Programação Orientada a Objetos

Programa Orientada a Objetos em Java/Android

www.comp.ita.br/~pauloac/ces22/

Referências sobre Android

- Professional Android Application Development. Reto Meier. Wrox. 2008. (PDF disponível na web)
- **Professional Android 2 Application Development** Reto Meier. Wrox. 2010.
- Google Android. Ricardo Lecheta . Ed. Novatec.. 2a. Ed. 2010. (Em português)
- Android Developers website: http://developer.android.com/index.html
- Informações sobre API Android (packages, classes, métodos,etc.)
  - http://www.androidjavadoc.com
- Há muitos outros livros, sites, blogs, foruns, etc. sobre Android ver na web(Google)

Sumário

- **Introdução a Plataforma Android**
- Primeiros Programas
- Criando Aplicações e Activities
- Criação de Interfaces de usuário
- Armazenamento e compartilhamento de dados
- Conexão a redes e outros programas
- Mapas, geolocalização e Serviços Baseados em Localização (Location-based Services)
- Multithreading
- Mecanismos de Acesso ao Hardware
- Comunicação ponto-a-ponto
- Desenvolvimento avançado em Android

Introdução

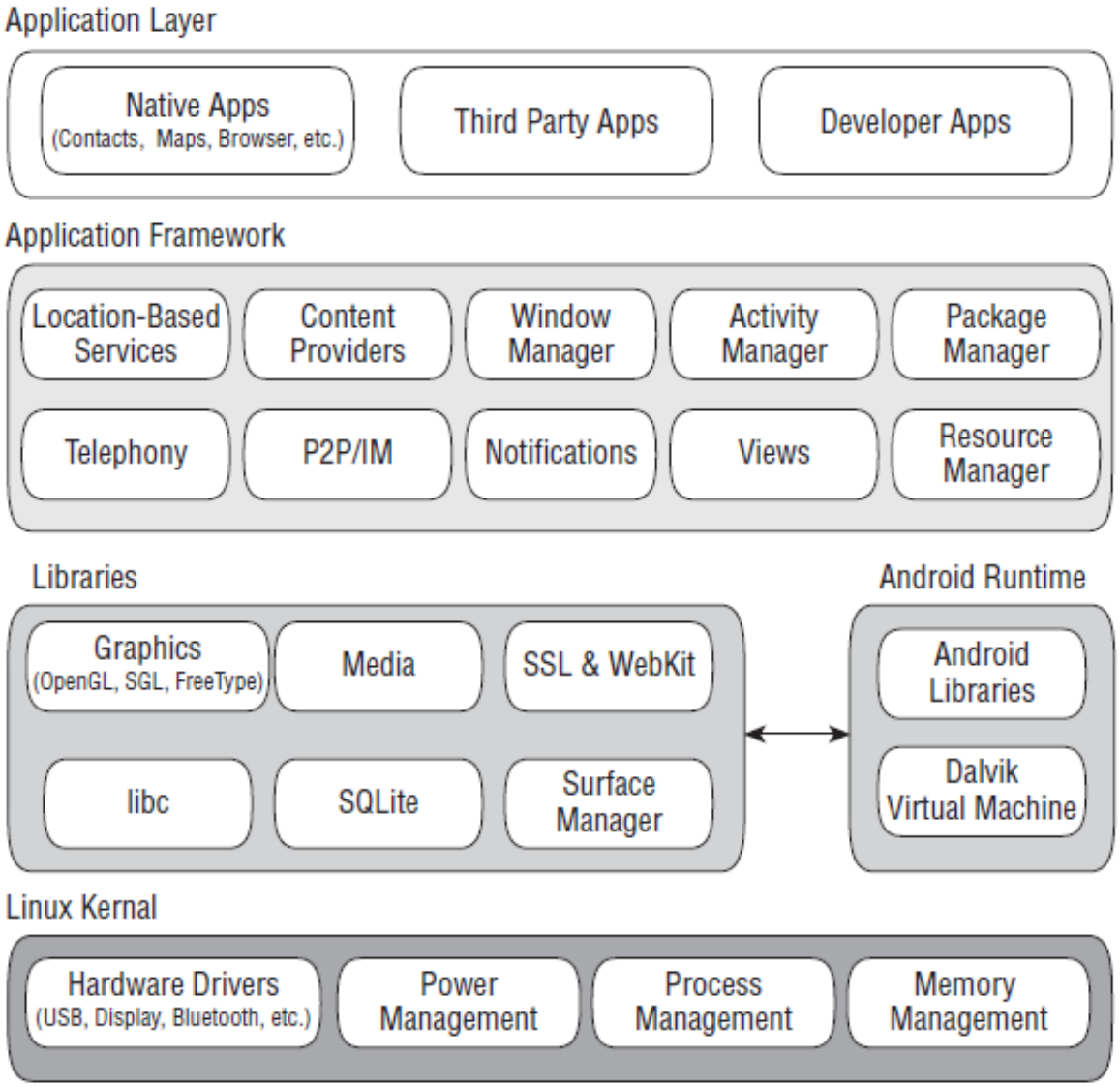
- The *Open Handset Alliance (OHA) is a collection of more than 30 technology companies including hardware* manufacturers, mobile carriers, and software developers. Of particular note are the prominent mobile technology companies Motorola, HTC, T-Mobile, and Qualcomm. In their own words, the OHA represents:
  - *A commitment to openness, a shared vision for the future, and concrete plans to make the vision a reality. To accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience.*

- The OHA hopes to deliver a better mobile software experience for consumers by providing the platform needed for innovative mobile development at a faster rate and a higher quality without licensing fees for software developers or handset manufacturers.

Android

- O que é Android? Segundo o Google:
  - *“The first truly open and comprehensive platform for mobile devices”*
- A plataforma pode ser dividida em quatro grandes camadas (layers)
  - Application Layer
  - Application Framework
  - Libraries / Android Run-time(inclui Dalvik Virtual Machine)
  - Linux Kernel

The Android Software Stack



Componentes da plataforma

- **Linux Kernel Core services (including hardware drivers, process and memory management,** security, network, and power management) are handled by a Linux 2.6 kernel. The kernel also provides an abstraction layer between the hardware and the remainder of the stack.

- **Libraries Running on top of the kernel, Android includes various C/C++ core libraries such** as libc and SSL, as well as:
  - A media library for playback of audio and video media
  - A Surface manager to provide display management
  - Graphics libraries that include SGL and OpenGL for 2D and 3D graphics
  - SQLite for native database support
  - SSL and WebKit for integrated web browser and Internet security

Componentes da plataforma - 2

- **Android Run Time** What makes an Android phone an Android phone rather than a mobile Linux implementation is the Android run time.

  - **Core Libraries** While Android development is done in Java, Dalvik is not a Java VM. The core Android libraries provide most of the functionality available in the core Java libraries as well as the Android-specific libraries.

  - **Dalvik Virtual Machine** Dalvik is a register-based virtual machine that's been optimized to ensure that a device can run multiple instances effi ciently. It relies on the Linux kernel for threading and low-level memory management.

Componentes da plataforma - 3

- **Application Framework** The application framework provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources.

- **Application Layer** All applications, both native and third party, are built on the application layer using the same API libraries. The application layer runs within the Android run time using the classes and services made available from the application framework.

Dalvik Virtual Machine

- One of the key elements of Android is the Dalvik virtual machine. Rather than use a traditional Java virtual machine (VM) such as Java ME (Java Mobile Edition), Android uses its own custom VM designed to ensure that multiple instances run efficiently on a single device.

- The Dalvik VM uses the device's underlying Linux kernel to handle low-level functionality including security, threading, and process and memory management.

Dalvik Virtual Machine - 2

- It's also possible to write C/C++ applications that run directly on the underlying Linux OS. While you *can do this, in most cases there's no reason* you should need to. The main reason would be achieve better performance with the same hardware

- The Dalvik VM executes Dalvik executable fi les, a format optimized to ensure minimal memory footprint.

- The .dex executables are created by transforming Java language compiled classes using the tools supplied within the Android SDK

Android Application Architecture

- Android's architecture encourages the concept of component reuse, allowing you to publish and share activities, services, and data with other applications with access managed by the security restrictions you put in place

- The following application services are the architectural cornerstones of all Android applications, providing the framework you'll be using for your own software:
  - **Activity Manager** Controls the life cycle of your activities, including management of the activity stack
  - **Views** Are used to construct the user interfaces for your activities
  - **Notification Manager** Provides a consistent and non-intrusive mechanism for signaling your users
  - **Content Providers** Lets your applications share data between applications
  - **Resource Manager** Supports non-code resources like strings and graphics to be externalized

Android Libraries

- Android offers a number of APIs for developing your applications. The following list of core APIs should provide an insight into what's available; all Android devices will offer support for at least these APIs:
  - **android.util** The core utility package contains low-level classes like specialized containers, string formatters, and XML parsing utilities.
  - **android.os** The operating system package provides access to basic operating system services like message passing, interprocess communication, clock functions, and debugging.
  - **android.graphics** The graphics API supplies the low-level graphics classes that support canvases, colors, and drawing primitives, and lets you draw on canvases.
  - **android.text** The text processing tools for displaying and parsing text.
  - **android.database** Supplies the low-level classes required for handling cursors when working with databases.
  - **android.content** The content API is used to manage data access and publishing by providing services for dealing with resources, content providers, and packages.
  - **android.view** Views are the core user interface class. All user interface elements are constructed using a series of Views to provide the user interaction components.

Android Libraries - 2

- **android.widget** Built on the View package, the widget classes are the “here’s one we created earlier” user-interface elements for you to use in your applications. They include lists, buttons, and layouts.
- **com.google.android.maps** A high-level API that provides access to native map controls that you can use within your application. Includes the MapView control as well as the Overlay and MapController classes used to annotate and control your embedded maps
- **android.app** A high-level package that provides access to the application model. The application package includes the Activity and Service APIs that form the basis for all your Android applications.
- ❑ **android.provider** To ease developer access to certain standard Content Providers (such as the contacts database), the Provider package offers classes to provide access to standard databases included in all Android distributions.
- ❑ **android.telephony** The telephony APIs give you the ability to directly interact with the device’s phone stack, letting you make, receive, and monitor phone calls, phone status, and SMS messages.
- ❑ **android.webkit** The WebKit package features APIs for working with Web-based content, including a WebView control for embedding browsers in your activities and a cookie manager.

C/C++ Libraries

- In addition to the Android APIs, the Android stack includes a set of C/C++ libraries that are exposed through the application framework. These libraries include:
  - **OpenGL** The library used to support 3D graphics based on the Open GL ES 1.0 API
  - **FreeType** Support for bitmap and vector font rendering
  - **SGL** The core library used to provide a 2D graphics engine
  - **libc** The standard C library optimized for Linux-based embedded devices
  - **SQLite** The lightweight relation database engine used to store application data
  - **SSL** Support for using the Secure Sockets Layer cryptographic protocol for secure Internet communications

Advanced Libraries

- Outras bibliotecas para usos específicos que podem estar disponíveis em alguns dispositivos são as seguintes:
  - **android.location The location-based services API gives your applications access to the** device's current physical location. Location-based services provide generic access to location information using whatever position-fixing hardware or technology is available on the device.
  - **android.media The media APIs provide support for playback and recording of audio and** video media files, including streamed media.
  - **android.opengl** Android offers a powerful 3D rendering engine using the OpenGL ES API that you can use to create dynamic 3D user interfaces for your applications.
  - **android.hardware** Where available, the hardware API exposes sensor hardware including the camera, accelerometer, and compass sensors
  - **android.bluetooth, android.net.wifi , and android.telephony** Android also provides low-level access to the hardware platform, including Bluetooth, Wi-Fi, and telephony hardware

Resumo

- Android offers an opportunity for developers to create innovative software applications for mobile devices **without the restrictions generally associated with the existing proprietary** mobile development frameworks.

- Android is a **software stack,** which includes not only an application layer and development toolkit but also the Dalvik VM, a custom run time, core libraries, and a Linux kernel; all of which will be available as open source.

Sumário

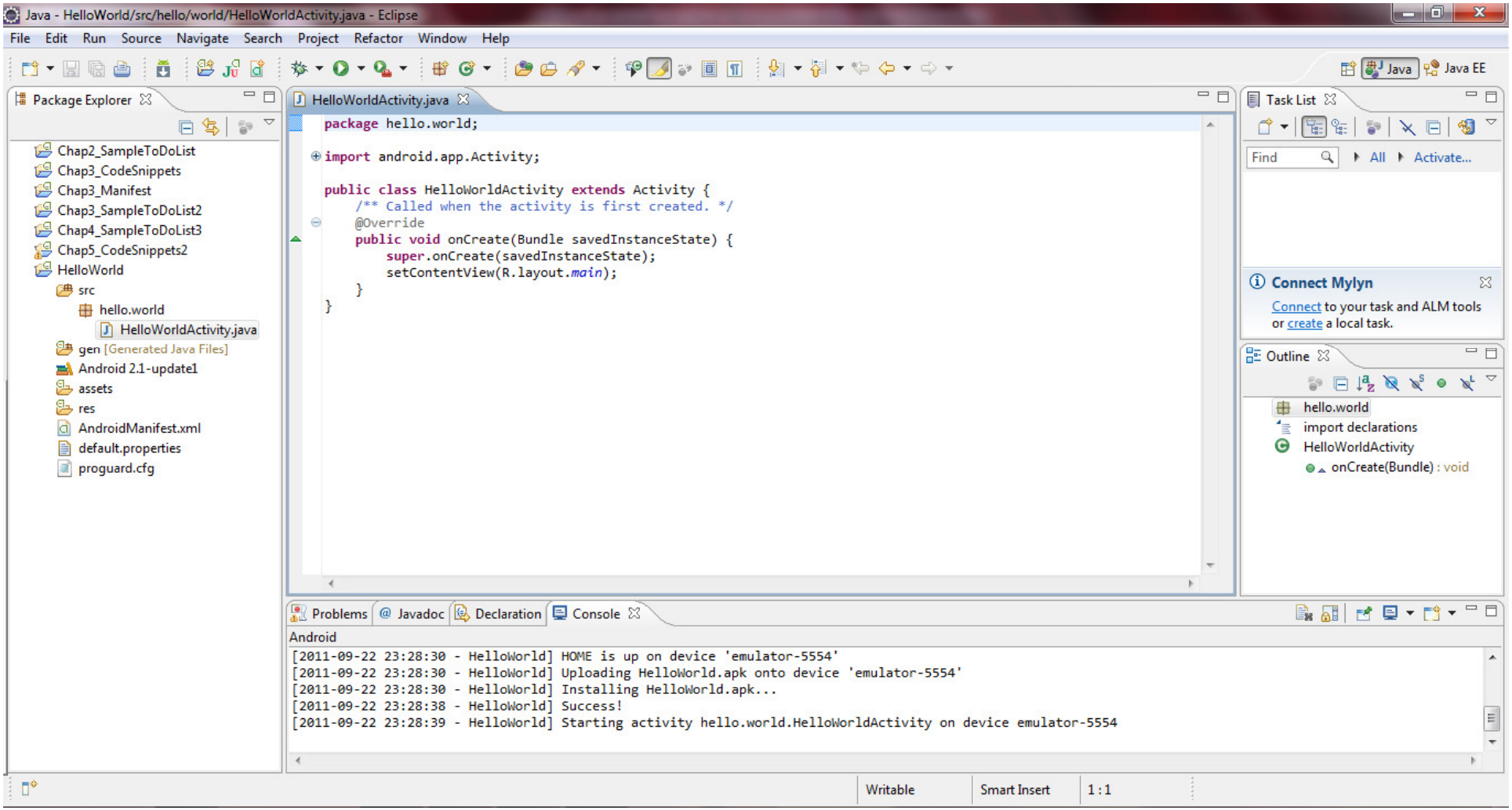
- Introdução a Plataforma Android
- **Primeiros Programas**
- Criando Aplicações e Activities
- Criação de Interfaces de usuário
- Conexão a redes e outros programas
- Armazenamento e compartilhamento de dados
- Mapas, geolocalização e Serviços Baseados em Localização (Location-based Services)
- Multithreading
- Comunicação ponto-a-ponto
- Mecanismos de Acesso ao Hardware
- Desenvolvimento avançado em Android

Iniciando em Android

- Para desenvolver para a plataforma é fundamental ter o Android SDK e um JDK (Java Development Kit). O uso de uma IDE embora não seja imprescindível é altamente recomendável por tornar o desenvolvimento mais produtivo.

- O IDE mais utilizado para desenvolver para Android é o Eclipse, que já conta com plugin específico para este fim.

Eclipse + ADT Plugin

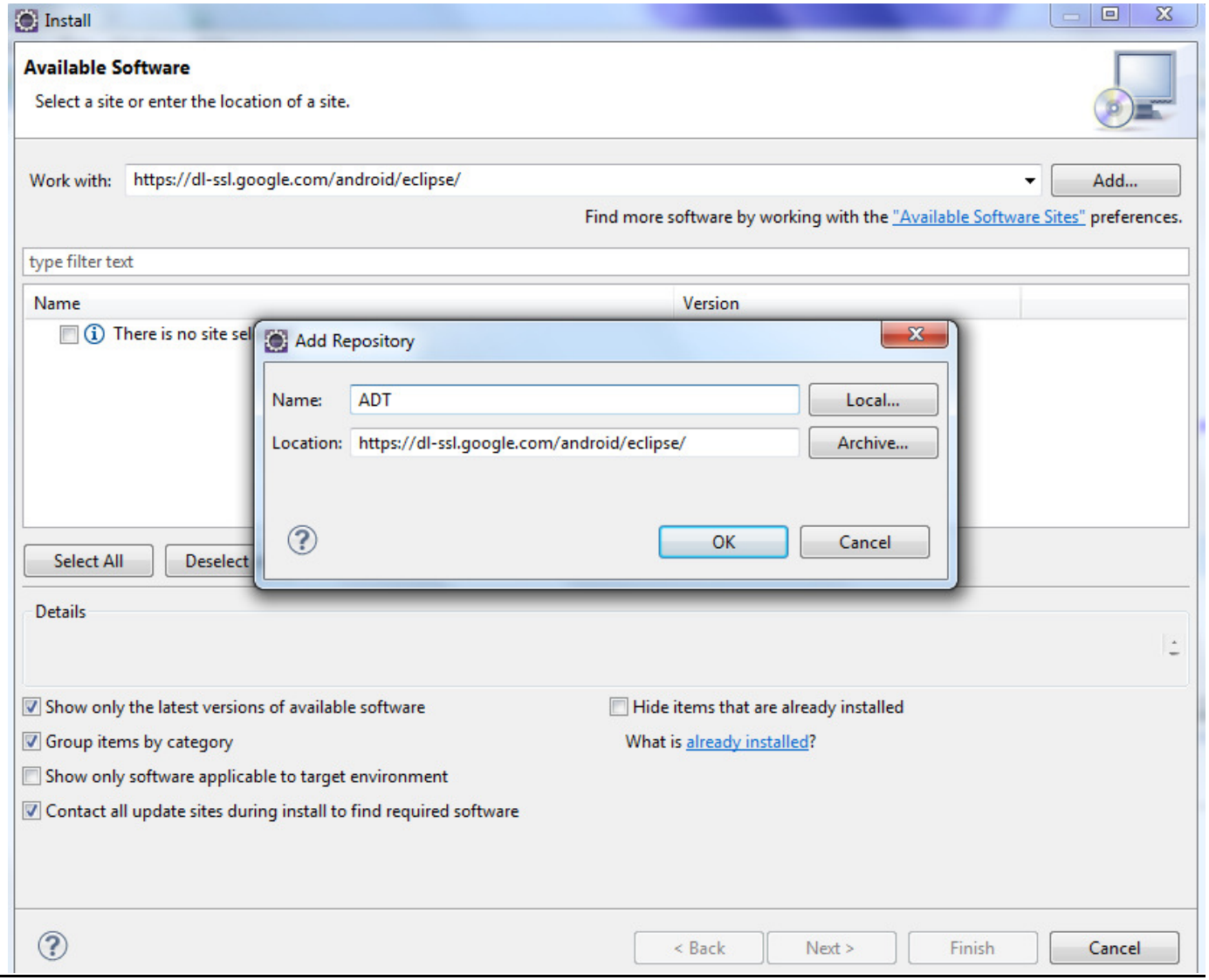


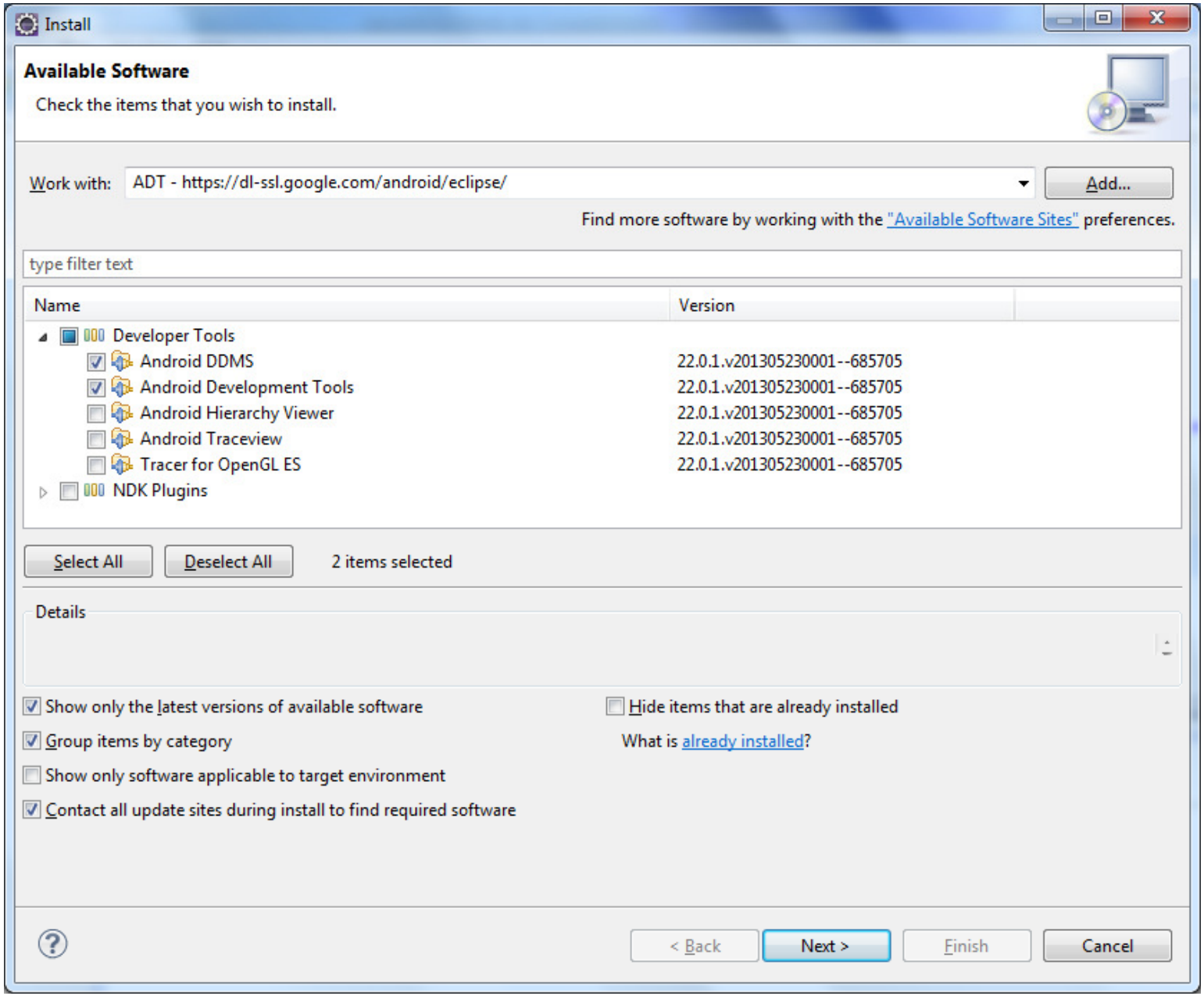
Eclipse+ADT Plugin+Android SDK

- Baixar o ADT Bundle de http://developer.android.com/sdk/
- Descompactar em D:\AndroidSDK
- Executar D:\AndroidSDK\eclipse\eclipse.exe

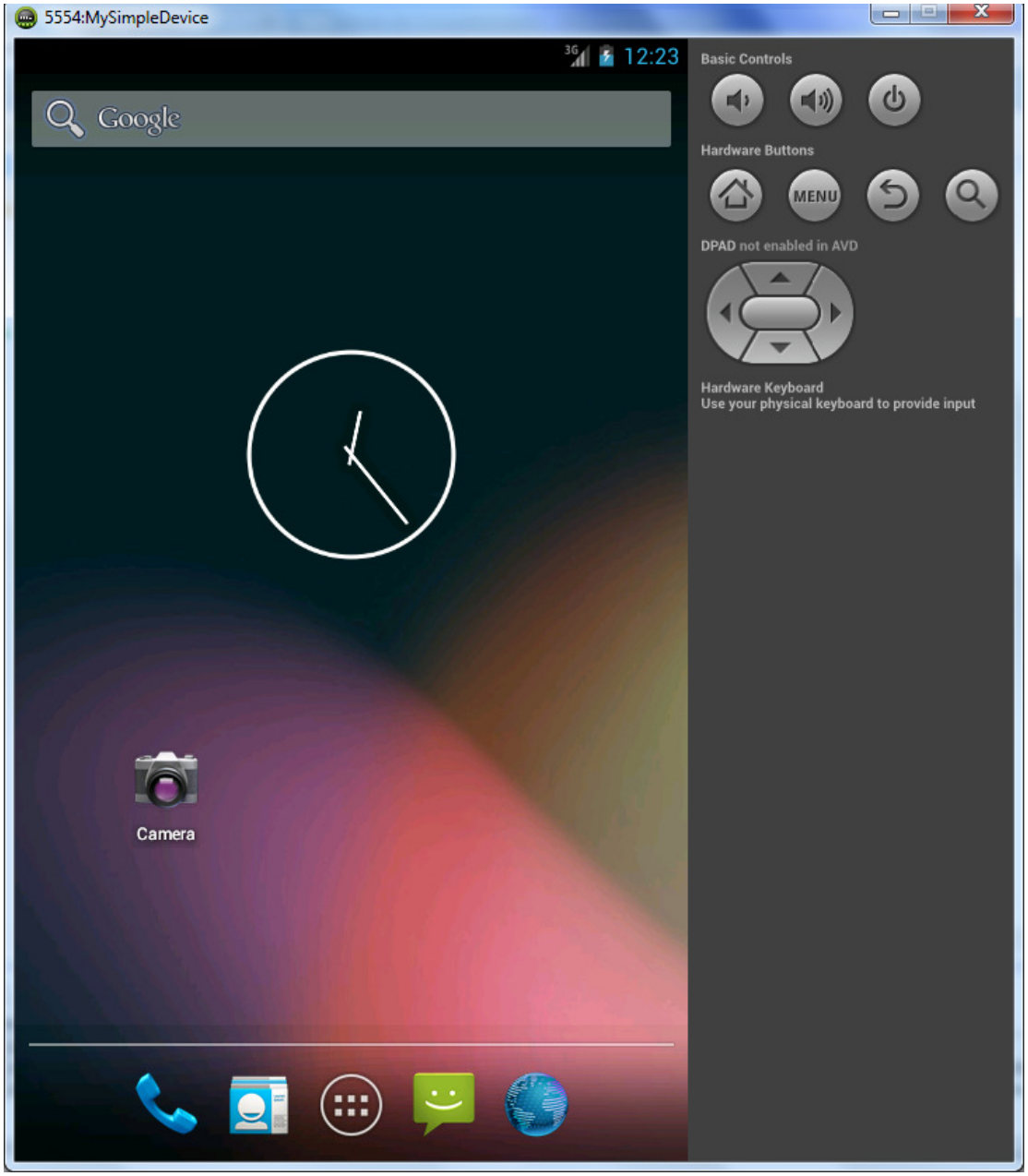
Instalando o Plugin ADT

Menu Help >>
Install New
Software

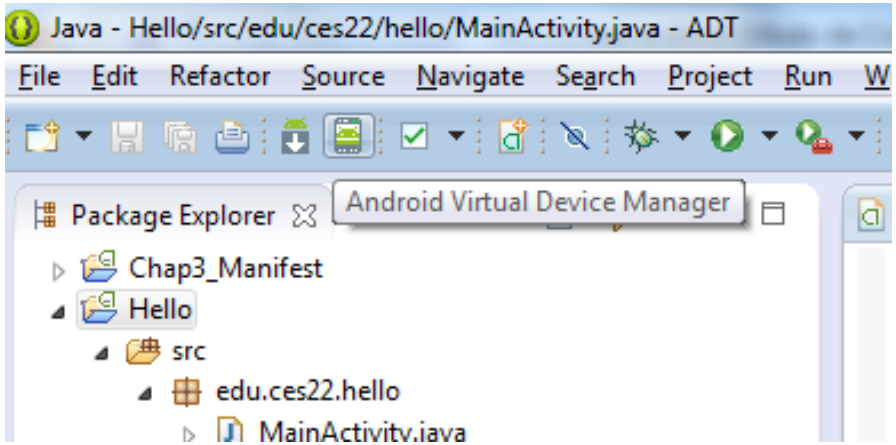


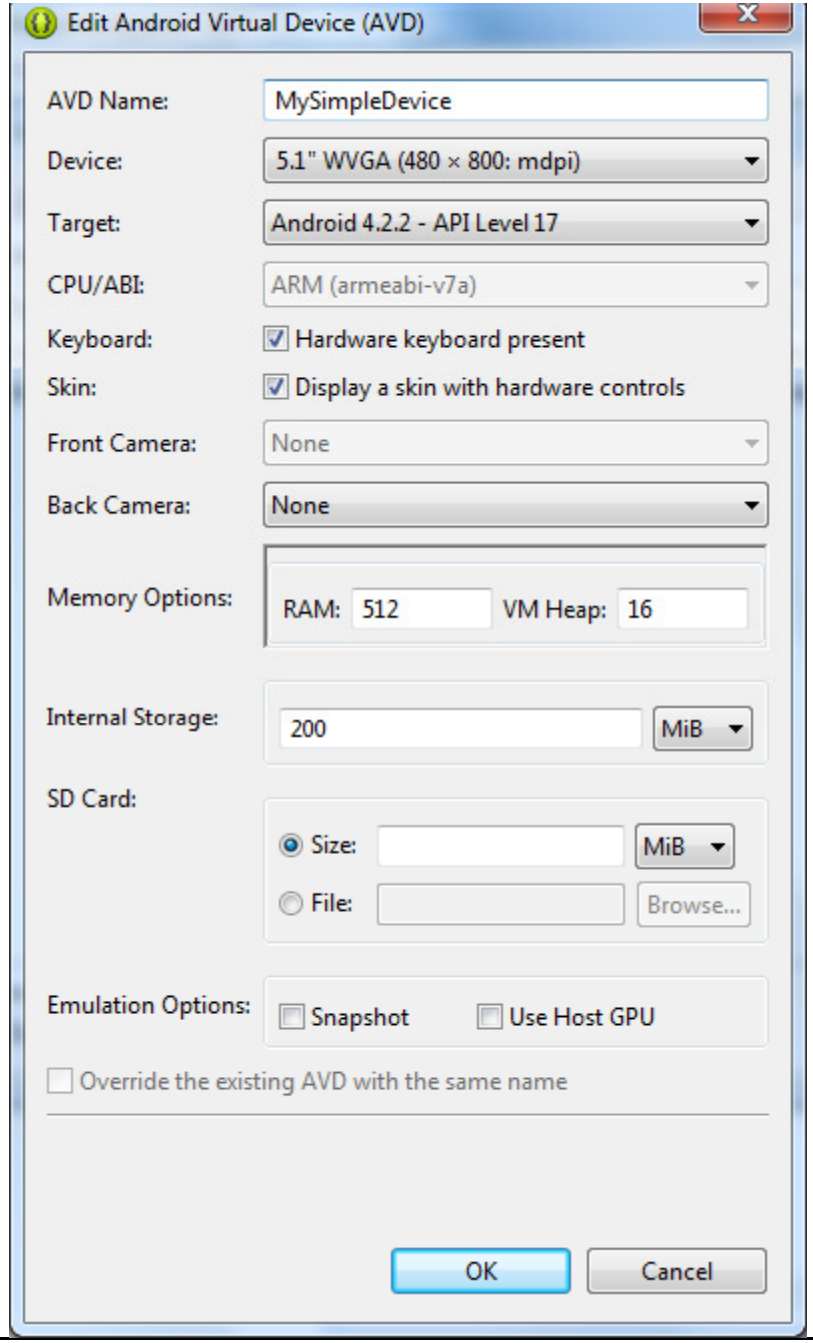

Install
Available Software
Check the items that you wish to install.
Work with:
ADT - https://dl-ssl.google.com/android/eclipse/
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
Developer Tools
Android DDMS
22.0.1.v201305230001--685705
Android Development Tools
22.0.1.v201305230001--685705
Android Hierarchy Viewer
22.0.1.v201305230001--685705
Android Traceview
22.0.1.v201305230001--685705
Tracer for OpenGL ES
22.0.1.v201305230001--685705
NDK Plugins
Select All
Deselect All
2 items selected
Details
Show only the latest versions of available software
Group items by category
Show only software applicable to target environment
Contact all update sites during install to find required software
Hide items that are already installed
What is already installed?
< Back
Next >
Finish
Cancel

Android Emulator

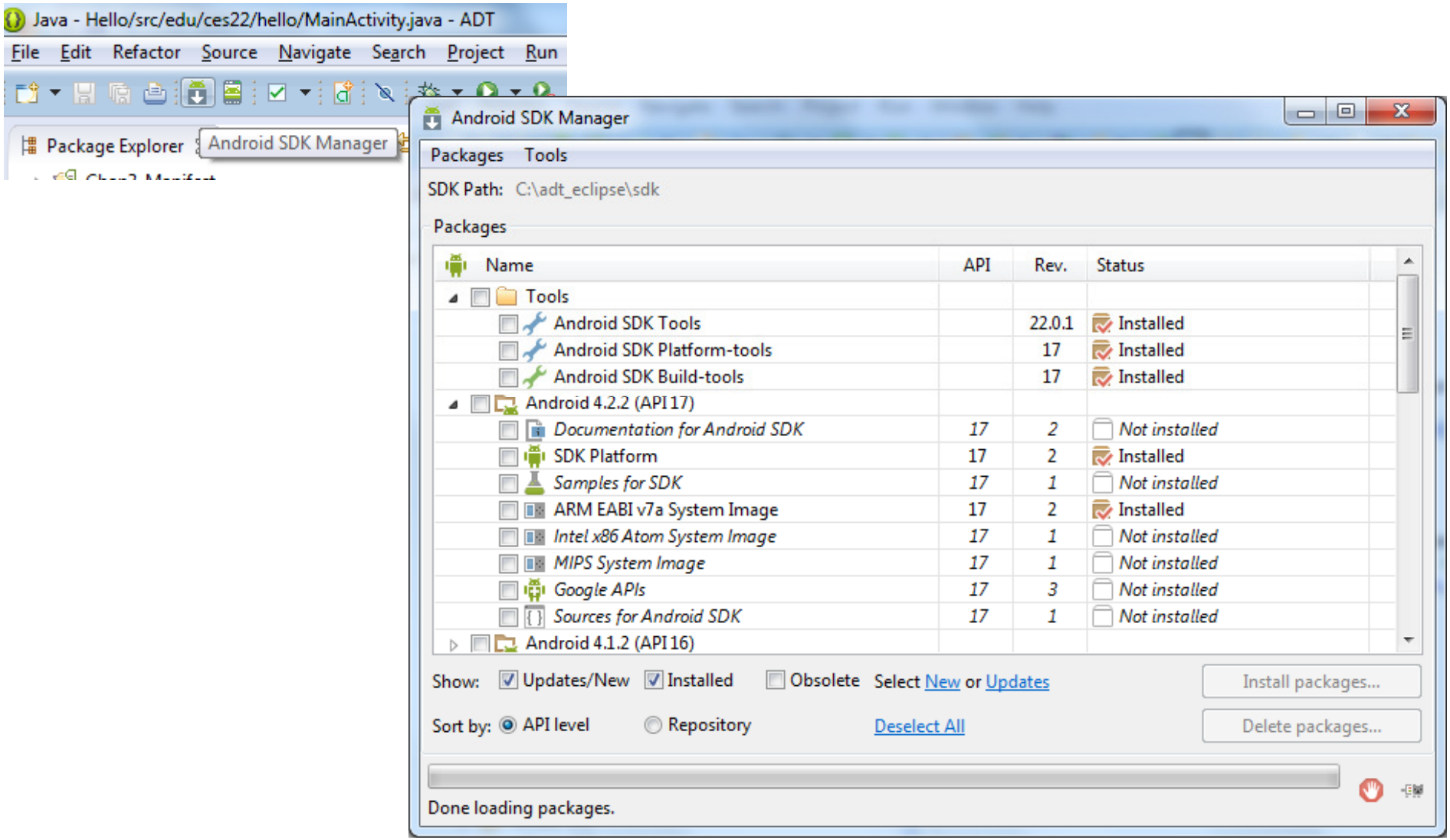


Android Virtual Device



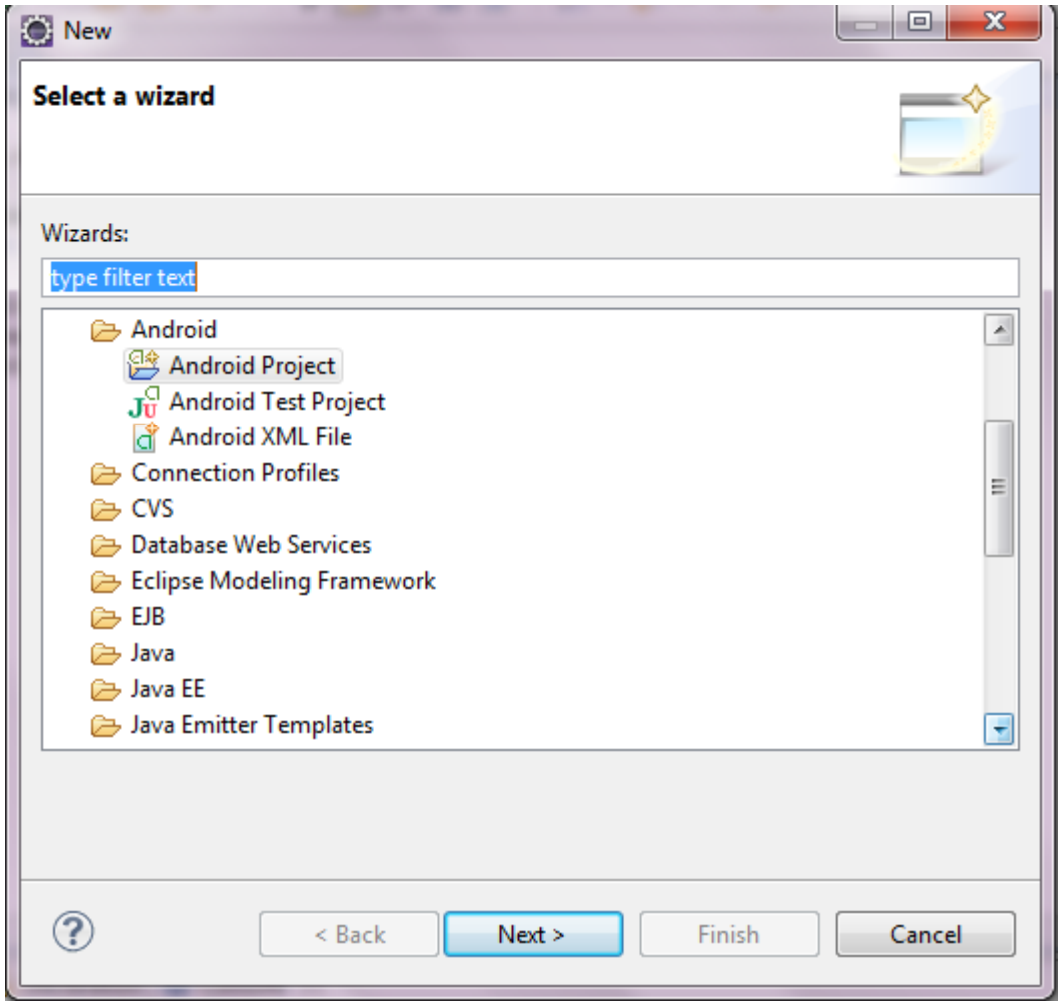


Android SDK Manager

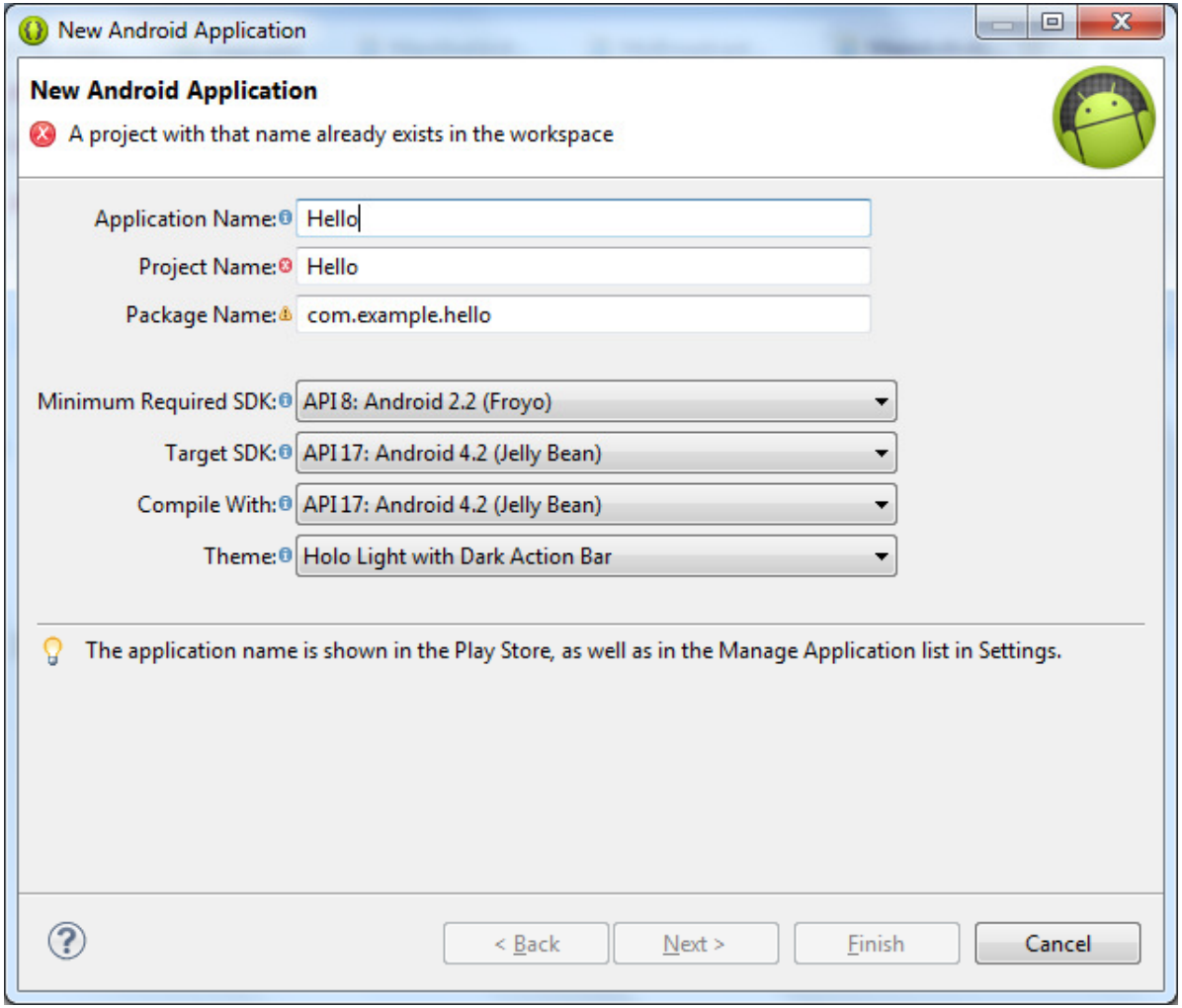


Projetos Android

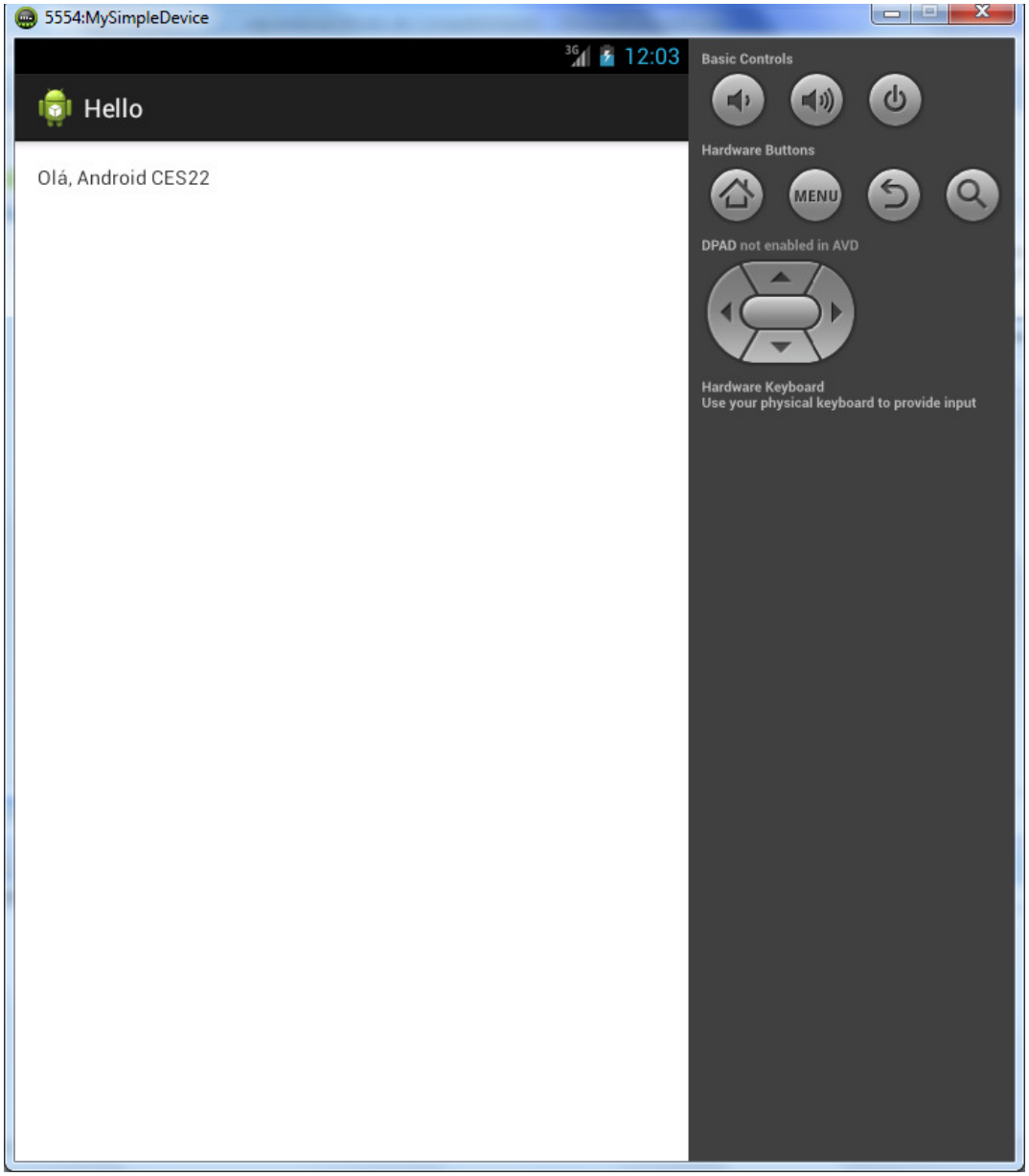
- New > Other



Novo Projeto Android



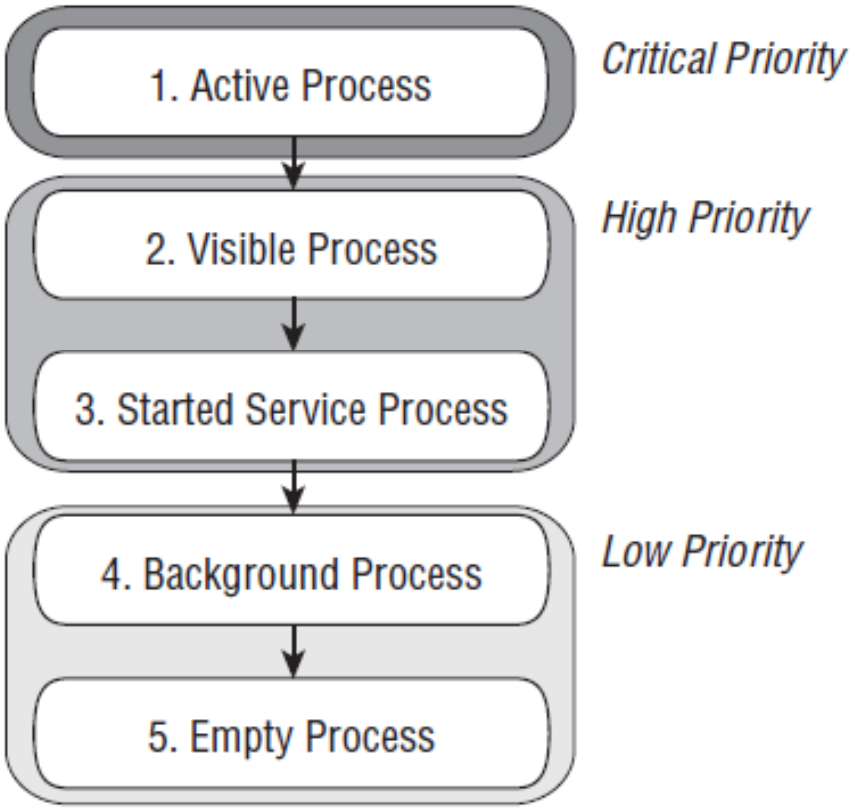
Hello World!


5554:MySimpleDevice
12:03
Hello
Olá, Android CES22
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

Sumário

- Introdução
- Primeiros Programas
- **Criando Aplicações e Activities**
- Criação de Interfaces de usuário
- Conexão a redes e outros programas
- Armazenamento e compartilhamento de dados
- Mapas, geolocalização e Serviços Baseados em Localização (Location-based Services)
- Multithreading
- Comunicação ponto-a-ponto
- Mecanismos de Acesso ao Hardware
- Desenvolvimento avançado em Android

Prioridade de Processos no Android



Active Processes

- Active (foreground) processes are those hosting applications with components currently interacting with the user. These are the processes Android is trying to keep responsive by reclaiming resources.
  - Active processes include:
  - Activities in an “active” state; that is, they are in the foreground and responding to user events. You will explore Activity states in greater detail later in this chapter.
  - Activities, Services, or Broadcast Receivers that are currently executing an onReceive event handler.
  - Services that are executing an onStart, onCreate, or onDestroy event handler.

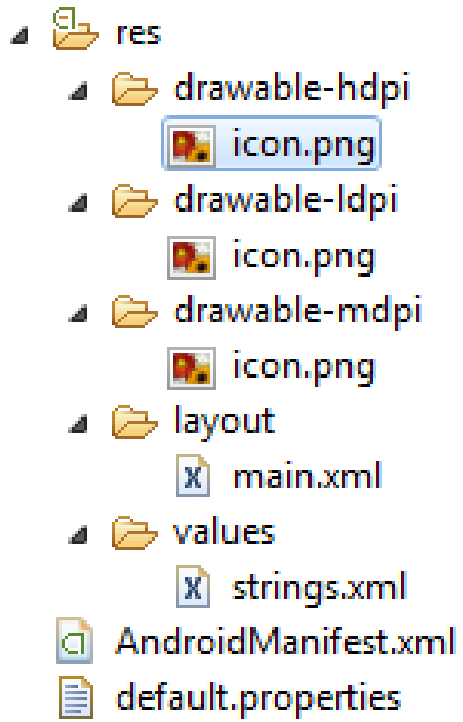
- **Visible Processes** Visible, but inactive processes are those hosting “visible” Activities. As the name suggests, visible Activities are visible, but they aren’t in the foreground or responding to user events. This happens when an Activity is only partially obscured (by a non-full-screen or transparent Activity).

- **Started Service Processes Processes hosting Services that have been started. Services support ongoing** processing that should continue without a visible interface. Because Services don’t interact directly with the user, they receive a slightly lower priority than visible Activities. They are still considered to be foreground processes and won’t be killed unless resources are needed for active or visible processes.

- **Background Processes** Processes hosting Activities that aren’t visible and that don’t have any Services that have been started are considered background processes. There will generally be a large number of background processes that Android will kill using a last-seen-first-killed pattern to obtain resources for foreground processes.

- **Empty Processes** To improve overall system performance, Android often retains applications in memory after they have reached the end of their lifetimes. Android maintains this cache to improve the start-up time of applications when they’re re-launched. These processes are routinely killed as required.

Onde ficam os recursos na App Android….



Sete tipos de Recursos

- There are seven primary resource types that have different folders:
  - simple values,
  - drawables,
  - layouts,
  - Animations,
  - XML
  - Styles,
  - raw resources

Simple Values

- Within each XML file, you indicate the type of value being stored using tags as shown in the sample XML file below:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">To Do List</string>
  <color name="app_background">#FF0000FF</color>
  <dimen name="default_border">5px</dimen>
  <array name="string_array">
    <item>Item 1</item>
    <item>Item 2</item>
    <item>Item 3</item>
  </array>
  <array name="integer_array">
    <item>3</item>
    <item>2</item>
    <item>1</item>
  </array>
</resources>
```

Como usar recursos simples no código

- Outros recursos Simples que podem ser adicionados são: Cores e Dimensões.

- Cores podem ser definidas em RGB.,RRGGBB,ARGB, ARRGGBB . (A=Alpha, define a transparência). Por exemplo:

```
<color name="opaque_blue">#00F</color>
<color name="transparent_green">#7700FF00</color>
```

- Dimensões podem ser definidas em px, in (inches) , Physical Points (pt) entre outros. Por exemplo:

```
<dimen name="standard_border">5px</dimen>
<dimen name="large_font_size">16sp</dimen>
```

Styles

- Style resources let your applications maintain a consistent look and feel by specifying the attribute values used by Views. The most common use of themes and styles is to store the colors and fonts for an application.
- You can easily change the appearance of your application by simply specifying a different style as the theme in your project manifest

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="BaseText">
    <item name="android:textSize">14sp</item>
    <item name="android:textColor">#111</item>
  </style>
  <style name="SmallText" parent="BaseText">
    <item name="android:textSize">8sp</item>
  </style>
</resources>
```

Figuras ou Drawables

- O formato preferível é o PNG, mas JPG e GIF também são suportados

- Utiliza-se ainda o formato NinePatch que são arquivos PNG que marcam as partes de uma figura que podem ser aumentadas com uma linha preta de fronteira de largura de um pixel em volta das partes da figura que são aumentáveis.

Layout

- Recursos de Layout permitem a separação entre a apresentação da sua aplicação e o restante do código. O layout é escrito em XML e não em Java!

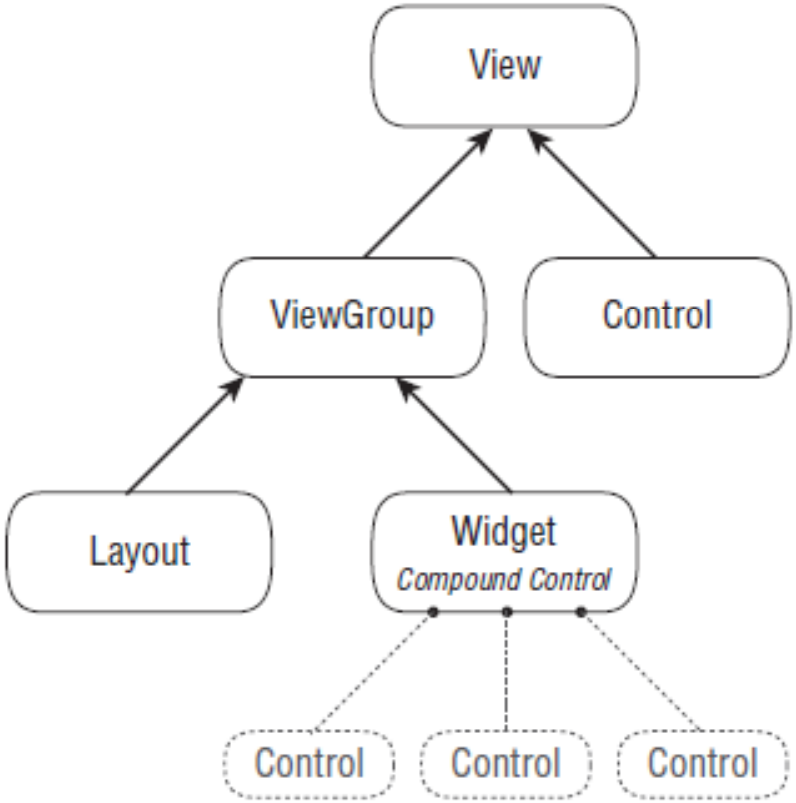
- Após ser definido em XML o layout é utilizado, ou “inflated” dentro de uma Activity usando o setContentView

Layout - 2

- Há Wizards no plugin ADT para facilitar a criação de layouts e os veremos em detalhes, juntamente com as Views que são os controles que populam os Layouts, mas a frente.
- Exemplo de Layout do Hello World!

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Hello World!"
  />
</LinearLayout>
```

Layout e outras classes relevantes na UI



Animações

- Há dois tipos de animações em Android: **tweened** animations e **frame-by-frame** animations
  - **Tweened animations** criados a partir de transformações sobre uma View. Transformações= Rodar, aumentar,reduzir, mover ou apagar (fade)
  - Frame-by-Frame Animations: permitem que você crie uma seqüência de imagens (drawable), cada uma será mostrada por um tempo específico no pano de fundo de uma View

```
<animation-list
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:oneshot="false">
  <item android:drawable="@drawable/rocket1" android:duration="500" />
  <item android:drawable="@drawable/rocket2" android:duration="500" />
  <item android:drawable="@drawable/rocket3" android:duration="500" />
</animation-list>
```

Acesso a Recursos no código Java

- Você pode acessar os recursos utilizando uma classe estática chamada R. Esta classe é gerada automaticamente pelo plugin ADT no eclipse. Esta classe R é gerada pelo compilador, por isso modificações manuais serão perdidas sempre que for gerada novamente.

- A classe R contém sub classes para cada tipo de recurso para o qual você definiu pelo menos um recurso.

- Exemplo:

```
// Inflate a layout resource.
setContentView(R.layout.main);
// Display a transient dialog box that displays the
// error message string resource.
Toast.makeText(this, R.string.app_error, Toast.LENGTH_LONG).show();
```

A classe Resources

- A classe Resources é utilizada quando é necessário obter referências (objetos) de recursos. Exemplo:

```
Resources myResources = getResources();
CharSequence styledText = myResources.getText(R.string.stop_message);
Drawable icon = myResources.getDrawable(R.drawable.app_icon);

int opaqueBlue = myResources.getColor(R.color.opaque_blue);

float borderWidth = myResources.getDimension(R.dimen.standard_border);

Animation tranOut;
tranOut = AnimationUtils.loadAnimation(this, R.anim.spin_shrink_fade);

String[] stringArray;
stringArray = myResources.getStringArray(R.array.string_array);

int[] intArray = myResources.getIntArray(R.array.integer_array);
```

Recursos referenciando recursos

- Recursos também podem referenciar outros recursos diretamente no XML. Isto é feito através da notação @[packagename]. O Android assume que você referencia no mesmo pacote, então só necessáro referenciar o nome completo de um pacote, quando se está dentro de outro.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:padding="@dimen/standard_border">
  <EditText
    android:id="@+id/myEditText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/stop_message"
    android:textColor="@color/opaque_blue"
  />
</LinearLayout>
```

Referenciando Recursos de Sistema

- Para referenciar recursos do sistema android, utilize a classe android R no código e o package @android: nos próprios recursos em XML
- Java:

```
CharSequence httpError = getString(android.R.string.httpErrorBadUrl);
```

- XML:

```
<EditText
  android:id="@+id/myEditText"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="@android:string/httpErrorBadUrl"
  android:textColor="@android:color/darker_gray"
/>
```

Exercício: Criar recursos adicionais para o Exemplo do Hello World!

- Crie duas imagens com 16x16 pixels,:

- Torne-as recursos endereçáveis dentro do Eclipse.

- Crie um tema baseado (herdado) do tema Theme.Black padrão do Android, setando o valor para um tamanho de texto default

- Aplique o novo tema ao seu projeto no arquivo de manifesto.

Localização de Aplicativos Android

- O uso de recursos torna mais fácil a internacionalização de aplicativos, adaptação ao hardware e também o compartilhamento destes recursos
- Por exemplo, podem ser criadas pastas padrões para strings que serão guardadas em diferentes línguas

```
Project/
  res/
    values/
      strings.xml
    values-fr/
      strings.xml
    values-fr-rCA/
      strings.xml
```

- Seguindo a um padrão de nomeaçõa dos recursos o próprio sistema Android poderá escolher entres os recursos adequados sem que sua aplicação precise se preocupar com isso

Padrão de Qualificação de Recursos

- **Language** Using the lowercase two-letter ISO 639-1 language code (e.g., `en`)
- **Region** A lowercase "r" followed by the uppercase two-letter ISO 3166-1-alpha-2 language code (e.g., `rUS`, `rGB`)
- **Screen Orientation** One of `port` (portrait), `land` (landscape), or `square` (square)
- **Screen Pixel Density** Pixel density in dots per inch (dpi) (e.g., `92dpi`, `108dpi`)
- **Touchscreen Type** One of `notouch`, `stylus`, or `finger`
- **Keyboard Availability** Either of `keysexposed` or `keyshidden`
- **Keyboard Input Type** One of `nokeys`, `qwerty`, or `12key`
- **UI Navigation Type** One of `notouch`, `dpad`, `trackball`, or `wheel`
- **Screen Resolution** Screen resolution in pixels with the largest dimension first (e.g., `320x240`)

Exemplos:

- **Valid:**

```
drawable-en-rUS
drawable-en-keyshidden
drawable-land-notouch-nokeys-320x240
```

- **Invalid:**

```
drawable-rUS-en (out of order)
drawable-rUS-rUK (multiple values for a single qualifier)
```

Mudanças de Configurações

- Se você quiser que seu aplicativo Android seja avisado de mudanças de configuração, você deve informar isso no manifesto da sua activity e especificar as mudanças sobre as quais quer ser avisado (Listener Pattern). As mudanças possíveis são as seguintes:

- `orientation` The screen has been rotated between portrait and landscape.
- `keyboardHidden` The keyboard has been exposed or hidden.
- `fontScale` The user has changed the preferred font size.
- `locale` The user has chosen a different language setting.
- `keyboard` The type of keyboard has changed; for example, the phone may have a 12 keypad that flips out to reveal a full keyboard.
- `touchscreen` or `navigation` The type of keyboard or navigation method has changed. Neither of these events should normally happen.

Mudanças de Configurações - 2

- The following XML snippet shows an activity node declaring that it will handle changes in screen orientation and keyboard visibility:

- Você pode selecionar vários eventos usando o símbolo |

```
<activity android:name=".TodoList"
          android:label="@string/app_name"
          android:theme="@style/TodoTheme"
          android:configChanges="orientation|keyboard"/>
```

Mudanças de Configurações - 2

- Ao definir no manifesto que uma determinada Activity deve ser avisada sobre mudança de configuração. Ao ocorrer um evento do(s) tipo(s) indicado(s) será chamado o método onConfigurationChanged da Activity. Exemplo:

```
@Override
public void onConfigurationChanged(Configuration _newConfig) {
  super.onConfigurationChanged(_newConfig);

  [ ... Update any UI based on resource values ... ]

  if (_newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
    [ ... React to different orientation ... ]
  }

  if (_newConfig.keyboardHidden == Configuration.KEYBOARDHIDDEN_NO) {
    [ ... React to changed keyboard visibility ... ]
  }
}
```

Trabalhando com Activities

- Each Activity represents a screen (similar to the concept of a Form in desktop development) that an application can present to its users. The more complicated your application, the more screens you are likely to need.

- Uma Activity será composta por uma classe e uma entrada no arquivo de manifesto. Sem a entrada no manifesto, jamais será iniciada.

Uma Activity sem Views

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

public class MyActivity extends Activity {

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
  }
}
```

Adicionando View a uma Activity

- Via código Java

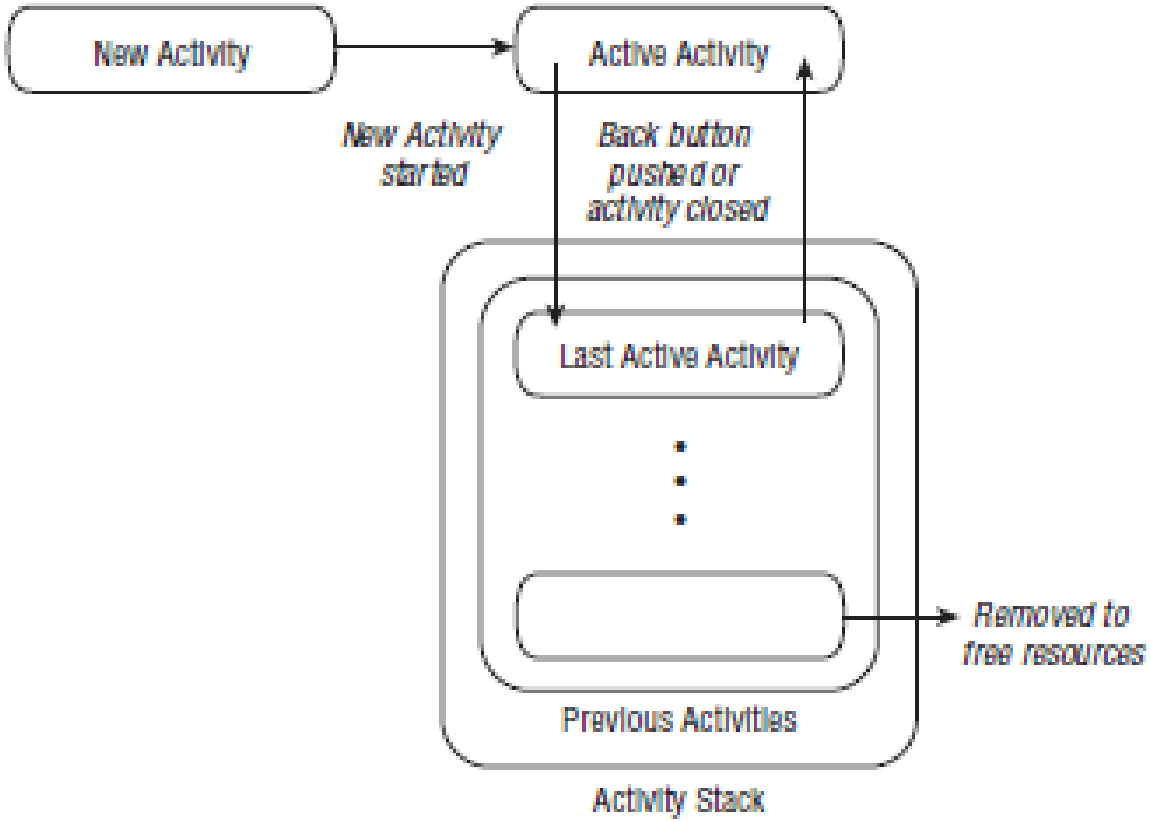
```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  MyView myView = new MyView(this);
  setContentView(myView);
}
```

- Via recursos (codificada em XML)

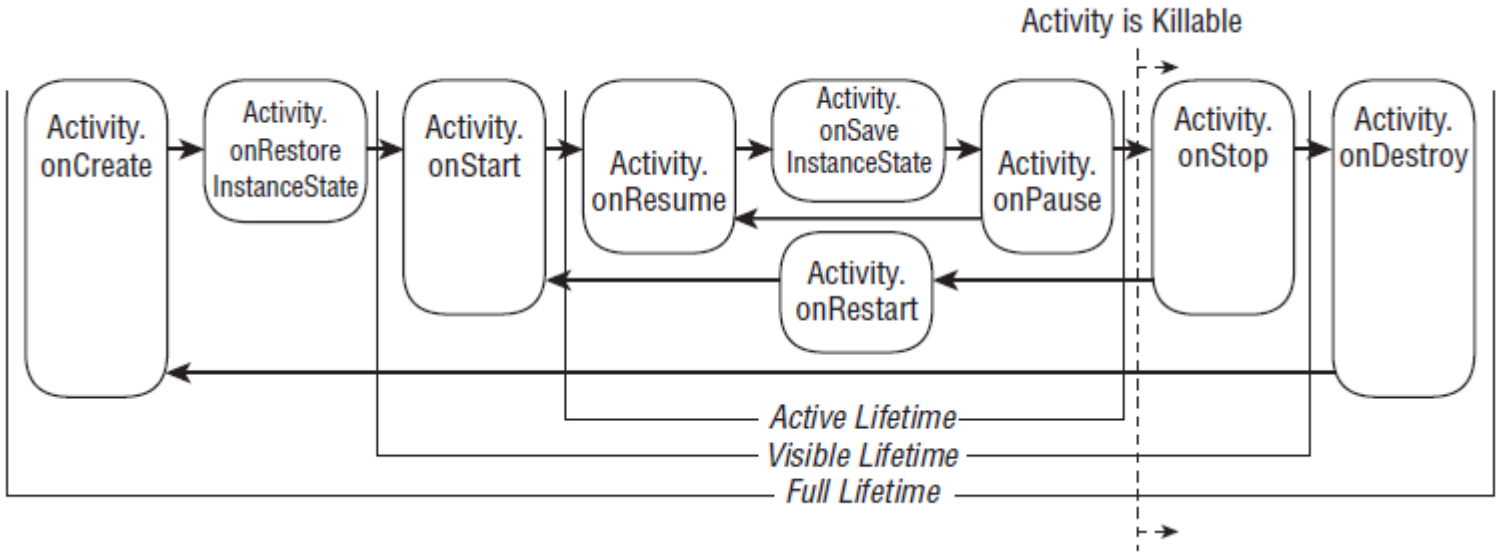
```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  setContentView(R.layout.main);
}
```

- Antes de tratar dos vários tipos de View e como criá-las vamos falar do ciclo de vida das Activities

Activity Stacks



Estados de um Activity



Esqueleto de código para Activity

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

public class MyActivity extends Activity {

  // Called at the start of the full lifetime.
  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    // Initialize activity.
  }

  // Called after onCreate has finished, use to restore UI state
  @Override
  public void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    // Restore UI state from the savedInstanceState.
    // This bundle has also been passed to onCreate.
  }

  // Called before subsequent visible lifetimes
  // for an activity process.
  @Override
  public void onRestart(){
    super.onRestart();
    // Load changes knowing that the activity has already
    // been visible within this process.
  }
```

Esqueleto de código para Activity - 2

```
// Called at the start of the visible lifetime.
@Override
public void onStart(){
  super.onStart();
  // Apply any required UI change now that the Activity is visible.
}

// Called at the start of the active lifetime.
@Override
public void onResume(){
  super.onResume();
  // Resume any paused UI updates, threads, or processes required
  // by the activity but suspended when it was inactive.
}

// Called to save UI state changes at the
// end of the active lifecycle.
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
  // Save UI state changes to the savedInstanceState.
  // This bundle will be passed to onCreate if the process is
  // killed and restarted.
  super.onSaveInstanceState(savedInstanceState);
}
```

Esqueleto de código para Activity - 3

```
  // Called at the end of the active lifetime.
  @Override
  public void onPause(){
    // Suspend UI updates, threads, or CPU intensive processes
    // that don't need to be updated when the Activity isn't
    // the active foreground activity.
    super.onPause();
  }

  // Called at the end of the visible lifetime.
  @Override
  public void onStop(){
    // Suspend remaining UI updates, threads, or processing
    // that aren't required when the Activity isn't visible.
    // Persist all edits or state changes
    // as after this call the process is likely to be killed.
    super.onStop();
  }

  // Called at the end of the full lifetime.
  @Override
  public void onDestroy(){
    // Clean up any resources including ending threads,
    // closing database connections etc.
    super.onDestroy();
  }
}
```

Fases da Vida de uma Activity

- The full lifetime of your Activity occurs between the fi rst call to onCreate and the fi nal call to onDestroy. It's possible, in some cases, for an Activity's process to be terminated *without the* onDestroy method being called.
- Visible Lifetime
  - An Activity's visible lifetimes are bound between calls to onStart and onStop. Between these calls, your Activity will be visible to the user, although it may not have focus and might be partially obscured. Activities are likely to go through several visible lifetimes during their full lifetime, as they move between the foreground and background. hile unusual, in extreme cases, the Android run time will kill an Activity during its visible lifetime without a call to onStop.

Fases da Vida de uma Activity - 2

- Active Lifetime
  - The active lifetime starts with a call to onResume and ends with a corresponding call to onPause.
  - An active Activity is in the foreground and is receiving user input events. Your Activity is likely to go through several active lifetimes before it's destroyed, as the active lifetime will end when a new Activity is displayed, the device goes to sleep, or the Activity loses focus.
  - Try to keep code in the onPause and onResume methods relatively fast and lightweight to ensure that your application remains responsive when moving in and out of the foreground.

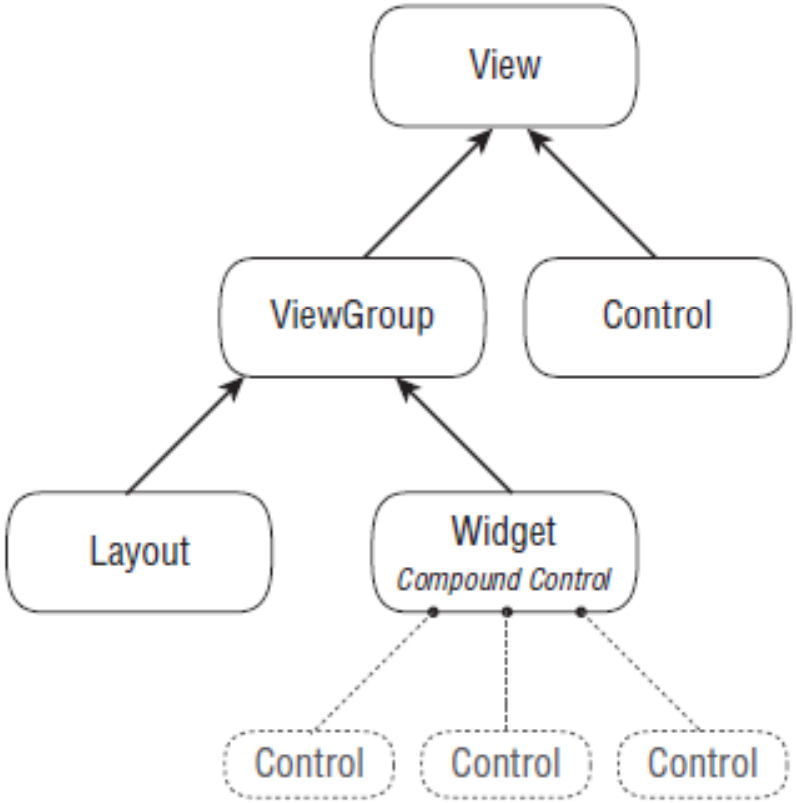
Sumário

- Introdução a Plataforma Android
- Primeiros Programas
- Criando Aplicações e Activities
- **Criação de Interfaces de usuário**
- Conexão a redes e outros programas
- Armazenamento e compartilhamento de dados
- Mapas, geolocalização e Serviços Baseados em Localização (Location-based Services)
- Multithreading
- Comunicação ponto-a-ponto
- Mecanismos de Acesso ao Hardware
- Desenvolvimento avançado em Android

Interfaces de Usuário em Android

- Os principais componentes de uma Interface de usuário em Android são as seguintes:
  - **Views** Views are the basic User Interface class for visual interface elements (commonly known as controls or widgets). All User Interface controls, and the layout classes, are derived from Views.
  - **ViewGroups** V*iew Groups are extensions of the View class that can contain multiple child*Views. By extending the ViewGroup class, you can create compound controls that are made up of interconnected child Views.
  - **Activities** *Activities represent the window or* screen being displayed to the user. Activities are the Android equivalent of a Form. To display a User Interface, you assign a View or layout to an Activity.

Classes nas UI



Criando UI com Activity e Views

- Opção 1:

- Opção 2:

```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);

  TextView myTextView = new TextView(this);
  setContentView(myTextView);
  myTextView.setText("Hello, Android");
}
```

A caixa de ferramentas para UI Android: Android widget toolbox

- Alguns dos controles mais comuns em UI Android são os seguintes (Controles similares podem ser encontrados em praticamente qualquer outro sistema de interface com usuário):
  - **TextView** A standard read only text label. It supports multiline display, string formatting, and automatic word wrapping.
  - **EditText** An editable text entry box. It accepts multiline entry and word wrapping.
  - **ListView** A View Group that creates and manages a group of Views used to display the items in a List. The standard ListView displays the string value of an array of objects using a Text View for each item.
  - **Spinner Composite** control that displays a TextView and an associated ListView that lets you select an item from a list to display in the textbox. It's made from a Text View displaying the current selection, combined with a button that displays a selection dialog when pressed.
  - **Button** Standard push-button
  - **CheckBox** Two-state button represented with a checked or unchecked box
  - **RadioButton** Two-state grouped buttons. Presents the user with a number of binary options of which only one can be selected at a time

Layouts e Layout Managers

Exemplo simples de Layout em XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Enter Text Below"
  />
  <EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Text Goes Here!"
  />
</LinearLayout>
```

- Especifica um TextView, um EditText em um layout linear e configurado para ser vertical.

Exemplo simples de Layout em Java

```
LinearLayout ll = new LinearLayout(this);
ll.setOrientation(LinearLayout.VERTICAL);

TextView myTextView = new TextView(this);
EditText myEditText = new EditText(this);

myTextView.setText("Enter Text Below");
myEditText.setText("Text Goes Here!");

int lHeight = LinearLayout.LayoutParams.FILL_PARENT;
int lWidth = LinearLayout.LayoutParams.WRAP_CONTENT;

ll.addView(myTextView, new LinearLayout.LayoutParams(lHeight, lWidth));
ll.addView(myEditText, new LinearLayout.LayoutParams(lHeight, lWidth));
setContentView(ll);
```

Modificando Views existentes - Herança

```
import android.content.Context;
import android.util.AttributeSet;
import android.widget.TextView;

public class MyTextView extends TextView {

  public MyTextView (Context context, AttributeSet attrs, int defStyle)
  {
    super(context, attrs, defStyle);
  }

  public MyTextView (Context context) {
    super(context);
  }

  public MyTextView (Context context, AttributeSet attrs) {
    super(context, attrs);
  }
}
```

Alterando o comportamento da View

```
public class MyTextView extends TextView {
    .       .       .
  @Override
  public void onDraw(Canvas canvas) {
    [ ... Draw things on the canvas under the text ... ]

    // Render the text as usual using the TextView base class.
    super.onDraw(canvas);

    [ ... Draw things on the canvas over the text ... ]
  }

  @Override
  public boolean onKeyDown(int keyCode, KeyEvent keyEvent) {
    [ ... Perform some special processing ... ]
    [ ... based on a particular key press ... ]

    // Use the existing functionality implemented by
    // the base class to respond to a key press event.
    return super.onKeyDown(keyCode, keyEvent);
  }
}
```

Controles Compostos

- *Compound controls (Controles Compostos) are atomic, reusable widgets that contain multiple child controls laid out and wired* together.

- Geralmente, controles compostos são herdados a partir de um Layout(que é um controle composto ou ViewGroup). Ex.:

```
public class MyCompoundView extends LinearLayout {

  public MyCompoundView(Context context) {
    super(context);
  }

  public MyCompoundView(Context context, AttributeSet attrs) {
    super(context, attrs);
  }

}
```

Controles Compostos -2

- Também se pode criar controles compostos em XML, através do layout resource. Ex.:

Exemplo de Controle Composto

```
public class ClearableEditText extends LinearLayout {
EditText editText;
Button clearButton;
public ClearableEditText(Context context) {
super(context);
// Set orientation of layout to vertical
setOrientation(LinearLayout.VERTICAL);
// Create the child controls.
editText = new EditText(getContext());
clearButton = new Button(getContext());
clearButton.setText(“Clear”);
// Lay them out in the compound control.
int lHeight = LayoutParams.WRAP_CONTENT;
int lWidth = LayoutParams.FILL_PARENT;
addView(editText, new LinearLayout.LayoutParams(lWidth, lHeight));
addView(clearButton, new LinearLayout.LayoutParams(lWidth, lHeight));
// Hook up the functionality
private void hookupButton() {
clearButton.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v) {
editText.setText(“”);
}
});}
```

Criando uma Nova View

Criando uma Nova View - 2

```
  private int measureWidth(int measureSpec) {
    int specMode = MeasureSpec.getMode(measureSpec);
    int specSize = MeasureSpec.getSize(measureSpec);

     [ ... Calculate the view width ... ]

     return specSize;
  }

  @Override
  protected void onDraw(Canvas canvas) {
    [ ... Draw your visual interface ... ]
  }
}
```

Desenhando um Controle

- The onDraw method is where the magic happens. If you're creating a new widget from scratch, it's because you want to create a completely new visual interface. The Canvas parameter available in the onDraw method is the surface you'll use to bring your imagination to life.

- Android provides a variety of tools to help draw your design on the Canvas using various Paint objects.

- The Canvas class includes helper methods to draw primitive 2D objects including circles, lines, rectangles, text, and Drawables (images).

- It also supports transformations that let you rotate, translate (move), and scale (resize) the Canvas while you draw on it.

Desenhando um Controle - 2

- The following code snippet shows how to override the onDraw method to display a simple text string in the center of the control

```
@Override
protected void onDraw(Canvas canvas) {
  // Get the size of the control based on the last call to onMeasure.
  int height = getMeasuredHeight();
  int width = getMeasuredWidth();

  // Find the center
  int px = width/2;
  int py = height/2;

  // Create the new paint brushes.
  // NOTE: For efficiency this should be done in
  // the widget's constructor
  Paint mTextPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
  mTextPaint.setColor(Color.WHITE);

  // Define the string.
  String displayText = "Hello World!";

  // Measure the width of the text string.
  float textWidth = mTextPaint.measureText(displayText);

  // Draw the text string in the center of the control.
  canvas.drawText(displayText, px-textWidth/2, py, mTextPaint);
}
```

Redimensionando um Controle

- Unless you conveniently require a control that always occupies 100 × 100 pixels, you will also need to override onMeasure.

- The onMeasure method is called when the control's parent is laying out its child controls. It asks the question, "How much space will you use?" and passes in two parameters — widthMeasureSpec and heightMeasureSpec. They specify the space available for the control and some metadata describing that space.

Exemplo de Redimensionamento de Controle - onMeasure

```
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
  int measuredHeight = measureHeight(heightMeasureSpec);

   int measuredWidth = measureWidth(widthMeasureSpec);

   setMeasuredDimension(measuredHeight, measuredWidth);
 }

 private int measureHeight(int measureSpec) {
   int specMode = MeasureSpec.getMode(measureSpec);
   int specSize = MeasureSpec.getSize(measureSpec);

   //  Default size if no limits are specified.
   int result = 500;

   if (specMode == MeasureSpec.AT_MOST) {
     // Calculate the ideal size of your
     // control within this maximum size.
     // If your control fills the available
     // space return the outer bound.
     result = specSize;
   } else if (specMode == MeasureSpec.EXACTLY) {
     // If your control can fit within these bounds return that value.
     result = specSize;
   }
   return result;
 }
```

Exemplo de Redimensionamento de Controle – onMeasure - 2

```
private int measureWidth(int measureSpec) {
  int specMode = MeasureSpec.getMode(measureSpec);
  int specSize = MeasureSpec.getSize(measureSpec);

  //  Default size if no limits are specified.
  int result = 500;

  if (specMode == MeasureSpec.AT_MOST) {
    // Calculate the ideal size of your control
    // within this maximum size.
    // If your control fills the available space
    // return the outer bound.
    result = specSize;
  } else if (specMode == MeasureSpec.EXACTLY) {
    // If your control can fit within these bounds return that value.
    result = specSize;
  }
  return result;
}
```

Respondendo a Ações do Usuário

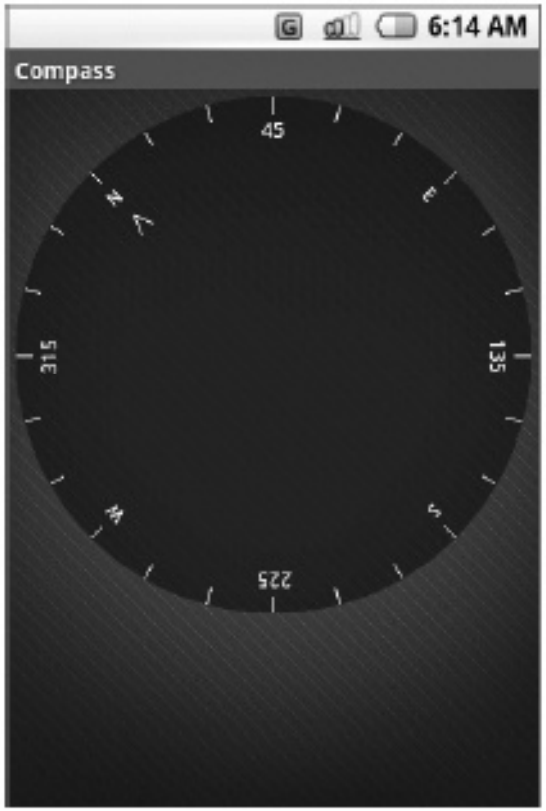
- To make your new widget interactive, it will need to respond to user events like key presses, screen touches, and button clicks.

- Android exposes several virtual event handlers, listed below, that let you react to user input:
  - onKeyDown Called when any device key is pressed; includes the D-pad, keyboard, hang-up, call, back, and camera buttons
  - onKeyUp Called when a user releases a pressed key
  - onTrackballEvent Called when the device's trackball is moved
  - onTouchEvent Called when the touch screen is pressed or released, or it detects movement

Como tratar os eventos?

- Os eventos são direcionados para a View ativa e são chamados os seguintes métodos:

Exemplo 3: Criando uma Bússola



Adicionando um Controle

- O uso de uma View (ou controle) criado por você é feito da mesma forma que se usa os controle previamente existentes.
- Através de código Java puro:

```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  CompassView cv = new CompassView(this);
  setContentView(cv);
  cv.setBearing(45);
 }
```

- Através de recursos em XML e referência ao recurso em Java

Execute o Projeto Compass

- Analise o código do projeto nas classes Compass e CompassView
- Uso da Classe android.graphics.Canvas e android.graphics.Paint (estilo)
- MeasureSpec em onMeasure trazem informações de modo e tamanho das dimensões de um controle

```
public class CompassView extends View {

  // Paints used to draw the Compass
  private Paint markerPaint;
  private Paint textPaint;
  private Paint circlePaint;

  // Cardinal point Strings
  private String northString;
  private String eastString;
  private String southString;
  private String westString;

  // Height of text
  private int textHeight;

  /** Get or set the bearing displayed by the compass **/
  public void setBearing(float _bearing) {
    bearing = _bearing;
  }
  public float getBearing() {
    return bearing;
  }
  private float bearing;

  /** Constructors **/
  public CompassView(Context context) {
    super(context);
    initCompassView();
  }
```

CompassView.java

```
public CompassView(Context context, AttributeSet attrs) {
    super(context, attrs);
    initCompassView();
  }
  public CompassView(Context context, AttributeSet attrs, int defaultStyle) {
    super(context, attrs, defaultStyle);
    initCompassView();
  }
  /** Initialize the Class variables **/
  protected void initCompassView() {
    setFocusable(true);
    // Get a reference to the external resources
    Resources r = this.getResources();
    northString = r.getString(R.string.cardinal_north);
    eastString = r.getString(R.string.cardinal_east);
    southString = r.getString(R.string.cardinal_south);
    westString = r.getString(R.string.cardinal_west);

    // Create the paints
    circlePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    circlePaint.setColor(R.color.background_color);
    circlePaint.setStrokeWidth(1);
    circlePaint.setStyle(Paint.Style.FILL_AND_STROKE);

    markerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    markerPaint.setColor(r.getColor(R.color.marker_color));

    textPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    textPaint.setColor(r.getColor(R.color.text_color));

    textHeight = (int)textPaint.measureText("yY");
  }
```

CompassView.java

```
@Override
 protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
  // The compass is a circle that fills as much space as possible.
  // Set the measured dimensions by figuring out the shortest boundary,
  // height or width.
  int measuredWidth = measure(widthMeasureSpec);
  int measuredHeight = measure(heightMeasureSpec);

  int d = Math.min(measuredWidth, measuredHeight);

  setMeasuredDimension(d, d);
 }

 private int measure(int measureSpec) {
  int result = 0;

  // Decode the measurement specifications.
  int specMode = MeasureSpec.getMode(measureSpec);
  int specSize = MeasureSpec.getSize(measureSpec);

  if (specMode == MeasureSpec.UNSPECIFIED) {
   // Return a default size of 200 if no bounds are specified.
   result = 200;
  } else {
   // As you want to fill the available space
   // always return the full available bounds.
   result = specSize;
  }
  return result;
 }
```

CompassView.java

```
@Override
 protected void onDraw(Canvas canvas) {
  int px = getMeasuredWidth() / 2;
  int py = getMeasuredHeight() / 2;            int radius = Math.min(px, py);

  // Draw the background
  canvas.drawCircle(px, py, radius, circlePaint);
     // Rotate our perspective so that the 'top' is facing the current bearing.
  canvas.save();
  canvas.rotate(-bearing, px, py);
  int textWidth = (int)textPaint.measureText("W");
  int cardinalX = px-textWidth/2;
  int cardinalY = py-radius+textHeight;

  // Draw the marker every 15 degrees and a text every 45.
  for (int i = 0; i < 24; i++) {
   // Draw a marker.
   canvas.drawLine(px, py-radius, px, py-radius+10, markerPaint);
   canvas.save();
   canvas.translate(0, textHeight);
   // Draw the cardinal points
   if (i % 6 == 0) {
    String dirString = "";
    switch (i) {
     case(0)  : {
                dirString = northString;
                int arrowY = 2*textHeight;
                canvas.drawLine(px, arrowY, px-5, 3*textHeight, markerPaint);
                canvas.drawLine(px, arrowY, px+5, 3*textHeight, markerPaint);
                break;
               }
     case(6)  : dirString = eastString; break;
     case(12) : dirString = southString; break;
     case(18) : dirString = westString; break;
    }
    canvas.drawText(dirString, cardinalX, cardinalY, textPaint);
   }
   else if (i % 3 == 0) {
    // Draw the text every alternate 45deg
```

CompassView.java

```
    else if (i % 3 == 0) {
      // Draw the text every alternate 45deg
      String angle = String.valueOf(i*15);
      float angleTextWidth = textPaint.measureText(angle);

      int angleTextX = (int)(px-angleTextWidth/2);
      int angleTextY = py-radius+textHeight;
      canvas.drawText(angle, angleTextX, angleTextY, textPaint);
    }
    canvas.restore();
    canvas.rotate(15, px, py);
   }

   canvas.restore();
  }  // end of method onDraw
} //end of the class CompassView
```

CompassView.java

Exercício :

- Altere o projeto Compass para que este responda a toques na tela e a toques de tela atualizando sua interface gráfica.
  - Ao tocar a tela deve-se avançar o “bearing” mais 5 graus, até chegar ao limite de 360o. Após isso, deve-se inverter o sentido do movimento
  - Escolha uma tecla (KEYCODE_ENTER) para movimento horário e outra tecla para movimento anti-horário (KEYCODE_DEL) e faça a aplicação movimentar a bússola no sentido indicado

Sumário

- Introdução a Plataforma Android
- Primeiros Programas
- Criando Aplicações e Activities
- Criação de Interfaces de usuário
- **Conexão a redes e outros programas**
- Mecanismos de Acesso ao Sensores
- Services (e Multithreading)
- Mapas, geolocalização e Serviços Baseados em Localização (Location-based Services)
- Instalação de Aplicativos Android em Dispositivos

Intents: Comunicação entre Aplicativos e Activities Android e o próprio Android

- *Intents are used as a message-passing mechanism that lets you declare your intention that an action be* performed, usually with (or on) a particular piece of data.
- You can use Intents to support interaction between any of the application components available on an Android device, no matter which application they're part of. This turns a collection of independent components into a single, interconnected system.

- One of the most common uses for Intents is to **start new Activities**, either ***explicitly*** *(by specifying the* class to load) or ***implicitly*** *(by requesting an action be performed on a piece of data).*

Intents: Comunicação entre Aplicativos Android e o próprio Android - 2

- Intents can also be used to **broadcast messages** across the system. Any application can register a **Broadcast Receiver to listen for, and react to, these broadcast Intents.** This lets you create event-driven applications based on internal, system, or third-party application events.

- Android uses broadcast Intents to announce **system events**, like changes in Internet connection status or battery charge levels. The native Android applications, such as the phone dialer and SMS manager, simply **register components** that listen for specific broadcast Intents — such as “incoming phone call” or “SMS message received” — and react accordingly.

- Using Intents to propagate actions — even within the same application — is a fundamental Android design principle. It encourages the decoupling of components, to allow the seamless replacement of application elements. It also provides the basis of a simple model for extending functionality.

Usando Intents para Invocar Activities

- Pode-se utilizar Intents para invocar Activities que você não conhece!!!
  - Implicit Intents são similares a idéia de Web Services ou páginas amarelas em sistemas multiagentes.
  - A busca é feita por serviço (página amarela) e não diretamente por um componente

Implicit Intents: Buscando por “serviços”

- Por exemplo, para comandar a chamada de uma Activity que preste o serviço de discagem você poderia usar um Implicit Intent:

```
Intent intent = new Intent(Intent.ACTION_DIAL,
                           Uri.parse("tel:555-2368"));
startActivity(intent);
```

- Ao iniciar uma Activity com startAcitivity você cria uma nova Activity totalmente indenpendente da primeira e fica impossibilitado de retornar resultados da segunda activity para a primeira.
- Para criar subactivities, deve-se utilizar startActivityForResult indicando um intent e um identificador único para a subactivity

```
Intent intent = new Intent(this, MyOtherActivity.class);
startActivityForResult(intent, SHOW_SUBACTIVITY);
```

Subactivities

As with regular Activities, sub-Activities can be started implicitly or explicitly. The following skeleton code uses an implicit Intent to launch a new sub-Activity to pick a contact:

```
private static final int PICK_CONTACT_SUBACTIVITY = 2;

Uri uri = Uri.parse("content://contacts/people");
Intent intent = new Intent(Intent.ACTION_PICK, uri);
startActivityForResult(intent, PICK_CONTACT_SUBACTIVITY);
```

- É possível retornar resultados com setResult, antes de finalizar sua subactivity defina o resultado com setResult passando o código do resultado (geralmente, Activity.RESULT_OK or Activity.RESULT_CANCELED.)e um Intent que pode ser usado para passar informação adicional de volta

Retornando Resultados de Subactivity

- Trecho de código de uma subactivity que retorna resultados. O Intent funciona como um

```
Button okButton = (Button) findViewById(R.id.ok_button);
  okButton.setOnClickListener(new View.OnClickListener() {

   public void onClick(View view) {
     Uri data = Uri.parse("content://horses/" + selected_horse_id);

     Intent result = new Intent(null, data);
     result.putExtra(IS_INPUT_CORRECT, inputCorrect);
     result.putExtra(SELECTED_PISTOL, selectedPistol);
     setResult(RESULT_OK, result);

     finish();
   }
});

Button cancelButton = (Button) findViewById(R.id.cancel_button);
cancelButton.setOnClickListener(new View.OnClickListener() {

  public void onClick(View view) {
    setResult(RESULT_CANCELED, null);

    finish();
  }
});
```

Tratando o retorno de Subacitivities na Activity "principal"

- Quando uma sub-Activity é fechada, sua Activity pai tem o método onActivityResult chamado. Para receber a informação basta sobrescrever tal método. Os parâmetros do método onAcitivityResult são os seguintes:

  - **The Request Code** The request code that was used to launch the returning sub-Activity
  - **A Result Code** The result code set by the sub-Activity to indicate its result. It can be any integer value, but typically will be either `Activity.RESULT_OK` or `Activity.RESULT_CANCELLED`.

  If the sub-Activity closes abnormally or doesn't specify a result code before it closes, the result code is `Activity.RESULT_CANCELED`.

  - **Data** An Intent used to bundle any returned data. Depending on the purpose of the sub-Activity, it will typically include a URI that represents the particular piece of data selected from a list. Alternatively, or additionally, the sub-Activity can return extra information as primitive values using the "extras" mechanism.

Tratando o retorno de Subacitivities na Activity “principal” - 2

- Trecho de código da Activity “principal”:

```
private static final int SHOW_SUB_ACTIVITY_ONE = 1;
private static final int SHOW_SUB_ACTIVITY_TWO = 2;

@Override
public void onActivityResult(int requestCode,
                             int resultCode,
                             Intent data) {

  super.onActivityResult(requestCode, resultCode, data);

  switch(requestCode) {
    case (SHOW_SUB_ACTIVITY_ONE) : {
      if (resultCode == Activity.RESULT_OK) {
        Uri horse = data.getData();
        boolean inputCorrect = data.getBooleanExtra(IS_INPUT_CORRECT,
                                                    false);
        String selectedPistol = data.getStringExtra(SELECTED_PISTOL);
      }
      break;
    }
    case (SHOW_SUB_ACTIVITY_TWO) : {
      if (resultCode == Activity.RESULT_OK) {
        // TODO: Handle OK click.
      }
      break;
    }
  }
}
```

IntentFilters

- Como o Android sabe qual Activity chamar em um Implicit Intent: IntentFilters
- Intent Filters are used to register Activities, Services, and Broadcast Receivers as being capable of performing an action on a particular kind of data.
  - To register an application component as an Intent handler, use the intent-filter tag within the component's manifest node.

```
<activity android:name=".EarthquakeDamageViewer"
          android:label="View Damage">
  <intent-filter>
    <action
      android:name="com.paad.earthquake.intent.action.SHOW_DAMAGE">
    </action>
    <category android:name="android.intent.category.DEFAULT"/>
    <category
      android:name="android.intent.category.ALTERNATIVE_SELECTED"
    />
    <data android:mimeType="vnd.earthquake.cursor.item/*"/>
  </intent-filter>
</activity>
```

Broadcast Events and Receivers

- **Intents** can also be used to broadcast messages anonymously *between components with the* ***sendBroadcast*** *method.*

```
Intent intent = new Intent(NEW_LIFEFORM_DETECTED);
intent.putExtra("lifeformName", lifeformType);
intent.putExtra("longitude", currentLongitude);
intent.putExtra("latitude", currentLatitude);

sendBroadcast(intent);
```

- *You can implement* Broadcast Receivers to listen for, and respond to, these broadcast Intents within your applications. Example:

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

public class MyBroadcastReceiver extends BroadcastReceiver {

  @Override
  public void onReceive(Context context, Intent intent) {
    //TODO: React to the Intent received.
  }

}
```

- BroadcastReceivers também devem ser registrados no AndroidManifest. Ver referências para mais informações sobre Broadcast events e receivers

```
<receiver android:name=".LifeformDetectedBroadcastReceiver">
  <intent-filter>
    <action android:name="com.paad.action.NEW_LIFEFORM"/>
  </intent-filter>
</receiver>
```

Acesso a rede

Modern mobile devices offer various alternatives for accessing the Internet. Looked at broadly, Android provides three connection techniques for Internet connectivity. Each is offered transparently to the application layer.

- **GPRS, EDGE, and 3G** Mobile Internet access is available through carriers that offer mobile data plans.
- **Wi-Fi** Wi-Fi receivers and mobile hotspots are becoming increasingly more common.

- No momento da instalação do aplicativo, o usuário deve autorizar o aplicativo a ter acesso a Internet.
- O acesso a Internet utiliza classes de stream tal como visto no Java padrão

Acesso a rede

- Trecho de código que exemplifica um acesso a Internet

```
String myFeed = getString(R.string.my_feed);
try {
  URL url = new URL(myFeed);

  URLConnection connection = url.openConnection();
  HttpURLConnection httpConnection = (HttpURLConnection)connection;

  int responseCode = httpConnection.getResponseCode();
  if (responseCode == HttpURLConnection.HTTP_OK) {
    InputStream in = httpConnection.getInputStream();

    [ ... Process the input stream as required ... ]
  }
}
catch (MalformedURLException e) { }
catch (IOException e) { }
```

Sumário

- Introdução a Plataforma Android
- Primeiros Programas
- Criando Aplicações e Activities
- Criação de Interfaces de usuário
- Conexão a redes e outros programas
- **Mecanismos de Acesso aos Sensores**
- Services (e Multithreading)
- Mapas, geolocalização e Serviços Baseados em Localização (Location-based Services)
- Instalação de Aplicativos Android em Dispositivos

Sensor Manager e Sensor Listener

- The Sensor Manager is used to manage the sensor hardware available on an Android device. Use getSystemService to get a reference to the Sensor Service as shown in the code snippet below:

```
String service_name = Context.SENSOR_SERVICE;
SensorManager sensorManager = (SensorManager)getSystemService(service_name);
```

- The availability of compass and accelerometer values depends on the hardware upon which your application runs. When available, they are exposed through the SensorManager class, allowing you to:
  - Determine the current orientation of the hardware.
  - Monitor for changes in orientation.
  - Know which direction the user is facing.
  - Monitor acceleration — changes in movement speed — in any direction: vertically, laterally, orlongitudinally

Sensor Manager - 2

The Sensor Manager includes constants to help identify the sensor triggering the change event. The following list includes the sensors for which constants are currently defined. Some or all of these sensors will be available to your applications depending on the hardware available on the host device:

- `SensorManager.SENSOR_ACCELEROMETER` Is an accelerometer sensor that returns the current acceleration along three axes in meters per second squared (m/s^2). The accelerometer is explored in greater detail later in this chapter.
- `SensorManager.SENSOR_ORIENTATION` Is an orientation sensor that returns the current orientation on three axes in degrees. The orientation sensor is explored in greater detail later in this chapter.
- `SensorManager.SENSOR_LIGHT` Is an ambient-light sensor that returns a single value describing the ambient illumination in lux.
- `SensorManager.SENSOR_MAGNETIC_FIELD` Is a sensor used to determine the current magnetic field measured in microteslas (μT) along three axes.
- `SensorManager.SENSOR_PROXIMITY` Is a proximity sensor that returns a single value describing the distance between the device and the target object in meters (m).
- `SensorManager.SENSOR_TEMPERATURE` Is a thermometer sensor that returns the ambient temperature in degrees Celsius (°C).

Precisão do Sensor

Implement `onAccuracyChanged` to react to changes in a sensor's accuracy. The `sensor` parameter again identifies the sensor that triggered the event, while the `accuracy` parameter indicates the new accuracy of that sensor using one of the constants:

- `SensorManager.SENSOR_STATUS_ACCURACY_HIGH` Indicates that the sensor is reporting with the highest possible accuracy.
- `SensorManager.SENSOR_STATUS_ACCURACY_LOW` Indicates that the sensor is reporting with low accuracy and needs to be calibrated.
- `SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM` Indicates that the sensor data is of average accuracy, and that calibration might improve the readings.
- `SensorManager.SENSOR_STATUS_UNRELIABLE` Indicates that the sensor data is unreliable, meaning that either calibration is required or readings are not currently possible.

Percebendo mudanças via Sensores

- Utiliza-se o design pattern Observer/Observable.
  - Cria-se um SensorListener (papel de observer)
  - Registra-se o SensorListener junto ao SensorManager(observable)
  - Ao ocorrerem eventos o SensorManager encarrega-se de avisar ao SensorListener. Exemplo:

```
SensorListener mySensorListener = new SensorListener() {
  public void onSensorChanged(int sensor, float[] values) {
    // TODO Deal with sensor value changes
  }

  public void onAccuracyChanged(int sensor, int accuracy) {
    // TODO Auto-generated method stub
  }
};

 sensorManager.registerListener(mySensorListener,
                                SensorManager.SENSOR_TRICORDER,
                                SensorManager.SENSOR_DELAY_FASTEST);
```

SensorListener tornou-se deprecated

- Deve-se utilizar a partir da API level 3, a interface SensorEventListener em substituição a SensorListener. Esta contém porém basicamente os mesmos métodos da anterior com pequenas alterações:

```
public Interface SensorEventListener {
public abstract void onAccuracyChanged(Sensor sensor, int accuracy);
// Called when the accuracy of a sensor has changed.

public abstract void onSensorChanged(SensorEvent event)
// Called when sensor values have changed.

}
```

Exemplo de Uso com SensorEventListener

-
```
public class SensorActivity extends Activity, implements SensorEventListener {
    private final SensorManager mSensorManager;
    private final Sensor mAccelerometer;

    public SensorActivity() {
        mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
        mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }

    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
    }

    protected void onPause() {
        super.onPause();
        mSensorManager.unregisterListener(this);
    }

    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    public void onSensorChanged(SensorEvent event) {
    }
}
```

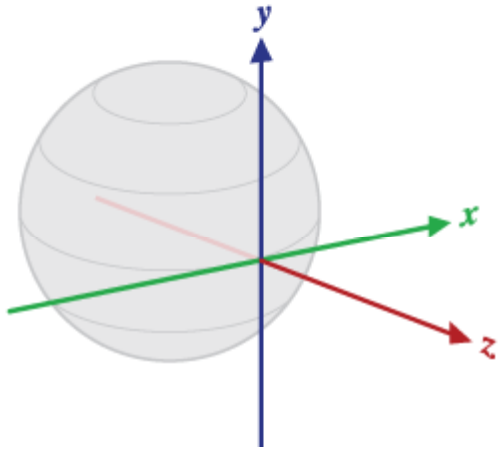
Sistema de Coordenadas usado pelo SensorManager

- Método estático disponível em SensorManager: getRotationMatrix

public static boolean **getRotationMatrix** (float[] R, float[] I, float[] gravity, float[] geomagnetic) Since: API Level 3

Computes the inclination matrix **I** as well as the rotation matrix **R** transforming a vector from the device coordinate system to the world's coordinate system which is defined as a direct orthonormal basis, where:

- X is defined as the vector product **Y.Z** (It is tangential to the ground at the device's current location and roughly points East).
- Y is tangential to the ground at the device's current location and points towards the magnetic North Pole.
- Z points towards the sky and is perpendicular to the ground.



Parameters

R	is an array of 9 floats holding the rotation matrix **R** when this function returns. R can be null.
I	is an array of 9 floats holding the rotation matrix **I** when this function returns. I can be null.
gravity	is an array of 3 floats containing the gravity vector expressed in the device's coordinate. You can simply use the `values` returned by a `SensorEvent` of a `Sensor` of type `TYPE_ACCELEROMETER`.
geomagnetic	is an array of 3 floats containing the geomagnetic vector expressed in the device's coordinate. You can simply use the `values` returned by a `SensorEvent` of a `Sensor` of type `TYPE_MAGNETIC_FIELD`.

Returns

`true` on success, `false` on failure (for instance, if the device is in free fall). On failure the output matrices are not modified.

Obtendo a Orientação do dispositivo:

- Método estático auxiliar getOrientation em SensorManager

public static float[] **getOrientation** (float[] R, float[] values) Since: API Level 3

Computes the device's orientation based on the rotation matrix.

When it returns, the array values is filled with the result:

- values[0]: *azimuth*, rotation around the Z axis.
- values[1]: *pitch*, rotation around the X axis.
- values[2]: *roll*, rotation around the Y axis.

The reference coordinate-system used is different from the world coordinate-system defined for the rotation matrix:

- X is defined as the vector product **Y.Z** (It is tangential to the ground at the device's current location and roughly points West).
- Y is tangential to the ground at the device's current location and points towards the magnetic North Pole.
- Z points towards the center of the Earth and is perpendicular to the ground.

All three angles above are in **radians** and **positive** in the **counter-clockwise** direction.

Parameters

R	rotation matrix see `getRotationMatrix(float[], float[], float[], float[])`.
values	an array of 3 floats to hold the result.

Sumário

- Introdução a Plataforma Android
- Primeiros Programas
- Criando Aplicações e Activities
- Criação de Interfaces de usuário
- Conexão a redes e outros programas
- Mecanismos de Acesso aos Sensores
- **Services (e Multithreading)**
- Mapas, geolocalização e Serviços Baseados em Localização (Location-based Services)
- Instalação de Aplicativos Android em Dispositivos

Services e Multithreading em Android

- A classe Service é utilizada para executar um serviço em segundo plano, geralmente vinculado a algum processo que deve executar por tempo indeterminado e tem um alto consumo de recursos, memória e CPU.

- O Android também conta com classes Thread e interface Runnable como o Java padrão

- Então por que utilizar a classe Service?

Service e Threads

- O motivo é simples: o Android conhece a classe Service e pode gerenciar seu ciclo de vida junto com os outros processos do sistema operacional. Se você utilizar uma thread, o Android pode eliminá-la porque não faz parte do ciclo devida conhecido do Android.

- A classe Service tem um ciclo de vida controlado e tem prioridade máxima sobre qualquer outro processo executando em segundo plano, que somente será encerrado em condições realmente críticas de memória.

- Para exemplificar, imagine que uma Activity iniciou uma nova thread e logo depois ela foi encerrada, o que pode ocorrer simplesmente se o usuário clicou no botão voltar. Nesse caso, no momento em que o Android for matar o processo da Activity, a thread corre um grande risco de ser finalizada também.

Definindo um Service

To define a Service, create a new class that extends the `Service` base class. You'll need to override `onBind` and `onCreate`, as shown in the following skeleton class:

```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

public class MyService extends Service {

  @Override
  public void onCreate() {

    // TODO: Actions to perform when service is created.
  }

  @Override
  public IBinder onBind(Intent intent) {
    // TODO: Replace with service binding implementation.
    return null;
  }
}
```

Once you've constructed a new Service, you have to register it in the application manifest.

Do this by including a `service` tag within the `application` node. You can use attributes on the `service` tag to enable or disable the Service and specify any permissions required to access it from other applications using a `requires-permission` flag.

Below is the `service` tag you'd add for the skeleton Service you created above:

```
<service android:enabled="true" android:name=".MyService"></service>
```

Criando e parando Services

To start a Service, call `startService`; you can either implicitly specify a Service to start using an action against which the Service is registered, or you can explicitly specify the Service using its class.

If the Service requires permissions that your application does not have, this call will throw a `SecurityException`. The snippet below demonstrates both techniques available for starting a Service:

```
// Implicitly start a Service
startService(new Intent(MyService.MY_ACTION));
// Explicitly start a Service
startService(new Intent(this, MyService.class));
```

To use this example, you would need to include a `MY_ACTION` *property in the* `MyService` *class and use an Intent Filter to register it as a provider of* `MY_ACTION`.

To stop a Service, use `stopService`, passing an Intent that defines the Service to stop. This next code snippet first starts and then stops a Service both explicitly and by using the component name returned when calling `startService`:

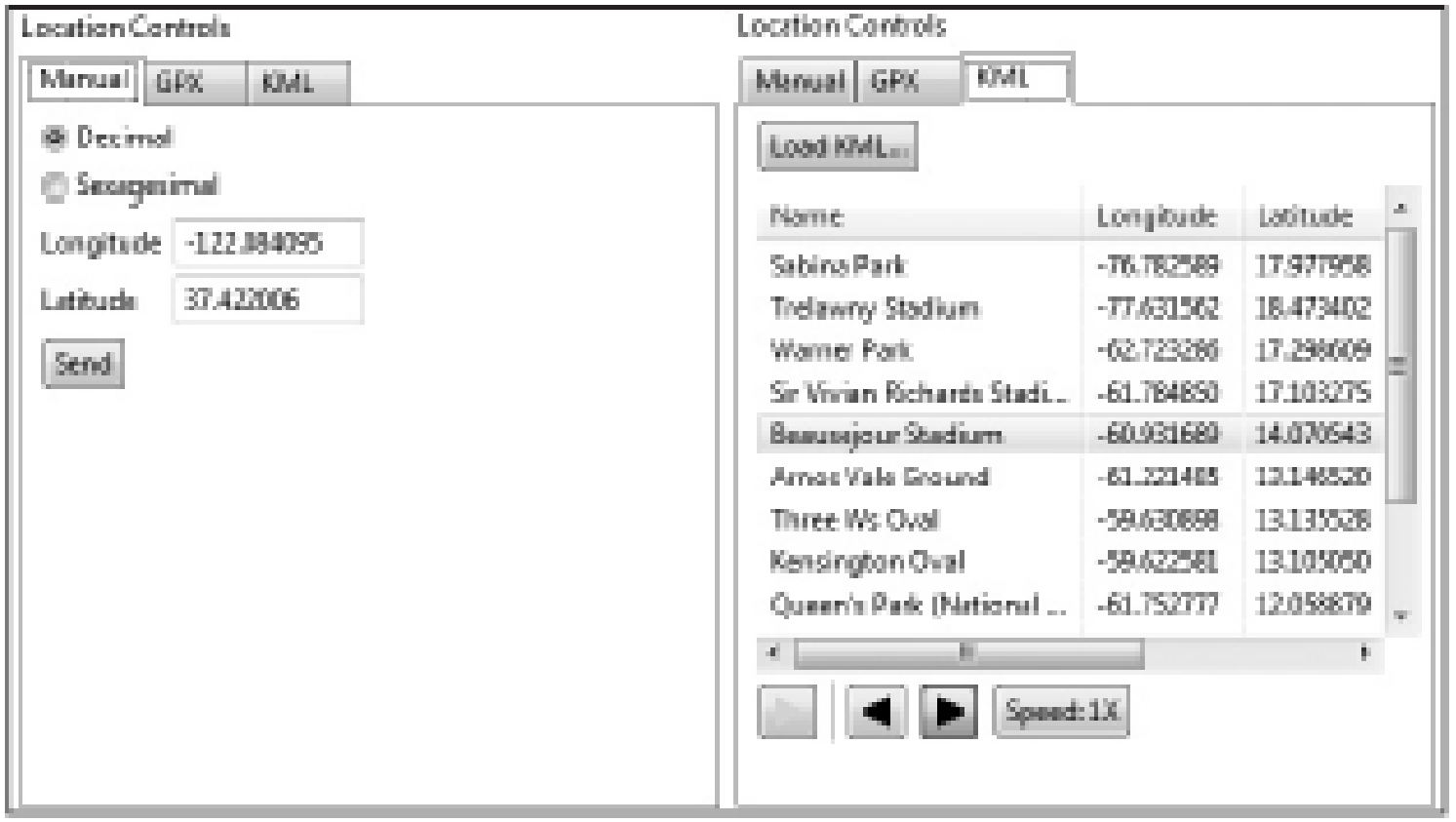
```
ComponentName service = startService(new Intent(this, BaseballWatch.class));
// Stop a service using the service name.
stopService(new Intent(this, service.getClass()));


// Stop a service explicitly.
try {
  Class serviceClass = Class.forName(service.getClassName());
  stopService(new Intent(this, serviceClass));
} catch (ClassNotFoundException e) {}
```

Sumário

- Introdução a Plataforma Android
- Primeiros Programas
- Criando Aplicações e Activities
- Criação de Interfaces de usuário
- Conexão a redes e outros programas
- Armazenamento e compartilhamento de dados
- Mecanismos de Acesso aos Sensores
- Services (e Multithreading)
- **Mapas, geolocalização e Serviços Baseados em Localização (Location-based Services)**
- Instalação de Aplicativos Android em Dispositivos

Emulando o GPS



- Acessível no Eclipse, Menu Window > Open Perspective > DDML

Exemplo de uso de Location Manager

```
LocationManager locationManager;

@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  setContentView(R.layout.main);

  String location_context = Context.LOCATION_SERVICE;
  locationManager = (LocationManager)getSystemService(location_context);
  testProviders();
}

public void testProviders() {}
```

- Para acessar o serviço de localização uma aplicação precisa de permissão, use no manifesto:

  <uses-permission android:name=*"android.permission.ACCESS_FINE_LOCATION"*/>
  <uses-permission android:name=*"android.permission.ACCESS_COARSE_LOCATION"*/>

- Agora vamos ver o método testProviders que obtém a posição dos Providers habilitados

Obtendo posição

```
public void testProviders() {
  TextView tv = (TextView)findViewById(R.id.myTextView);
  StringBuilder sb = new StringBuilder("Enabled Providers:");

  List<String> providers = locationManager.getProviders(true);

  for (String provider : providers) {

    locationManager.requestLocationUpdates(provider, 1000, 0,
                                           new LocationListener() {
      public void onLocationChanged(Location location) {}
      public void onProviderDisabled(String provider){}
      public void onProviderEnabled(String provider){}
      public void onStatusChanged(String provider, int status,
                                  Bundle extras){}
    });

    sb.append("\n").append(provider).append(": ");

    Location location = locationManager.getLastKnownLocation(provider);
    if (location != null) {
      double lat = location.getLatitude();
      double lng = location.getLongitude();
      sb.append(lat).append(", ").append(lng);
    } else {
      sb.append("No Location");
    }
  }
  tv.setText(sb);
}
```

Resultado (apresentado em um TextView identificado por R.id.myTextView)

- Descrição do TextView no Manifesto Android

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <TextView
    android:id="@+id/myTextView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
  />
</LinearLayout>
```

Sumário

- Introdução a Plataforma Android
- Primeiros Programas
- Criando Aplicações e Activities
- Criação de Interfaces de usuário
- Conexão a redes e outros programas
- Armazenamento e compartilhamento de dados
- Mecanismos de Acesso aos Sensores
- Services (e Multithreading)
- Mapas, geolocalização e Serviços Baseados em Localização (Location-based Services)
- **Instalação de Aplicativos Android em Dispositivos**

Instalações de Aplicativos Android em Dispositivos

- Após testar no seu Emulador você pode instalar sua aplicação em um dispositivo de três modos distintos
  - Android Market
  - Android SDK e cabo USB
  - Copiando aplicativo (.apk) através de um SD Card e instalando através de um programa Instalador (p.ex.: App Installer)

Installing Applications With Android SDK

- It is possible to install APK files without utilizing the Android Market, although the process is more difficult and complex. To avoid the Android Market, you need to use Android SDK.
- Download and install the Google Android SDK program and the Android USB drivers.

- You need to modify your Android's settings to allow the installation of applications from other sources. Under "Settings," select "Application Settings" and then enable "Unknown Sources." Also under "Settings," select "SD Card" and "Phone Storage," and finally enable "Disable Use for USB Storage"
- This last step is easy. Open Command Prompt and type the following:
  - adb install [Path\FileName.apk]
- You're done! Your application is now ready for your use and enjoyment.

Copiando aplicativos através de um SD Card

- A maneira mas simples de instalar um aplicativo em um dispositivo Android é através de um SD Card
  - Crie sua aplicação e exporte-a através do Eclipse (Android Tools | Export signed application package)
  1. Copie o arquivo APK em seu memory card e insira-o no dispositivo
  2. Download e instalar um aplicativo instalador de aplicativos via Android Market (por exemplo: App Installer, mas há várias opções gratuitas e...outras pagas)
  3. Através do instalador, veja os arquivos APK no seu memory card
  4. Clique e instale seu aplicativo.