MÓVILES

Tema: La Clase `SharedPreferences`.

CONCEPTOS

La clase `SharedPreferences` permite guardar datos persistentes. Cada dato se almacena con un identificador y un valor asociado a él, por ejemplo, mes="Marzo", donde mes es la clave y Marzo es el valor. Los datos se guardan en archivos XML. Las siguientes constantes administran el acceso a las preferencias:

`MODE_PRIVATE`	Sólo la aplicación puede tener acceso a estas preferencias.
`MODE_WORLD_READABLE`	Las preferencias puede leerlas cualquier aplicación y sólo ésta puede modificarlas.
`MODE_WORLD_WRITABLE`	Cualquier aplicación puede leer y modificar las preferencias.
`MODE_MULTI_PROCESS`	Comprueba la modificación de las preferencias.
`MODE_APPEND`	Agrega las nuevas preferencias con las preferencias ya existentes.
`MODE_ENABLE_WRITE_AHEAD_LOGGING`	Bandera de apertura de los datos. Si se asigna, se habilita la escritura predeterminada anticipada.

La clase `SharedPreferences` permite guardar y recuperar pares de valores clave persistentes de tipos primitivos: `boolean`, `float`, `int`, `long`, `String`. Estos datos persisten en distintas sesiones, incluso si la aplicación se cierra. También la clase `PreferenceActivity` permite crear datos preferentes del usuario.

Para tener acceso a los datos del objeto `SharedPreferences` se utilizan los siguientes métodos:

`getSharedPreferences()`	Para múltiples archivos de preferencias, identificados por su nombre, el cual se especifica con el primer parámetro.
`getPreferences()`	Para un solo archivo de preferencias de la actividad. Debido a que este será el único archivo de preferencias de la actividad, no se proporciona un nombre.

Con el método `getSharedPrefences()` se accede a las preferencias, pasándole el identificador y el acceso. Por ejemplo:

```
SharedPreferences sp = getSharedPreferences("Datos",Context.MODE_PRIVATE);
```

- Para escribir los valores:
  a. Invocar a `edit()` para obtener un `SharedPreferences.Editor`.
  b. Añadir los valores con los métodos `putBoolean()`, `putString()`, o algún otro, según el tipo de dato.
  c. Guardar los nuevos valores con `commit()`.
- Para leer los valores:
  a. Leer los datos con los métodos `getBoolean()`, `getString()`, o el que corresponda, según el tipo de dato.

Android permite otras opciones para almacenar datos persistentes, la elección depende de las necesidades de la aplicación.

I. **ALMACÉN INTERNO**. En el almacén interno se almacenan datos privados en la memoria interna del dispositivo. Por definición, los archivos guardados en la memoria interna son privados para la aplicación y otras aplicaciones no pueden acceder a ellos (tampoco el usuario). Cuando el usuario desinstala la aplicación, estos archivos se eliminan.

Para crear y escribir un archivo privado al almacenamiento interno:

i. Invocar a `openFileOutput ()` con el nombre del archivo y el modo de acceso. Esto devuelve un `FileOutputStream`.
ii. Escribir en el archivo con `write()`.
iii. Cerrar el flujo con `close()`.

Por ejemplo:

```
String FILENAME = "misdatos";
String s = "ESCOM";
FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

Con `MODE_PRIVATE` se crea el archivo (o se sustituye el archivo con el mismo nombre) y lo convierte en privado para la aplicación. Otros modos son `MODE_APPEND`, `MODE_WORLD_READABLE` y `MODE_WORLD_WRITEABLE` (estos dos últimos son obsoletos).

Para leer un archivo del almacenamiento interno:

i. Invocar a `openFileInput` () y pasarle el nombre del archivo a leer. Regresa un `FileInputStream`.
ii. Leer los bytes del archivo con `read`().
iii. Enseguida, cerrar el flujo con `close`().

Si se necesita guardar datos sin persistencia, se utiliza `getCacheDir`() para abrir un archivo que represente la carpeta interna en la que la aplicación guardaría los archivos caché temporales.

Otros métodos útiles son:

`getFilesDir`() Obtiene la ruta absoluta de la carpeta en la que se guardaron los archivos.
`getDir`() Crea (o abre, si existe) la carpeta en el espacio del almacén interno.
`deleteFile`() Borra un archivo en al almacén interno.
`fileList`() Regresa un arreglo de archivos guardados por la aplicación.

II. **ALMACÉN EXTERNO**. Es un medio extraíble (una tarjeta SD) o una memoria interna (no extraíble). Los archivos guardados en el almacenamiento externo son legibles por el mundo y puede modificarlos el usuario cuando se permite el almacenamiento masivo USB para transferir archivos en una computadora.

Para acceder al almacén externo se necesitan los permisos del sistema con `READ_EXTERNAL_STORAGE or WRITE_EXTERNAL_STORAGE`. Por ejemplo, en el archivo AndroidManifest.xml:

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    :
</manifest>
```

Se debe verificar si existe almacén suficiente para guardar archivos con `getExternalStorageState`(). Por ejemplo:

```
public boolean isExternalStorageWritable(){// Verifica almacén para read y write.
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}
public boolean isExternalStorageReadable(){// Verifica almacén al menos para read
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

Por ejemplo, para crear una nueva carpeta para las imágenes:

```
  public File getAlbumStorageDir(String albumName) {
      File file = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES), albumName); if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
  return file;
  }
```

DESARROLLO

EJEMPLO 1.

En el siguiente ejemplo se emplea la clase `NotificationCompat` para construir la notificación.

Paso 1. Crear un nuevo proyecto Preferencias. En la carpeta java/com.example.mipaquete, abrir y modificar el archivo MainActivity.java con el siguiente código:

```
import android.content.Context;
import android.content.SharedPreferences;
import android.os.*;
import android.app.*;
import android.widget.*;
public class MainActivity extends Activity{
    SharedPreferences sp;
    EditText   jetn, jetx, jety;
    String    s;
    float  x, y;
    public void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.activity_main);
        jetn = (EditText) findViewById(R.id.xetn);
        jetx = (EditText) findViewById(R.id.xetx);
        jety = (EditText) findViewById(R.id.xety);
        sp = getSharedPreferences("preferencias", Context.MODE_PRIVATE);
        s = sp.getString("titulo", "ESCOM");
        x = sp.getFloat("x", 0);
        y = sp.getFloat("y", 0);
        jetn.setText(s);
        jetx.setText("" + x);
        jety.setText("" + y);
    }
    protected void onPause(){
        super.onPause();
        s = jetn.getText().toString();
        x = Float.parseFloat(jetx.getText().toString());
        y = Float.parseFloat(jety.getText().toString());
        SharedPreferences.Editor miEditor = sp.edit();
        miEditor.putString("titulo", s);
        miEditor.putFloat("x", x);
        miEditor.putFloat("y", y);
        miEditor.commit();
        Toast.makeText(this, "Preferencias guardadas", Toast.LENGTH_LONG).show();
    }
}
```

Paso 2. En la carpeta res/layout, abrir y modificar el archivo activity_main.xml con el siguiente código:

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

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:id="@+id/layout1" >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Cambiar sus preferencias"
        android:id="@+id/text1" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Nombre" />
    <EditText
        android:id="@+id/xetn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="EditText" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Eje X:" />
    <EditText
        android:id="@+id/xetx"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="EditText" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Eje Y:" />
    <EditText
        android:id="@+id/xety"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="EditText" />
</LinearLayout>
```

Paso 3. Ahora, ejecutar la aplicación. Cuando la aplicación se ejecuta por primera vez se muestra la inicialización de los datos de las preferencias, ver la figura 1. El usuario puede ingresar nuevos datos, ver la figura 2.

Cuando la aplicación se cierra, por ejemplo, digitando el botón de la barra inferior del móvil, los datos se guardarán indicándose ello con el mensaje momentáneo Preferencias guardadas del `Toast`, ver la figura 3.

Nota: No cerrar el proyecto ni el emulador, ello impediría la apertura del archivo de datos.

De esa forma los datos permanecerán persistentes.

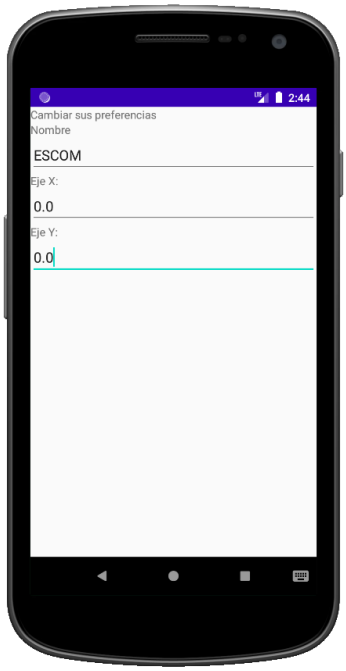

Figura 1. Inicio de la aplicación.

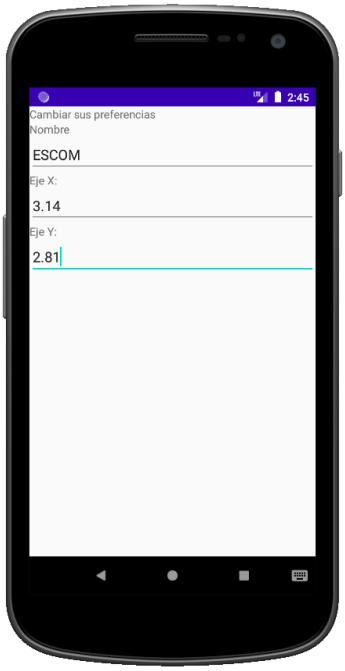

Figura 2. Modificar los datos.

Figura 3. Cerrar para mostrar el `Toast`.

Paso 4. Para verificar la recuperación de los datos, se abre el archivo XML correspondiente. En el menú principal, seleccionar la opción `View > Tool Windows > Device Explorer`, o es también es posible la opción `Device File Explorer` como se indica en la figura 4:

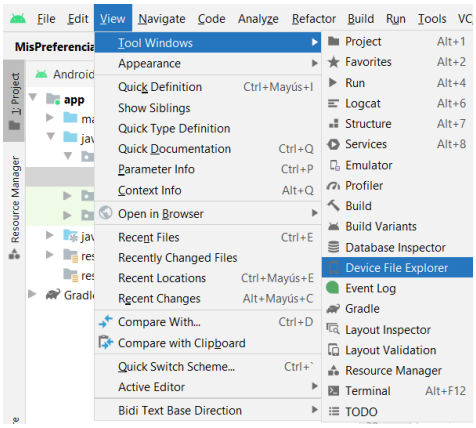

Figura 4. Seleccionar del menú principal la opción `View > Tool Windows > Device Explorer`. o también la opción `Device File Explorer`.

Al digitar en la opción Device Explorer se muestra, en la sección a la derecha del IDE, la ventana con los proyectos desarrollados. Seleccionar: data > data >. Buscar y seleccionar el nombre del paquete del proyecto, en este caso com.example.mispreferencias, como se indica en la siguiente figura:

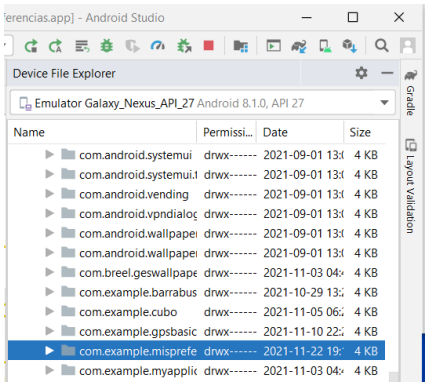

Figura 5. Seleccionar la carpeta del proyecto.

Abrir la carpeta del proyecto. Abrir la carpeta shared_prefs. Seleccionar y abrir el archivo preferencias.xml para mostrar su contenido. **Nota**: Los nombres del proyecto y del archivo de preferencias pueden ser diferentes.

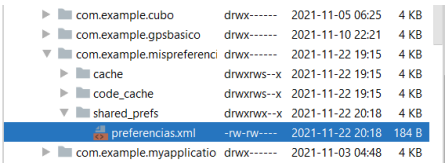

Figura 6. El archivo que contiene los datos actualizados.

Ahora, en el editor se muestra el contenido del archivo preferencias.xml y sus datos persistentes. Si la aplicación se ejecuta nuevamente los datos iniciales serán los mismos extraídos de este archivo, como se indica en la figura 7.

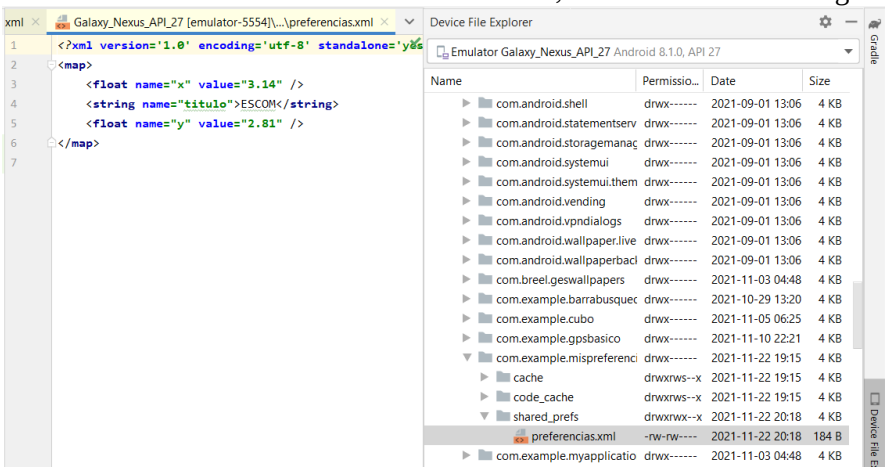

Figura 7. El editor muestra los datos persistentes.

EJERCICIO 1. Diseñar una aplicación que solicite las coordenadas de latitud y longitud de la ciudad de México para almacenarlas en el dispositivo. Después, mostrar en un `TextView` los datos almacenados a través de un `Intent`.

EJERCICIO 2. Diseñar una aplicación para almacenar una imagen, seleccionada desde la galería de imágenes, en un almacén del dispositivo. Después, mostrar en un `Canvas` la imagen almacenada a través de un `Intent`.

NOTA. Generar el reporte y documentarlo con las imágenes de cada una de las ejecuciones de los ejercicios. Guardar el documento con la sintaxis AlumnoPreferenciasGrupo.PDF, por ejemplo, y enviarlo al sitio indicado por el profesor.