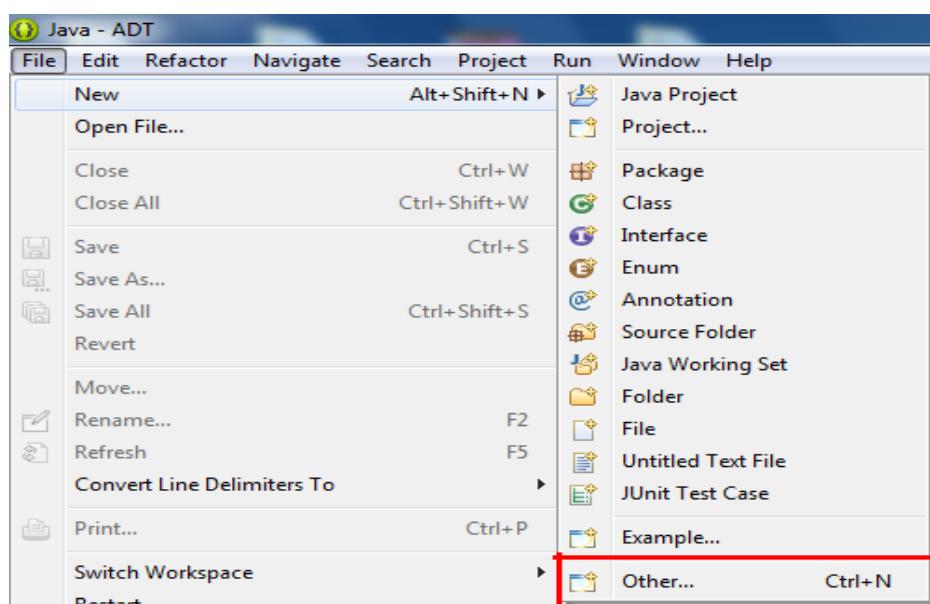
Para iniciarmos o exemplo, vamos criar um novo projeto clicando em File:



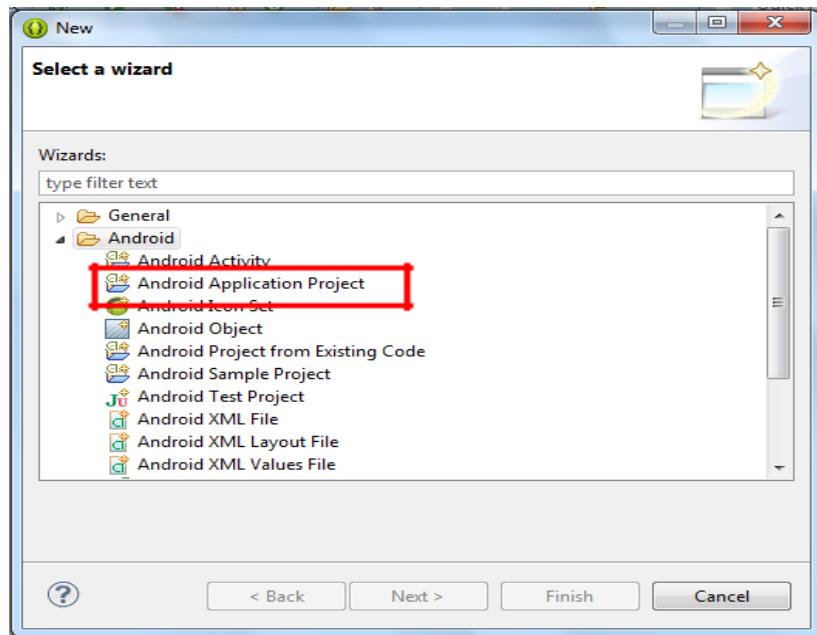
Mova o cursor do mouse até “New”.



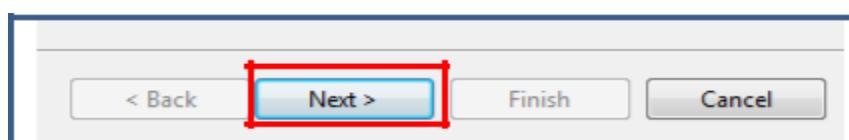
Clique em “Other”.



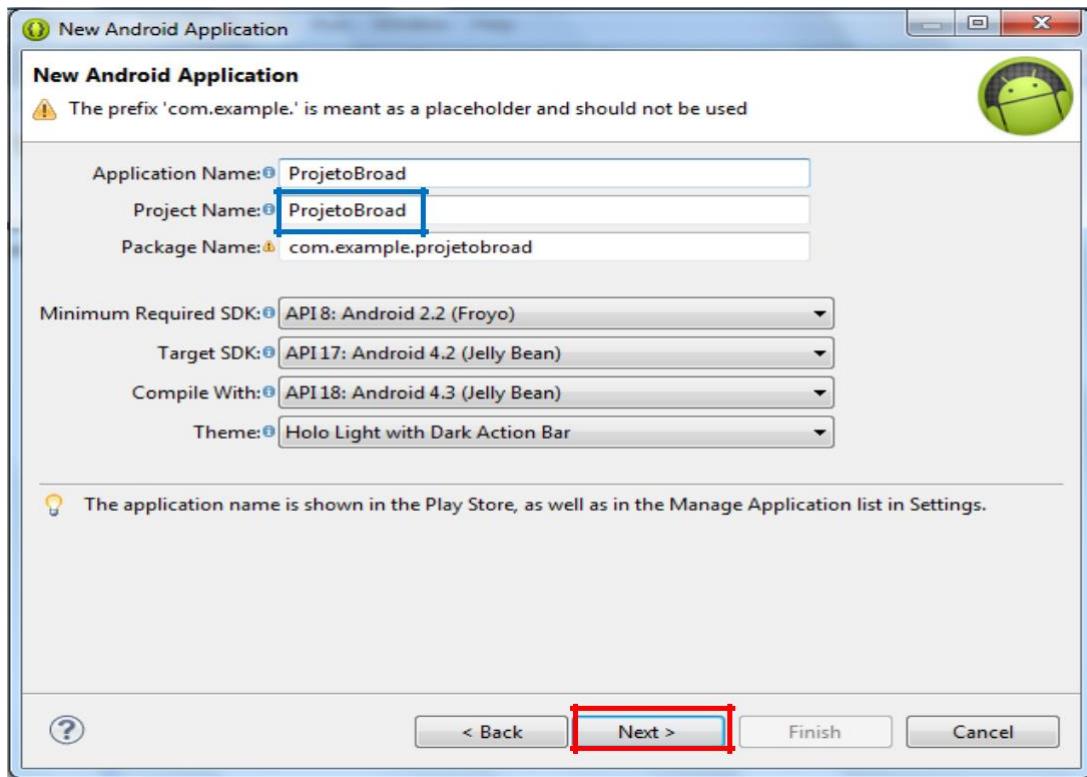
Selecione o projeto Android.



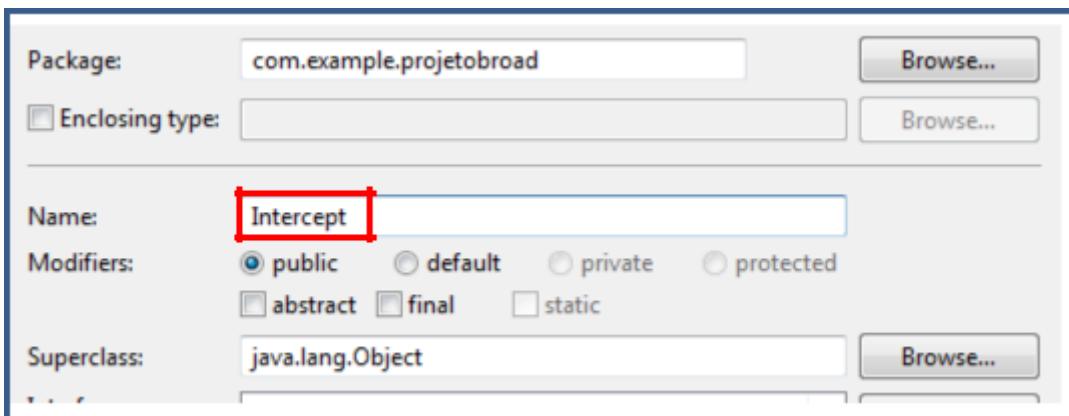
Clique em Next.



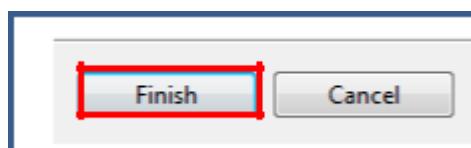
Em Application name, digitamos o nome ProjetoBroad, depois continuamos clicando em Next.



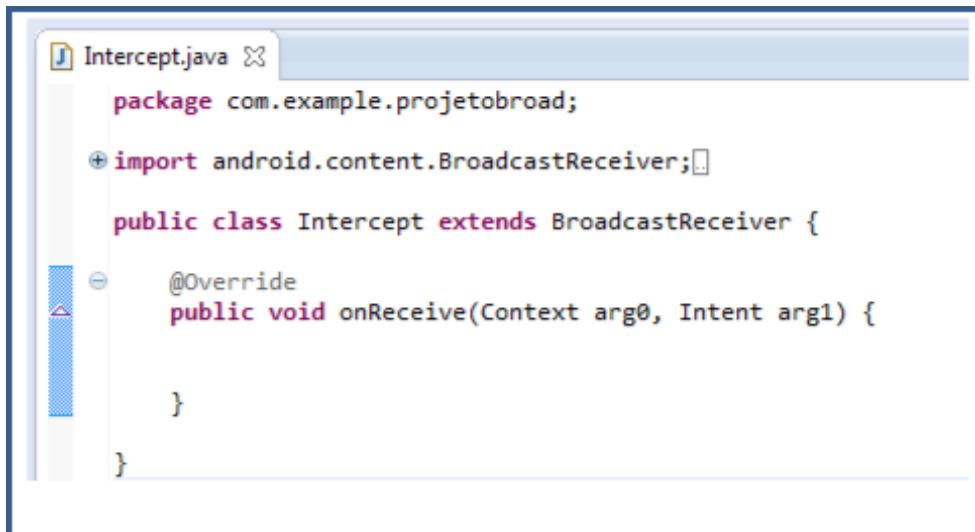
Agora vamos criar a classe que estende o BroadcastReceiver.  
 Como nome da classe digitamos “Intercept”, sem aspas.



Clique em Finish.



Pronto. O evento a ser “escutado” será o recebimento de uma mensagem de texto.



```

package com.example.projetobroad;

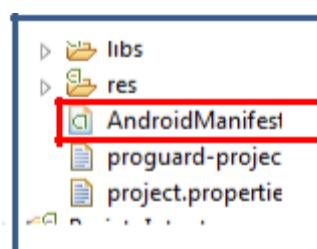
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class Intercept extends BroadcastReceiver {

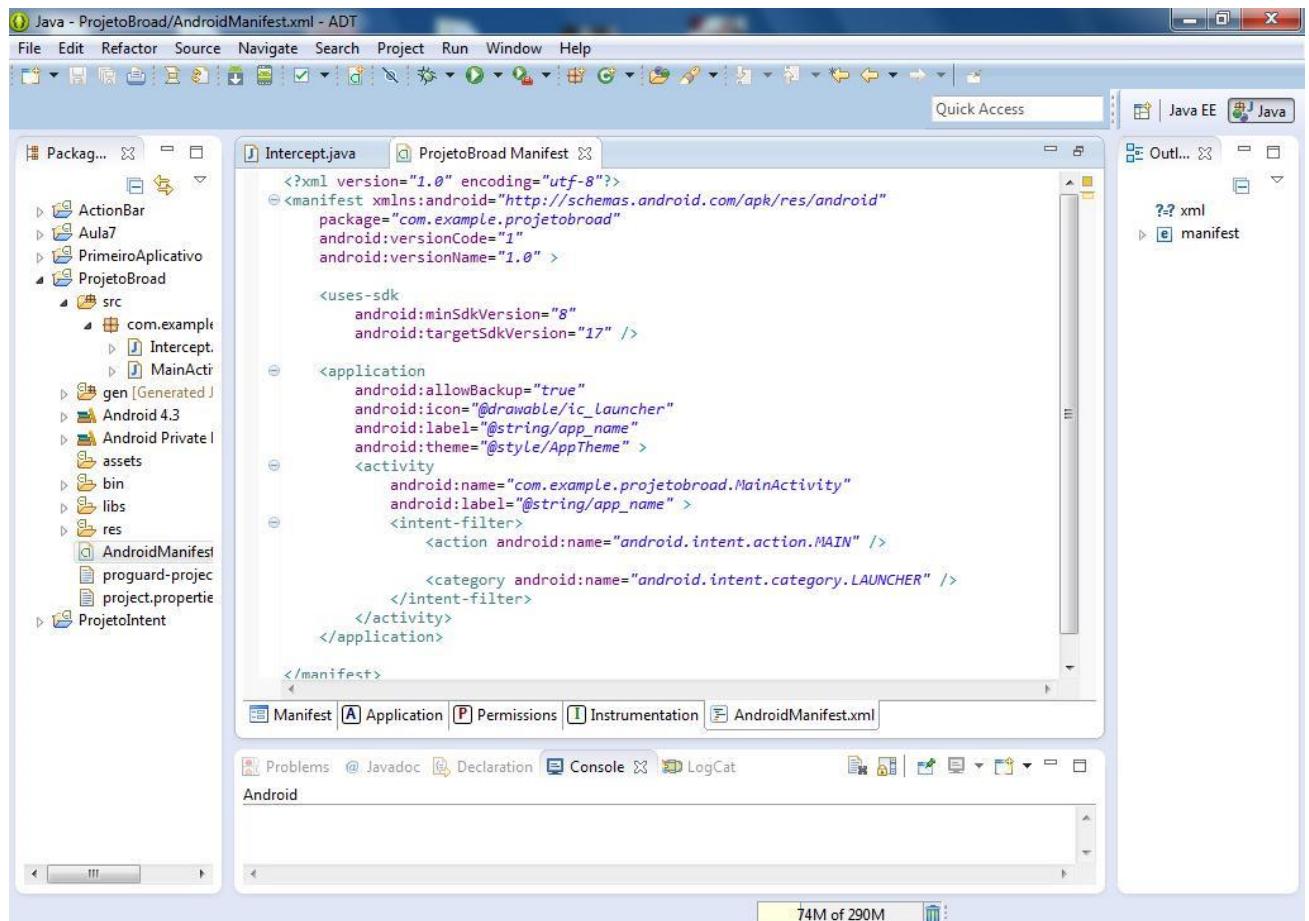
    @Override
    public void onReceive(Context arg0, Intent arg1) {
    }
}

```

Neste caso, é preciso registrar o receiver e para isso, clicamos duas para editar o AndroidManifest.



Vamos adicionar o receiver e também as permissões necessárias para escutar o evento do recebimento de mensagens de texto e leitura de contatos.



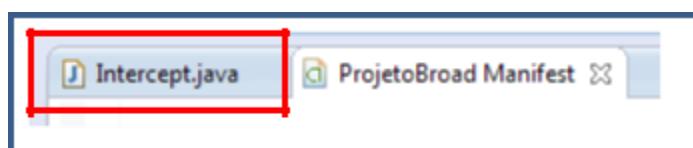
Note que fizemos isso para você:

```

<receiver android:name=".Intercept">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</receiver>
</application>
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />

```

Clique para voltar ao receiver.



Vamos implementar o método `onReceive`. Adicionamos o código abaixo:

Preste atenção no código, pois agora vamos detalhar para você.

```
public void onReceive(Context context, Intent intent) {  
    Bundle extras = intent.getExtras();  
    if (extras == null)  
        return;  
  
    Object[] pdus = (Object[]) extras.get("pdus");  
  
    if (pdus.length > 0) {  
        SmsMessage message = SmsMessage.createFromPdu( (byte[]) pdus[0]);  
        String fromAddress = message.getOriginatingAddress();  
        String messageBody = message.getMessageBody().toString();  
        String fromDisplayName = fromAddress;  
  
        Uri uri;  
        String[] projection = null;
```

```

        uri = Uri.withAppendedPath(
            ContactsContract.PhoneLookup.CONTENT_FILTER_URI,
            Uri.encode(fromAddress));

        Cursor cursor = context.getContentResolver().query(uri, projection, null, null, null);
        if(cursor !=null){
            if (cursor.moveToFirst ())
                fromDisplayName = cursor.getString(0);
            cursor.close();
        }
        Intent intencao = new Intent(context, MainActivity.class);
        intencao.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

        intencao.putExtra("number", fromAddress);
        intencao.putExtra("name", fromDisplayName);
        intencao.putExtra("message", messageBody);

        context.startActivity(intencao);
    }

}

```

Primeiro pegamos os dados passados pelo Intent, se ele não possuir nenhum dado, cancela-se a execução do método.

```

Bundle extras = intent.getExtras();
if (extras == null)
    return;

Object[] pdus = (Object[]) extras.get("pdus");

```

O valor pdus é o padrão do Android – vem da sigla PDU (Protocol Data Unit) que é um padrão para envio de dados em telecomunicações.

Logo em seguida é verificado se alguma mensagem foi recebida, se sim, cria-se um objeto SmsMessage.

```

if (pdus.length > 0) {
    SmsMessage message = SmsMessage.createFromPdu( (byte[]) pdus[0]);
}

```

Depois criam-se duas variáveis para armazenar o número e o texto da mensagem.

Logo em seguida o número é atribuído ao nome de remetente.

```

String fromAddress = message.getOriginatingAddress();
String messageBody = message.getMessageBody().toString();
String fromDisplayName = fromAddress;

```

Adiante será verificado se o número do remetente se encontra na agenda de citados, se sim, este nome será o nome atribuído a variável *DisplayName*.

Para pesquisar o contato, são criados o uri e o projection para depois atribuir valores às variáveis.

```
Uri uri;
String[] projection;

uri = Uri.withAppendedPath(
    ContactsContract.PhoneLookup.CONTENT_FILTER_URI,
    Uri.encode(fromAddress));
projection = new String[] { ContactsContract.PhoneLookup.DISPLAY_NAME};
```

Na primeira variável foi criado um filtro para os contatos, de acordo com o número de telefone da mensagem. Já na segunda, foi informado

Em seguida, foi criado um cursor para pesquisar os contatos do celular de acordo com o filtro. Se ele encontrar algum contato pegará o nome retornado e atribuirá à variável *fromDisplayName*.

```
Cursor cursor = context.getContentResolver().query(uri, projection, null, null, null);
if(cursor !=null){
    if (cursor.moveToFirst ())
        fromDisplayName = cursor.getString(0);
    cursor.close();
```

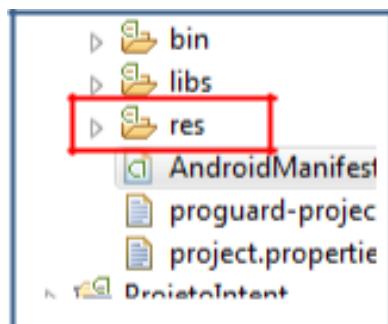
Com todos os dados, cria-se um Intent para o MainActivity, adicionam-se os dados da mensagem e inicia-se o activity.

```
Intent intencao = new Intent(context, MainActivity.class);
intencao.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

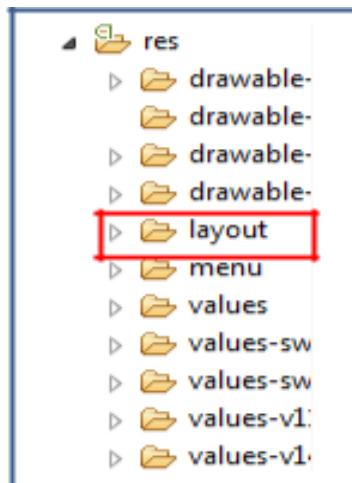
intencao.putExtra("number", fromAddress);
intencao.putExtra("name", fromDisplayName);
intencao.putExtra("message", messageBody);

context.startActivity(intencao);
```

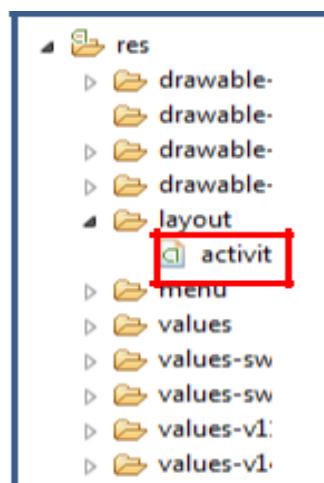
Agora, vamos alterar o layout. Para isto clicamos na pasta *res*.



Depois, clique em layout:



Clique duas vezes para abrir o activity\_main.



Adicionamos quatro TextViews nele para você.

```
<TextView
    android:id="@+id/tViewTitulo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/titulo" />

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

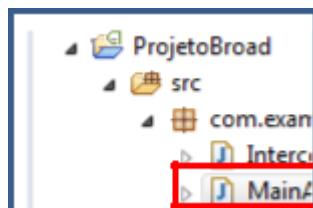
<TextView
    android:id="@+id/tViewNumero"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/numero" />

<TextView
    android:id="@+id/tViewMensagem"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/mensagem" />
```

Atribuímos as Strings em destaque à elas.

```
<string name="app_name">ProjetoBroad</string>
<string name="action_settings">Settings</string>
<string name="titulo">Dados da mensagem:</string>
<string name="remetente">Remetente:</string>
<string name="numero">Número:</string>
<string name="mensagem">Mensagem:</string>
```

Clique duas vezes para abrir o activity.



Implementamos o método onCreate para você.

```
public class MainActivity extends Activity {

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

        Bundle extra = getIntent().getExtras();

        if (extra == null)
            return;

        TextView viewRemetente = (TextView) findViewById(R.id.tViewRemetente);
        TextView viewNumero = (TextView) findViewById(R.id.tViewNumero);
        TextView viewMensagem = (TextView) findViewById(R.id.tViewMensagem);

        String remetente = "";
        String numero = "";
        String mensagem = "";

        if (extra.getString("name") != null){
            remetente = "remetente: " + extra.getString("name");
            numero = "Número: " + extra.getString("number");
            mensagem = extra.getString("message");
        }
    }
}
```

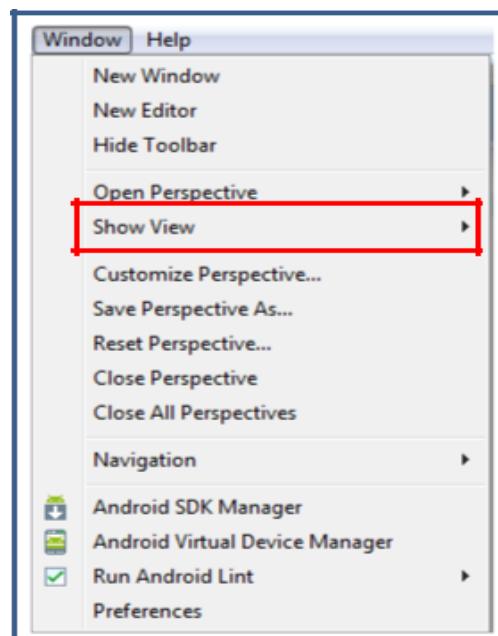
Neste caso, utilizamos o onCreate apenas como construtor, atribuindo valores para os objetos.

Pois bem, agora é preciso emular um envio de mensagem de texto para testar o aplicativo.

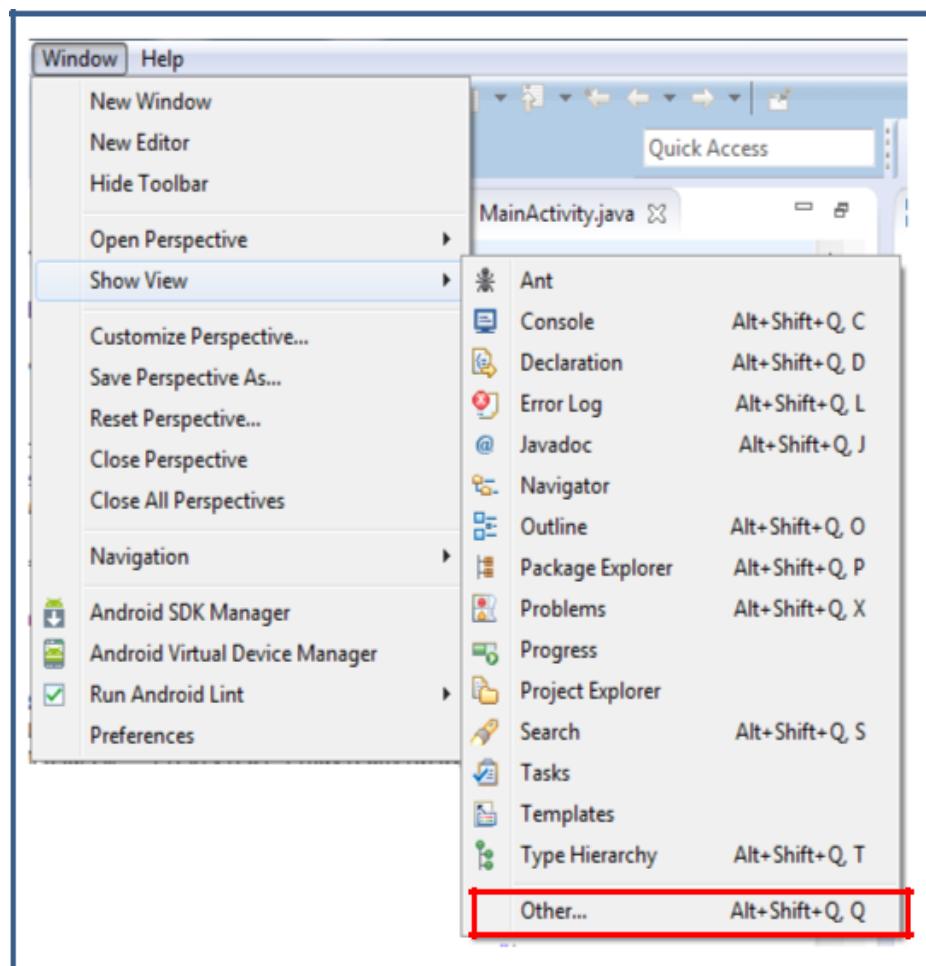
Clicamos em Window:



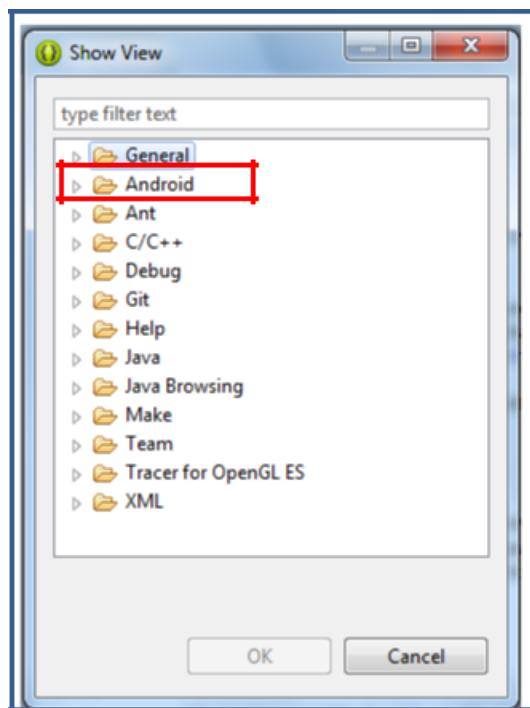
Mova o cursor do mouse até Show View:



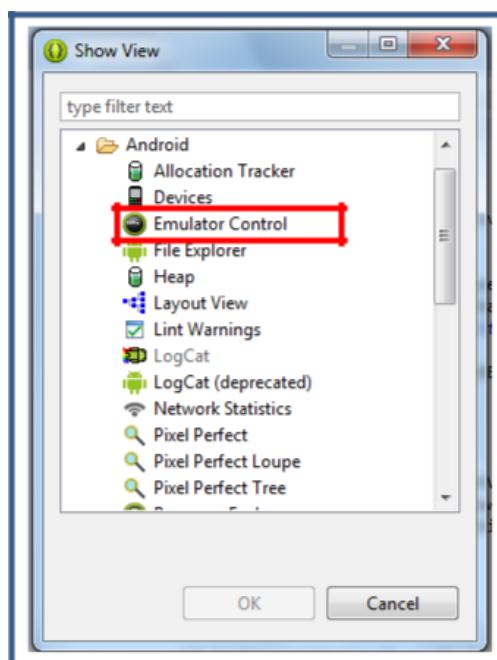
Clique em Other:



Agora, abra a pasta Android:



Selecione Emulator Control:



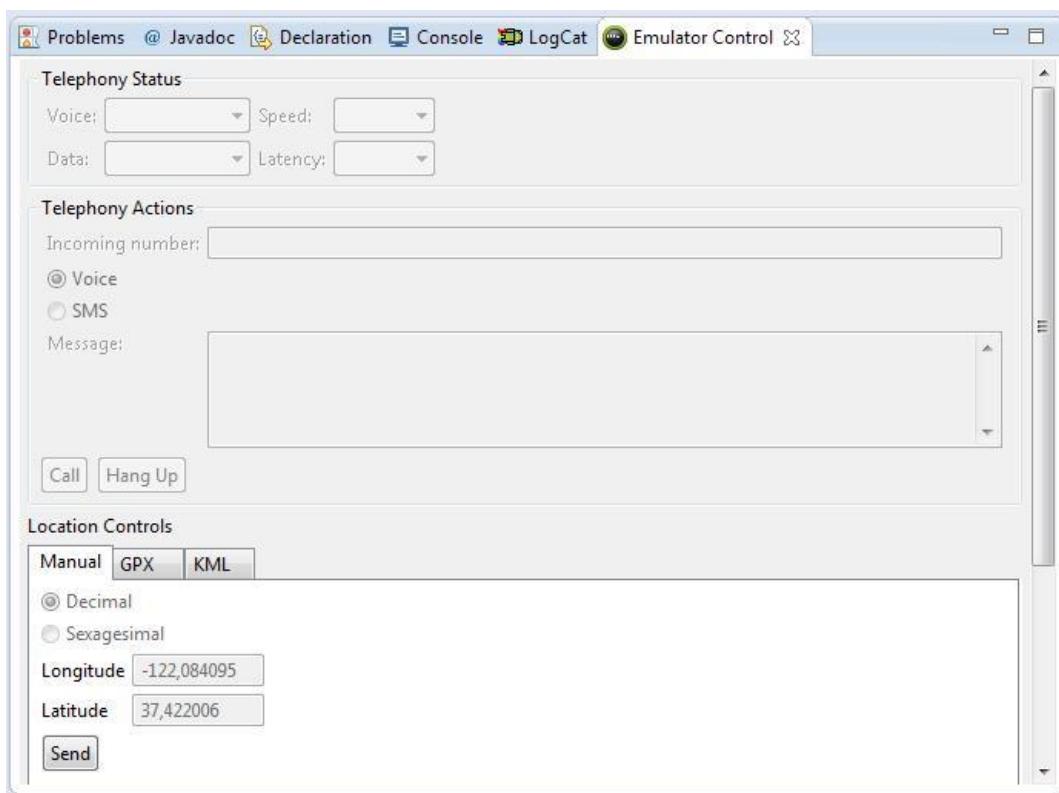
Depois de selecionado, clique em Ok.



Faça conforme descrito:



Veja que está tudo desabilitado.



Para habilitar, devemos deixar o emulador sempre aberto:



Clique no campo *Incoming number* para informar o telefone:

Telephony Status

Voice: home ▾ Speed: Full ▾

Data: home ▾ Latency: None ▾

Telephony Actions

Incoming number:

Digite 99009900 como número simbólico:

Telephony Actions

Incoming number: 99009900

Agora, selecione a opção de enviar SMS:

Problems @ Javadoc Declaration Console LogCat Emulator Control

Telephony Status

Voice: home ▾ Speed: Full ▾

Data: home ▾ Latency: None ▾

Telephony Actions

Incoming number: 99009900

Voice

SMS

Message:

Digitamos “Mensagem teste” sem aspas e clicamos em *Send* para enviar.

**Telephony Actions**

Incoming number: 99009900

Voice  
 SMS

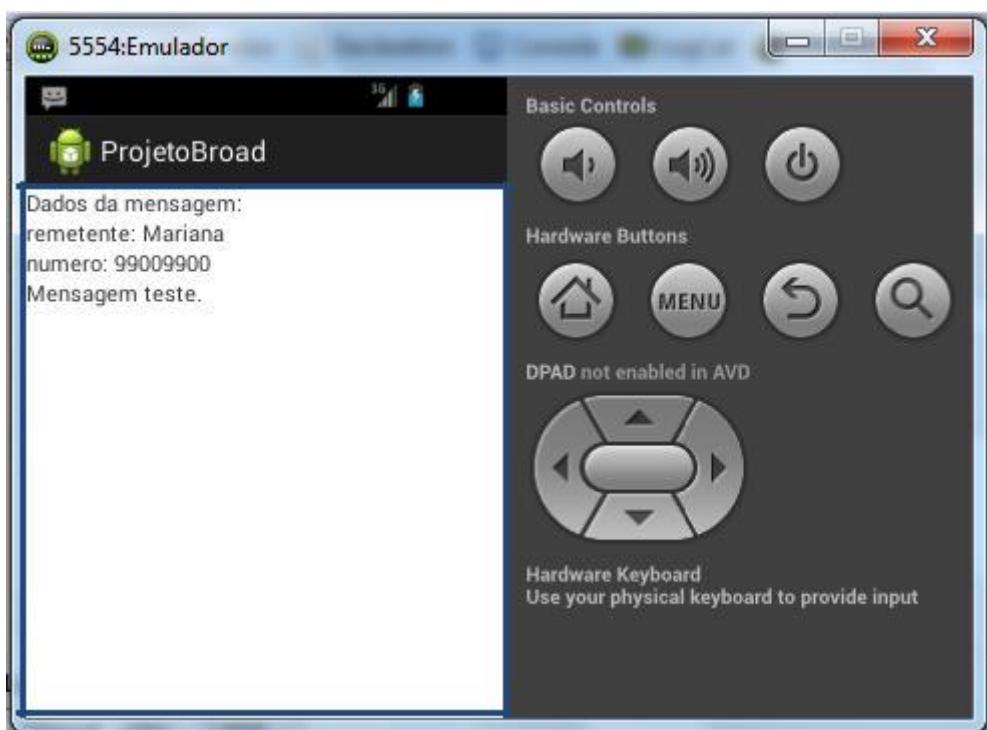
Message: Mensagem teste

**Send** **Hang Up**

Ótimo. Agora clique no emulador para verificarmos se a mensagem foi enviada corretamente.



Pronto. Como você pode perceber nosso aplicativo recebeu a mensagem, informando o nome do contato que contém o número e a própria mensagem.



Assim encerramos o conteúdo deste curso. Agora é somente com você. Programação depende muito mais da sua pretensão e interesse em buscar novos meios de codificação. Lembre-se que sem pretensão e prática você não terá o proveito que o curso proporciona.

## ***EXERCÍCIO DE FIXAÇÃO***

- 1)** Abra o Eclipse e crie um novo projeto.
- 2)** Altere o nome do programa para “Setimo Projeto”.
- 3)** Crie uma classe de nome Intercept como receiver.
- 4)** Implemente o método onReceive e o layout de modo que o aplicativo mostre a mensagem, o nome e o número do contato que enviou.
- 5)** Adicione as permissões de receber SMS e ler contatos ao AndroidManifest.

Adicione um contato qualquer na lista de contatos do emulador e envie uma mensagem deste número através do Emulator Control para testar aplicativo.

## NOTAÇÕES









