# Adapters

Marco Ronchetti
Università degli Studi di Trento

## Adapter - AdapterView

AdapterView:

a view whose children are determined by an Adapter.

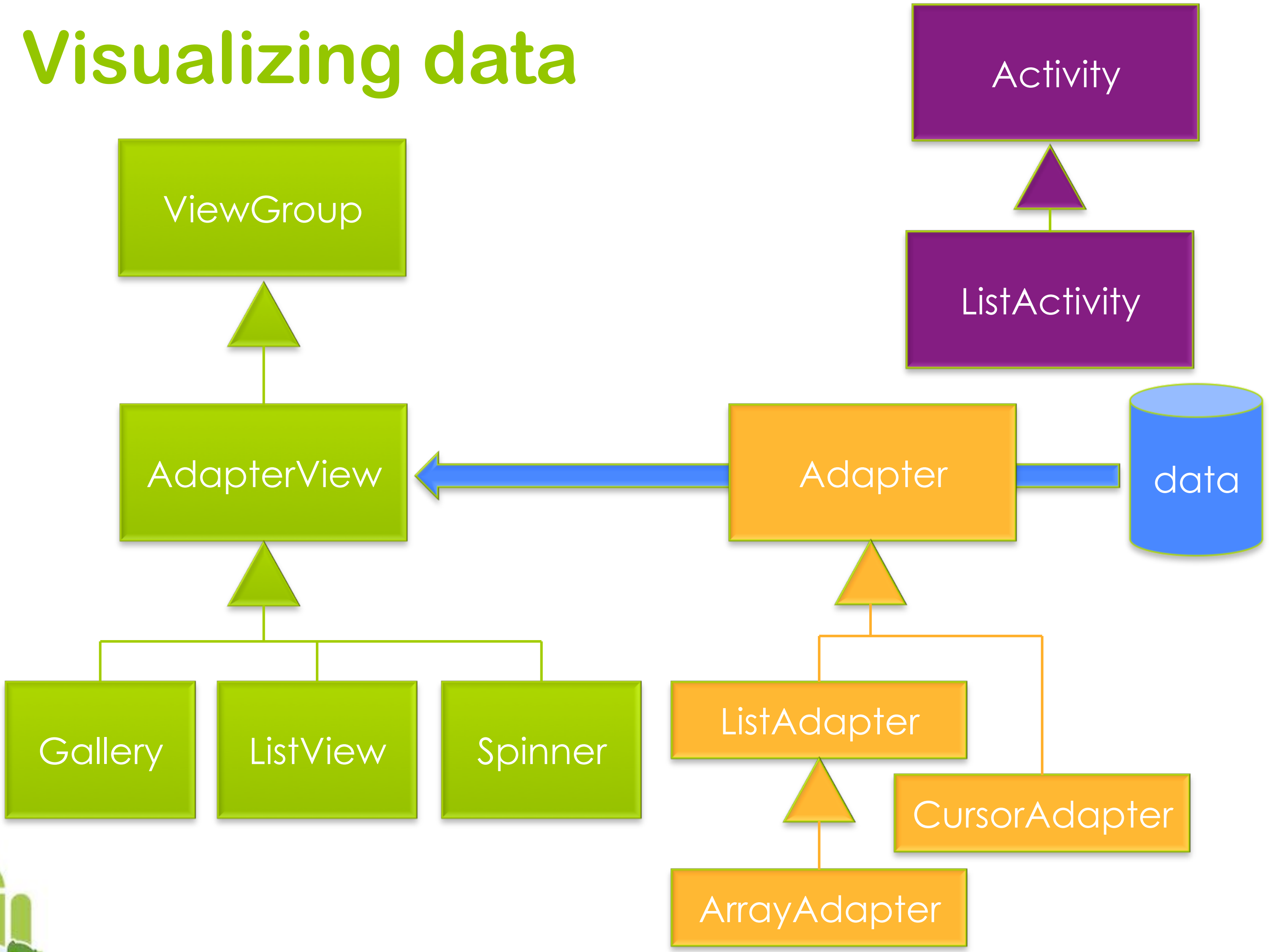
Adapter:

a bridge between an AdapterView and the underlying data for that view.

The Adapter:

- provides access to the data items.
- makes a View for each item in the data set.

## Visualizing data



Whenever you use Activity you set your_layout.xml as your Activity's ContentView. So the ListView should be in your_layout.xml.

That ListView should have an id attribute defined in xml file say:
android:id="@+id/list"
You get your ListView object some thing like this way:

```
setContentView(R.layout.your_layout);
ListView list = (ListView)findViewById(R.id.list);
list.addFooterView(view);
```

When you use ListActivity you get your ListView by calling method

```
ListView list = getListView();
```

OR you can do
```
ListView list = (ListView)findViewById(android.R.id.list);  //note the android prefix.
```

and please note that while defining any layout.xml for ListActivity you would have a ListView in your layout having id of something like this: android:id="@android:id/list"

## Example: ListView + ArrayAdapter

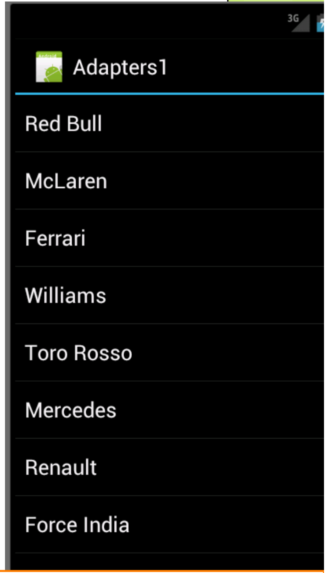
```
public class Adapters1Activity extends ListActivity {
@Override
   public void onCreate(Bundle state) {
      super.onCreate(savedInstanceState);
      ListView listView = getListView();
      Data x= new Data();
      String[] values= x.getValues();
      ArrayAdapter<String> adapter =
            new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1,
            values);
      listView.setAdapter(adapter);
   }
}
```

Extends Activity, designed to simplify the handling of ListViews.

No setContentView!

A default layout,
You can see its code here:
http://www.netmite.com/android/mydroid/frameworks/base/core/res/res/layout/

```
public class Data {
  private String[] values = new String[] {
       "Red Bull", "McLaren", "Ferrari",
       "Williams", "Toro Rosso", "Mercedes",
        "Renault", "Force India", "Sauber" };
   public String[] getValues(){return values;}
}
```


Adapters1

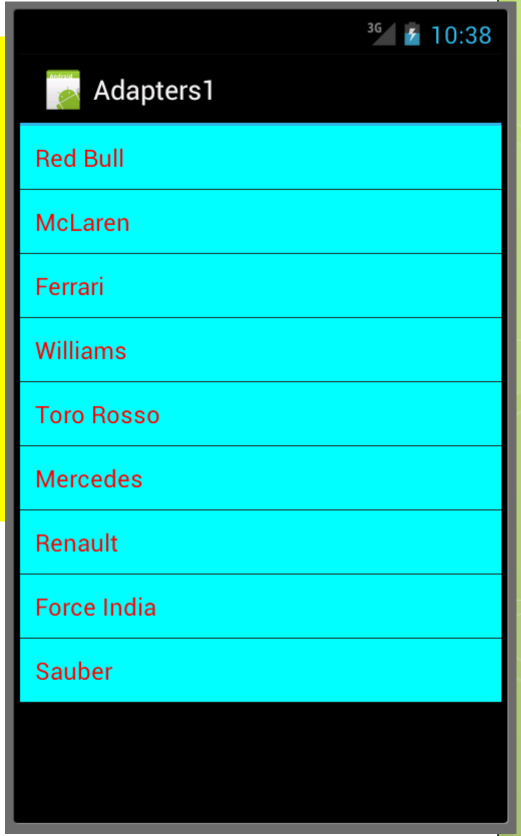
Red Bull
McLaren
Ferrari
Williams
Toro Rosso
Mercedes
Renault
Force India

# Add non-default row-layout

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                                        R.layout.simple_element, values);
```

*simple_element.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:padding="10dp"
   android:textSize="16sp"
   android:textColor="#FF0000"
   android:background="#00FFFF">
</TextView>
```

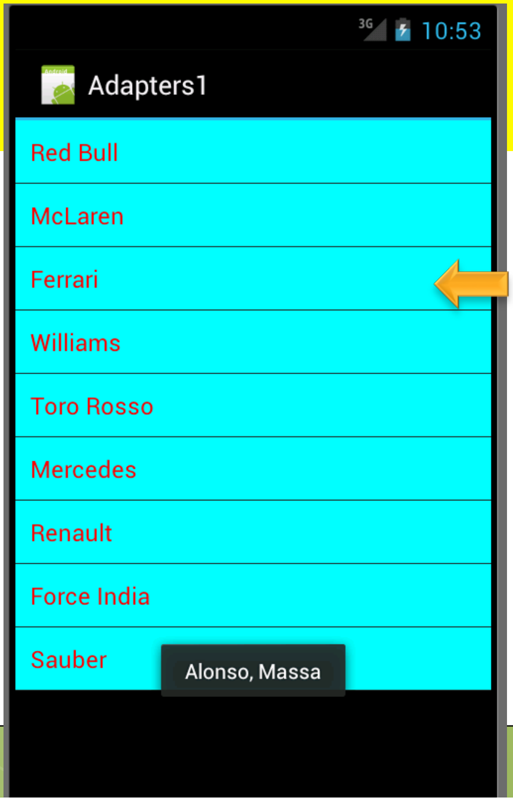


Red Bull
McLaren
Ferrari
Williams
Toro Rosso
Mercedes
Renault
Force India
Sauber

# Add a click behaviour

```
listView.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent,
        View view, int position, long id) {
     // When clicked, show a toast
     Toast.makeText(getApplicationContext(),
       x.getMapped(((TextView) view).getText().toString()),
       Toast.LENGTH_SHORT).show();
    }
  });
```

Add this fragment to onCreate



```
public class Data {
  private String[] values = new String[] {
      "Red Bull", "McLaren", "Ferrari",
      "Williams", "Toro Rosso", "Mercedes",
       "Renault", "Force India", "Sauber" };
    private HashMap<String,String> hm;
      Data() {
          hm=new HashMap<String,String>();
          hm.put("Ferrari", "Alonso, Massa");
      }

    public String[] getValues(){return values;}
    public String getMapped(String key){
       return hm.get(key);
     }
}
```

## Toast

A toast is a view containing a quick little message for the user (shown for a time interval).

When the view is shown to the user, appears as a floating view over the application. It will never receive focus.

It is as unobtrusive as possible, while still showing the user the information you want them to see.

setGravity(), setDuration(), set Text(), view()

## Adapter

int getCount()

- How many items are in the data set represented by this Adapter.

boolean isEmpty()

- true if dataset is empty

Object getItem(int k)

- Get the k-th data item

### What to do when the data change?

void registerDataSetObserver(DataSetObserver o)

- Register an observer that is called when changes happen to the data used by this adapter.

void unregisterDataSetObserver(DataSetObserver o)

- Unregister an observer that has previously been registered

Class DataSetObserver
onChanged()
- called when the entire data set has changed

## AdapterView

int getFirstVisiblePosition()

int getLastVisiblePosition()

- Returns the position within the adapter's data set for the first (last) item displayed on screen.

getAdapter() / setAdapter()

setOnItemClickListener(AdapterView.OnItemClickListener listener)

setOnItemSelectedListener(AdapterView.OnItemSelectedListener listener)

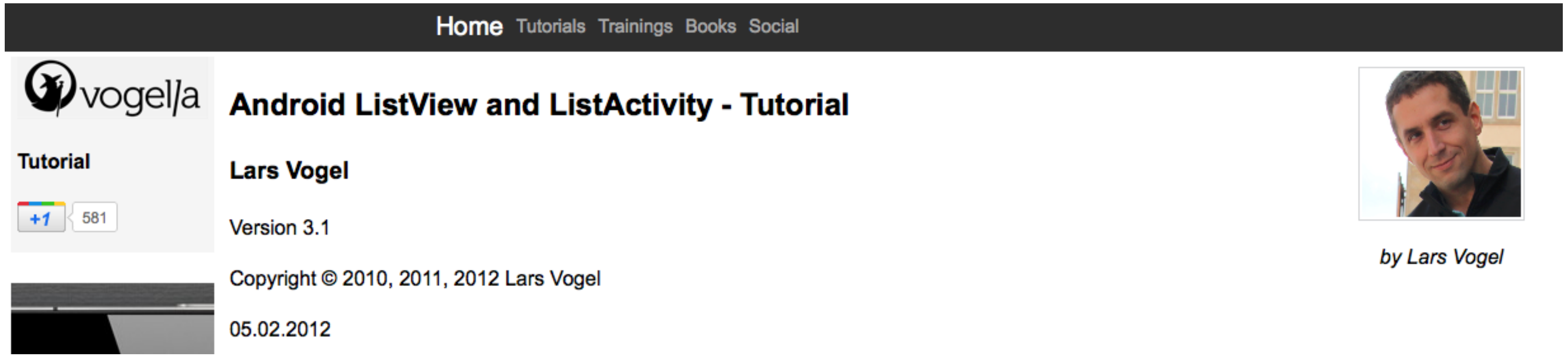
Object getItemAtPosition(int position)

setSelection(int position)

## For more demos and details…

See Lars Vogel:

http://www.vogella.de/articles/AndroidListView/article.html



**Android ListView and ListActivity - Tutorial**

**Lars Vogel**

Version 3.1

Copyright © 2010, 2011, 2012 Lars Vogel

05.02.2012

*by Lars Vogel*

In the next lectures, we'll discuss Filters and Observers

# Threads

Marco Ronchetti
Università degli Studi di Trento

## Threads

When an application is launched, the system creates a thread of execution for the application, called "main" or "UI thread"

This thread dispatches events to the user interface widgets, and draws (uses the android.widget and android.view packages).

Unlike Java AWT/Swing, separate threads are NOT created automatically.

Methods that respond to system callbacks (such as onKeyDown() to report user actions or a lifecycle callback method) always run in the UI thread.

If everything is happening in the UI thread, performing long operations such as network access or database queries will block the whole UI. When the thread is blocked, no events can be dispatched, including drawing events. From the user's perspective, the application appears to hang.

If the UI thread is blocked for more than 5 sec the user is presented with the"ANR - application not responding" dialog.

## the Andoid UI toolkit is not thread-safe !

Consequence:

you must not manipulate your UI from a worker thread—all manipulation to the user interface must be done within the UI thread.

You MUST respect these rules:

- Do not block the UI thread
- Do not access the Android UI toolkit from outside the UI thread

## An example from android developers

```
public void onClick(View v) {
        Bitmap b = loadImageFromNetwork(
                        "http://example.com/image.png");
        myImageView.setImageBitmap(b);
}
```

WRONG!
Potentially
Slow
Operation!

```
public void onClick(View v) {
 new Thread(new Runnable() {
      public void run() {
          Bitmap b = loadImageFromNetwork(
              "http://example.com/image.png");
          myImageView.setImageBitmap(b);
     })
            .start();
}
```

WRONG!
A non UI thread
accesses the UI!

## Still not the solution…

```
public void onClick(View v) {
  Bitmap b;
      new Thread(new Runnable() {
          public void run() {
              b = loadImageFromNetwork(
                  "http://example.com/image.png");
          })
      .start();
      myImageView.setImageBitmap(b);
}
```

**WRONG!**
**This does not wait for the thread to finish!**

## The solution

public boolean post (Runnable action)

- Causes the Runnable to be sent to the UI thread and to be run therein. It is invoked on a View from outside of the UI thread.

public boolean postDelayed (Runnable action, long delayMillis)

```
public void onClick(View v) {
  new Thread(new Runnable() {
      public void run() {
          Bitmap b = loadImageFromNetwork(
              "http://example.com/image.png");
          myImageView.post(
              new Runnable() {
                  public void run() {
                      mImageView.setImageBitmap(bitmap);
                  }
              }
      })
          .start();
}
```

OK! This code will be run in the UI thread

## Java reminder: varargs

void f(String pattern, Object... arguments);

The three periods after the final parameter's type indicate that the final argument may be passed

- as an array *or*
- as a sequence of arguments.

Varargs can be used *only* in the final argument position.

```
Object a, b, c, d[10];
...
f("hello",d);
f("hello",a,b,c);
```

# Varargs example

```
public class Test {
    public static void main(String args[]){ new Test(); }

    Test(){
        String k[]={"uno","due","tre"};
        f("hello",k);
        f("hello",“alpha“,“beta“);
        // f("hello“,“alpha“,“beta“,k); THIS DOES NOT WORK!
    }

    void f(String s, String... d){
        System.out.println(d.length);
         for (String k:d) {
             System.out.println(k);
         }
    }

}
```

## AsyncTask<Params,Progress,Result>

Creates a new asynchronous task. The constructor must be invoked on the UI thread.

AsyncTask must be subclassed, and instantiated in the UI thread.

Methods to be overridden:

| method | where | when |
|---|---|---|
| void onPreExecute() | UI Thread | before |
| Result doInBackground(Params...) | Separate new thread | during |
| void onProgressUpdate(Progress…) | UI Thread | |
| void onPostExecute(Result) | UI Thread | after |

## The more elegant solution

```
public void onClick(View v) {
  new DownloadImageTask().execute("http://example.com/image.png");
}
```

```
private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
    protected Bitmap doInBackground(String... urls) {
        return loadImageFromNetwork(urls[0]);
    }
    protected void onPostExecute(Bitmap result) {
        mImageView.setImageBitmap(result);
    }
}
```

# Using Progress

```
package it.unitn.science.latemar;
import ...
```

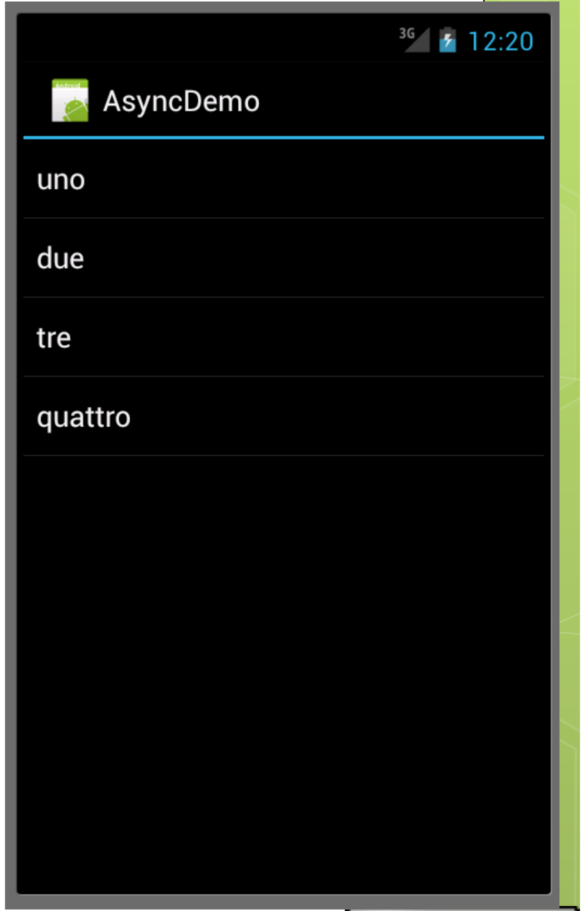
```
public class AsyncDemoActivity extends ListActivity {
 private static final String[] item{"uno","due","tre","quattro",
          "cinque","sei", "sette","otto","nove",
          "dieci","undici","dodici",};

 @Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  ListView listView = getListView();

  setListAdapter(new ArrayAdapter<String>(this,
          android.R.layout.simple_list_item_1,
          new ArrayList<String>()));

  new AddStringTask().execute();
 }
```

Adapted from the source code of http://commonsware.com/Android/



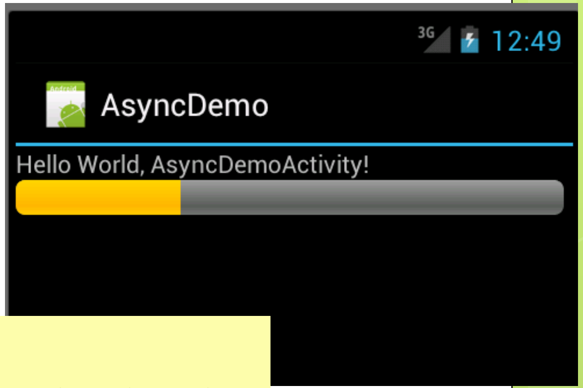
## Using Progress

This is an inner class!

```
class AddStringTask extends AsyncTask<Void, String, Void> {
  @Override
  protected Void doInBackground(Void... unused) {
    for (String item : items) {
      publishProgress(item);
      SystemClock.sleep(1000);
    }
    return(null);
  }
  @SuppressWarnings("unchecked")
  @Override
  protected void onProgressUpdate(String... item) {
    ((ArrayAdapter<String>)getListAdapter()).add(item[0]);
  }
```

```
@Override
    protected void onPostExecute(Void unused) {
      Toast
        .makeText(AsyncDemoActivity.this,
                  "Done!", Toast.LENGTH_SHORT)
        .show();
    }
  }
}
```

# Using the ProgressBar



Hello World, AsyncDemoActivity!

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

   <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello" />
   <ProgressBar
        android:id="@+id/pb1"
        android:max="10"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        style="@android:style/Widget.ProgressBar.Horizontal"
        android:layout_marginRight="5dp" />
</LinearLayout>
```

```
public class AsyncDemoActivity2
                extends Activity {
 ProgressBar pb;
 @Override
 public void onCreate(Bundle state) {
  super.onCreate(state);
  setContentView(R.layout.main);
  pb=(ProgressBar) findViewById(R.id.pb1);
  new AddStringTask().execute();
 }
```

## Using the ProgressBar

```
  class AddStringTask extends AsyncTask<Void, Integer, Void> {
  @Override
  protected void doInBackground(Void... unused) {
   int item=0;
   while (item<10 ){
    publishProgress(++item);
    SystemClock.sleep(1000);
   }
 }
  @Override
  protected void onProgressUpdate(Integer... item) {
   pb.setProgress(item[0]);
  }
}
```

# Sensors

Marco Ronchetti
Università degli Studi di Trento

## Sensor categories

### Motion sensors

- measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes.

### Environmental sensors

- measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.

### Position sensors

- measure the physical position of a device. This category includes orientation sensors and magnetometers.

## Basic code for managing sensors

```
public class SensorActivity extends Activity, implements SensorEventListener {
   private final SensorManager sm;
   private final Sensor sAcc;
   public SensorActivity() {
      sm= (SensorManager)getSystemService(SENSOR_SERVICE);
      sAcc= sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
   }
   protected void onPause() {
      super.onPause();
      sm.unregisterListener(this);
   }
   protected void onResume() {
      super.onResume();
      sm.registerListener(this, sAcc, SensorManager.SENSOR_DELAY_NORMAL);
   }
   public void onAccuracyChanged(Sensor sensor, int accuracy) { }
   public void onSensorChanged(SensorEvent event) { }
}
```

## SensorManager

SensorManager sm=Context.getSystemService(SENSOR_SERVICE);

List<Sensor> getSensorList(int type)

- get the list of available sensors of a certain type. Sensor

Sensor getDefaultSensor(int type)

- Use this method to get the default sensor for a given type

void registerListener(SensorEventListener listener, Sensor sensor, int rate)

- Registers a SensorEventListener for the given sensor.

void unregisterListener(SensorEventListener listener, Sensor sensor)

- Unregisters a listener for the sensors with which it is registered.

void unregisterListener(SensorEventListener listener)

- Unregisters a listener for all sensors.

- Some methods for transforming data (Vector to matrix representation etc.)

## Sensor types

int constants of the Sensor class describing sensor types:

| | |
|---|---|
| TYPE_ACCELEROMETER | |
| TYPE_ALL | A constant describing all sensor types. |
| TYPE_AMBIENT_TEMPERATURE | |
| TYPE_GRAVITY | |
| TYPE_GYROSCOPE | |
| TYPE_LIGHT | |
| TYPE_LINEAR_ACCELERATION | |
| TYPE_MAGNETIC_FIELD | |
| TYPE_PRESSURE | |
| TYPE_PROXIMITY | |
| TYPE_RELATIVE_HUMIDITY | |
| TYPE_ROTATION_VECTOR | |

## Accelerometer

*"Sensor's values are in* ***meters/second^2*** *units. A sensor measures the acceleration applied to the device. For this reason, when the device is sitting on a table (and obviously not accelerating), the accelerometer reads a magnitude of* ***g*** *= 9.81 m/s^2. Similarly, when the device is in free-fall and therefore dangerously accelerating towards to ground at 9.81 m/s^2, its accelerometer reads a magnitude of 0 m/s^2."* (Android Developers – sensors)

## Orientation sensor

*"A compass is a navigational instrument for determining **direction** relative to the Earth's **magnetic poles**. It consists of a magnetized pointer (usually marked on the North end) free to align itself with **Earth's magnetic field**."* (Compass EN Wiki)

In Android's terminology it is called **Orientation sensor**.

## Gyroscope

*"A gyroscope is an instrument consisting of a rapidly **spinning wheel** so mounted as to use the tendency of such a wheel to maintain a fixed position in space, and to resist any force which tries to change it. The way it will move if a twisting force is applied depends on the extent and orientation of the force and the way the gyroscope is mounted. A free vertically spinning gyroscope remains vertical as the carrying vehicle tilts, so providing an artificial horizon. A horizontal gyroscope will maintain a certain bearing, and therefore indicate a vessel's heading as it turns. Modern gyroscopes (including those built-in in smartphones) no longer have a spinning wheel."* (Gyroscope Cambridge Encyclopedia)

*"All values are in **radians/second** and measure the rate of rotation around the **X**, **Y** and **Z** axis. The coordinate system is the same as is used for the acceleration sensor."* (Android Developers – sensors) Rotation is positive in the **counter-clockwise** direction.

## Sensor class

float getMaximumRange()

- maximum range of the sensor in the sensor's unit.

int getMinDelay()

- minimum delay allowed between two events in microsecond or zero if this sensor only returns a value when the data it's measuring changes

String getName()

float getPower()

- the power in mA used by this sensor while in use

float getResolution()

- resolution of the sensor in the sensor's unit.

int getType()

String getVendor()

int getVersion()

```
SensorManager sm= (SensorManager)getSystemService(SENSOR_SERVICE);
List<Sensor> sensorList = sm.getSensorList(Sensor.TYPE_ALL);
StringBuilder sensorString = new StringBuilder("Sensors:\n");
for(int i=0; i<sensorList.size(); i++) {
    sensorString.append(sensorList.get(i).getName()).append(", \n");
}
```

**HTC EVO 4G**
BMA150 3-axis Accelerometer
AK8973 3-axis Magnetic field sensor
AK8973 Orientation sensor
CM3602 Proximity sensor
CM3602 Light sensor

**Samsung Nexus-S**
KR3DM 3-axis Accelerometer
AK8973 3-axis Magnetic field sensor
AK8973 Orientation sensor
GP2A Light sensor
GP2A Proximity sensor
K3G Gyroscope sensor
Gravity Sensor
Linear Acceleration Sensor
Rotation Vector Sensor

## Interface SensorEventListener

abstract void onAccuracyChanged(Sensor sensor, int accuracy)

- Called when the accuracy of a sensor has changed.

abstract void onSensorChanged(SensorEvent event)

- Called when sensor values have changed.

## Code examples

- http://www.vogella.com/articles/AndroidSensor/article.html accelerometer and compass examples

- http://developer.android.com/guide/topics/sensors/sensors_overview.html and following pages

## Sensors limitation - 1

From Jim Steele

**Using available sensors in the Android platform: current limitations and expected improvements**

Comparing the sensors on these two phones demonstrates the **sensor fragmentation** now found in Android:

1) **Non-standard sensor availability**: The Nexus-S has a gyroscope (from ST Micro), but the EVO does not. In fact, most Android devices do not have a gyroscope. There is no standard availability of sensors across devices.

2) **Non-standard sensor capability**: The BMA150 is a Bosch Sensortec 10-bit accelerometer, and the KR3DM is a ST Micro 12-bit accelerometer (using a special part number). In fact, there is no standard capability requirement for sensors across devices to ensure consistentresolution, noise floor, or update rate.

## Sensors limitation - 2

3) **Sensors not fully specified**: The AK8973 is an AKM magnetometer, which is only 8-bits. Analyzing this data stream shows it is low-pass filtered. This fact is not published on the phone or even the sensor datasheet. Many sensors have characteristics not specified such as bias changes, non-uniform gain, and skew (coupling between measurement axes). **Algorithms that use sensors without knowing these extra characteristics may produce incorrect information**.

4) **Broken virtual sensors**: The AKM sensor driver abstracts out an orientation virtual sensor which is derived from the combination of two sensors: the accelerometer and magnetometer. However, support for this virtual sensor was dropped early on, so the TYPE_ORIENTATION sensor is deprecated and the method **SensorManager.getOrientation()** should be used instead. Furthermore, the new virtual sensors introduced in Android 2.3 (Gingerbread) are not supported on all devices.

The sensor differences between just these two phones is substantial. So when a developer is faced with **writing apps utilizing sensors across as many devices as possible, it is a daunting task**.

Furthermore, **the Android platform is not optimized for real-time sensor data acquisition**.

## What can you do with accelerometer and gyroscope?

http://www.starlino.com/imu_guide.html

**A Guide To using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications.**

“This guide is intended to everyone interested in inertial MEMS (Micro-Electro-Mechanical Systems) sensors, in particular Accelerometers and Gyroscopes as well as combination IMU devices (Inertial Measurement Unit).”

- what does an accelerometer measure
- what does a gyroscope (aka gyro) measure
- how to convert analog-to-digital (ADC) readings that you get from these sensor to physical units (those would be g for accelerometer, deg/s for gyroscope)
- how to combine accelerometer and gyroscope readings in order to obtain accurate information about the inclination of your device relative to the ground plane

http://www.starlino.com/dcm_tutorial.html

**DCM Tutorial – An Introduction to Orientation Kinematics**

## Emulator limits

The emulator does not emulate sensors, so what can you do without a physical device?

BUT…

There is an app that emulates many sensors, and that you can use as data provider!

## SensorSimulator

OpenIntents SensorSimulator lets you simulate sensor events from accelerometer, compass, orientation, temperature, light, proximity, pressure, linear acceleration, gravity, gyroscope and rotation vector sensors.

Moreover, you can simulate your battery level and your gps position too, using a telnet connection.

It transmits the simulated sensor data to an Android emulator.

Also, it can record sensor data from an real Android device

See https://github.com/openintents/sensorsimulator