1 Ficheiros

1.1 Sumario

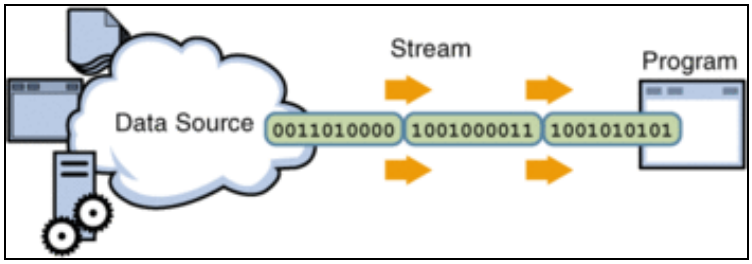
1.2 Introdución

- O tratamento dos ficheiros en Android é idéntico a Java.

1.2.1 Introdución á E/S en Java

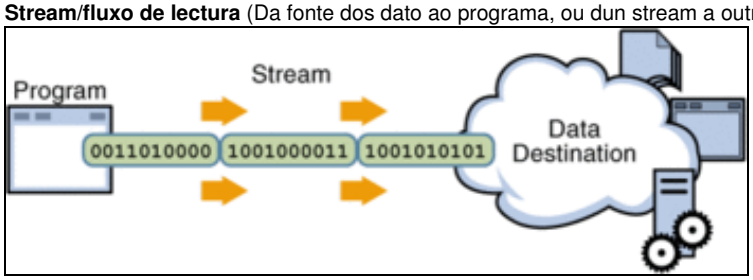
- O paquete **java.io** contén as clases para manipular a E/S.
- En java a entrada/saída xestionase a través de **streams (fluxos)** , e estes poden interactuar cun teclado, a consola, un porto, un ficheiro, outro stream, etc.
- Todo stream ten un orixe e un destino.

- Streams
-



Stream/fluxo de lectura (Da fonte dos dato ao programa, ou dun stream a outro stream).

-

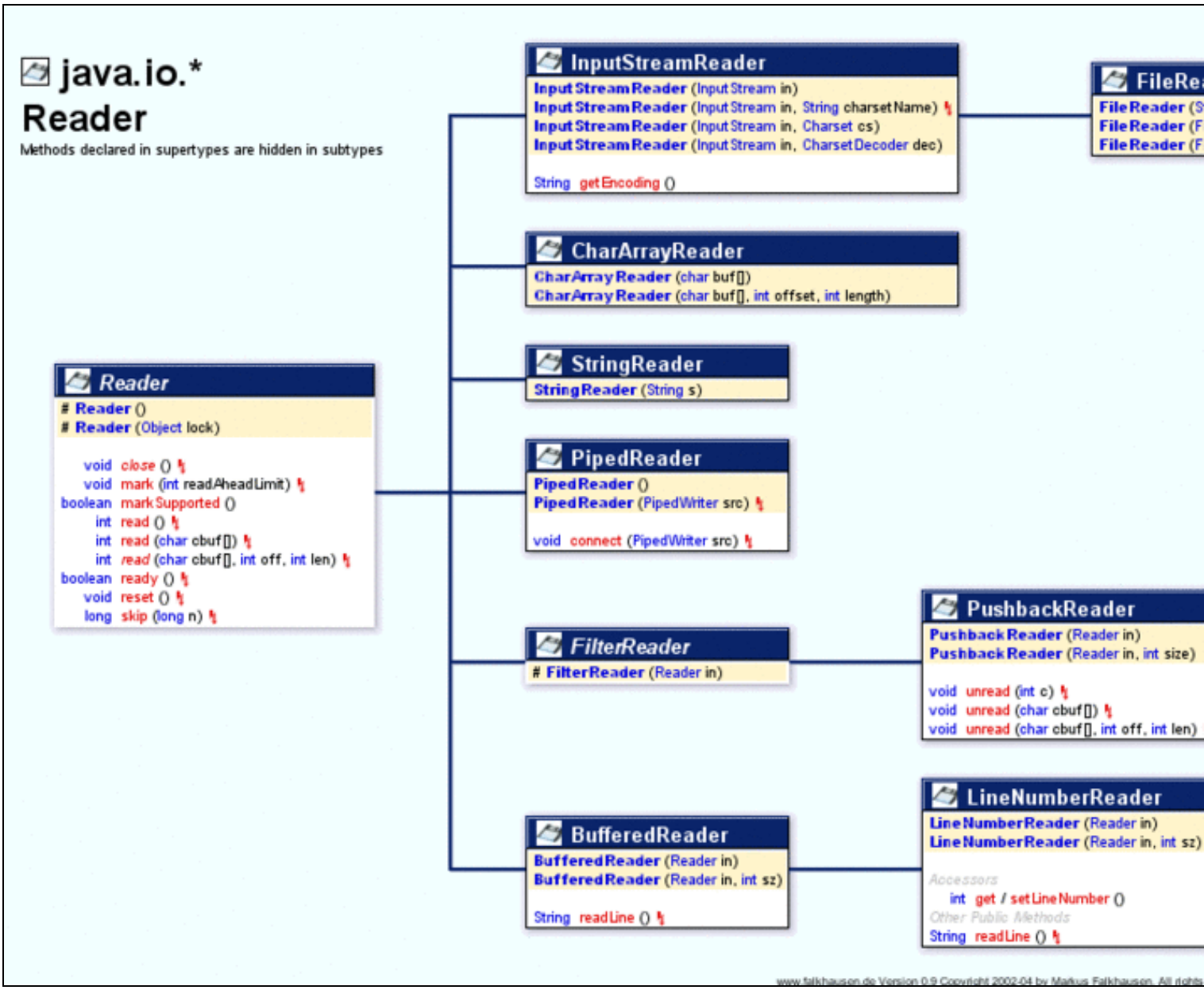


Stream/fluxo de escritura (Do programa ao destino dos datos, ou dun stream a outro stream).

- O fluxo máis básico de E/S son os fluxos de bytes, pero un fluxo de bytes pode ser a entrada doutro fluxo máis complexo, até chegar a ter fluxos de caracteres, e de buffers e o mesmo á inversa.

- Para aquel usuario que desexe repasar ou afondar na E/S en java déixanse os seguintes enlaces:

- Curso de Java nos Manuais do IES San Clemente: Entrada/Saída
- Diagramas moi gráficos (valga a redundancia) das xerarquías de clases de E/S, onde se poden ver as clases, atributos, construtores, métodos, etc dun modo moi claro:
- http://www.falkhausen.de/en/diagram/html/java.io.Writer.html (Neste caso da xerarquía writer).
- A modo de exemplo amósase un exemplo de diagrama da xerarquía *Reader*:
- Observar no diagrama como os costrutores da clase **InputStreamReader** reciben como parámetro outro stream/fluxo de tipo **InputStream** (Que está noutra xerarquía).



1.2.2 Introdución aos ficheiros en Android

- Os ficheiros en Android poden servirnos para almacenar información de modo temporal, para pasar información entre dispositivos, para ter unha "mini" base de datos, etc.
- En Android os ficheiros poden almacenarse en tres sitios (e dentro dun deles en 2 directorios distintos).
    - Na **propia aplicación** a modo de recurso (como cando incluímos unha imaxe): **/res/raw/ficheiro...** (raw significa cru).
    - Na **memoria interna**, no subdirectorio *files* da carpeta da aplicación: **/data/data/paquete_java/files/ficheiro...**
    - Na **tarxeta SD**, se existe, en 2 posibles subdirectorios:
        - ◊ **/storage/sdcard/directorio que indique o programador, se indica/ficheiro...**
        - ◊ **/storage/sdcard/Android/data/paquete_java/files/ficheiro...** (Algo parecido á memoria interna). Deste xeito se se desinstala a aplicación, tamén se borraría o ficheiro automaticamente, cousa que non pasaría no caso anterior

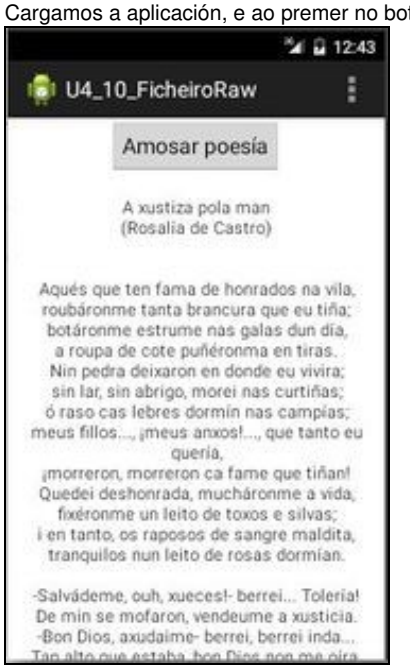
1.3 Ficheiro como recurso da aplicación

- Este ficheiro xa vai no instalable, no *.apk.
- O ficheiro debe estar en: /res/raw/ficheiro...
- Imos traballar cun poema de Rosalía de Castro, que máis dun século despois está de vigorosa actualidade.

1.3.1 Ficheiro raw: Caso práctico

- Crear o proxecto: **U4_10_FicheiroRaw**

- Ficheiro raw
- 

Cargamos a aplicación, e ao premer no botón ...
- 

para ler o ficheiro no recurso e amosámolo no TextView. Dispúxose un scroll para poder ver todo o poema.

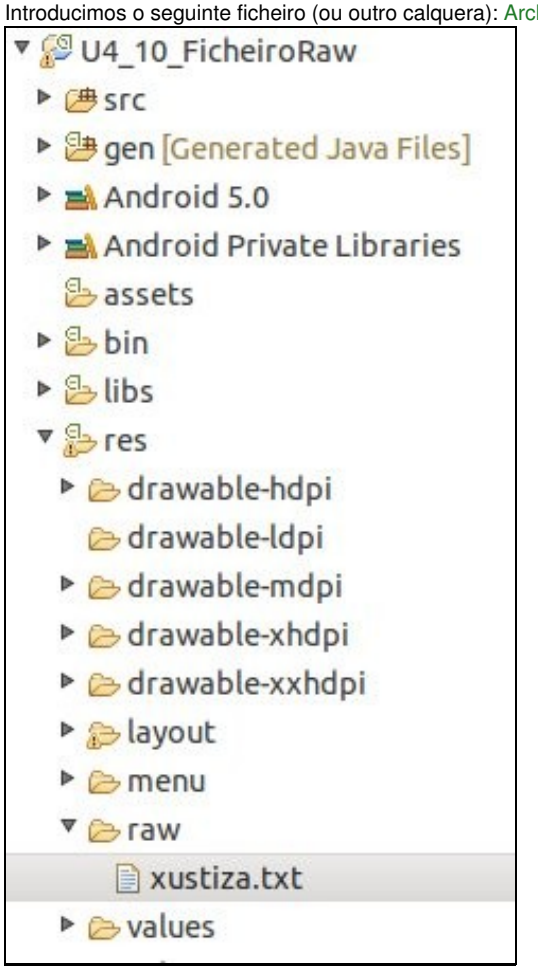
1.3.2 Ficheiro raw: O XML do layout

- Observar como envolvemos o TextView nun scroll.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="onButtonClick"
        android:text="Amosar poesía" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/tv"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="Aquí amosarase unha poesía de\nRosalía de Castro:\n&apos;A xustiza pola Man&apos;" />
    </ScrollView>

</LinearLayout>
```

1.3.3 Ficheiro raw: O recurso raw

- Creamos o cartafol **raw** dentro da carpeta /**res**
- Introducimos o seguinte ficheiro (ou outro calquera): Archivo:Xustiza.txt (Ollo que o ficheiro debe ter o nome en minúscula).
- 

1.3.4 Ficheiro raw: Código java da aplicación

```
package com.example.u4_10_ficheiroraw;

import java.io.BufferedReader;
```

```
import java.io.InputStream;
import java.io.InputStreamReader;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.TextView;

public class U4_10_FicheiroRaw extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_u4_10__ficheiro_raw);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.u4_10__ficheiro_raw, menu);
return true;
}

public void onButtonClick(View v) {
TextView tv = (TextView) findViewById(R.id.tv);
String verso;

tv.setText("");

try {
InputStream is = getResources().openRawResource(R.raw.xustiza);
BufferedReader br = new BufferedReader(new InputStreamReader(is));

while ((verso = br.readLine()) != null)
tv.append(verso + "\n");

br.close();
is.close();
} catch (Exception ex) {
Log.e("FICHEIROS", "Error ao ler ficheiro dende recurso raw");
}
}
}
```

- **Liña 36**: Creamos un fluxo de tipo InputStream cuxa entrada é o ficheiro xustiza. Ollo que non se lle puxo a extensión.
- **Liña 37**: A partir do fluxo anterior creamos un fluxo de tipo BufferReader.
- **Liña 38**: Lemos cada liña do ficheiro e asignámoslla a *verso*, até que sexa fin de ficheiro.
- **Liña 39**: Engádese ao TextView cada unha das liñas cun retorno de carro ao final de cada unha delas.
- **Liñas 42-43**: Pechamos os fluxos abertos no momento da súa creación.
- **Liñas 35 e 44**: habilitamos un control de excepcións, por se hai problemas cos fluxos.

1.4 Ficheiro na memoria interna

- Como sabemos cando se instala a aplicación no dispositivo esta instálase na memoria interna (salvo que se diga o contraio) na ruta **/data/data/paquete_java**.
- No subdirectorio **files** dese directorio é onde se crea o ficheiro por defecto.
- Ollo que cando se almacena na memoria interna debemos ter en conta o espazo que ten o dispositivo asignado a esta memoria.

- Para **crear** un ficheiro na memoria interna, Android facilítanos o método: **openFileOutput (ficheiro, modo_acceso_ao_ficheiro)**.
    - ♦ Este método abre o ficheiro indicado no modo indicado, que pode ser:
        - ◊ **MODE_PRIVATE**: para acceso privado dende a nosa app, e non dende outras.
        - ◊ **MODE_APPEND**: para engadir datos a un ficheiro existente
        - ◊ **MODE_WORLD_READABLE**: permitir que outras app lean o ficheiro
        - ◊ **MODE_WORLD_WRITEABLE**: permitir que outras app lean/escriban o ficheiro
    - ♦ O método devolve un stream asociado ao ficheiro de tipo **FileOutputStream**, a partir de aquí xa podemos operar con ese fluxo como o faríamos en Java.

- O método crea o ficheiro no directorio: **/data/data/paquete_java/files/ficheiro**.
- Tamén poderíamos crear ese ficheiro cunha clase tradicional de java como é: **FileOutputStream(ficheiro)** ou **fileOutputStream(ficheiro,engadir)**, onde:
  - **Ficheiro:** é a ruta ao ficheiro que lle temos que indicar nós ou ben a *lume* (/data/data/.../files) ou facendo uso do método: **getFilesDir()**, que devolve a ruta ao directorio **files** da aplicación.
  - **Engadir**: se o ficheiro se abre en modo sobrescritura ou append.
  - Recoméndase consultar a clase Java FileOutputStream se se desexa operar con ela.

- Para **ler** un ficheiro da memoria interna, temos o método: **openFileInput(ficheiro)**
  - E xa abre directamente o *ficheiro* que se atopa en:**/data/data/paquete_java/files/ficheiro** (Se existe, claro).
  - Devolve un stream, manipulable dende Java, do tipo **InputStreamReader**.

- Nos dous casos operaremos aínda con fluxos de nivel superior (OutputStreamWriter e BufferReader, respectivamente) para poder manipular cadeas de texto directamente.

- Referencias:
  - http://developer.android.com/reference/android/content/Context.html#openFileOutput%28java.lang.String,%20int%29
  - http://developer.android.com/reference/android/content/Context.html#openFileInput%28java.lang.String%29
  - http://developer.android.com/reference/android/content/Context.html#deleteFile%28java.lang.String%29
  - http://developer.android.com/reference/android/content/ContextWrapper.html#getFilesDir%28%29

1.4.1 Memoria Interna: Caso práctico

- Crear o proxecto: **U4_11_FicheiroInterna**
- Imos realizar nun só proxecto todas as operacións con ficheiros na memoria interna:
  - **Escribir**: tanto en modo *append* como sobrescribindo.
  - **Ler**: o ficheiro se existe, e senón dar un aviso
  - **Borrar**: o ficheiro se existe, e senón dar un erro
  - **Listar**: o contido dun directorio.

- Memoria Interna
- 

Entramos na aplicación.

- 

Prememos en **Ler** e non hai ningún ficheiro para ler.

- 

Prememos en **Borrar** e non hai ningún ficheiro para borrar.

- 

Escribimos un texto e prememos en **Escribir**/**Engadir**.
- 

Escribimos outro texto e prememos en **Escribir**/**Engadir**.

- 

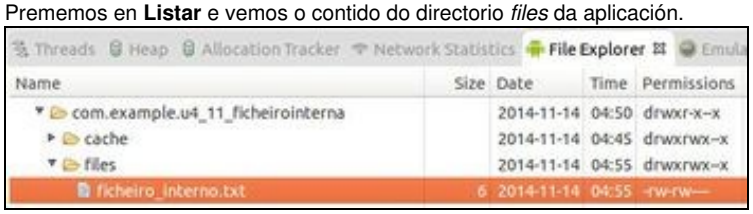
Prememos en **Ler** e lemos o ficheiro, que amosa o que escribimos antes.

- 

Marcamos **Sobrescribir**, escribimos un novo texto e prememos en **Escribir**/**Engadir**.

- 

Prememos **Ler** e vemos que o ficheiro foi sobrescrito co novo texto.
- 

Prememos en **Listar** e vemos o contido do directorio *files* da aplicación.
- 

A través do DDMS, pódese ver o ficheiro que se creou dende a aplicación.

1.4.2 Memoria Interna: XML do Layout

- Observar que os botóns, neste caso, foron organizados facendo uso dun **TableLayout**.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="2dp" >

    <EditText
        android:id="@+id/etTexto"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Escribe algún texto"
        android:inputType="textMultiLine" />

    <CheckBox
        android:id="@+id/cbSobrescribir"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Sobrescribir ficheiro" />

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:stretchColumns="*" >

        <TableRow>

            <Button
                android:id="@+id/bEscribirEngadir"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="onEscribirEngadirClick"
                android:text="Escribir/\nEngadir" />

            <Button
                android:id="@+id/bLer"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="onLerClick"
                android:text="Ler\n" />

            <Button
                android:id="@+id/bBorrar"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="onBorrarClick"
                android:text="Borrar\n" />

            <Button
                android:id="@+id/bListar"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="onListarClick"
                android:text="Listar\n" />
        </TableRow>
    </TableLayout>

    <ScrollView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/tvAmosar"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    </ScrollView>

</LinearLayout>
```

- **Liñas 33,40,47,54**: observar a que métodos chaman ao facer Click nos botóns.
- **Liñas 59-67**: O TextView está dentro dun **ScrollView** por se desbordamos a pantalla pola parte inferior á hora de amosar o contido do ficheiro.

1.4.3 Memoria Interna: O código java da aplicación

- Imos analizar cada un dos bloques de código.

```
package com.example.u4_11_ficheirointerna;
```

```
import java.io.BufferedReader;
import java.io.File;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class U4_11_FicheiroInterna extends Activity {
TextView tv;
public static String nomeFicheiro = "ficheiro_interno.txt";

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_u4_11__ficheiro_interna);

tv = (TextView) findViewById(R.id.tvAmosar);

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.u4_11__ficheiro_interna, menu);
return true;
}

public void onEscribirEngadirClick(View v) {
EditText etTexto = (EditText) findViewById(R.id.etTexto);
CheckBox cbSobrescribir = (CheckBox) findViewById(R.id.cbSobrescribir);
int contexto;
tv.setText("");

if (cbSobrescribir.isChecked())
contexto = Context.MODE_PRIVATE;
else
contexto = Context.MODE_APPEND;

try {

OutputStreamWriter osw = new OutputStreamWriter(openFileOutput(nomeFicheiro, contexto));

osw.write(etTexto.getText() + "\n");
osw.close();

etTexto.setText("");

} catch (Exception ex) {
Log.e("INTERNA", "Error escribindo no ficheiro");
}
}

public void onLerClick(View v) {
String linha = "";
TextView tv = (TextView) findViewById(R.id.tvAmosar);
tv.setText(linha);

try {

BufferedReader br = new BufferedReader(new InputStreamReader(openFileInput(nomeFicheiro)));

while ((linha = br.readLine()) != null)
tv.append(linha + "\n");
```

```
br.close();

} catch (Exception ex) {
Toast.makeText(this, "Problemas lendo o ficheiro", Toast.LENGTH_SHORT).show();
Log.e("INTERNA", "Erro lendo o ficheiro. ");

}
}

public void onBorrarClick(View v) {
File directorio_app = getFilesDir();
File ruta_completa = new File(directorio_app, "/" + nomeFicheiro);

if (ruta_completa.delete())
Log.i("INTERNA", "Ficheiro borrado");
else {
Log.e("INTERNA", "Problemas borrando o ficheiro");
Toast.makeText(this, "Problemas borrando o ficheiro", Toast.LENGTH_SHORT).show();

}
}

public void onListarClick(View v) {
tv.setText("");
File directorio_app = getFilesDir();
// File directorio_app = new File ("/");

tv.append(directorio_app.getAbsolutePath() + "\nContido:");
try {
String[] files = directorio_app.list();

for (int i = 0; i < files.length; i++) {
File subdir = new File(directorio_app, "/" + files[i]);
if (subdir.isDirectory())
tv.append("\n  Subdirectorio: " + files[i]);
else
tv.append("\n  Ficheiro: " + files[i]);
}
Log.i("INTERNA", "Listado realizado");

} catch (Exception ex) {
Log.e("INTERNA", "Erro listando o directorio");
}

}
}
```

- **Liñas 39-62**: Escribir/Engadir no ficheiro
    - **Liñas 45-48**: revisamos o estado do botón Sobrescribir e actuamos en consecuencia
    - **Liña 52**: obtemos un fluxo de tipo OuputStreamWriter que nos permite manipular cadeas de texto. Pero como parámetro recibe o ficheiro creado en función do contexto. Observar que non lle indicamos ningunha ruta para o ficheiro.
    - **Liña 54**: Escribimos no ficheiro o contido do EditText. Pero **Ollo!!!** engadimos ao final un retorno de carro, para que cada entrada vaia nunha única liña e non concatenadas.
    - **Liña 55**: Pechamos o fluxo.
    - **Liña 60**: Se se produciu algunha excepción no manipulación do fluxo sacamos unha mensaxe a través de LogCat.

- **Liñas 64-83**: Ler o ficheiro.
    - **Liña 67**: Limpamos o TextView
    - **Liña 71**: Creamos un fluxo de tipo BufferReader para poder manipular cadeas de texto. Este fluxo recibe como a apertura do ficheiro indicado. Observar que non lle indicamos ningunha ruta para o ficheiro.
    - **Liñas 73-74**: Mentres non sexa fin de ficheiro imos lendo liña a liña e presentándoa no TextView. Observar que introducimos un retorno de carro ao final de cada liña.
    - **Liñas 79-80**: se se produciu algunha excepción, por exemplo o ficheiro non existe, sacamos un Toast e unha mensaxe por LogCat.

- **Liñas 85-96**: Borrar o ficheiro

- Neste caso hai método (**deleteFile(ficheiro)**) que xa nos borra o ficheiro e devolve un boolean indicando o éxito da operación.
- Pero para introducir a clase **File** imos facelo de outra maneira.
    - ◊ Información sobre a clase File: http://developer.android.com/reference/java/io/File.html
    - ◊ Esta clase permite representar obxectos do sistema de ficheiros (directorios e ficheiros) a través das rutas relativas ou absolutas.
- **Liñas 86-87**: creamos unha ruta completa até o ficheiro. Para iso usamos o método **getFilesDir()** que nos devolve a ruta até o directorio *files* da aplicación. E logo construímos un novo obxecto File concatenando esa ruta coa barra de directorio e o nome do ficheiro.
- **Liña 89**: comprobamos o éxito do proceso de borrado do ficheiro. Esa liña podería ser substituída por "**if (deleteFile(nomeFicheiro))**" e non precisaríamos o código das liñas 86 e 87.
- Tamén controlamos as posibles excepcións.

- **Liñas 98-120**: Listar o contido dun directorio.
    - Ao igual que no caso anterior existe un método que xa nos devolve a lista de ficheiros do directorio *files* da aplicación: **fileList()**. Pero imos apoiarnos outra vez na clase File, para obter o listado de ficheiros dun directorio.
    - **Liña 100**: Obtemos a ruta ao directorio **files** da aplicación.
    - **Liña 103**: Amosamos a ruta completa a ese directorio.
    - **Liña 105**: Obtemos un array de obxectos (directorios e ficheiros) que contén o directorio en cuestión. Esta liña podería ser substituída por: **String[] files = fileList(nomeFicheiro);** e non precisaríamos a liña 100.
    - **Liñas 107-113**: Percorremos o array anterior e comprobamos se cada elemento é un ficheiro o un directorio e amosamos o seu nome.
    - **Liña 101**: descomentar esa liña e realizar un listado da raíz do sistema.

1.5 Memoria Externa - Tarxeta SD

- Todo canto se vai ver nesta parte apóiase no visto no apartado anterior de Memoria Interna.
- Vaise realizar o mesmo proceso que no caso anterior, so que neste caso na Memoria Externa.
- Co cal antes de pasar a este caso asegurarse de ter asimilado o referente a Memoria Interna.
- Aquí simplemente imos explicar as diferenzas co caso anterior.

1.5.1 Memoria Externa: Caso práctico

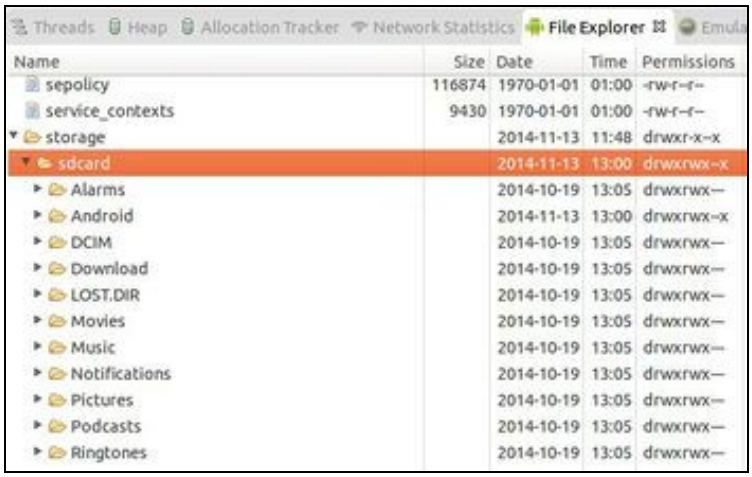
- Comezar creando o proxecto: **U4_12_FicheiroSD**.
- As seguintes imaxes amosan un aplicación semellante á anterior, so que esta traballa coa tarxeta SD no canto de coa memoria interna.

- Memoria Externa
-

A operativa será igual que no caso anterior.

- 

Se usamos o método: **getExternalFilesDir(null)**. O ficheiro gardarase na ruta da SD Card **/storage/sdcard/Android/data/paquete_java/files/ficheiro**.
Esta ruta é a ruta da aplicación na SD card, de modo que, se se desinstala a aplicación tamén se vai borrar esta ruta no proceso de desinstalación.

- 

Se usamos o método: **Environment.getExternalStorageDirectory()** o ficheiro gardaríase na ruta da raíz da SD Card (/storage/sdcard), salvo que se indique outra cousa.
Se se desinstala a aplicación non se vai borrar o ficheiro creado da SD Card.

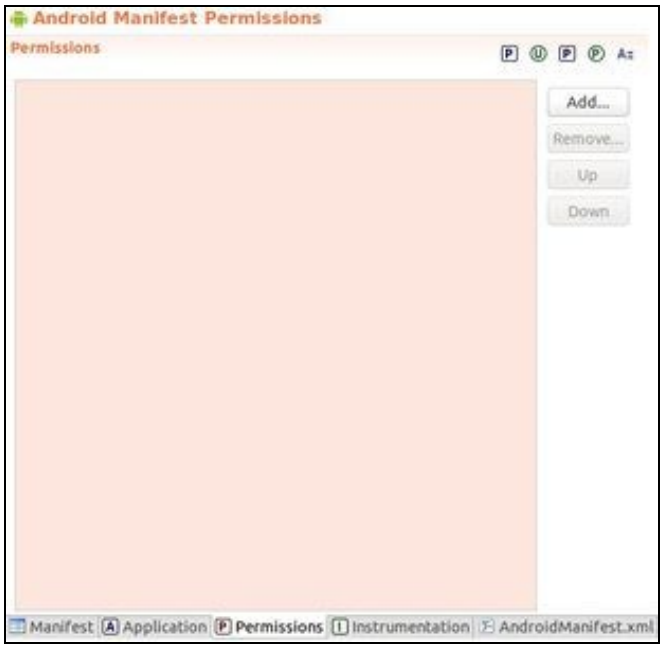
- En ámbolos dous casos o ficheiro pode ser borrado, manipulado polo usuario, ben dende o propio dispositivo usando calquera explorador de ficheiros ou ben montando a tarxeta SD nun ordenador, por exemplo, e actuando dende aí.

1.5.2 Memoria Externa: permisos de escritura na tarxeta SD

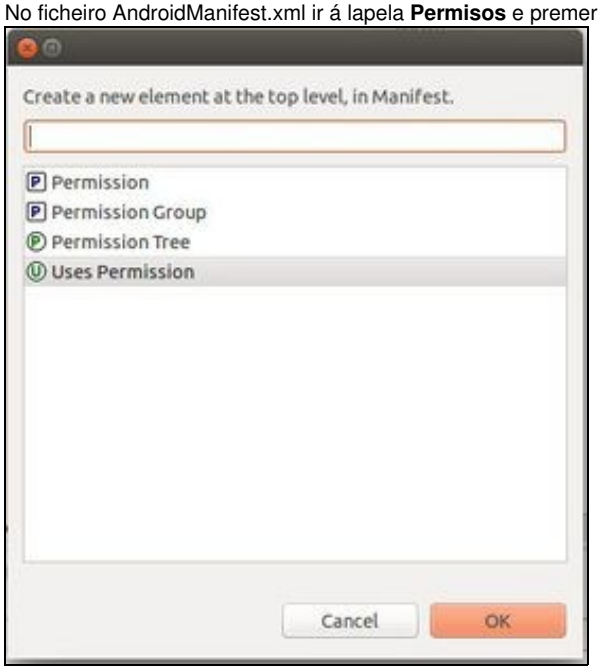
- Se imos ler na tarxeta SD:
  - ◊ Se a versión do S.O. Android é inferior á 4.1 non precisamos ningún permiso.
  - ◊ Se a versión do S.O. Android é superior ou igual á 4.1 debemos engadir o permiso: **<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>**
- Se imos escribir na tarxeta SD:
  - ◊ Se a versión do S.O. Android é inferior á 4.4 o permiso é: **<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>** .
  - ◊ Se a versión do S.O. Android é a 4.4 ou superior. Podemos poñer o mesmo permiso anterior pero as aplicacións dispoñen dun cartafol para escribir na SD (cartafol Android/data/paquete/) sen necesidade de ter o permiso anterior.

Os permisos necesarios son postos no ficheiro **AndroidManifest.xml** da aplicación.

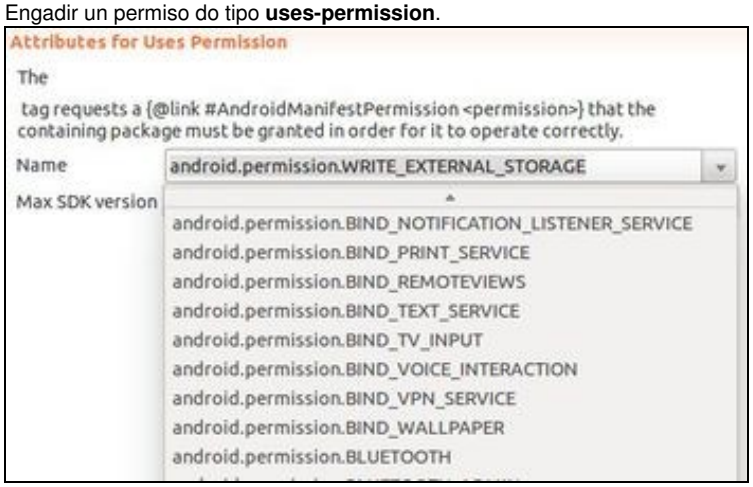
- Permiso escritura na Memoria Externa

- 

No ficheiro AndroidManifest.xml ir á lapela **Permisos** e premer en Engadir.

- 

Engadir un permiso do tipo **uses-permission**.

- 

Engadir o permiso: **android.permission.WRITE_EXTERNAL_STORAGE**.

-
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.u4_12_ficheirosd"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="16"
        android:targetSdkVersion="21" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".U4_12_FicheiroSD"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```
Comprobar que o permiso está no ficheiro XML.

1.5.3 Memoria Externa: XML do Layout

- O layout, neste caso é o mesmo, que o que se usou para a aplicación de Memoria Interna.

1.5.4 Memoria Externa: o código Java da Aplicación

- O código é o mesmo que o da aplicación Memoria Interna, salvo nos detalles que a continuación se relatan.
- No caso de usar a tarxeta SD, é preciso comprobar se esta está dispoñible e en que estado: modo lectura ou escritura.

- Para iso faremos uso do método: **Environment.getExternalStorageState()**, que nos pode devolver un dos seguintes estados:
    - MEDIA_UNKNOWN, MEDIA_REMOVED, MEDIA_UNMOUNTED, MEDIA_CHECKING, MEDIA_NOFS, MEDIA_MOUNTED, MEDIA_MOUNTED_READ_ONLY, MEDIA_SHARED, MEDIA_BAD_REMOVAL, ou MEDIA_UNMOUNTABLE.
    - Imos quedarnos con:
        - **MEDIA_MOUNTED**: indica que a tarxeta está dispoñible e ademais que se pode escribir nela.
        - **MEDIA_MOUNTED_READ_ONLY**: indica que a tarxeta está dispoñible, pero só en modo lectura.
    - Referencias: http://developer.android.com/reference/android/os/Environment.html#getExternalStorageState%28java.io.File%29

```
package com.example.u4_12_ficheirosd;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

import android.app.Activity;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class U4_12_FicheiroSD extends Activity {
boolean sdDisponhible = false;
boolean sdAccesoEscritura = false;
File dirFicheiroSD;
File rutaCompleta;
```

```
public static String nomeFicheiro = "ficheiro_SD.txt";

TextView tv;


@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_u4_12__ficheiro_sd);

tv = (TextView) findViewById(R.id.tvAmosar);

comprobarEstadoSD();
establecerDirectorioFicheiro();

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.u4_12__ficheiro_sd, menu);
return true;
}

public void comprobarEstadoSD() {
String estado = Environment.getExternalStorageState();
Log.e("SD", estado);

if (estado.equals(Environment.MEDIA_MOUNTED)) {
sdDisponhible = true;
sdAccesoEscritura = true;
} else if (estado.equals(Environment.MEDIA_MOUNTED_READ_ONLY))
sdDisponhible = true;
}

public void establecerDirectorioFicheiro() {

if (sdDisponhible) {
// dirFicheiroSD = Environment.getExternalStorageDirectory();
dirFicheiroSD = getExternalFilesDir(null);
rutaCompleta = new File(dirFicheiroSD.getAbsolutePath(), nomeFicheiro);

}
}

public void onEscribirEngadirClick(View v) {

EditText etTexto = (EditText) findViewById(R.id.etTexto);
CheckBox cbSobrescribir = (CheckBox) findViewById(R.id.cbSobrescribir);

boolean sobrescribir = false;

sobrescribir = !(cbSobrescribir.isChecked());

tv.setText("");

if (sdAccesoEscritura) {

try {

OutputStreamWriter osw = new OutputStreamWriter(new FileOutputStream(rutaCompleta, sobrescribir));

osw.write(etTexto.getText() + "\n");
osw.close();

etTexto.setText("");

} catch (Exception ex) {
Log.e("SD", "Error escribindo no ficheiro");
}
} else
Toast.makeText(this, "A tarxeta SD non está en modo acceso escritura", Toast.LENGTH_SHORT).show();
```

```
}

public void onLerClick(View v) {
String linha = "";
TextView tv = (TextView) findViewById(R.id.tvAmosar);
tv.setText(linha);

if (sdDisponhible) {
try {

BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(rutaCompleta)));

while ((linha = br.readLine()) != null)
tv.append(linha + "\n");

br.close();

} catch (Exception ex) {
Toast.makeText(this, "Problemas lendo o ficheiro", Toast.LENGTH_SHORT).show();
Log.e("SD", "Erro lendo o ficheiro. ");

}
} else
Toast.makeText(this, "A tarxeta SD non está dispoñible", Toast.LENGTH_SHORT).show();

}

public void onBorrarClick(View v) {

if (sdAccesoEscritura) {

if (rutaCompleta.delete())
Log.i("SD", "Ficheiro borrado");
else {
Log.e("SD", "Problemas borrando o ficheiro");
Toast.makeText(this, "Problemas borrando o ficheiro", Toast.LENGTH_SHORT).show();

}
} else
Toast.makeText(this, "A tarxeta SD non está en modo acceso escritura", Toast.LENGTH_SHORT).show();
}

public void onListarClick(View v) {
tv.setText("");

if (sdDisponhible) {

tv.append(dirFicheiroSD.getAbsolutePath() + "\nContido:");

try {
String[] files = dirFicheiroSD.list();

for (int i = 0; i < files.length; i++) {
File subdir = new File(dirFicheiroSD, "/" + files[i]);
if (subdir.isDirectory())
tv.append("\n  Subdirectorio: " + files[i]);
else
tv.append("\n  Ficheiro: " + files[i]);
}
Log.i("SD", "Listado realizado");

} catch (Exception ex) {
Log.e("SD", "Erro listando o directorio");
}

} else
Toast.makeText(this, "A tarxeta SD non está dispoñible", Toast.LENGTH_SHORT).show();
}

}
```

- **Liñas 22-26**: Definición de atributos.

- **Liña 22**: sdDisponhible: boolean que usaremos para antes de realizar calquera operación na SD card comprobar se está dispoñible.
- **Liña 23**: sdAccesoEscritura: boolean que usaremos para antes de escribir na SD card comprobar se poida realizar esa operación.
- **Liña 24**: dirFicheiroSD: imos usar esta variable para indicar para decidir se o ficheiro se vai crear na raíz da SD Card ou no directorio de aplicación na SD Card.
- **Liña 25**: rutaCompleta: nesta variable teremos a ruta ao directorio concatenada co nome do ficheiro.

• **Comprobar estado da SD Card**
- **Liña 28**: chamamos ao método que comproba o estado da SD Card.
- **Liñas 50-58**: comprobamos o estado
  - **Liña 51**: obtemos o estado da tarxeta.
  - **Liña 54**: comprobamos se a tarxeta está en modo escritura.
  - **Liña 57**: comprobamos se a tarxeta está accesible en modo lectura.

• **Determinar o directorio no que escribir/ler o ficheiro na SD Card**.
- **Liña 27**: chamamos ao método.
- **Liñas 61-69**: definimos as rutas ao directorio e ao ficheiro.
  - **Liña 64**: //**dirFicheiroSD = Environment.getExternalStorageDirectory();** devolvería a ruta da raíz da SD card:(/storage/sdcard)
  - **Liña 65**: **dirFicheiroSD = getExternalFilesDir(null);** devolve a ruta de *files* no directorio da aplicación na SD card (/storage/sdcard/Android/data/paquete_java/files).

• **Liñas 71-99**: Escribir/Engadir no ficheiro
- É basicamente igual ao proceso de Memoria Interna, salvo:
- **Liña 78**: como imos usar un fluxo dos de Java imos indicarlle no construtor se o ficheiro se abre en modo escritura ou append a través dun boolean.
- **Liña 82**: Comprobamos se a tarxeta SD está dispoñible en modo escritura, en caso contrario sacamos un Toast.
- **Liña 86**: non dispomos dun método que nos permita abrir o ficheiro, con simplemente indicarlle o nome, por tanto usamos a clase **FileOutputStream** pasándolle a ruta completa ao ficheiro e se se abre en modo append ou non.

• **Liñas 101-124**: Ler o ficheiro.
- É basicamente igual ao proceso de Memoria Interna, salvo:
- **Liña 82**: Comprobamos se a tarxeta SD está dispoñible (dá igual o modo), en caso contrario sacamos un Toast.
- **Liña 86**: non dispomos dun método que nos permita abrir o ficheiro, con simplemente indicarlle o nome, por tanto usamos a clase **FileInputStream** pasándolle a ruta completa ao ficheiro.

• **Liñas 126-139**: Borrar o ficheiro
- A estas alturas o usuario xa debe ser quen de interpretar ese código, estudando as explicacións anteriores e a correspondente para Memoria Interna.

• **Liñas 141-166**: Listar o contido dun directorio.
- A estas alturas o usuario xa debe ser quen de interpretar ese código, estudando as explicacións anteriores e a correspondente para Memoria Interna.

-- Ángel D. Fernández González e Carlos Carrión Álvarez -- (2015).