КАЛУЖСКИЙ ФИЛИАЛ
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Э. БАУМАНА (национальный исследовательский университет)»

Факультет «Информатика и управление"

Кафедра "Программное обеспечение ЭВМ, информационные технологии"

Хранение данных в Android-приложениях

Калуга

Хранение данных. Preferences.

В Android есть несколько способов хранения данных:

Preferences - в качестве аналогии можно привести Windows INI-файлы

Обычные файлы - внутренние и внешние (на SD карте)

SQLite - база данных, таблицы

Рассмотрим Preferences. Значения сохраняются в виде пары: имя, значение.

Разработаем приложение. В нем будет поле для ввода текста и две кнопки – Save и Load. По нажатию на Save значение из поля будет сохраняться, по нажатию на Load – загружаться.

Откроем activity_main.xml и создадим такой экран:

```
<?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"
    android:id="@+id/lo" >

     <LinearLayout


        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

<EditText
    android:id="@+id/etText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"/>

</LinearLayout>
```

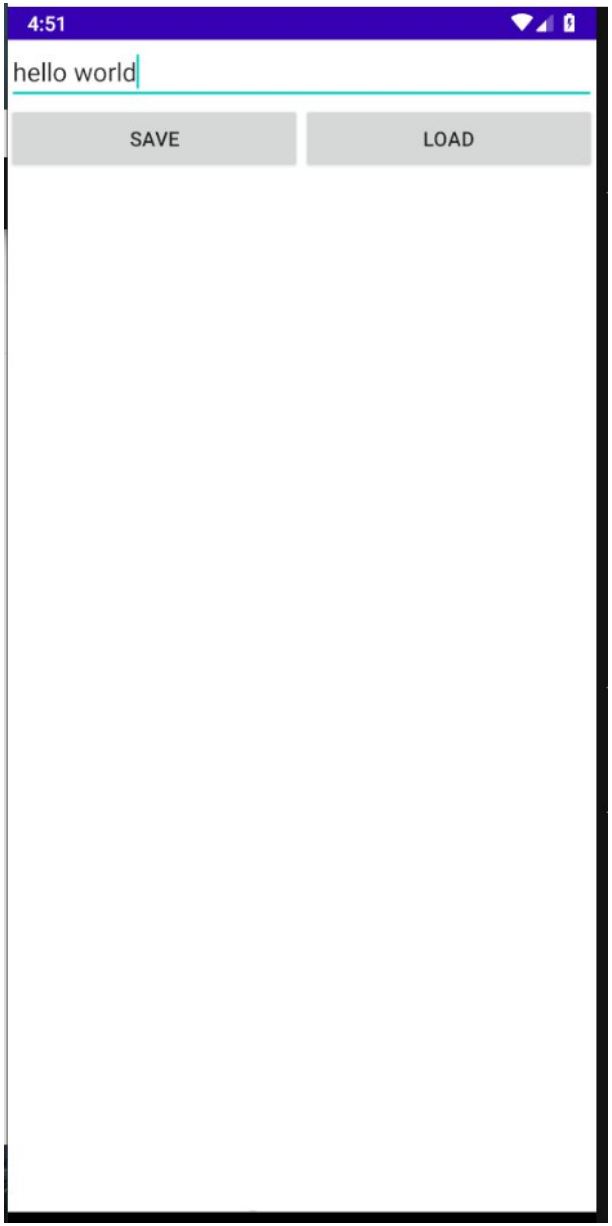
```
<LinearLayout


    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <Button
        android:id="@+id/btnSave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Save" >

    </Button>

    <Button
        android:id="@+id/btnLoad"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Load" >


    </Button>
</LinearLayout>
```

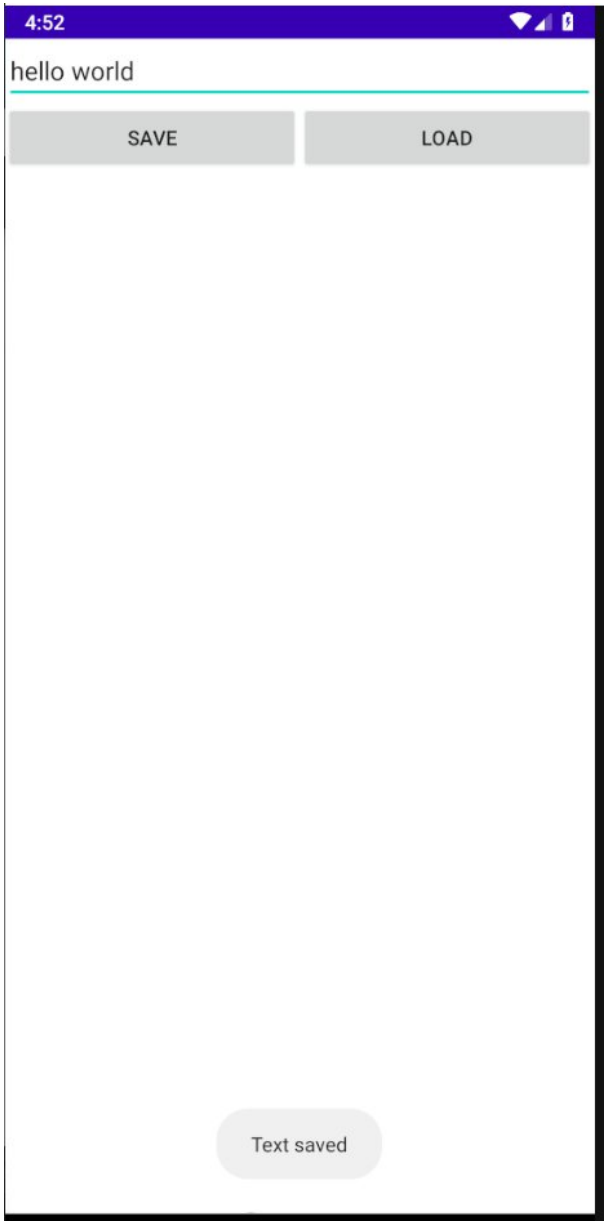
код в MainActivity.java

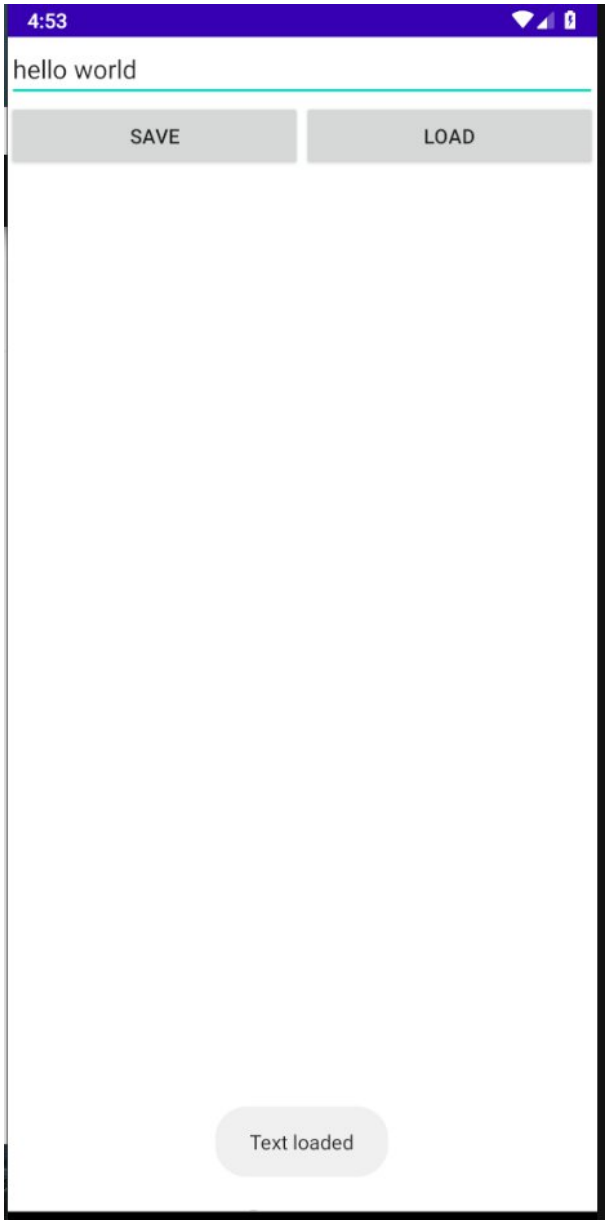
```
//Java
public class MainActivity extends Activity
implements View.OnClickListener {
  EditText etText;
  Button btnSave, btnLoad;
  SharedPreferences sPref;
  final String SAVED_TEXT = "saved_text";

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  etText = (EditText) findViewById(R.id.etText);
  btnSave = (Button) findViewById(R.id.btnSave);
  btnSave.setOnClickListener(this);
  btnLoad = (Button) findViewById(R.id.btnLoad);
  btnLoad.setOnClickListener(this);
}

public void onClick(View v) {
  switch (v.getId()) {
    case R.id.btnSave:
      saveText();
      break;
    case R.id.btnLoad:
      loadText();
      break;
    default:
      break;
}
}
```

```
void saveText() {
  sPref = getPreferences(MODE_PRIVATE);
  Editor ed = sPref.edit();
  ed.putString(SAVED_TEXT,
          etText.getText().toString());
  ed.commit();
  Toast.makeText(this, "Text saved",
          Toast.LENGTH_SHORT).show();
}

void loadText() {
  sPref = getPreferences(MODE_PRIVATE);
  String savedText =
          sPref.getString(SAVED_TEXT, "");
  etText.setText(savedText);
  Toast.makeText(this, "Text loaded",
          Toast.LENGTH_SHORT).show();
}
}
```

```kotlin
//Kotlin
class MainActivity : AppCompatActivity() {

    val SAVED_TEXT = "saved_text"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        btnSave.setOnClickListener(this::onClick)
        btnLoad.setOnClickListener(this::onClick)
    }

    fun onClick(view: View){
        when (view.id){
            R.id.btnSave -> saveText()
            R.id.btnLoad -> loadText()
        }
    }

    fun saveText() {
        var sPref = getPreferences(MODE_PRIVATE)
        var ed = sPref.edit()
        var etText = findViewById(R.id.etText) as EditText
        ed.putString(SAVED_TEXT, etText.getText().toString())
        ed.commit()
        Toast.makeText(this, "Text saved", Toast.LENGTH_SHORT).show()
    }

    fun loadText() {
        var sPref = getPreferences(MODE_PRIVATE)
        var savedText = sPref.getString(SAVED_TEXT, "")
        var etText = findViewById(R.id.etText) as EditText
        etText.setText(savedText)
        Toast.makeText(this, "Text loaded", Toast.LENGTH_SHORT).show()
    }
}
```


4:51
hello world
SAVE
LOAD


4:52
hello world
SAVE
LOAD
Text saved


4:53
hello world
SAVE
LOAD
Text loaded

Рассмотрим методы, которые вызываются в onClick

saveText – сохранение данных. Сначала с помощью метода getPreferences получаем объект sPref класса SharedPreferences, который позволяет работать с данными (читать и записывать). Константа MODE_PRIVATE используется для настройки доступа и означает, что после сохранения, данные будут видны только этому приложению. Далее, чтобы редактировать данные, необходим объект Editor – он получается из sPref. В метод putString указывается наименование переменной – это константа SAVED_TEXT, и значение – содержимое поля etText. Чтобы данные сохранились, необходимо выполнить commit. И для наглядности выводится сообщение, что данные сохранены.

```java
//Java
void saveText() {
sPref = getPreferences(MODE_PRIVATE);
Editor ed = sPref.edit();
ed.putString(SAVED_TEXT,
etText.getText().toString());
ed.commit();
Toast.makeText(this, "Text saved",
Toast.LENGTH_SHORT).show();
}
```

```kotlin
//Kotlin
fun saveText() {
var sPref = getPreferences(MODE_PRIVATE)
var ed = sPref.edit()
var etText = findViewById(R.id.etText) as
EditText
ed.putString(SAVED_TEXT,
        etText.getText().toString())
ed.commit()
Toast.makeText(this, "Text saved",
Toast.LENGTH_SHORT).show()
}
```

loadText – загрузка данных. Так же, как и saveText, с помощью метода getPreferences получаем объект sPref класса SharedPreferences. MODE_PRIVATE снова указывается, хотя и используется только при записи данных. Здесь Editor не используется т.к. сейчас необходимо только чтение данных. Чтение выполняется с помощью метода getString – в параметрах указываем константу - это имя, и значение по умолчанию (пустая строка). Далее записываем значение в поле ввода etText и выводим сообщение, что данные считаны.

```
//Java
void loadText() {
  sPref = getPreferences(MODE_PRIVATE);
  String savedText = sPref.getString(SAVED_TEXT, "");
  etText.setText(savedText);
  Toast.makeText(this, "Text loaded", Toast.LENGTH_SHORT).show();
}
```

```
//Kotlin
fun loadText() {
        var sPref = getPreferences(MODE_PRIVATE)
        var savedText = sPref.getString(SAVED_TEXT, "")
        var etText = findViewById(R.id.etText) as EditText
        etText.setText(savedText)
        Toast.makeText(this, "Text loaded", Toast.LENGTH_SHORT).show()
    }
```

Режимы доступа к sharedPreferences

MODE_APPEND
MODE_ENABLE_WRITE_AHEAD_LOGGING
MODE_MULTI_PROCESS
MODE_WORLD_READABLE
MODE_WORLD_WRITEABLE

Сделаем так, чтобы сохранение и загрузка происходили автоматически при закрытии и открытии приложения и не надо было нажимать кнопки. Для этого метод loadText будет вызываться в onCreate.

//Java

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  etText = (EditText) findViewById(R.id.etText);
  btnSave = (Button) findViewById(R.id.btnSave);
  btnSave.setOnClickListener(this);
  btnLoad = (Button) findViewById(R.id.btnLoad);
  btnLoad.setOnClickListener(this);
  loadText();
}
```

//Kotlin

```
override fun onCreate(savedInstanceState: Bundle?)
{
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    btnSave.setOnClickListener(this::onClick)
    btnLoad.setOnClickListener(this::onClick)
    loadText()
}
```

а метод saveText - в onDestroy

//Java

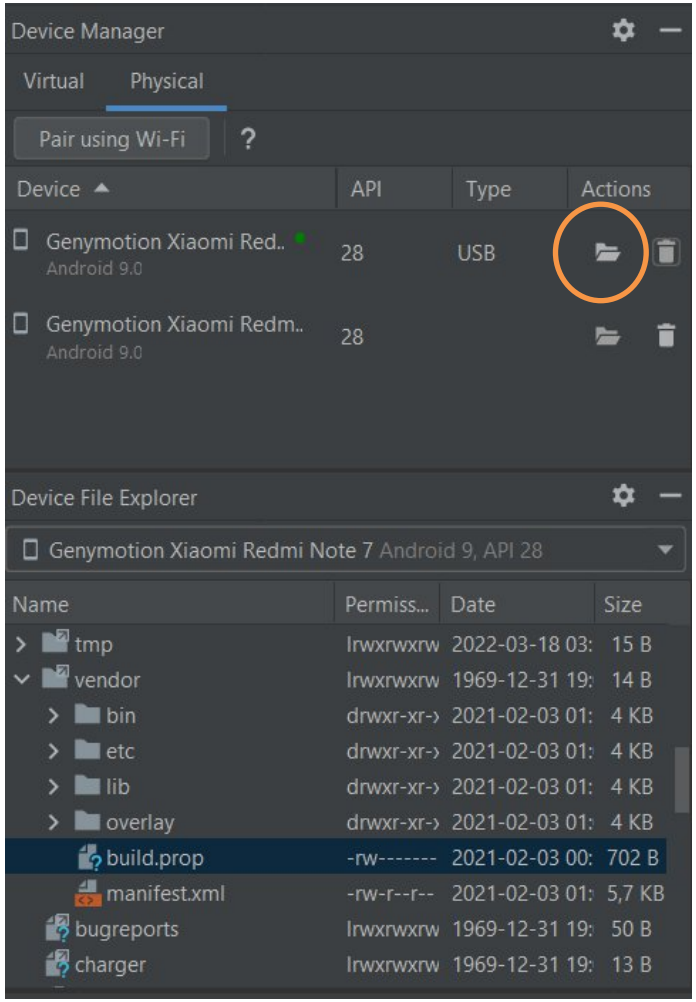
```
@Override
protected void onDestroy() {
  saveText();
  super.onDestroy();
}
```

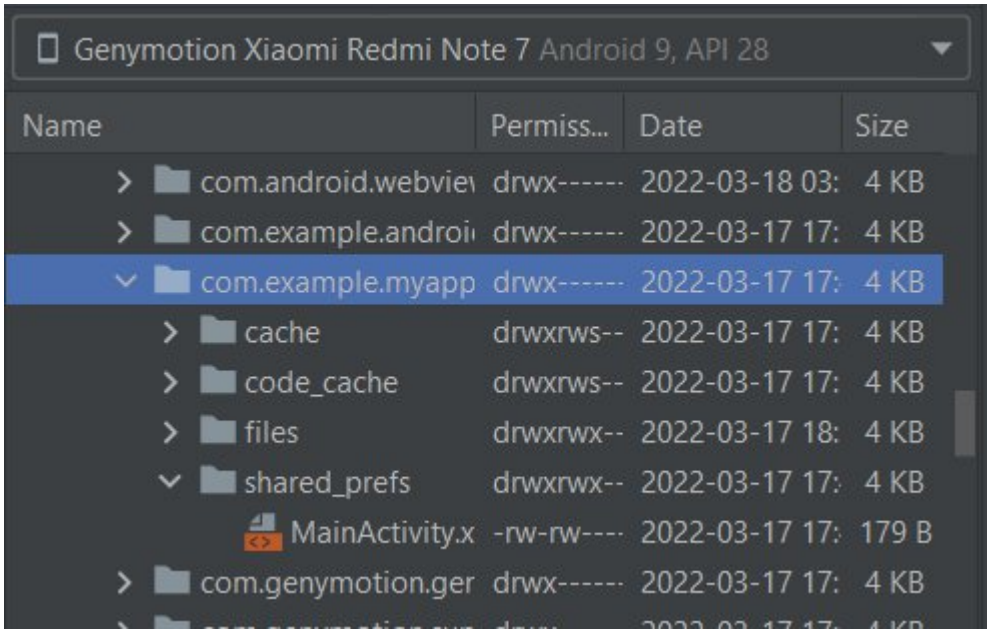
//Kotlin

```
override fun onDestroy() {
    saveText()
    super.onDestroy()
}
```

Теперь можно вводить данные, закрывать приложение, снова открывать и данные не потеряются. Кнопки Save и Load также работают. В какой момент сохранять данные все зависит от логики работы приложения. По нажатию кнопки, при закрытии программы или еще по какому-либо событию.

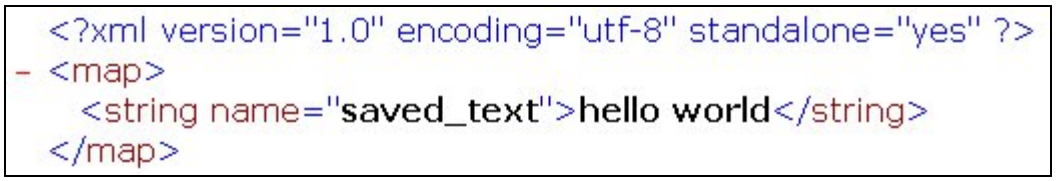
Preferences-данные сохраняются в файлы и их можно просмотреть. Для этого откройте файловую систему эмулятора. Открываем *data/data/com.example.myapplication/shared_prefs* и видим там файл MainActivity.xml.





Если его выгрузить на ПК и открыть – можно увидеть следующее:

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
 <map>
  <string name="saved_text">hello world</string>
  </map>
```



Обратите внимание, что в пути к файлу используется имя package.

Рассмотрим, откуда появилось наименование файла MainActivity.xml. Кроме метода getPreferences, который использовался, есть метод getSharedPreferences. Он выполняет абсолютно те же функции, но позволяет указать имя файла для хранения данных. Т.е., например, если бы в saveText использовался для получение SharedPreferences такой код:

```
sPref = getSharedPreferences("MyPref", MODE_PRIVATE);
```

то данные сохранились бы в файле MyPref.xml, а не в MainActivity.xml.

Теперь если рассмотреть исходный код метода getPreferences, то видим следующее:

```
public SharedPreferences getPreferences(int mode) {
  return getSharedPreferences(getLocalClassName(), mode);
}
```

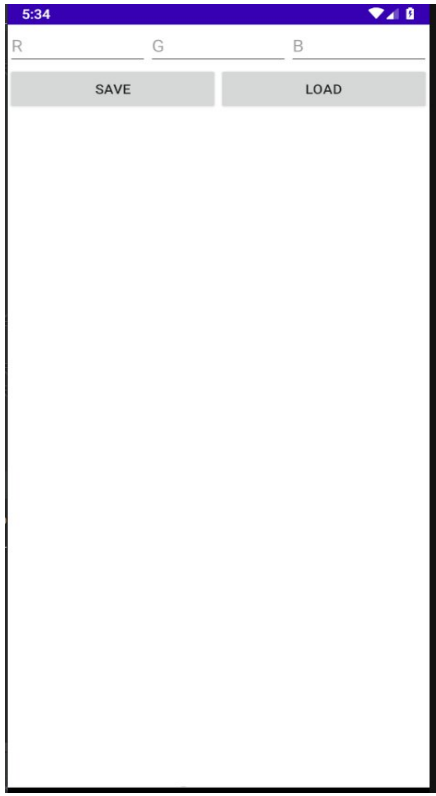
Используется метод getSharedPreferences, а в качестве имени файла берется имя класса текущего Activity. Отсюда и появилось имя файла MainActivity.xml.

-используете getPreferences, если работаете с данными для текущего Activity и не хотите выдумывать имя файла.

- используете getSharedPreferences, если сохраняете, например, данные - общие для нескольких Activity и сами выбираете имя файла для сохранения.

```
<?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"
    android:id="@+id/lo" >

     <LinearLayout



        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

<EditText
    android:id="@+id/etTextr"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:hint="R" >

    </EditText>

       <EditText
           android:id="@+id/etTextg"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_weight="1"
           android:hint="G" >

    </EditText>

    <EditText
        android:id="@+id/etTextb"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="B" >

    </EditText>
    </LinearLayout>
```

```
    <LinearLayout


        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btnSave"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Save" >

        </Button>

        <Button
            android:id="@+id/btnLoad"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Load" >


        </Button>
    </LinearLayout>
```



//Java

```
package com.example.lec9_1;

import android.app.Activity;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Toast;

public class MainActivity extends Activity
implements OnClickListener {
  EditText etTextr,etTextg, etTextb;
  Button btnSave, btnLoad;
  SharedPreferences sPref;
  final String SAVED_TEXT = "saved_text";
  final String R_Color="red";
  final String G_Color="green";
  final String B_Color="blue";
  LinearLayout lo;
  int col;
  Color myRGB;
```

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  etTextr = (EditText)
          findViewById(R.id.etTextr);
  etTextg = (EditText)
          findViewById(R.id.etTextg);
  etTextb = (EditText)
          findViewById(R.id.etTextb);

  btnSave = (Button) findViewById(R.id.btnSave);
  btnSave.setOnClickListener(this);
  btnLoad = (Button) findViewById(R.id.btnLoad);
  btnLoad.setOnClickListener(this);
}

public void onClick(View v) {
  switch (v.getId()) {
    case R.id.btnSave:
      saveColor();
      break;
    case R.id.btnLoad:
      loadColor();
      break;
    default:
      break;
}
}
```

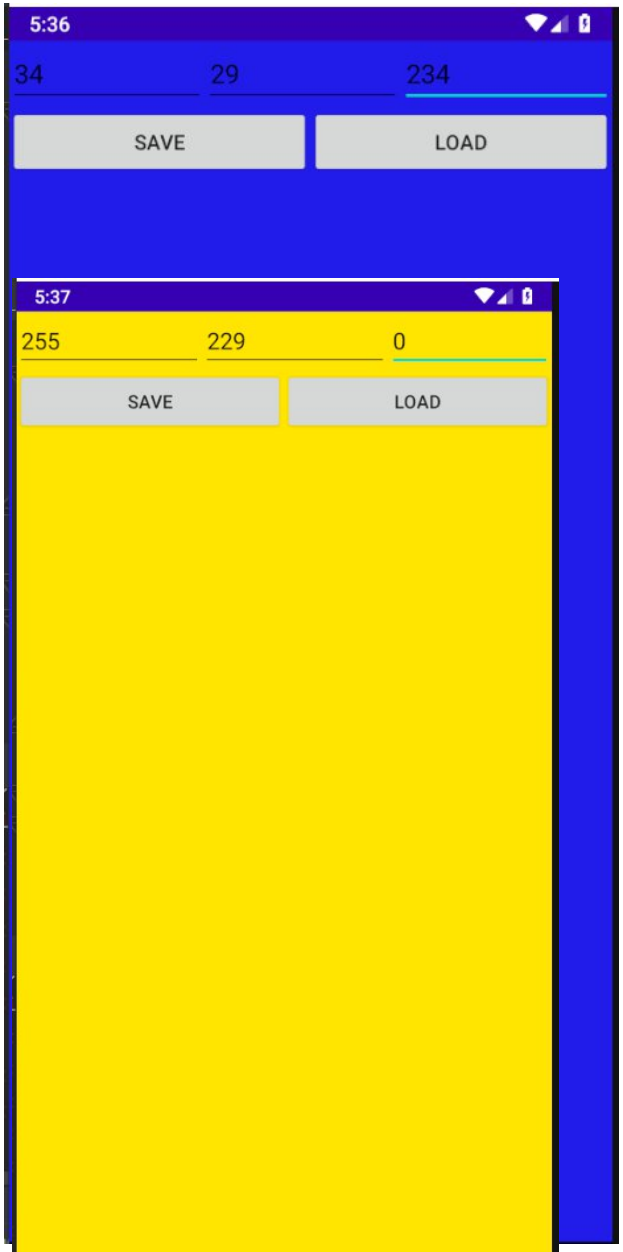
```
void saveColor() {
  sPref = getPreferences(MODE_PRIVATE);
  Editor ed = sPref.edit();
  ed.putString(R_Color,
          etTextr.getText().toString());
  ed.putString(G_Color,
          etTextg.getText().toString());
  ed.putString(B_Color,
          etTextb.getText().toString());
  ed.commit();
  Toast.makeText(this, "Text saved",
          Toast.LENGTH_SHORT).show();
}

void loadColor() {

  sPref = getPreferences(MODE_PRIVATE);
  String rr = sPref.getString(R_Color, "");
  String gg = sPref.getString(G_Color, "");
  String bb = sPref.getString(B_Color, "");

  lo=  (LinearLayout)findViewById(R.id.lo);
  col = myRGB.rgb(Integer.parseInt(rr),
          Integer.parseInt(gg),
          Integer.parseInt(bb));
  lo.setBackgroundColor(col);
  etTextr.setText(rr);
  etTextg.setText(gg);
  etTextb.setText(bb);
  Toast.makeText(this, "Text loaded",
          Toast.LENGTH_SHORT).show();
}
```

```
@Override
protected void onDestroy() {
  saveColor();
  super.onDestroy();
}
}
```

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <map>
    <string name="blue">120</string>
    <string name="green">240</string>
    <string name="saved_text">hello world</string>
    <string name="red">0</string>
  </map>
```

```kotlin
//Kotlin
import android.graphics.Color
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.EditText
import android.widget.LinearLayout
import android.widget.Toast
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

    val R_color = "red"
    val G_color = "green"
    val B_color = "blue"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        btnSave.setOnClickListener(this::onClick)
        btnLoad.setOnClickListener(this::onClick)
        loadColor()
    }

    override fun onDestroy() {
        saveColor()
        super.onDestroy()
    }

    fun onClick(view: View){
        when (view.id){
            R.id.btnSave -> saveColor()
            R.id.btnLoad -> loadColor()
        }
    }
```

```
 fun saveColor() {
        var sPref = getPreferences(MODE_PRIVATE)
        var ed = sPref.edit()
        var etTextr = findViewById(R.id.etTextr) as EditText
        var etTextg = findViewById(R.id.etTextg) as EditText
        var etTextb = findViewById(R.id.etTextb) as EditText
        ed.putString(R_color, etTextr.getText().toString())
        ed.putString(G_color, etTextg.getText().toString())
        ed.putString(B_color, etTextb.getText().toString())
        ed.commit()
        Toast.makeText(this, "Text saved", Toast.LENGTH_SHORT).show()
    }

    fun loadColor() {
        var sPref = getPreferences(MODE_PRIVATE)
        var rr = sPref.getString(R_color, "")
        var gg = sPref.getString(G_color, "")
        var bb = sPref.getString(B_color, "")
        var lo = findViewById(R.id.lo) as LinearLayout
        lo.setBackgroundColor(Color.rgb(rr!!.toInt(), gg!!.toInt(),
bb!!.toInt()))
        var etTextr = findViewById(R.id.etTextr) as EditText
        var etTextg = findViewById(R.id.etTextg) as EditText
        var etTextb = findViewById(R.id.etTextb) as EditText
        etTextr.setText(rr)
        etTextg.setText(gg)
        etTextb.setText(bb)
        Toast.makeText(this, "Text loaded", Toast.LENGTH_SHORT).show()
    }
}
```


5:36
34
29
234
SAVE
LOAD

5:37
255
229
0
SAVE
LOAD


5:38
0
229
0
SAVE
LOAD

5:38
200
30
20
SAVE
LOAD


5:40
70
70
70
SAVE
LOAD

5:41
0
255
230
SAVE
LOAD

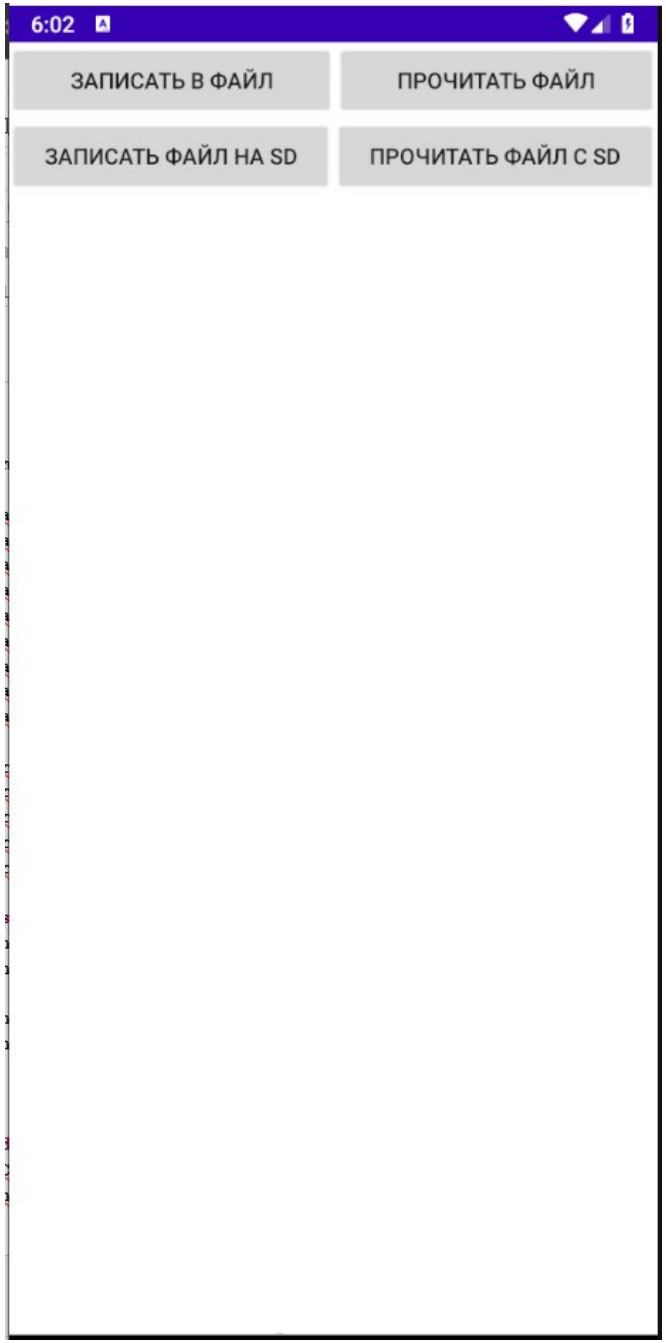
Хранение данных. Работа с файлами

Работа с файлами в Android не сильно отличается от таковой в Java. Рассмотрим, как записать/прочитать файл во внутреннюю память и на flash карту.

```
<?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" >

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

        <Button
            android:id="@+id/btnWrite"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="onclick"
            android:text="@string/write_file" >
        </Button>

        <Button
            android:id="@+id/btnRead"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="onclick"
            android:text="@string/read_file" >
        </Button>
    </LinearLayout>
```

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/btnWriteSD"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="onclick"
            android:text="@string/write_file_sd" >
        </Button>

        <Button
            android:id="@+id/btnReadSD"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="onclick"
            android:text="@string/read_file_sd" >
        </Button>
    </LinearLayout>
      <EditText
            android:id="@+id/editText"
            android:layout_width="match_parent"
            android:layout_height="1000px"/>

</LinearLayout>
```

```
//Java
package com.example.lec9_4;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
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.View;

public class MainActivity extends Activity {
final String LOG_TAG = "myLogs";
final String FILENAME = "file"; //Имя файла вот тута
final String DIR_SD = "MyFiles";
final String FILENAME_SD = "fileSD"; //И тута тоже
String text = “”; //Для текстового поля
EditText editText;
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  editText = findViewById(R.id.editText);
}
```



```
public void onclick(View v) {
  switch (v.getId()) {
    case R.id.btnWrite:
      writeFile();
      break;
    case R.id.btnRead:
      readFile();
      break;
    case R.id.btnWriteSD:
      writeFileSD();
      break;
    case R.id.btnReadSD:
      readFileSD();
      break;
}
}

void writeFile() {
  try {
// отрываем поток для записи
    BufferedWriter bw = new BufferedWriter(new
          OutputStreamWriter(
          openFileOutput(FILENAME,
          MODE_PRIVATE)));
// пишем данные
    bw.write("Содержимое файла");
// закрываем поток
    bw.close();
    Log.d(LOG_TAG, "Файл записан");
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

```
void readFile() {
  try {
// открываем поток для чтения
    BufferedReader br = new BufferedReader(new
          InputStreamReader(openFileInput(FILENAME)));
    String str = "";
// читаем содержимое
    while ((str = br.readLine()) != null) {
      //text += str;
      Log.d(LOG_TAG, str);
    }
    // editText.setText(text); //Для вывода на экране
  } catch (FileNotFoundException e) {
     e.printStackTrace();
  } catch (IOException e) {
     e.printStackTrace();
  }
}
```

```
void readFileSD() {
// проверяем доступность SD

if(!Environment.getExternalStorageState().eq
          uals(Environment.MEDIA_MOUNTED)) {
   Log.d(LOG_TAG, "SD-карта не доступна: "
   + Environment.getExternalStorageState());
   return;
 }
// получаем путь к SD
  File sdPath =
          Environment.getExternalStorageDire
          ctory();
// добавляем свой каталог к пути
  sdPath = new File(sdPath.getAbsolutePath()
          + "/" + DIR_SD);
// формируем объект File, который содержит
путь к файлу
  File sdFile = new File(sdPath,
          FILENAME_SD);
  try {
// открываем поток для чтения
    BufferedReader br = new
          BufferedReader(new
          FileReader(sdFile));
    String str = "";
// читаем содержимое
    while ((str = br.readLine()) != null) {
       Log.d(LOG_TAG, str);
    }
  } catch (FileNotFoundException e) {
      e.printStackTrace();
  } catch (IOException e) {
      e.printStackTrace();
  }
}
```

```
void writeFileSD() {
// проверяем доступность SD
  if (!Environment.getExternalStorageState().equals(
          Environment.MEDIA_MOUNTED)) {
     Log.d(LOG_TAG,"SD-карта не доступна: "
          + Environment.getExternalStorageState());
     return;
   } // получаем путь к SD
   File sdPath = Environment.getExternalStorageDirectory();
// добавляем свой каталог к пути
   sdPath = new File(sdPath.getAbsolutePath() + "/" +
          DIR_SD);
   sdPath.mkdirs(); // создаем каталог
// формируем объект File, который содержит путь к файлу
   File sdFile = new File(sdPath, FILENAME_SD);
   try {// открываем поток для записи
      BufferedWriter bw = new BufferedWriter(new
          FileWriter(sdFile));
// пишем данные
      bw.write("Содержимое файла на SD");
// закрываем поток
      bw.close();
      Log.d(LOG_TAG, "Файл записан на SD: " +
          sdFile.getAbsolutePath());
   } catch (IOException e) {
      e.printStackTrace();
   }
 }
}
```

```kotlin
//Kotlin
package com.example.lec5_1

import android.content.Context
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Environment
import android.view.View
import kotlinx.android.synthetic.main.activity_main.*
import android.util.Log
import java.io.*

class MainActivity : AppCompatActivity() {

    val LOG_TAG = "myLogs"
    val FILENAME = "file"

    val DIR_SD = "MyFiles"
    val FILENAME_SD = "fileSD"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        btnWrite.setOnClickListener(this::onClick)
        btnRead.setOnClickListener(this::onClick)
        btnWriteSD.setOnClickListener(this::onClick)
        btnReadSD.setOnClickListener(this::onClick)
    }
    fun onClick(view: View){
        when (view.id){
            R.id.btnWrite -> writeFile()
            R.id.btnRead -> readFile()
            R.id.btnWriteSD -> writeFileSD()
            R.id.btnReadSD -> readFileSD()
        }
    }
```

```
fun writeFile() {
    try {
        // отрываем поток для записи
        var bw = BufferedWriter(OutputStreamWriter(openFileOutput(FILENAME,
            Context.MODE_PRIVATE)))
        // пишем данные
        bw.write("Содержимое файла")
        // закрываем поток
        bw.close()
        Log.d(LOG_TAG, "Файл записан")
    } catch (e: FileNotFoundException) {
        e.printStackTrace()
    } catch (e: IOException) {
        e.printStackTrace()
    }
}

fun readFile() {
    try {
        // открываем поток для чтения
        val br = BufferedReader(InputStreamReader(openFileInput(FILENAME)))
        // читаем содержимое
        while (br.readLine() != null) {
            Log.d(LOG_TAG, br.readLine())
        }
    } catch (e: FileNotFoundException) {
        e.printStackTrace()
    } catch (e: IOException) {
        e.printStackTrace()
    }
}
```

```
fun writeFileSD() {
    // проверяем доступность SD
    if (!Environment.getExternalStorageState().equals(
            Environment.MEDIA_MOUNTED)) {
        Log.d(LOG_TAG,
            "SD-карта не доступна: "
                    + Environment.getExternalStorageState())
        return;
    }
    // получаем путь к SD
    var sdPath = Environment.getExternalStorageDirectory()
    // добавляем свой каталог к пути
    sdPath = File(sdPath.getAbsolutePath() + "/" + DIR_SD)
    // формируем объект File, который содержит путь к файлу
    var sdFile = File(sdPath, FILENAME_SD)
    try {
        // открываем поток для чтения
        var br = BufferedReader(FileReader(sdFile))
        // читаем содержимое
        while (br.readLine() != null) {
            Log.d(LOG_TAG, br.readLine())
        }
    } catch (e : FileNotFoundException) {
        e.printStackTrace()
    } catch (e : IOException) {
        e.printStackTrace()
    }
}
```

```
fun readFileSD() {
        // проверяем доступность SD
        if (Environment.getExternalStorageState() != Environment.MEDIA_MOUNTED) {
            Log.d(
                LOG_TAG,
                "SD-карта не доступна: " + Environment.getExternalStorageState()
            )
            return
        }
        // получаем путь к SD
        var sdPath = Environment.getExternalStorageDirectory()
        // добавляем свой каталог к пути
        sdPath = File(sdPath.absolutePath + "/" + DIR_SD)
        sdPath.mkdirs() // создаем каталог
        // формируем объект File, который содержит путь к файлу
        val sdFile = File(sdPath, FILENAME_SD)
        try {
            // открываем поток для записи
            val bw = BufferedWriter(FileWriter(sdFile))
            // пишем данные
            bw.write("Содержимое файла на SD")
            // закрываем поток
            bw.close()
            Log.d(LOG_TAG, "Файл записан на SD: " + sdFile.absolutePath)
        } catch (e: IOException) {
            e.printStackTrace()
        }
    }
```

В onclick обрабатываются нажатия 4-х кнопок и вызываются соответствующие методы.

writeFile – запись файла во внутреннюю память. Используется метод openFileOutput, который принимает имя файла и режим записи: MODE_PRIVATE – файл доступен только этому приложению, MODE_WORLD_READABLE – файл доступен для чтения всем, MODE_WORLD_WRITEABLE - файл доступен для записи всем, MODE_APPEND – запись в файл будет продолжена.

readFile – чтение файла из внутренней памяти. Используем метод openFileInput, принимающий на вход имя файла. Здесь вы можете задать только имя файла.

```
void writeFile() {
  try {
// отрываем поток для записи
    BufferedWriter bw = new BufferedWriter(new
          OutputStreamWriter(
          openFileOutput(FILENAME,
          MODE_PRIVATE)));
// пишем данные
    bw.write("Содержимое файла");
// закрываем поток
    bw.close();
    Log.d(LOG_TAG, "Файл записан");
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
void readFile() {
  try {
// открываем поток для чтения
    BufferedReader br = new BufferedReader(new
          InputStreamReader(openFileInput(FILENAM
    String str = "";
// читаем содержимое
    while ((str = br.readLine()) != null) {
      Log.d(LOG_TAG, str);
    }
  } catch (FileNotFoundException e) {
     e.printStackTrace();
  } catch (IOException e) {
     e.printStackTrace();
  }
}
```

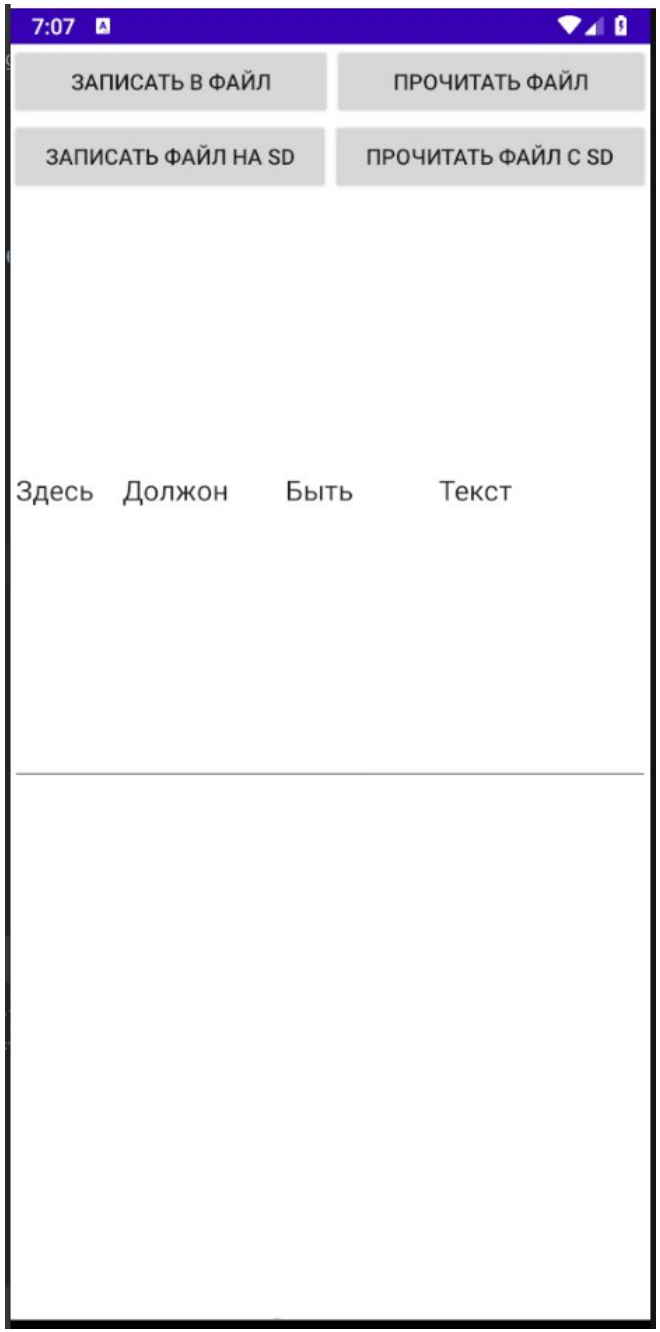
writeFileSD – запись файла на SD. Используется метод getExternalStorageState для получения состояния SD-карты. В данном случае необходимо состояние MEDIA_MOUNTED – когда SD-карта вставлена и готова к работе. Далее необходимо получить путь к SD-карте (метод getExternalStorageDirectory), добавить свой каталог и имя файла, после чего создается каталог и данные записываются в файл.

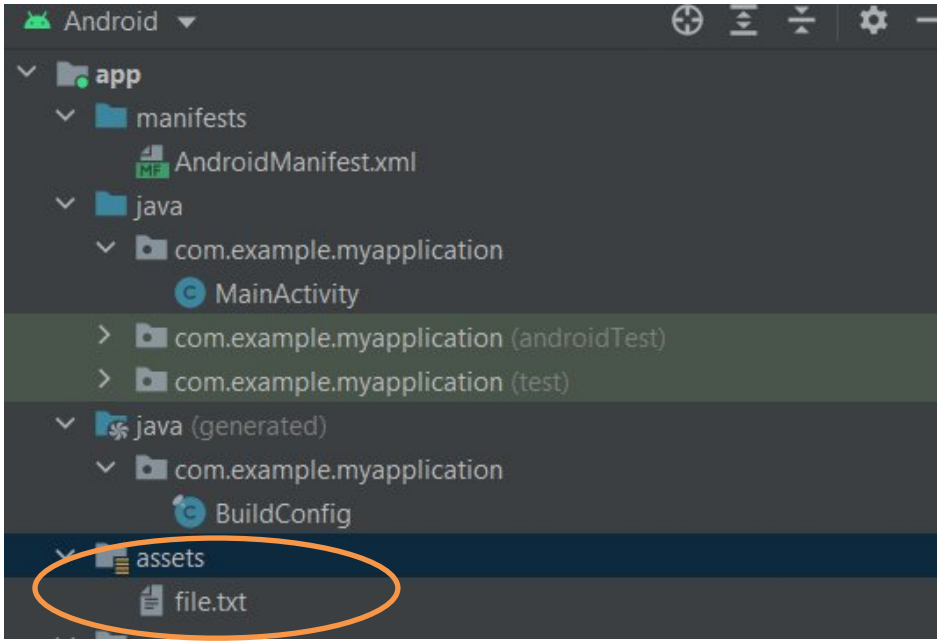
```
void writeFileSD() {
// проверяем доступность SD
  if (!Environment.getExternalStorageState().equals(
          Environment.MEDIA_MOUNTED)) {
     Log.d(LOG_TAG,"SD-карта не доступна: "
          + Environment.getExternalStorageState());
     return;
   } // получаем путь к SD
   File sdPath = Environment.getExternalStorageDirectory();
// добавляем свой каталог к пути
   sdPath = new File(sdPath.getAbsolutePath() + "/" +       DIR_SD);
   sdPath.mkdirs(); // создаем каталог
// формируем объект File, который содержит путь к файлу
   File sdFile = new File(sdPath, FILENAME_SD);
   try {// открываем поток для записи
      BufferedWriter bw = new BufferedWriter(new
          FileWriter(sdFile));
// пишем данные
      bw.write("Содержимое файла на SD");
// закрываем поток
      bw.close();
      Log.d(LOG_TAG, "Файл записан на SD: " +
          sdFile.getAbsolutePath());
   } catch (IOException e) {
      e.printStackTrace();
   }
 }
```

readFileSD – чтение файла с SD карты. Все аналогично предыдущему методу, только файл не пишем, а читаем.

```
void readFileSD() {
// проверяем доступность SD
  if(!Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
   Log.d(LOG_TAG, "SD-карта не доступна: « + Environment.getExternalStorageState());
   return;
  }
// получаем путь к SD
  File sdPath = Environment.getExternalStorageDirectory();
// добавляем свой каталог к пути
  sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR_SD);
// формируем объект File, который содержит путь к файлу
  File sdFile = new File(sdPath, FILENAME_SD);
  try {
// открываем поток для чтения
    BufferedReader br = new BufferedReader(new FileReader(sdFile));
    String str = "";
// читаем содержимое
    while ((str = br.readLine()) != null) {
       Log.d(LOG_TAG, str);
    }
  } catch (FileNotFoundException e) {
      e.printStackTrace();
  } catch (IOException e) {
      e.printStackTrace();
  }
}
```

Осталось в манифест добавить разрешение на работу с файлами на SD - android.permission. **WRITE_EXTERNAL_STORAGE.**





Чтение из файла

В Android Q изменен способ доступа приложений к файлам во внешнем хранилище устройства. Android Q заменяет разрешения READ_EXTERNAL_STORAGE и WRITE_EXTERNAL_STORAGE более детальными, специфичными для носителя разрешениями (например, для доступа к файлам других приложений в общей коллекции "Фото и видео" требуется разрешение READ_MEDIA_IMAGES или READ_MEDIA_VIDEO, в зависимости от типа файла, к которому должно обращаться ваше приложение), а приложениям, получающим доступ к собственным файлам на внешнем устройстве хранения, не требуются специальные разрешения.