

# Google Android



**Ricardo Cherobin**  
**[ricardo@cherobin.com.br](mailto:ricardo@cherobin.com.br)**

# Programação

- Preparação do ambiente
- Introdução
- Estrutura geral da plataforma Android
- Activities
- Tasks
- Services
- Broadcast Receivers
- Content Providers
- XML
- Widgets:
  - Text View, Edit Text, Button, Check box, Radio Button, Spinner, List View, ProgressBar.
- Publicando Seu aplicativo no Google Play



# Instalando o Eclipse e o Android



- <http://www.eclipse.org/downloads/>
- <http://developer.android.com/sdk/index.html>

# Configurar o Eclipse para o Android

- O Plugin
- Android oferece um plugin personalizado para o IDE Eclipse, chamado Android Development Tools (ADT). Este plugin é concebido para lhe fornecer um ambiente integrado poderoso, que permite desenvolver aplicativos para Android. Ela estende as capacidades do Eclipse para que você rapidamente possa criar novos projetos Android, construir uma interface de usuário do aplicativo, depurar seu aplicativo, e exportação de app (APKs) para distribuição assinados (ou não assinado).
- Certifique-se que você tem uma versão adequada do Eclipse instalado em seu computador, conforme descrito pelos requisitos do sistema.
- Se você precisa instalar o Eclipse, você pode baixá-lo <http://www.eclipse.org/downloads/> . Recomendamos o “Eclipse Classic” versão. Caso contrário, você deve usar uma versão Java ou RCP do Eclipse.”

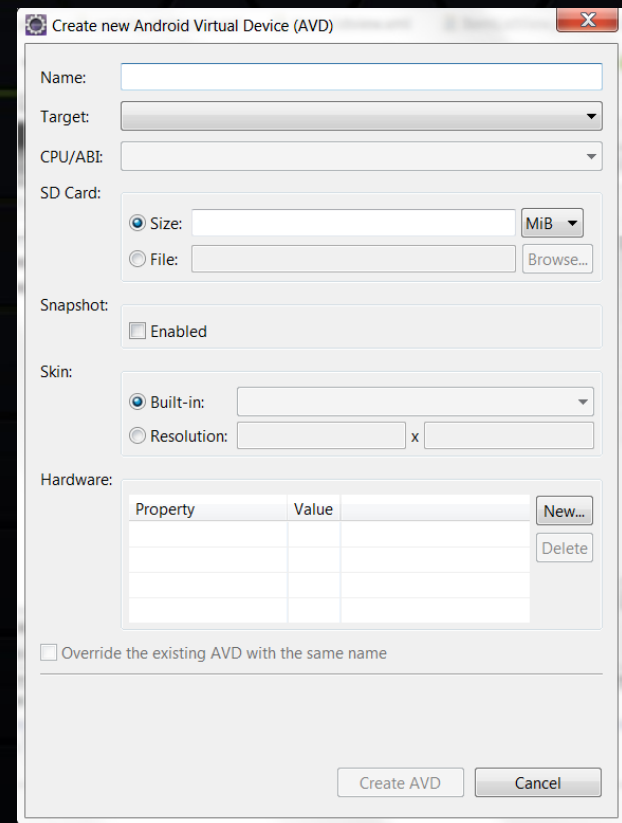
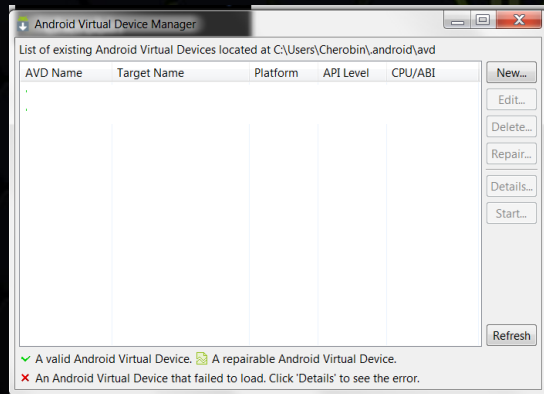
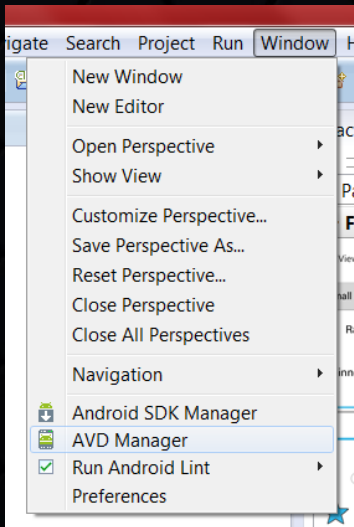


- Como baixar o plugin ADT e instalar
- Inicie o Eclipse, em seguida, selecione Help > Install New Software (Ajuda> Instale Software Novo) ....
- Clique em Add (Adicionar), no canto superior direito.
- Na caixa de diálogo Add Repository que aparece, digite “ADT Plugin” para o nome ea URL a seguir para o local:
- <https://dl-ssl.google.com/android/eclipse/>
- Clique em OK.
- Nota: Se você tiver problemas para adquirir o plugin, tente usar o “http” na URL Localização, em vez de “https” (https é preferido por razões de segurança)
- Na caixa de diálogo Available Software (Software disponíveis), selecione a caixa de seleção Developer Tools e clique em Next (Avançar).
- Na próxima janela, você verá uma lista das ferramentas para ser baixado. Clique em Next (Avançar).
- Leia e aceite os acordos de licença, clique em Finish (Concluir).
- Nota: Se você receber um aviso de segurança dizendo que a autenticidade ou validade do software não pode ser estabelecida, clique em OK.
- Quando a instalação estiver concluída, reinicie o Eclipse.

- Configurar o plugin ADT
- Depois de instalar o ADT e reiniciado Eclipse, você deve especificar a localização do seu diretório SDK do Android:
- Selecione Window> Preferences ... para abrir o painel de Preferências.
- Selecione Android no painel da esquerda.
- Você pode ver uma caixa de diálogo perguntando se você deseja enviar estatísticas de uso para o Google. Se assim for, faça a sua escolha e clique em Proceed (Continuar).
- Para o local do SDK no painel principal, clique em Browse (Procurar) ... e localize o diretório do Android SDK .
- Clique em Apply (Aplicar) e depois em OK.

# Instalando o Eclipse e o Android

- Para criarmos um dispositivo virtual clique no botão “New”, e será aberta uma tela conforme mostra a figura : ----->





# Instalando o Eclipse e o Android

- De início, vamos configurar o básico pra executarmos a nossa aplicação. Em “Name” você define o nome do AVD, vamos chamá-lo de “Emulador”.
- Em “Target” definirmos a plataforma-alvo a ser executada, neste caso “Android 2.3.3 - API Level 10” Vamos selecioná-la.



# O que é o **Android**



# Android

- É um sistema operacional mobile baseado em Linux, originalmente desenvolvido por uma empresa de mesmo nome, comprada pela Google em 2005.
- É fundamentado em código open source, sob licença Apache. Por isso, admite que fabricantes de hardware insiram códigos proprietário para diferenciar seus produtos.

# Android

- Android é em geral mas não completamente Java.
- API escrita baseada em Java para maquina virtual Dalvik.
- Arquivos .class viram .dex e o pacote é .apk



# Android

- A plataforma Androi é mais do que um S.O., é todo um ecossistema que envolve métricas de compatibilidade e padrões para fabricantes, kit para o desenvolvedor (SDK), middleware e plugins para ambientes integrados de desenvolvimentos (IDEs).

# Android

- O Android conta com o Android Market, rebatizado de Google Play Store, uma central de vendas e distribuição de aplicativos voltados para a plataforma.
- <https://play.google.com>

# Com o que posso contar no Android



# Android

- Ampla API e acesso a serviços nativos como chamadas telefônicas, lista de contatos, SMS, GSM etc.
- Interface multitouch adequada a telas de resoluções e densidades diferentes (Views e Fragments)
- Extenso suporte a mídias como img, gráficos 2D e 3D (OpenGL), audio, vídeo e live streaming

# Android

- Incrementos de experiências de usuário no ambiente operacional com widgets e live wallpapers
- Aplicações multitarefas, visíveis ou em background e serviços de notificações.
- Acesso a hardwares como câmeras (frontal e traseira), vibrações, flashes, etc.
- Persistências em bancos de dados SQLite e Content Providers



# Android

- Compartilhamento de dados inter-aplicação como intenções (Intents)
- Suporte a comandos remotos para o aparelho – Cloud To Device Manager (C2DM)
- Mapas e serviços de geolocalização utilizando GPS

# Suporte a sensores:

- Iluminação
- Proximidade
- Pressão
- Temperatura ambiente
- Acelerômetro
- Giroscópio
- Transferência de dados e detecção de redes GSM, 3G, 4G, Wi-Fi, Bluetooth entre outras.

# Quem está por trás do Android?

- Open Handset Alliance

<http://www.openhandsetalliance.com/>

Grupo de 84 Empresas: Google, Samsung, Asus, Dell, Intel, nvidia etc.

# Mercado Mobile no Mundo

- Para cada nascimento, quase 2 aparelhos Android são vendidos.
- No segundo trimestre de 2013, 79% dos dispositivos móveis vendidos foram Android.
- 31% só da Samsung, mais do que o dobro dos iPhones.

Fontes:

[www.Lukew.com/ff/entry.asp?1506](http://www.Lukew.com/ff/entry.asp?1506)

[www.gartner.com/newsroom/id/2573415](http://www.gartner.com/newsroom/id/2573415)

# Mercado Mobile no Brasil

- Em 2012, o número de celulares superou o de brasileiros
- Somos o 4º maior mercado mobile do mundo
- 84% dos brasileiros (16+) possuem celular
- 48% possuem mais de um aparelho
- 10% das vendas online vem de dispositivos móveis

Fontes: <http://www.mobilizado.com.br/mercado/infografico-mostra-importancia-do-mercado-mobile-no-brasil>



# Mobile



- Smartphones
- Tablets
- Wearable devices
- Internet das coisas (IoT)



# Para Qual versão do Android devo desenvolver?

## ANDROID



# Android 1.0

- HTC Dream (ou T-Mobile G1)
- API nível 1
- 23 de setembro de 2008
- Suporte a câmera
- Apps do Google (calendar, maps, youtube, search, talk, sysnc)
- Suporte WI-FI e bluetooth
- Webkit browser



# Android 1.1 Petit-Four

- API nível 2
- 09 de fevereiro de 2009
- Linux Kernel 2.6
- Apenas pequenas melhorias nas funcionalidades anteriores.
- Embora esse não tenha “pego”. Iniciou a brincadeira de chamar cada versão por um nome de sobremesa.

-



# Android 1.5 Cupcake

- API nível 3
- 30 de abril de 2009
- Linux Kernel 2.6.27
- Considerado o primeiro marco comercial do Android
- Grava e reproduz vídeo
- Suporte a teclados virtuais de terceiros com previsão de texto



# Android 1.6 Donut

- API nível 4
- 15 de setembro de 2009
- Linux Kernel 2.6.27
- Passou a suportar telas WVGA (800x480)
- Melhoria nos aplicativos no Android Market

# Android 2.0, 2.0.1, 2.1 Eclair

- API nível 5, 6, 7
- 26/10/2009, 03/12/2009, 12/01/2010
- Linux Kernel 2.6.29
- Primeira versão a entrar forte no Brasil (por meio da Vivo)
- Diversas melhorais e novas funcionalidades. Live Wallpapers

# Android 2.2 Froyo (Frozen yogurt)

- API nível 8
- 20/05/2010
- Linux Kernel 2.6.32
- USB e WI-FI hotspot
- Suporte a Flash e browser com o Chrome V8 (Javascript)
- Atualizações automáticas e instalações de apps no SD


<http://www.youtube.com/watch?v=yAZYSVr2Bhc>

# Android 2.2 Froyo (Frozen yogurt)

- API nível 8
- 20/05/2010
- Linux Kernel 2.6.32
- USB e WI-FI hotspot
- Suporte a Flash e browser com o Chrome V8 (Javascript)
- Atualizações automáticas e instalações de apps no SD

<http://www.youtube.com/watch?v=yAZYSVr2Bhc>

# Android 2.3 - 2.3.7 Gingerbread


- API nível 9 e 10
- 20/05/2010
- Linux Kernel 2.6.35 
- Garbage Collector concorrente
- Múltiplas câmaras
- Novos sensores giroscópio e barômetro




# Android 3.0 – 3.2 Honeycomb

- API nível 11, 12 e 13
- 22/02/2011
- Linux Kernel 2.6.36
- Primeira versão feita para tablets. Não usada em smartphones
- Nova funcionalidade barra de ações (Action Bar)
- Diversas melhorias de API

# Android 4.0 – 4.0.4 Icecream Sandwich

- API nível 14 e 15
- 19 de outubro de 2011
- Linux Kernel 3.0.1 
- Reagrupou os esforços em uma única versão para tablets smartphones
- Possível instalar o próprio Google Chrome como browser
- Outras melhorias

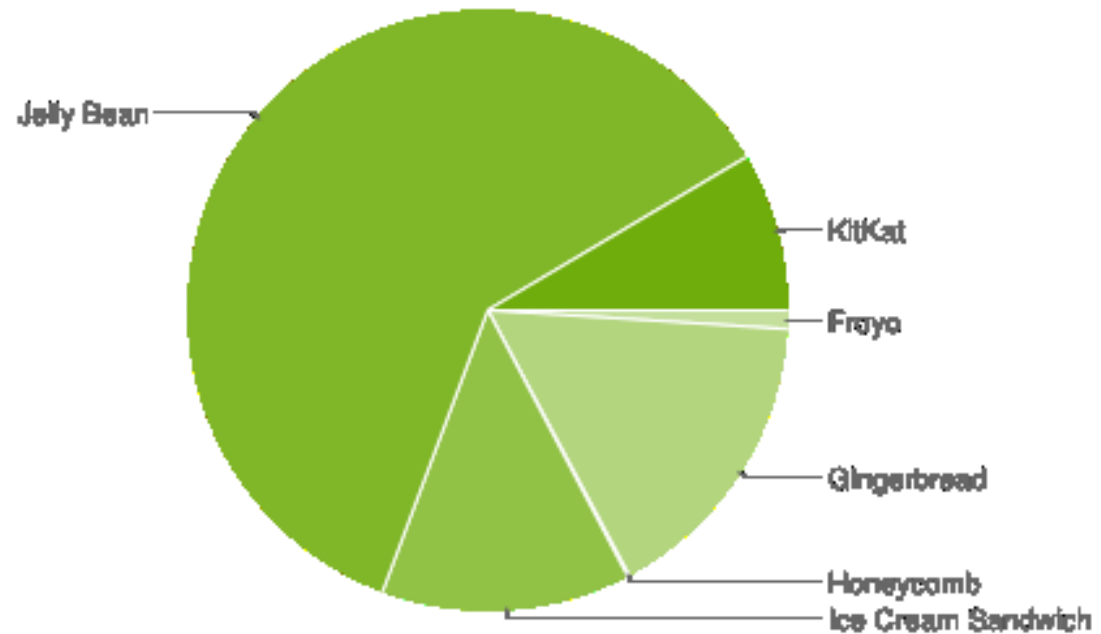
# Android 4.1, 4.2, 4.3 Jelly Bean

- API nível 16, 17 e 18
- 09/07/2012, 13/11/2012, 24/07/ 2013
- Linux Kernel 3.0.31 
- Trouxe o Chrome com navegador padrão, e não mais suporta Flash devido ao abandono da Adobe

# Android 4.4 KitKat

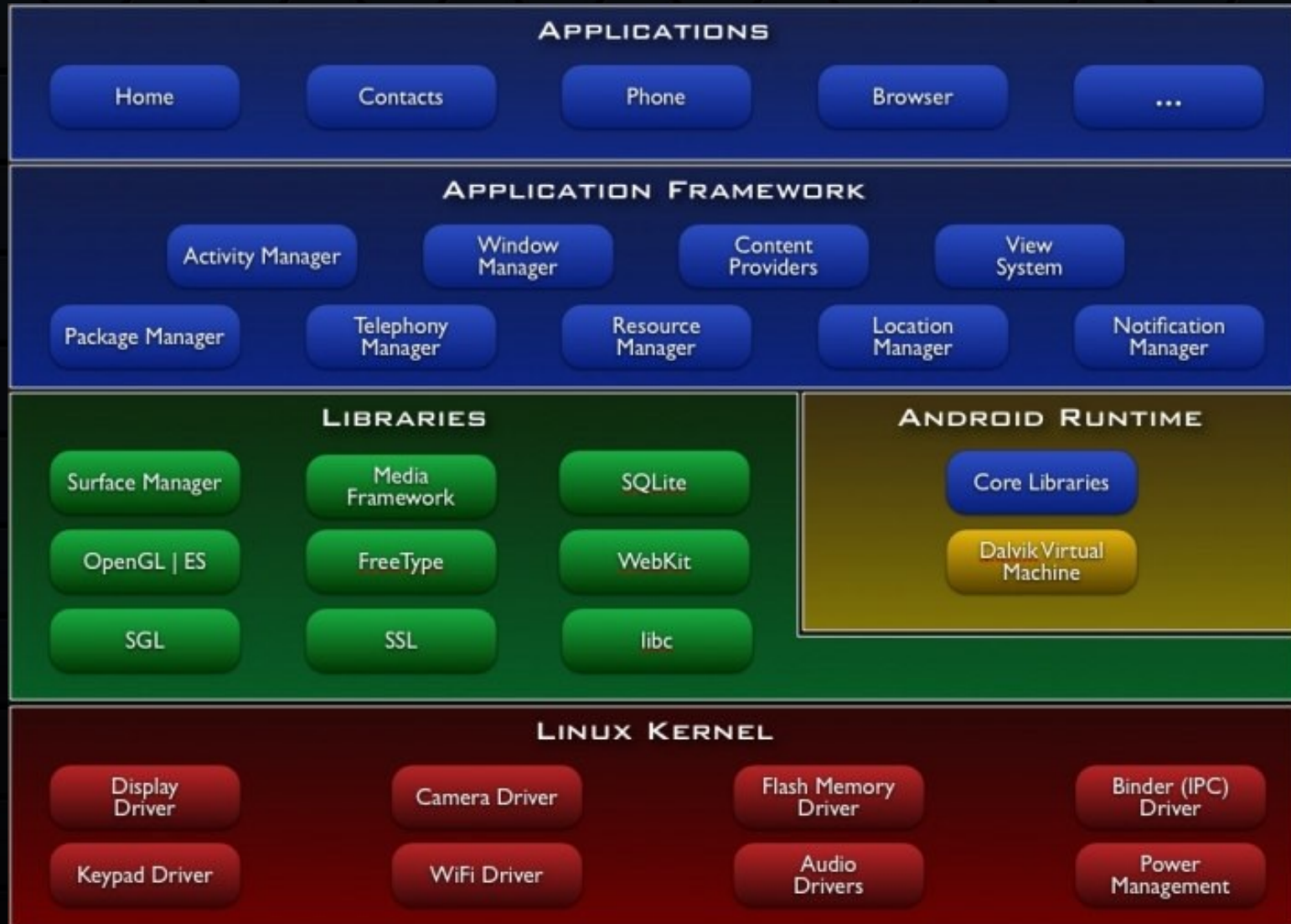
- API nível 19
- 03 de setembro de 2013
- Linux Kernel 3.8
- Apesar de já haver planos para o 4.5 (ainda kitkat), provavelmente, esta será última versão antes do Android 5, Key Lime Pie

Version	Codename	API	Distribution
<a href="#">2.2</a>	Froyo	8	1.0%
<a href="#">2.3.3 - 2.3.7</a>	Gingerbread	10	16.0%
<a href="#">3.2</a>	Honeycomb	13	0.1%
<a href="#">4.0.3 - 4.0.4</a>	Ice Cream Sandwich	15	13.0%
<a href="#">4.1.x</a>	Jelly Bean	16	33.0%
<a href="#">4.2.x</a>		17	18.0%
<a href="#">4.3</a>		18	8.5%
<a href="#">4.4</a>	KitKat	19	8.5%





# A arquitetura do Android



# Tipos de aplicativos

- Foreground
  - Aplicativos úteis apenas enquanto estão visíveis
  - Muitos games se enquadram nessa categoria
  - O mais importante aqui é o pleno entendimento do ciclo de vida das Activities

# Tipos de aplicativos

- Background
  - Aplicativos menos comuns com um tempo proporcional de interação com o usuário muito baixo
  - Exemplos: chamada de voz, sms, relógios, alarmes
  - Importante conhecer bem Services e Broadcast Receivers

# Tipos de aplicativos

- Intermittent
  - Misto dos dois primeiros tipos
  - A maioria dos aplicativos bem escritos precisa reagir a inputs do usuário e a eventos em background
  - Interessante conhecer também Notifications

# Tipos de aplicativos

- Widgets e Live Wallpapers
  - Esses tipos de aplicativos invadem a home screen do usuário proporcionando uma experiência mais agradável e maior praticidade no acesso
  - Há uma API específica para esses tipos de componentes



# Noções sobre aplicações Android

Por padrão, **cada aplicação é executada como um processo separado, com ID único e máquina virtual própria**, isolando o seu código das demais aplicações.

# Componentes da Aplicação

Um recurso fundamental do Android é o **reuso de componentes**. Caso uma aplicação precise disponibilizar uma lista de imagens com *scrolling* e outra aplicação apresenta tal componente e o disponibilizou para as demais, pode-se invocar esse componente.

# Componentes da Aplicação

Devido a essa organização dos aplicativos em componentes, as aplicações Android não possuem um único ponto de entrada (**não há um método `main()`**, por exemplo). Ao invés disso, as aplicações apresentam componentes essenciais que o sistema pode instanciar e executar quando necessário. **Esses componentes podem ser de quatro tipos:**

# Componentes da Aplicação

- **Activities** (apresenta uma interface visual para o usuário)
- **Services** (roda em segundo plano por um período de tempo indeterminado)
- **Broadcast Receivers** (recebe e reage a eventos do sistema)
- **Content providers** (dados do aplicativo disponíveis para os demais)

# Ativando e desativando componentes

**Content Providers** são ativados por meio de uma requisição de um **ContentResolver**. Os demais componentes (Activities, Services e Broadcast Receivers) são ativados por mensagens assíncronas denominadas **intents**. Trata-se de um objeto da classe Intent que armazena o conteúdo da mensagem.



# Ativando e desativando componentes

Para **activities** e **services**, o objeto apresenta o nome da ação que está sendo requisitada bem como o endereço do dado em que atuará, além de outras coisas.

No caso de uma **Activity**, por exemplo, pode conter uma requisição para apresentar uma imagem ao usuário ou permitir que o usuário edite algum texto. Já para **Broadcast Receivers**, o objeto Intent pode anunciar que um botão da câmera foi pressionado, por exemplo.

# Ativando e desativando componentes

**Content Provider** e **BroadCast Receiver** não precisam ser desativados, uma vez que permanecem ativos somente enquanto estão respondendo requisições.

Já **Activities** e **Services** podem permanecer em execução por um longo período de tempo, daí a necessidade de finalizá-los por meio dos métodos **finish()** e **stopSelf()**, espectivamente.

# Activities

O componente mais comum de uma aplicação é um **Activity**.

É implementado como uma **subclasse de Activity** e uma aplicação pode conter uma ou mais *activities*.

# Activities

Cada qual **representa uma interface visual** e uma delas é marcada como sendo a inicial que deve ser apresentada quando a aplicação é iniciada.

Mover-se de uma **Activity** para outra consiste em fazer com que a **Activity atual invoque a próxima**.

# Activities

Cada **Activity** possui uma janela padrão para desenhar. Normalmente a janela ocupa a tela toda mas também pode ser menor e flutuar sobre outras janelas.

Uma **Activity** pode conter janelas adicionais, como por exemplo um **dialog** que exige uma resposta do usuário ou mostra um aviso quando um dado item é selecionado.



# Activities

O visual da janela é composto por uma **hierarquia de views**, objetos derivados da classe base **View**.

As **views** também são responsáveis por responder às **ações** do usuário direcionadas ao **seu espaço**.

# Activities e Tasks

Conforme dito anteriormente, uma **Activity** pode **iniciar** outra, inclusive uma que pertença a **outro aplicativo**.

Por exemplo: uma aplicação deseja mostrar o mapa de algum local. Já existe uma **Activity** que o faz, então tudo o que a sua **Activity** precisa fazer é chamar o método **startActivity()** passando como **parâmetro** o objeto da **classe Intent** com as informações necessárias. O mapa será mostrado e, quando o usuário pressionar **a tecla voltar**, sua **Activity** será mostrada novamente na **tela**.

# Activities e Tasks

Para o usuário, isso é transparente. O Android mantém ambas as *activities* na mesma **Task**.

Uma **Task**, para o usuário, é como se fosse uma *aplicação*.

Tecnicamente é um **grupo de Activities relacionadas que foram adicionadas em uma pilha**.

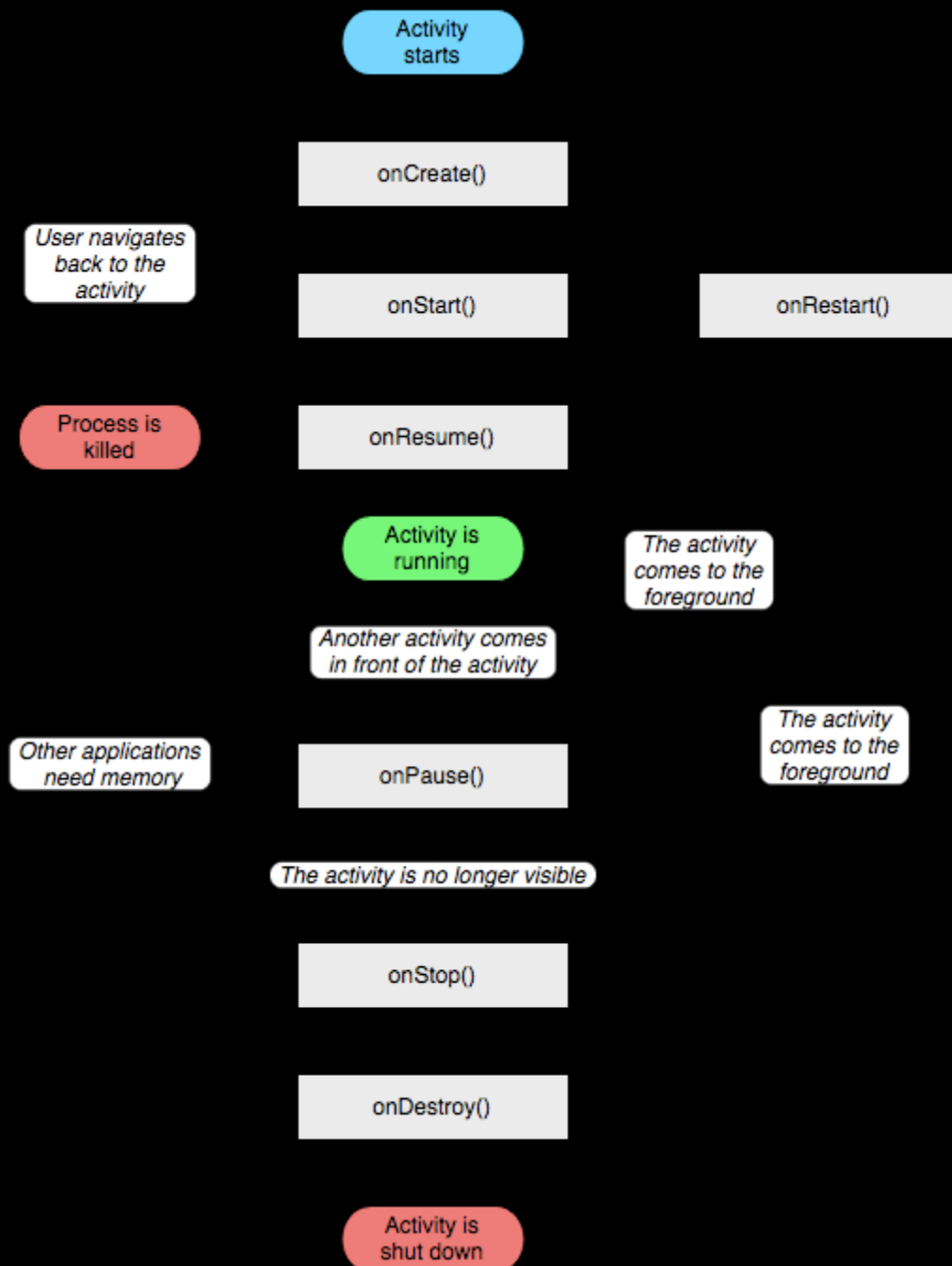
# Activities e Tasks

A **base** da pilha é a **primeira Activity** que é mostrada quando a **Task é iniciada**, enquanto que o **topo** é a **Activity** que está sendo executada no momento, ou seja, que está recebendo o foco das ações do usuário.

# Ciclo de vida de uma Activity

- Basicamente, uma **Activity** possui **três estados**:
  - **Ativo**: quando a Activity está no topo da pilha (visível na tela e recebendo as ações do usuário);
  - **Pausado**: quando a Activity perdeu o foco mas permanece visível ao usuário. Ou seja, há outra Activity no topo que é transparente ou não ocupa toda a tela;
  - **Parado**: quando a Activity está ofuscada por outra. Mantém o estado mas pode ser eliminada pelo sistema em caso de falta de memória.





# Ciclo de vida de uma Activity

Method	Description	Killable?	Next
<a href="#"><code>onCreate()</code></a>	Called when the activity is first created. This is where you should do all of your normal static set up — create views, bind data to lists, and so on. This method is passed a <code>Bundle</code> object containing the activity's previous state, if that state was captured (see <a href="#">Saving Activity State</a> , later). Always followed by <code>onStart()</code> .	No	<code>onStart()</code>
<a href="#"><code>onRestart()</code></a>	Called after the activity has been stopped, just prior to it being started again. Always followed by <code>onStart()</code> .	No	<code>onStart()</code>
<a href="#"><code>onStart()</code></a>	Called just before the activity becomes visible to the user. Followed by <code>onResume()</code> if the activity comes to the foreground, or <code>onStop()</code> if it becomes hidden.	No	<code>onResume()</code> or <code>onStop()</code>
<a href="#"><code>onResume()</code></a>	Called just before the activity starts interacting with the user. At this point the activity is at the top of the activity stack, with user input going to it. Always followed by <code>onPause()</code> .	No	<code>onPause()</code>
<a href="#"><code>onPause()</code></a>	Called when the system is about to start resuming another activity. This method is typically used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, and so on. It should do whatever it does very quickly, because the next activity will not be resumed until it returns. Followed either by <code>onResume()</code> if the activity returns back to the front, or by <code>onStop()</code> if it becomes invisible to the user.	Yes	<code>onResume()</code> or <code>onStop()</code>
<a href="#"><code>onStop()</code></a>	Called when the activity is no longer visible to the user. This may happen because it is being destroyed, or because another activity (either an existing one or a new one) has been resumed and is covering it. Followed either by <code>onRestart()</code> if the activity is coming back to interact with the user, or by <code>onDestroy()</code> if this activity is going away.	Yes	<code>onRestart()</code> or <code>onDestroy()</code>
<a href="#"><code>onDestroy()</code></a>	Called before the activity is destroyed. This is the final call that the activity will receive. It could be called either because the activity is finishing (someone called <code>finish()</code> on it), or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the <code>isFinishing()</code> method.	Yes	nothing

# Services

Diferentemente de **activities**, os **services** não possuem interface e executam em segundo plano por um período de tempo indeterminado.

Cada serviço é uma classe que herda de **Service**.

# Services

Um exemplo clássico de **Service** é um **tocador de músicas**.

A aplicação deve consistir de uma ou mais *activities* que permitem ao usuário selecionar as músicas e começar a tocá-las.

Contudo, a execução das músicas em si não faz parte da Activity mas sim de um Service, uma vez que o usuário espera que a música continue a ser tocada após sair da tela.

# Ciclo de vida de um Service

Service is  
started by  
`startService()`

`onCreate()`

`onStart()`

Service is  
running

*The service  
is stopped  
(no callback)*

`onDestroy()`

Service is  
shut down

Service is  
created by  
`bindService()`

`onCreate()`

`onBind()`

*Client interacts with the service*

`onRebind()`

`onUnbind()`

`onDestroy()`

Service is  
shut down



# Broadcast Receivers

Componente que recebe e reage a anúncios de broadcast, geralmente oriundos do sistema.

Cada *receiver* é uma classe que herda de **BroadcastReceiver**.

# Broadcast Receivers

Mudança no fuso horário, anúncio de bateria fraca e mudança da linguagem por parte do usuário são exemplos de anúncios que podem ser **capturados por Broadcast Receivers**.

Uma aplicação pode conter quantos receptores quiser. Os receptores podem iniciar uma **Activity** ou utilizar o **NotificationManager** para alertar o usuário (acender a luz do aparelho, vibrar, executar um som, etc).

# Ciclo de vida de Broadcast Receivers



Quando uma mensagem de broadcast chega ao receptor, o Android invoca o método **onReceive()**, passando como parâmetro um objeto **Intent** contendo a mensagem.

```
void onReceive(Context curContext, Intent broadcastMsg)
```

O **Broadcast Receiver** fica ativo apenas enquanto está executando esse método.

# Content Providers

Componente que torna um conjunto específico de dados da aplicação disponível para outras aplicações.

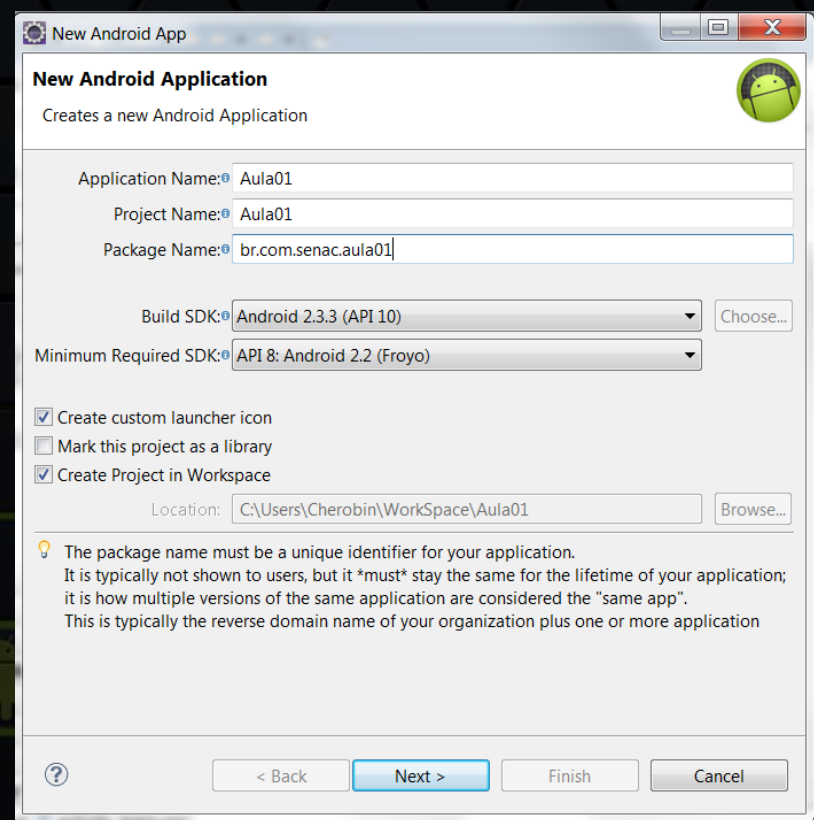
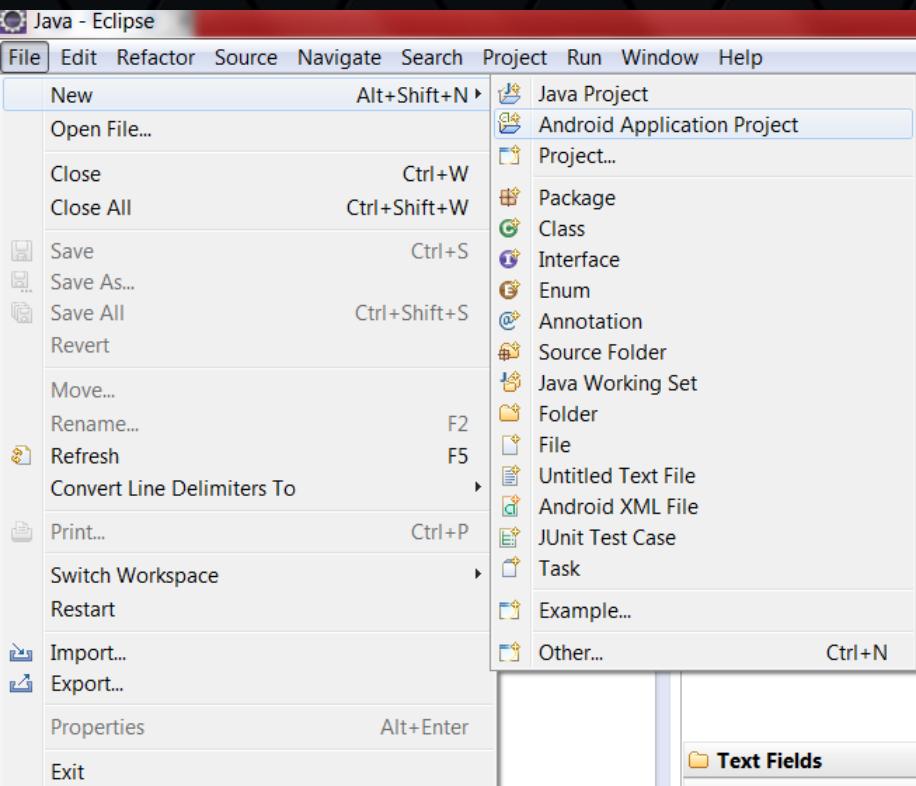
Cada *provider* é uma classe que herda de **ContentProvider** e disponibiliza um conjunto padrão de métodos para que outras aplicações possam recuperar e armazenar dados do tipo que o provedor controla.

# Content Providers

As aplicações não podem acessar os métodos de um **Content Provider** diretamente.

Para isso, elas precisam de um objeto **ContentResolver**, o qual pode conversar com qualquer **Content Provider**.





# Android Application Project

Bom, agora irei descrever a estrutura de um projeto Android.

Observem que dentro da pasta “Aula01” existe uma pasta chamada “src” e dentro dela é que ficam os códigos fonte java das aplicações.

Observem que o arquivo “MainActivity.java” se encontra dentro do pacote “br.com.xxx.aula01” (Esse pacote também é uma pasta). Esse arquivo é a nossa aplicação Android. Vou descrever em detalhes o arquivo “MainActivity.java” (Veja o código abaixo):

```
package br.com.senac.aula01;

import android.os.Bundle;

public class MainActivity extends Activity {

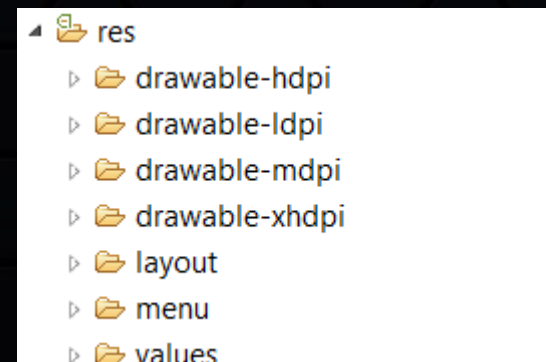
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

Diferentemente das aplicações comuns de Java, toda classe para aplicação Android deve ser derivada da classe Activity (Atividade) e possui como método principal, o método onCreate. Dentro desse método ele invoca o método onCreate da super classe passando mesmo parâmetro (o savedInstanceState), logo após esse método, vem o método setContentView, responsável por exibir a tela da minha aplicação, baseado nos layouts xml. Por padrão ele chama o arquivo "main.xml".

# Android Application Project

- Dentro da pasta “Aula01” existe um diretório chamado “res”, onde ficam armazenados todos os recursos utilizados pela aplicação. Dentro do diretório “res” existem cinco diretórios, cada um deles com uma finalidade, que descreverei agora:
- Os diretórios “drawable”



# Android Application Project

Cada um desses diretórios só será utilizado de acordo com a resolução do Android que você está utilizando, ou seja, qual modelo de emulador de você estiver usando.

Por exemplo, quando você usa uma resolução de 480x800 no seu emulador, é utilizado o diretório “drawable-hdpi” para buscar a imagem que vai representar o ícone da sua aplicação Android. Se você for usar uma resolução 320x480 (que é a resolução padrão do emulador Android), é utilizado o diretório “drawable-mdpi”



# Android Application Project

O diretório “layout” armazena todas os layouts da aplicação Android, que normalmente são arquivos “.xml”.

Para quem conhece a combinação HTML + JavaScript, o Android é similar, é a combinação de XML + Java, logo todos os nosso componentes vão ser adicionados usando tags XML. Por padrão, o arquivo de layout é o main.xml.

MainActivity.java activity\_main.xml

default Nexus One AppTheme MainAct

Palette

Form Widgets

TextView Large Medium Small Button

Small OFF CheckBox

RadioButton CheckedTextView

Spinner

QuickContactBadge

Custom & Library Views

Text Fields

Layouts

Composite

Images & Media

Time & Date

Transitions

Advanced

Graphical Layout activity\_main.xml

Aula01

Hello world!

# Android Application Project

Para visualizarmos o código do arquivo main.xml, simplesmente clique na guia “main.xml”, que se encontra abaixo da seção “Views”, como demonstra a figura:



```
1 <RelativeLayout xmlns:android="http://schemas.android.com/
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent" >
5
6     <TextView
7         android:layout_width="wrap_content"
8         android:layout_height="wrap_content"
9         android:layout_centerHorizontal="true"
10        android:layout_centerVertical="true"
11        android:text="@string/hello_world"
12        tools:context=".MainActivity" />
13
14 </RelativeLayout>
15
```

Graphical Layout activity\_main.xml

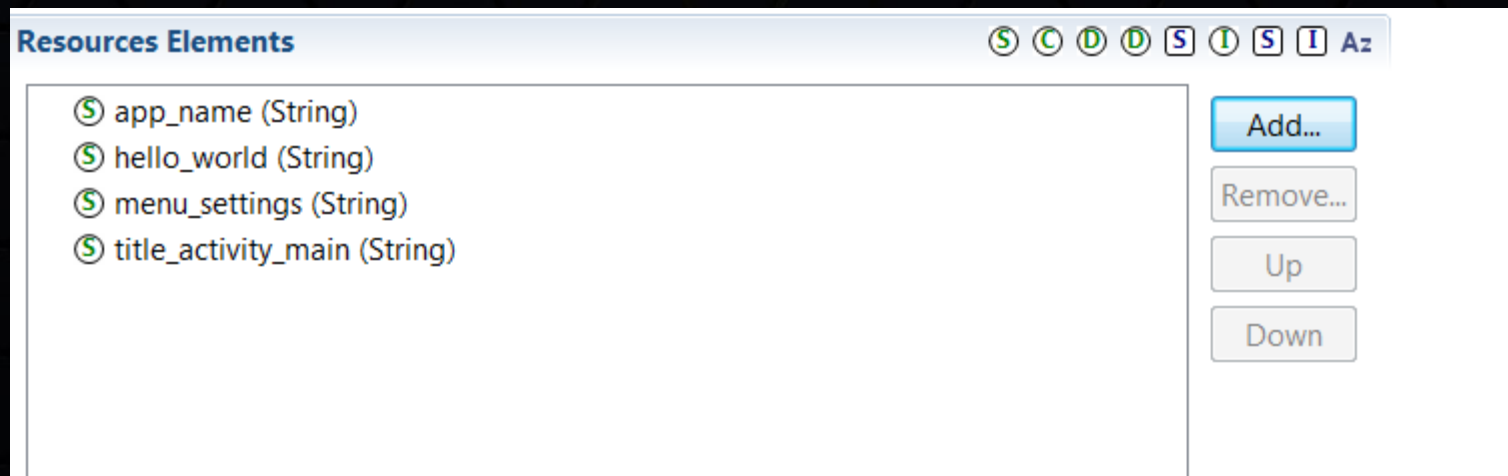
# Android Application Project

Observe que após a primeira linha (prólogo xml), existe uma tag chamada `RelativeLayout`, responsável por organizar os componentes exibidos na tela, por padrão os componentes são distribuídos na vertical pelo atributo `android:orientation="vertical"`.

Dentro desta tag, existe um componente chamado `TextView`, que representa um texto a ser exibido na tela, por padrão, ele irá exibir “Hello World” através do atributo `android:text="@string/hello_world"`, onde o valor `“@string/hello_world”` equivale a uma constante, que está definida no arquivo `strings.xml`, que se encontra no diretório “values”, que iremos ver agora.

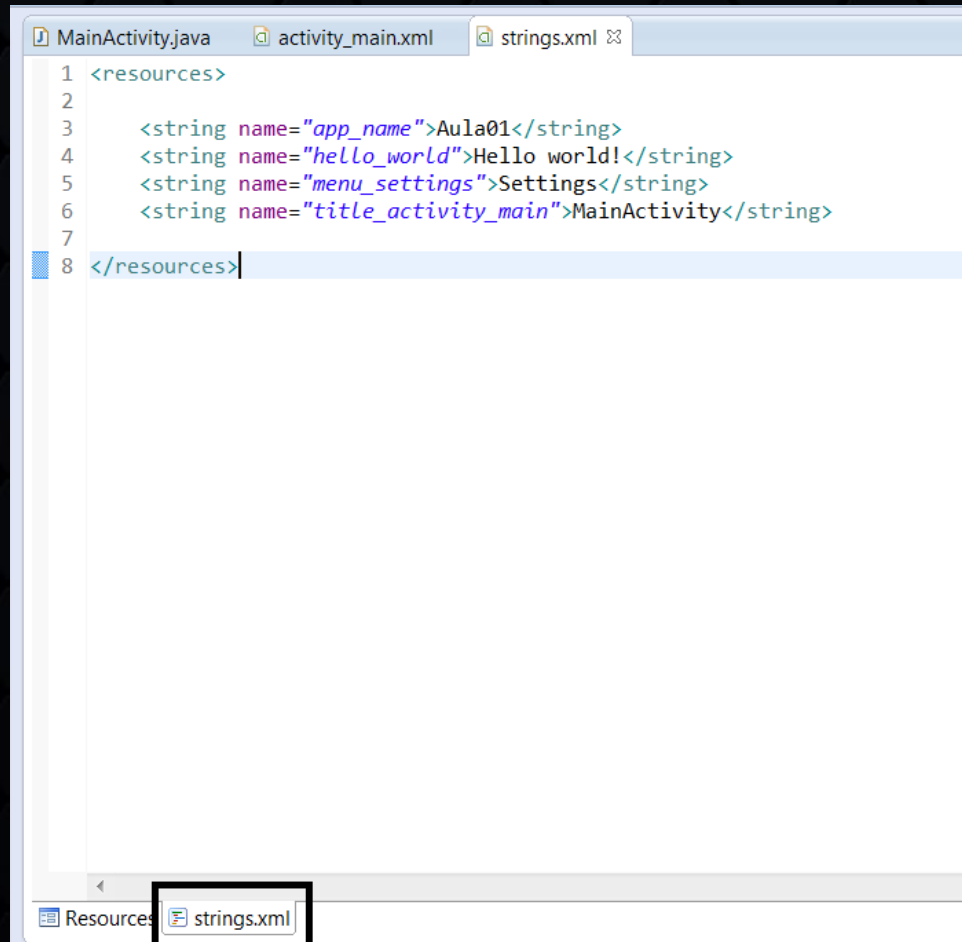
# Android Application Project

O diretório “values” armazena valores estáticos que podem ser utilizados por um arquivo “.XML”. Normalmente esses valores estáticos devem ser armazenados no arquivo “strings.xml”. Vá no diretório “res/values” e de um duplo clique no arquivo “strings.xml”, e será mostra o gerenciador desse arquivo.





# Android Application Project



```
1 <resources>
2
3     <string name="app_name">Aula01</string>
4     <string name="hello_world">Hello world!</string>
5     <string name="menu_settings">Settings</string>
6     <string name="title_activity_main">MainActivity</string>
7
8 </resources>
```

# Android Application Project

Observe que nas propriedades do atributo “hello\_world”, está atribuído um valor a ela, que é o valor “Hello World!”, isso quer dizer que lá no arquivo XML, no componente TextView, tem uma propriedade chama “android:text”, com o valor “@string/hello\_world”, que equivale a na verdade a string “Hello World!”. Para ver a sua estrutura, clique na guia “strings.xml”.

# Android Application Project

Observem que dentro desse arquivo eu declaro um valor estático chamado `app_name`, cujo valor é “Aula01”.

Dentro da pasta Aula01 existe um arquivo chamado “AndroidManifest.xml” Esse arquivo é o sistema nervoso de uma aplicação Android. É nele que ficam as definições referentes à aplicação. De um duplo clique nesse arquivo para abri-lo, feito isso será mostrado o seu gerenciador.

# Android Application Project

MainActivity.java activity\_main.xml strings.xml Aula01 Manifest

## Android Manifest

### Manifest General Attributes

Defines general information about the AndroidManifest.xml

**Package** br.com. :aula01 Browse...

**Version code** 1

**Version name** 1.0 Browse...

**Shared user id** Browse...

**Shared user label** Browse...

**Install location** ▼

### Manifest Extras

U S P U C U O Az

U Uses Sdk

Add...

Remove...

Up

Down

### Exporting

To export the application for distribution, you have the following options:

Manifest A Application P Permissions I Instrumentation AndroidManifest.xml

# Android Application Project

Bom, o que nos interessa aqui é o código. Para visualizarmos seu código, clique na seção “AndroidManifest.xml”.

```
MainActivity.java  activity_main.xml  strings.xml  Aula01 Manifest
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2   package="br.com.senac.aula01"
3   android:versionCode="1"
4   android:versionName="1.0" >
5
6   <uses-sdk
7       android:minSdkVersion="8"
8       android:targetSdkVersion="15" />
9
10  <application
11     android:icon="@drawable/ic_launcher"
12     android:label="@string/app_name"
13     android:theme="@style/AppTheme" >
14    <activity
15        android:name=".MainActivity"
16        android:label="@string/title_activity_main" >
17      <intent-filter>
18        <action android:name="android.intent.action.MAIN" />
19
20        <category android:name="android.intent.category.LAUNCHER" />
21      </intent-filter>
22    </activity>
23  </application>
24
25 </manifest>
```



# Android Application Project

Observem algumas tags interessantes. A tag `<application>` possui o atributo `android:icon`, no qual especifico o ícone da aplicação. Como havia citado anteriormente, todas as imagens ficam no diretório `drawable` e nesse diretório existe um arquivo de chamado `"ic_launcher.png"` que será o ícone da minha aplicação. Logo, para usar esse ícone neste atributo, deve-se passar o valor

[@drawable/ic\\_launcher](#). Observem que quando informamos o ícone, ele deve ser informado sem a extensão (nesse caso, PNG).

# Android Application Project

Observem agora a tag `<activity>`, ela define uma atividade (Activity).. Dentro desta tag, eu possuo o atributo chamado `android:label` que define o título da minha aplicação.

O título que será exibido é o valor que está armazenado no valor estático `app_name`. Isso é obtido pela atribuição `android:label="@string/app_name"`.

# Android Application Project

```
MainActivity.java  activity_main.xml  strings.xml  Aula01 Manifest  ⌵
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2   package="br.com.senac.aula01"
3   android:versionCode="1"
4   android:versionName="1.0" >
5
6   <uses-sdk
7     android:minSdkVersion="8"
8     android:targetSdkVersion="15" />
9
10  <application
11    android:icon="@drawable/ic_launcher"
12    android:label="@string/app_name"
13    android:theme="@style/AppTheme" >
14    <activity
15      android:name=".MainActivity"
16      android:label="@string/title_activity_main" >
17      <intent-filter>
18        <action android:name="android.intent.action.MAIN" />
19
20        <category android:name="android.intent.category.LAUNCHER" />
21      </intent-filter>
22    </activity>
23  </application>
24
25 </manifest>
```

# Android Application Project

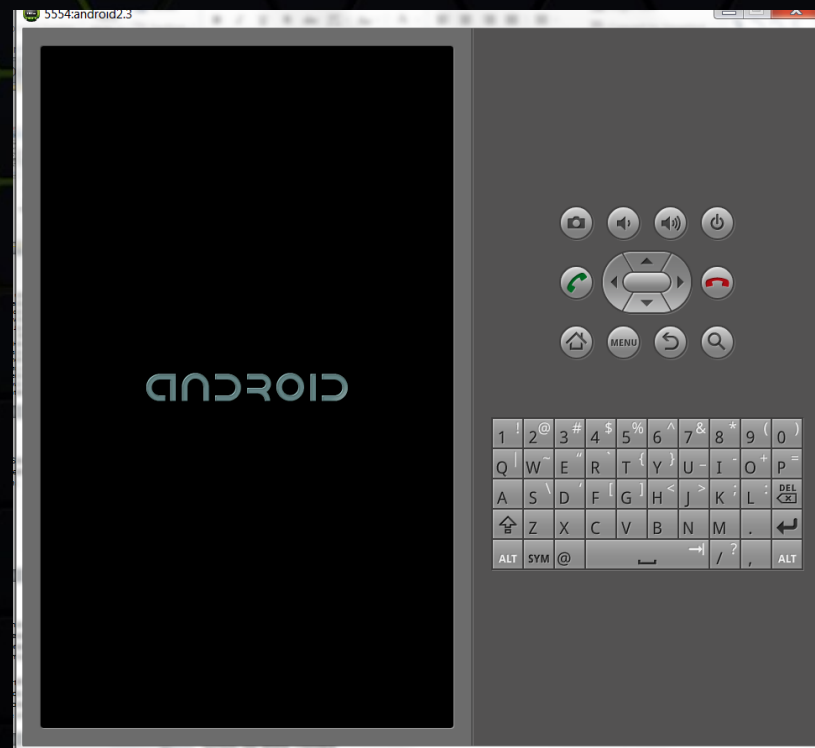
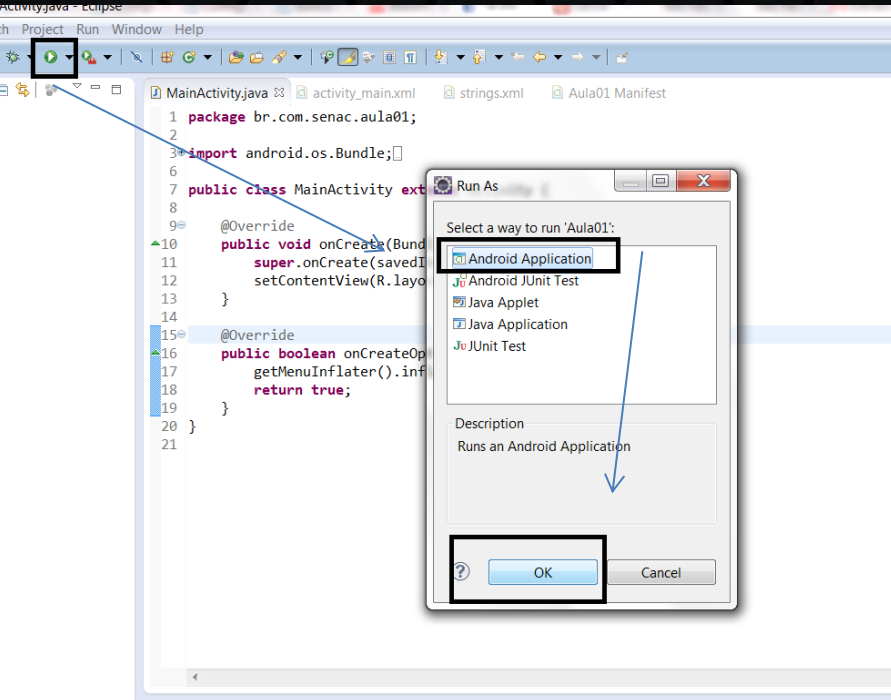
Como havia falado a aplicação Android nada mais é do que a combinação Java + XML.

Agora, como um código Java vai acessar um componente que está escrito em XML ? Ah, essa é a finalidade do arquivo R.java (que fica dentro do pacote “gen” , situado no projeto), ele funciona como uma “interface” entre o código Java e o código XML, logo, se eu quiser manipular em tempo de execução um componente via Java, tenho que fazer interface com esse arquivo.

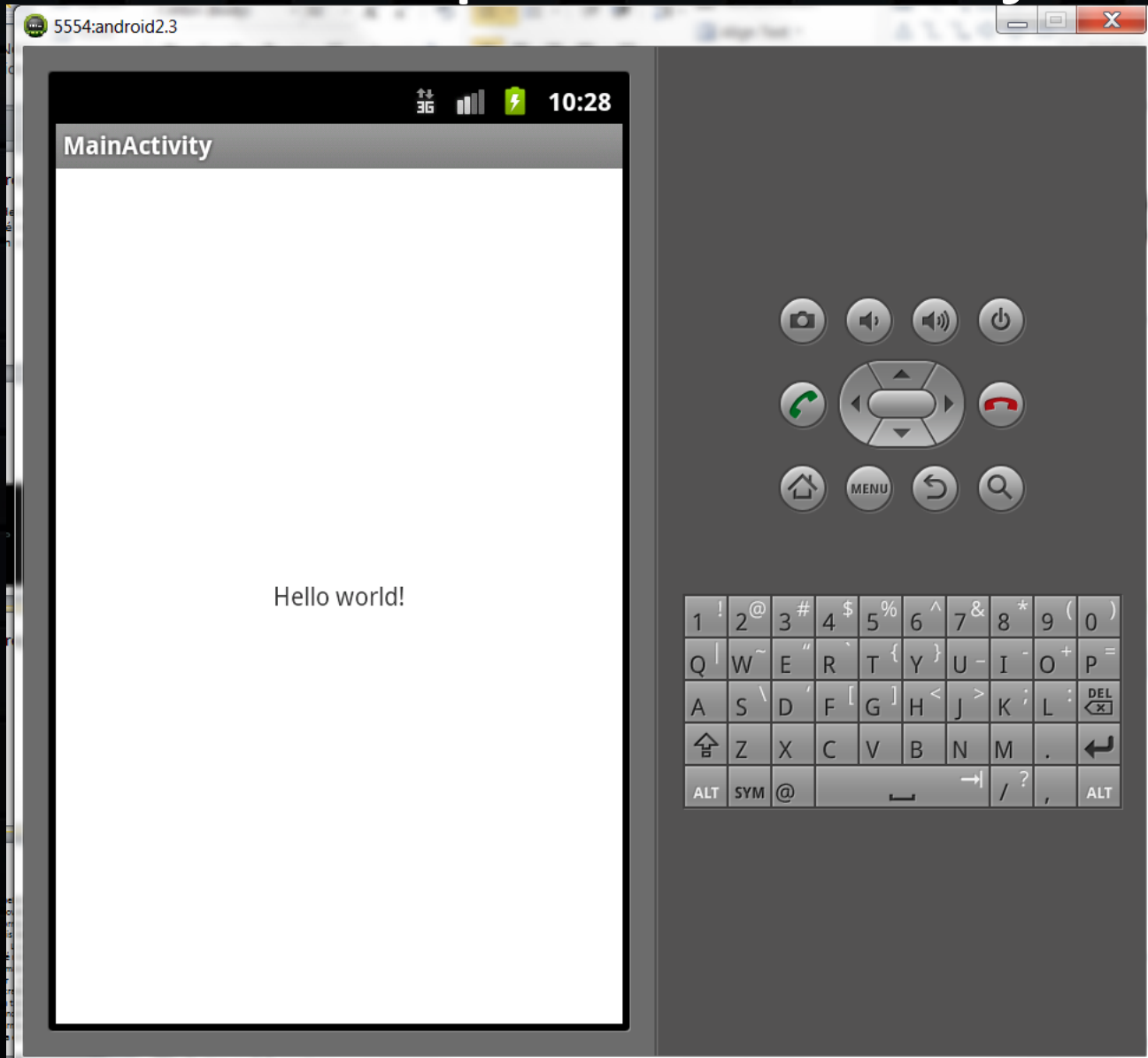
# Android Application Project

- **OBS:** O arquivo R.java não pode ser modificado **manualmente**. Ele é modificado automaticamente de acordo com as mudanças feitas no projeto.





# Android Application Project



# Android Application Project

- Esse emulador já vem com uma série de recursos como Navegador, Aplicações de demonstração, Mapas, Lista de contatos e etc.
- Se você neste exato momento fechou o emulador após a execução da aplicação, vou te dizer uma coisa: “**Não era para você ter feito isso**”.

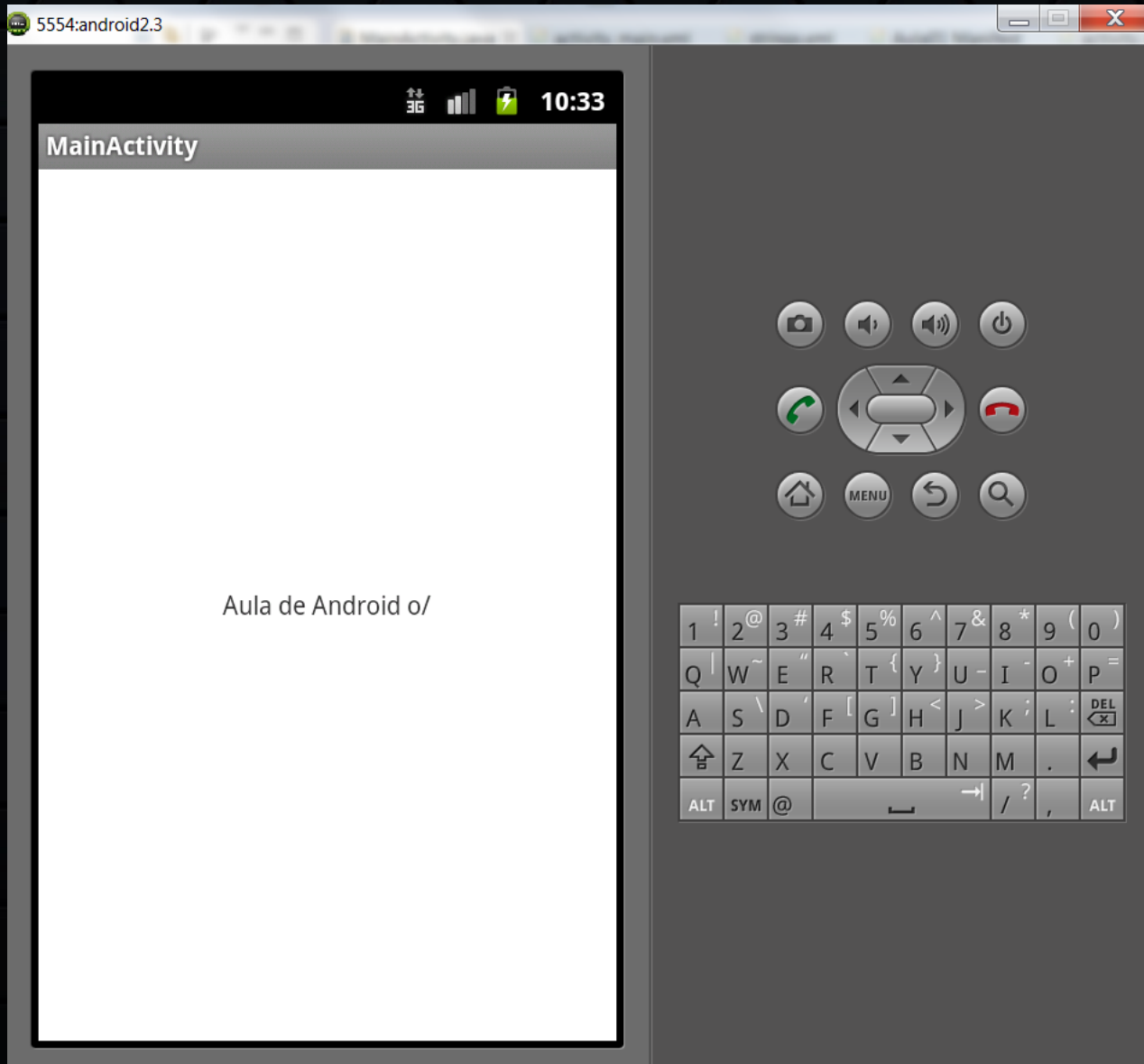
# Android Application Project

Vamos modificar essa aplicação. Minimize o emulador e vamos abrir o arquivo “activity\_main.xml”.

Na tag TextView que já havia explicado a vocês, possui um atributo chamado android:text, onde nele defino o título que será exibido, modifique agora essa propriedade com o seguinte valor (título), conforme o código abaixo:

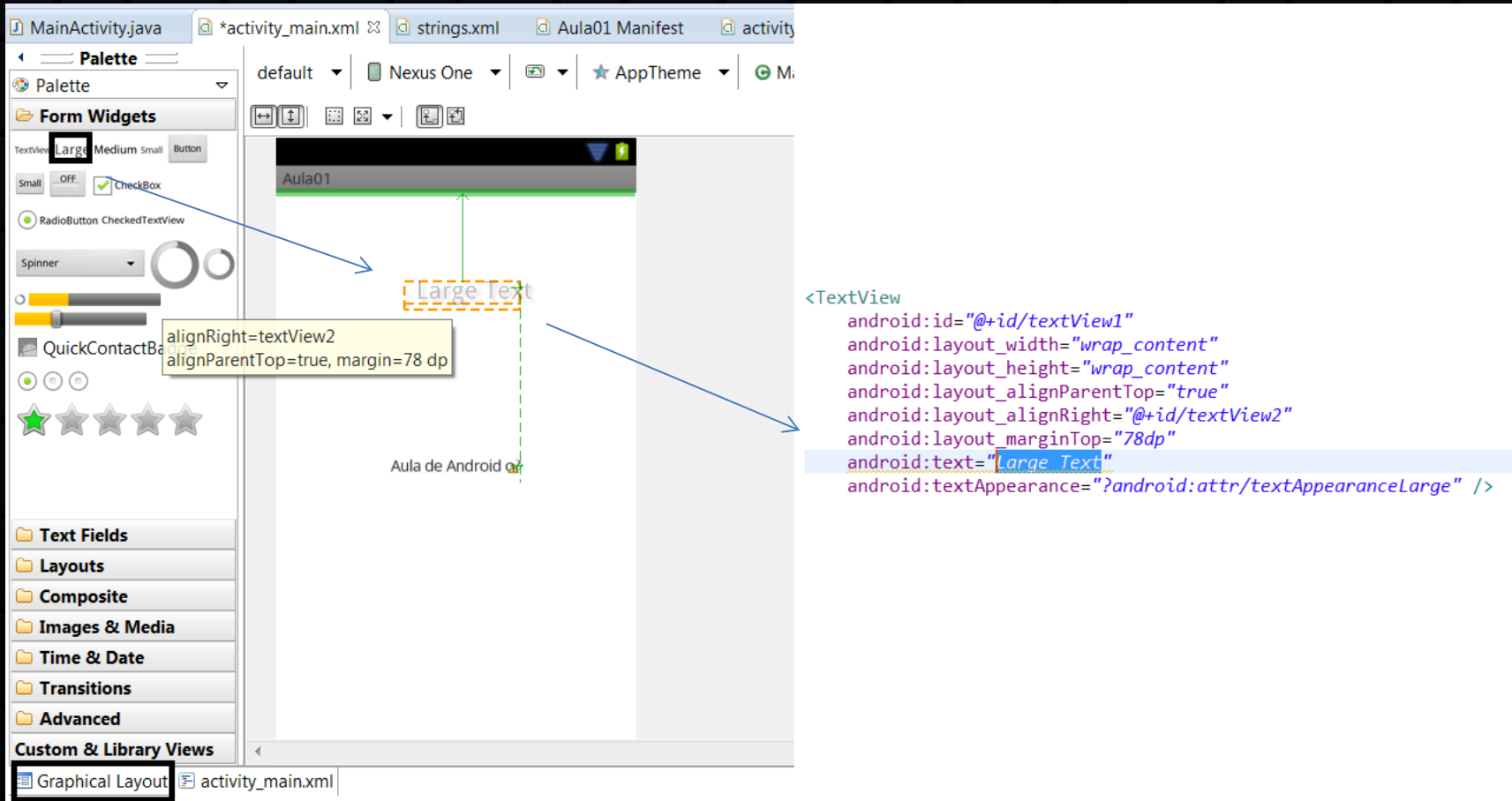
```
android:text="Aula de Android o/"
```

# Android Application Project





# Android Application Project



default | Nexus One | AppTheme | M.

Palette

Form Widgets

TextView Large Medium Small Button

Small OFF ☒ CheckBox

RadioButton CheckedTextView

Spinner

QuickContactBadge

alignRight=textView2  
alignParentTop=true, margin=78 dp

Aula01

Large text

Aula de Android

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignRight="@+id/textView2"
    android:layout_marginTop="78dp"
    android:text="Large Text"
    android:textAppearance="?android:attr/textAppearanceLarge" />
```

Text Fields

Layouts

Composite

Images & Media

Time & Date

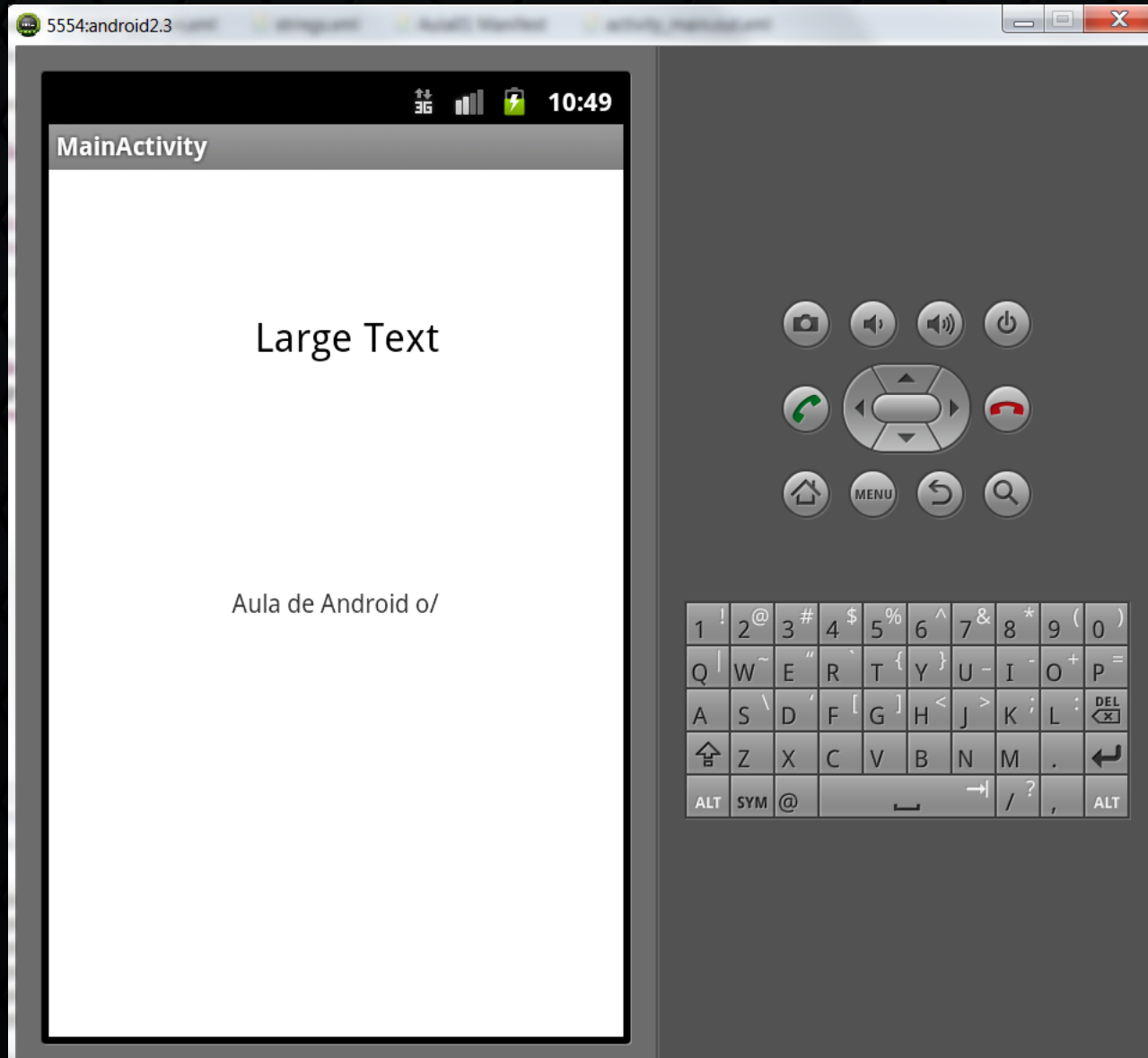
Transitions

Advanced

Custom & Library Views

Graphical Layout activity\_main.xml

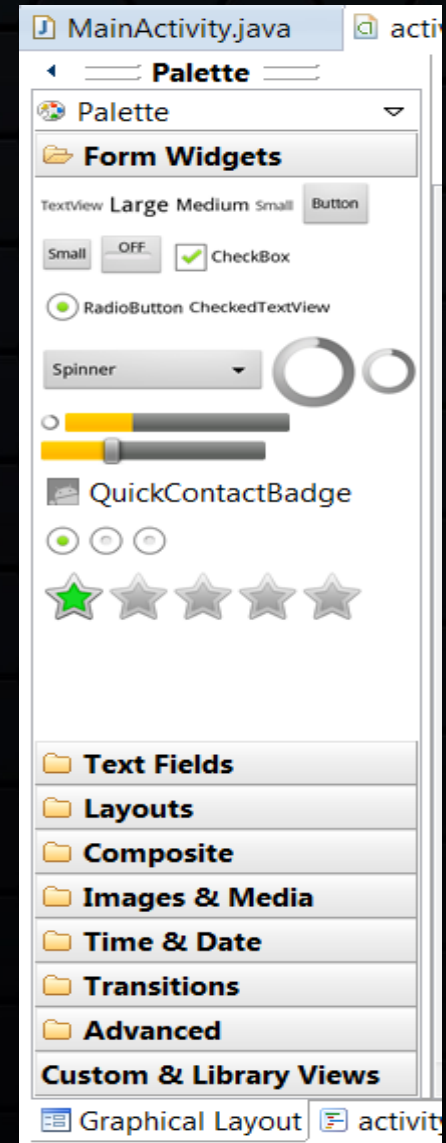
# Android Application Project



# Usando Widgets

Toda aplicação Android é constituída por widgets, que são componentes gráficos que constituem uma aplicação Android.

A partir de agora iremos conhecer os widgets básicos que constituem a plataforma android, para o desenvolvimento das aplicações. De acordo com alguns widgets que fomos conhecendo, vamos desenvolver aplicações que demonstrem o uso deles.



# A widget **TextView**

A widget **TextView** funciona como se fosse uma Label (“rotulo”), onde nele podemos mostrar alguma informação, mensagem e etc. Na nossa primeira aplicação, tivemos a oportunidade de usarmos esse componente.

# A widget **EditText**

A widget **EditText** funciona como se fosse caixa onde podemos digitar nela dados do teclado.



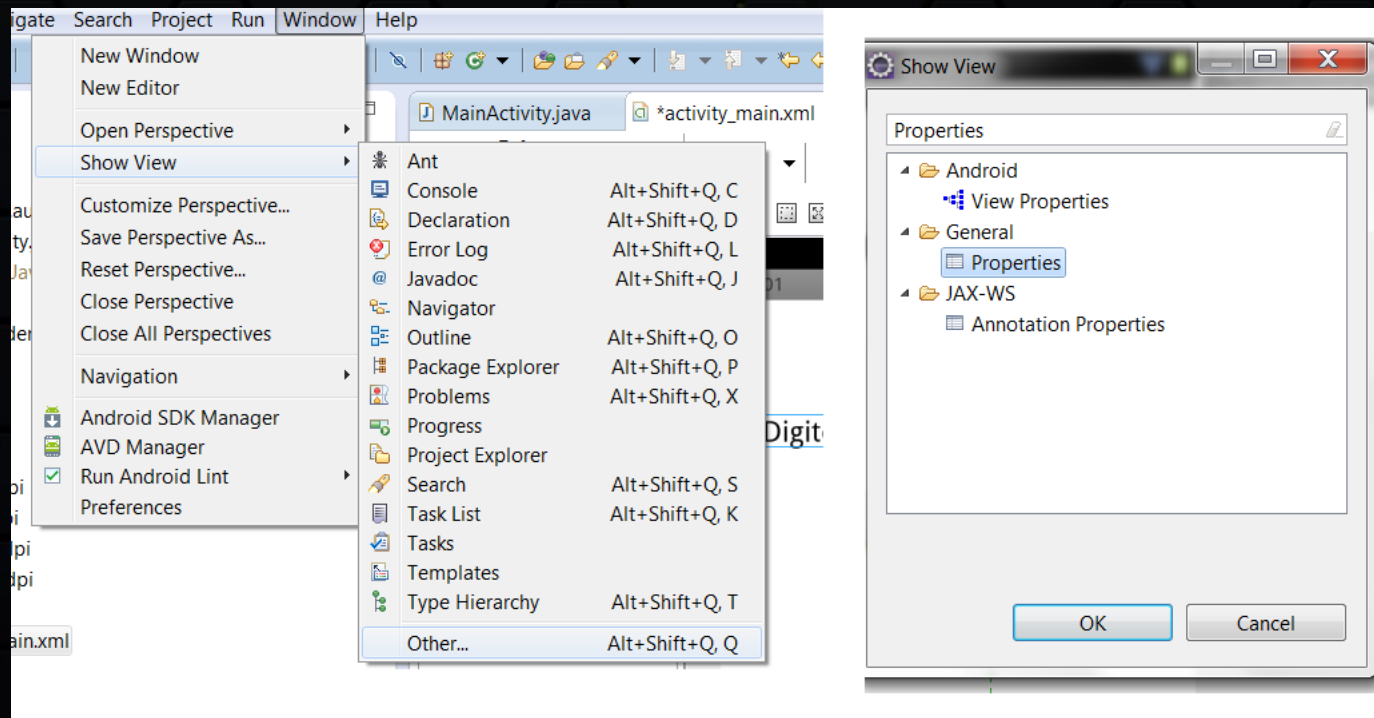


# A widget **Button**

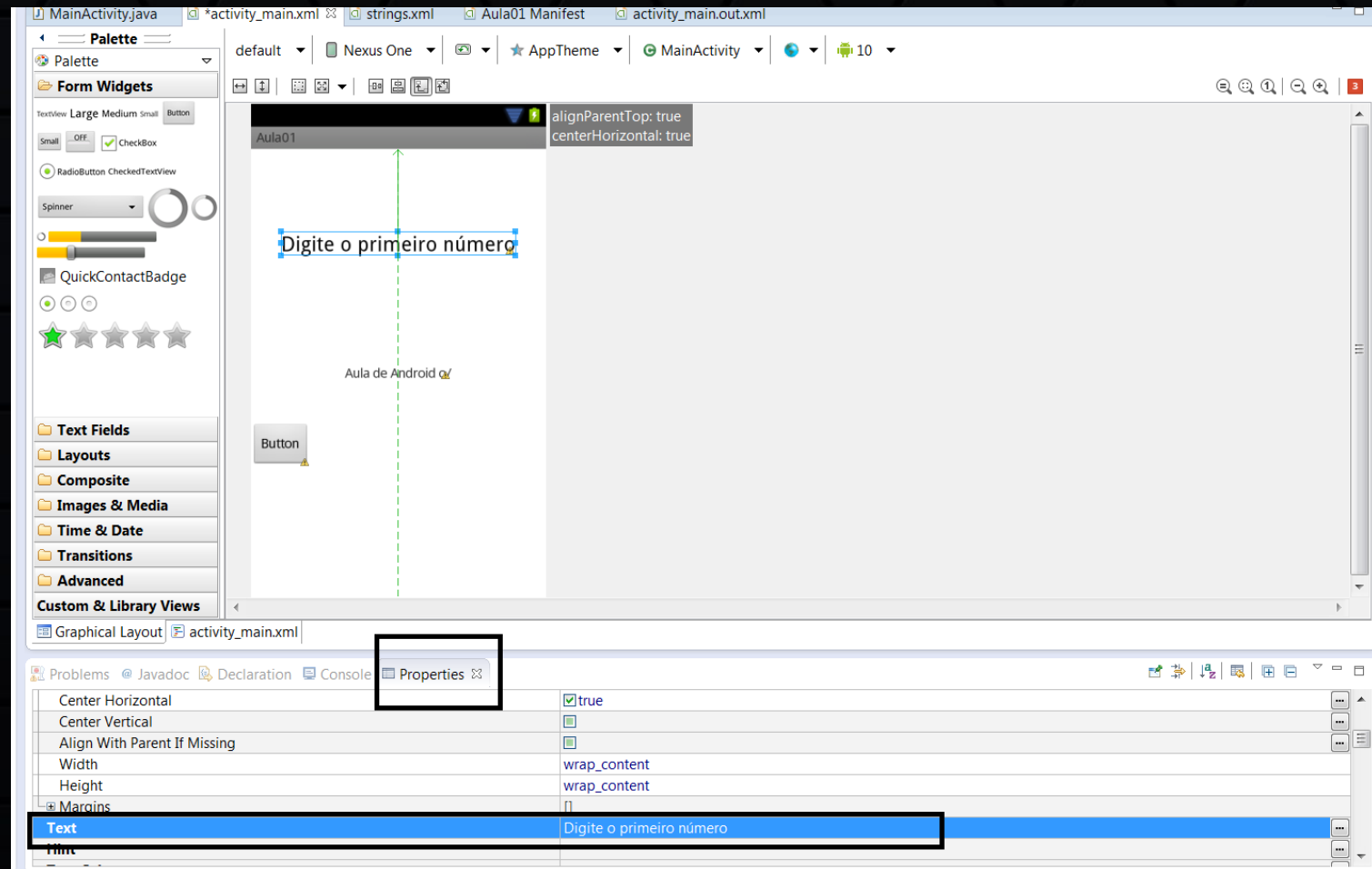
A widget **Button** nada mais é do que um Botão de comando , que quando clicado, dispara uma ação, um evento.



# Desenvolvendo uma aplicação que soma números



# Desenvolvendo uma aplicação que soma números



# Desenvolvendo uma aplicação que soma números



The screenshot shows an Android application window titled "Aula01". It contains two text input fields for numbers, each with a "Text Fields" label and a warning icon. Below the inputs is a large "SOMAR" button and a smaller "Button" button. The interface is in Portuguese.

Aula01

Digite o primeiro número

Text Fields

Digite o segundo número

Text Fields

SOMAR

Button

# Desenvolvendo uma aplicação que soma números



```
public class MainActivity extends Activity {  
  
    EditText ednumero1, ednumero2;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        ednumero1 = (EditText) findViewById(R.id.editText1);  
        ednumero2 = (EditText) findViewById(R.id.editText2);  
        final Button btsonar = (Button) findViewById(R.id.button1);  
    }  
}
```

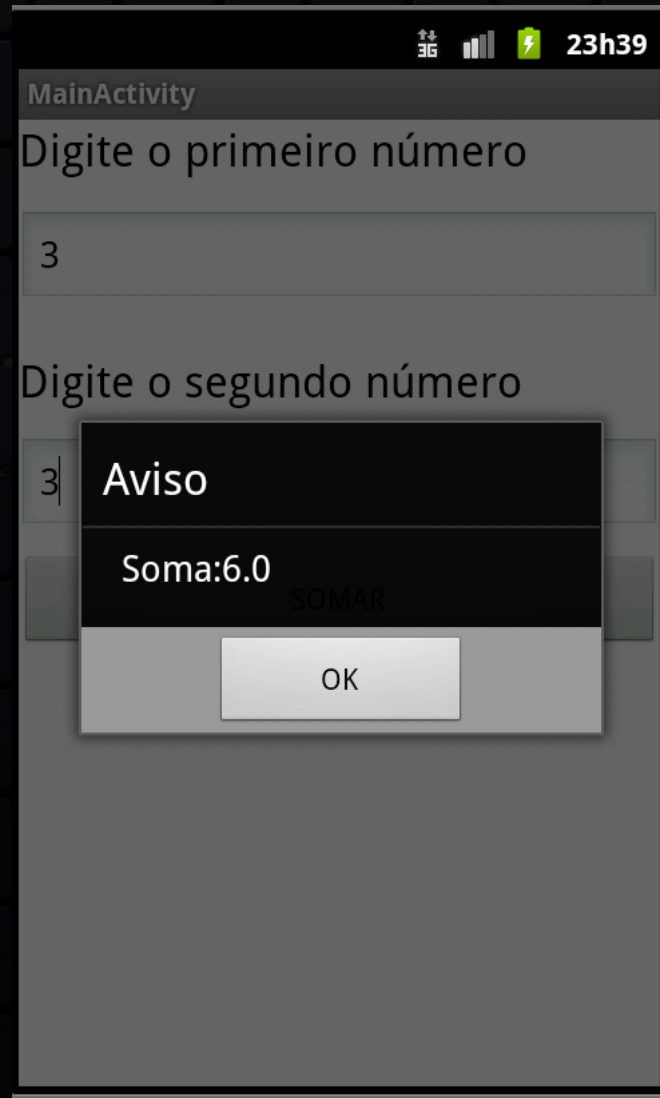


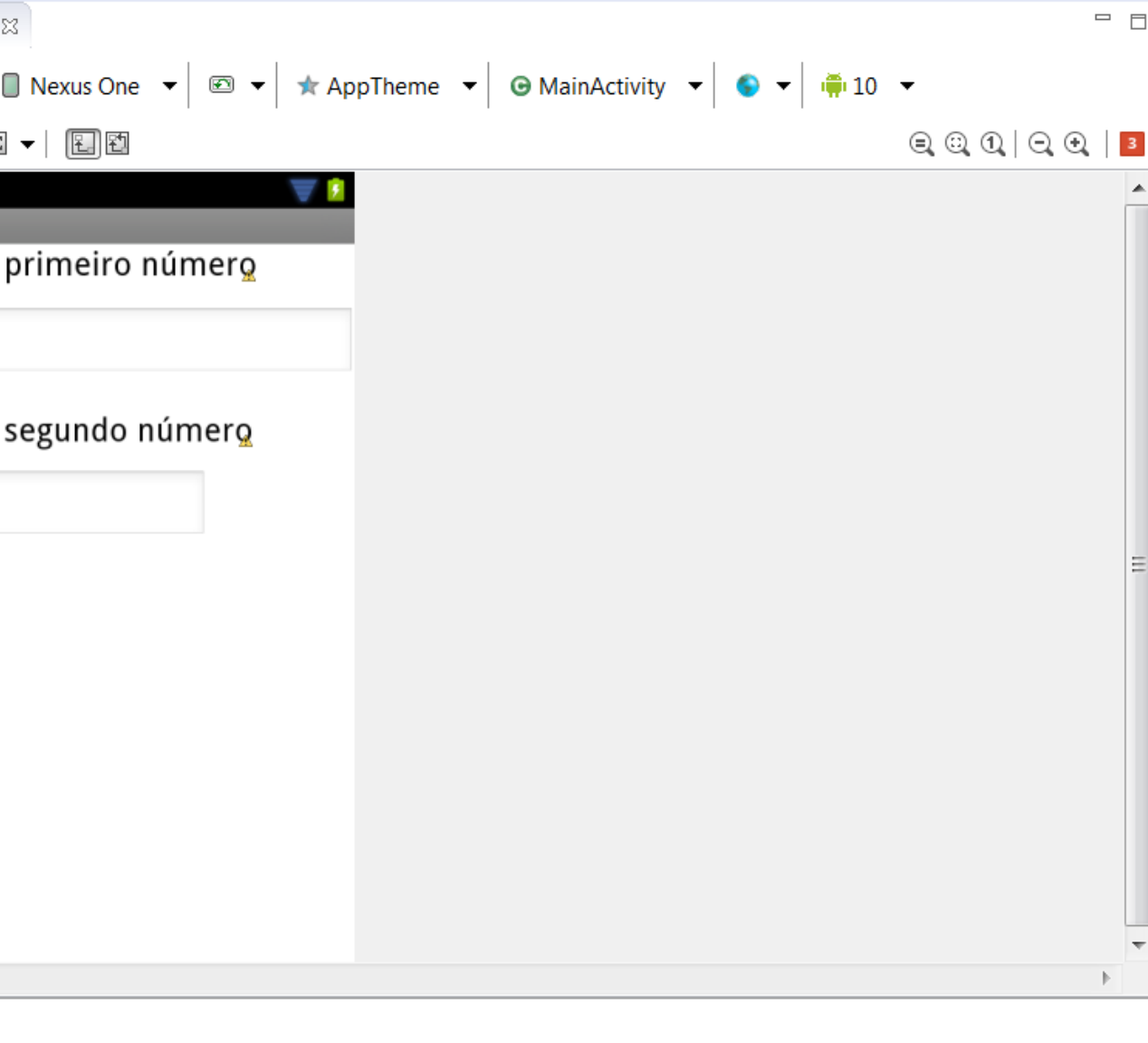
# Desenvolvendo uma aplicação que soma números



```
btsomar.setOnClickListener(new View.OnClickListener() {  
  
    public void onClick(View arg0) {  
  
        double num1 = Double  
            .parseDouble(ednumero1.getText().toString());  
        double num2 = Double  
            .parseDouble(ednumero2.getText().toString());  
        double res = num1 + num2;  
  
        AlertDialog.Builder dialogo = new AlertDialog.Builder(  
            MainActivity.this);  
  
        dialogo.setTitle("Aviso");  
        dialogo.setMessage("Soma:" + res);  
        dialogo.setNeutralButton("OK", null);  
        dialogo.show();  
  
    }  
  
});  
  
}
```

# Desenvolvendo uma aplicação que soma números





**Outline**

RelativeLayout

- textView1 - "Digite o primeiro número"
- editText1
- textView2 - "Digite o segundo número"
- linearLayout1
- button1 - "SOMAR"
- editText2

**Properties**

Center In Par...☐

Center Horiz...☐

Center Vertical☐

Align With P...☐

Widthwrap\_content

Heightwrap\_content

Margins[]

**Text**

**Hint**

**Input Type**number

**Content Desc...**

TextView

Text

**LogCat**

messages. Accepts Java regexes. Prefix with pid; app; tag; or text: to limit scope.

verbose

	PID	TID	Application	Tag	Text
09 23:32:5...	331	331	br.com.senac.a...	KeyCharact...	No keyboard for id 0
09 23:32:5...	331	331	br.com.senac.a...	KeyCharact...	Using default keypad: /system/usr/keychars/qwerty.kcm.bin

# A widget **CheckBox**

A widget **CheckBox** funciona como um componente que pode ser marcado e desmarcado, e que possui também um rótulo.



# Desenvolvendo uma aplicação simples de compras



Agora vamos fazer uma outra aplicação Android que vai fazer uso da widget CheckBox.

Nossa aplicação consiste em um simples sistemas de compras onde possuo cinco produtos, Arroz (R\$2,69) , Leite (R\$ 5,00) , Carne (R\$ 10,00), Feijão (R\$ 2,30) e Refrigerante (R\$ 2,00). Nessa aplicação eu marco os itens que quero comprar e no final o sistema mostra o valor total das compras.



## Aula02

☐ Arroz (R\$ 2,69)☐ Leite (R\$ 5,00)☐ Carne (R\$ 9,70)☐ Feijão (R\$ 2,30)

Total das compras

CheckBox chkarroz, chkleite, chkcarne, chkfeijao;

@Override

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    chkarroz = (CheckBox) findViewById(R.id.checkBox1);  
    chkleite = (CheckBox) findViewById(R.id.checkBox2);  
    chkcarne = (CheckBox) findViewById(R.id.checkBox3);  
    chkfeijao = (CheckBox) findViewById(R.id.checkBox4);  
  
    Button bttotal = (Button) findViewById(R.id.button1);  
}
```

```
btttotal.setOnClickListener(new View.OnClickListener() {  
  
    public void onClick(View arg0) {  
  
        double total = 0;  
  
        if (chkarroz.isChecked())  
            total += 2.69;  
  
        if (chkleite.isChecked())  
            total += 5.00;  
        if (chkcarne.isChecked())  
            total += 9.7;  
  
        if (chkfeijao.isChecked())  
            total += 2.30;  
  
        AlertDialog.Builder dialogo = new AlertDialog.Builder(  
            MainActivity.this);  
        dialogo.setTitle("Aviso");  
        dialogo.setMessage("Valor total da compra :"  
            + String.valueOf(total));  
  
        dialogo.setNeutralButton("OK", null);  
  
        dialogo.show();  
  
    }  
}
```



0h55

## MainActivity

- ☒ Arroz (R\$ 2,69)
- ☒ Leite (R\$ 5,00)
- ☐ Carne (R\$ 9,70)
- ☐ Feijão (R\$ 2,30)

Total das compras

### Aviso

Valor total da compra :12.0

OK

# Exercício

Deixar o usuário editar o valor dos produtos.





# A widget **RadioButton**

A widget **RadioButton** é um componente muito utilizado em opções de múltipla escolha, onde somente uma única opção pode ser selecionada.

# Desenvolvendo uma aplicação de cálculo de salário (Com RadioButton)



Bom, agora vamos fazer uma outra aplicação. Essa aplicação que vamos desenvolver agora consiste em um sistema que vai ler o salário de um funcionário e vai permitir que você escolha o seu percentual de aumento que pode ser de 40% , 45% e 50% e no final o sistema irá mostrar o salário reajustado com o novo aumento.

Aula3

Digite seu salário (R\$)

Qual é o seu percentual ?

☐ 40%

☐ 45%

☐ 50%

Calcular novo salário

```
Button btcalcular = (Button) findViewById(R.id.button1);
btcalcular.setOnClickListener(new View.OnClickListener() {
```

```
    public void onClick(View arg0) {
        double salario, novo_sal;
        EditText edsalario = (EditText) findViewById(R.id.editText1);
        salario = Double.parseDouble(edsalario.getText().toString());
        RadioGroup rg = (RadioGroup) findViewById(R.id.RadioGroup);
        int op = rg.getCheckedRadioButtonId();

        if (op == R.id.radioButton1)
            novo_sal = salario + (salario * 0.4);
        else if (op == R.id.radioButton2)
            novo_sal = salario + (salario * 0.45);
        else
            novo_sal = salario + (salario * 0.50);

        AlertDialog.Builder dialog = new AlertDialog.Builder(
            MainActivity.this);

        dialog.setTitle("Novo salário");
        dialog.setMessage("Seu novo salário é : R$"
            + String.valueOf(novo_sal));
        dialog.setNeutralButton("OK", null);
        dialog.show();
    }
}
```

```
});
```

# Exercício





# A widget **Spinner**

A widget **Spinner** é um componente do tipo caixa de combinação (“ComboBox”) onde nele é armazenado vários itens a serem selecionados. Para que um componente possa ser selecionado, é preciso clicarmos na seta , para que os itens possam ser mostrados e , por consequência, serem selecionados

# Desenvolvendo uma aplicação de cálculo de salário (Com Spinner)

Bom, agora vamos criar a nossa segunda versão do aplicativo acima, usando agora o componente Spinner.



```
private static final String[] percentual = { "De 40%", "De 45%", "De 50%" };  
ArrayAdapter<String> aPercentual;  
Spinner spnsal;
```



```
Button btmostrar = (Button) findViewById(R.id.button1);  
aPercentual = new ArrayAdapter<String>(this,  
    android.R.layout.simple_spinner_item, percentual);
```

```
spnsal = (Spinner) findViewById(R.id.spinner1);  
spnsal.setAdapter(aPercentual);
```

```
btmostrar.setOnClickListener(new View.OnClickListener() {  
  
    public void onClick(View arg0) {  
  
        double salario = 0, novo_sal = 0;  
  
        EditText edsalario = (EditText) findViewById(R.id.editText1);  
  
        salario = Double.parseDouble(edsalario.getText().toString());  
  
        switch (spnsal.getSelectedItemPosition()) {  
            case 0:  
                novo_sal = salario + (salario * 0.4);  
                break;  
            case 1:  
                novo_sal = salario + (salario * 0.45);  
                break;  
            case 2:  
                novo_sal = salario + (salario * 0.5);  
                break;  
        }  
    }  
}
```

```
AlertDialog.Builder dialogo = new AlertDialog.Builder(  
    MainActivity.this);  
dialogo.setTitle("Novo salário");  
dialogo.setMessage("Seu novo salário é : R$" +  
    + String.valueOf(novo_sal));  
dialogo.setNeutralButton("OK", null);  
dialogo.show();  
  
}  
  
});
```

The screenshot shows an Android application interface. At the top, the status bar displays '3G', signal strength, battery level, and the time '3h06'. The app's title bar is labeled 'MainActivity'. The main content area contains the following elements:

- A text label 'Digite seu salário (R\$)' followed by a text input field containing the value '580'.
- A text label 'Qual é o seu percentual ?' followed by a dropdown menu currently showing 'De 40%'.
- A button labeled 'Calcular novo salário'.

At the bottom of the screen is a numeric keypad with buttons for digits 1 through 6, each with its corresponding letters (1, 2 ABC, 3 DEF, 4 GHI, 5 JKL, 6 MNO), a minus sign, and a decimal point.



# Exercício



# A widget **ListView**

A Widget **ListView** é um componente que possui vários itens a serem selecionados, similar ao componente **Spinner**. A única diferença entre o **ListView** e o **Spinner** é que no componente **ListView**, os itens já são mostrados sem nenhuma necessidade de se clicar em alguma parte dele para que os mesmos possam ser mostrados.

# Desenvolvendo uma aplicação de lista telefônica



Agora vamos fazer uma nova aplicação em Android. Essa aplicação consiste em uma lista telefônica já pronta com contatos. Quando selecionamos um contato, ele mostra na tela uma mensagem com o nome selecionado. A nossa aplicação vai fazer uso do widget chamado ListView, que exiba uma lista contendo valores que podem ser selecionados.

```

public ListView lista;
static final String[] contatos = new String[] { "Ricardo","Lucas","Rafael","Gabriela","Silvana"}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, contatos);
    lista = (ListView) findViewById(R.id.textView1);
    lista.setAdapter(adapter);
    lista.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
            if (lista.getSelectedItem() != null) {
                AlertDialog.Builder dialogo = new AlertDialog.Builder(MainActivity.this);
                dialogo.setTitle("Contato selecionado");
                dialogo.setMessage(lista.getSelectedItem().toString());
                dialogo.setNeutralButton("OK", null);
                dialogo.show();
            }
        }
    });
}

```

# Exercício

