**Чтение и запись файлов**

В Android можно сохранять файлы непосредственно на мобильном устройстве или на внешнем носителе данных (*например, SDкарте*). По умолчанию другие приложения не могут обращаться к этим файлам.

Операции вводавывода в Android аналогичны операциям в стандартных ***Java-программах.***Android реализует потоки с помощью иерархии классов, определенных в пакете ***java.io***. Кроме этих классов в пакете***java.io*** определено множество специализированных классов для операций ввода вывода.

Чтобы прочитать данные из файла, необходимо вызвать **метод Context.openFileinput()** и передать в качестве параметра имя файла. Метод возвращает стандартный ***Java-объект*Fileinputstream**. Например, код для чтения данных из текстового файла может выглядеть так:

InputStream inStream openFiieinput(file.txt);

InputStreamReader sr = new InputStreamReader(inStream);

// создаем буфер для чтения файла

BufferedReader reader new BufferedReader(sr);

String str;

StringBuffer buffer new StringBuffer();

// читаем данные в буфер

while ((str reader.readLine()) null) buffer.append(str + "\n");

inStream.close();

Чтобы записывать в файл, необходимо вызвать **метод context.openFileOutput()**, передав ему имя файла как параметр. Этот метод возвращает **объект FileOutputstream**. Вызов этих методов для данного файла из другого приложения не будет работать, обратиться вы можете только к своим файлам. Пример записи строки данных в файл***file.txt*** может быть следующим:

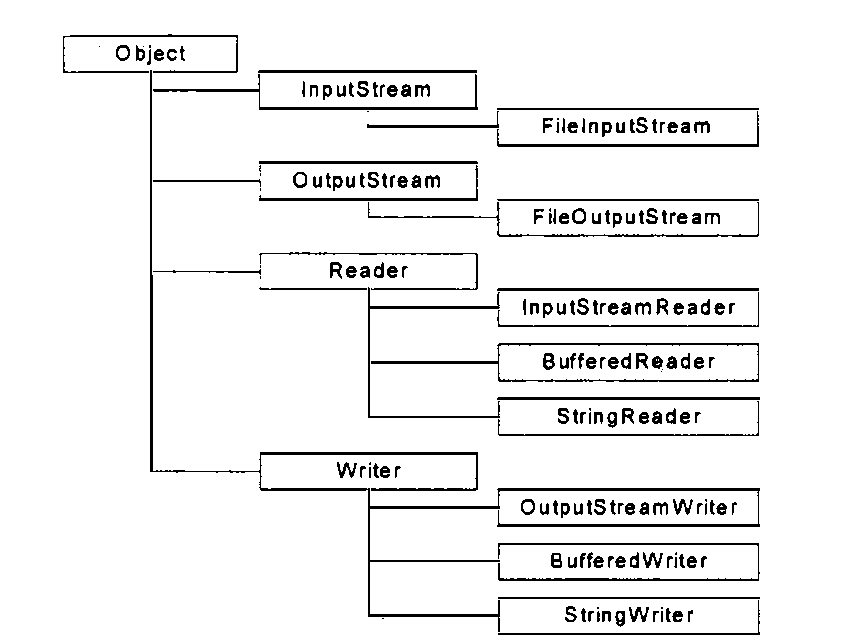
String data;

OutputStream outStream = openFileOutput(file.txt, 0);

OutputStreamWriter sw new OutputStreamWriter(outStream);

sw.write (data);

sw.close();



Если имеется статический файл, который надо упаковать с вашим приложением во время компиляции проекта, можно сохранить его в каталоге проекта в папке *res/raw/*, а затем открыть его **методом Resources.openRawResource()**. Он возвращает **объект Inputstrearn**, который можно использовать для чтения файла. После окончания работы с потоком не забудьте его закрыть, вызвав **метод close()**.

**Обрагиге внимание:**Методы ***openFilelnput()*** и ***openFileOutput()*** не принимают полного пути к файлу (*например, path/files/file.txt*), только простые имена файла.

В качестве примера приложения, записывающего и читающего данные из файлов, создадим простейший текстовый редактор с виджетом EditText и меню для открытия файла и сохранения его после редактирования. Создайте в Eclipse новый проект, заполнив поля в окне *New Android Project*:

* Project name — ContactEditor;
* Application name — ReadWrite File Sample;
* Package name — com.samples.filesrw;
* Create Activity — EditorActivity.

Код XMLсхемы разметки main.xml, в которой находится единственный элемент EditText:

**Файл разметки main.xml**

<?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">

<EditText

android:id="@+id/edit"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:singleLine="false"/>

</LinearLayout>

В классе деятельности приложения **EditorActivity**определим меню из трех пунктов— *Open, Save и Exit.* Во внутренних методах **openFile()** и **saveFile()** реализуем операции по открытию и сохранению файла, приведенные ранее.

**Файл класса деятельности EditorActivity.java**

package com.samples.filesrw;

import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import android.app.Activity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.EditText;

import android.widget.Toast;

public class EditorActivity extends Activity {

public static final int IDM\_OPEN = 101;

public static final int IDM\_SAVE = 102;

public static final int IDM\_EXIT = 103;

private final static String FILENAME = "file.txt";

private EditText mEdit;

@Override

public void onCreate(Bundle icicle) {

super.onCreate(icicle);

setContentView(R.layout.main);

mEdit = (EditText)findViewById(R.id.edit);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

menu.add(Menu.NONE, IDM\_OPEN, Menu.NONE, "Open")

.setIcon(R.drawable.ic\_menu\_open)

.setAlphabeticShortcut('o');

menu.add(Menu.NONE, IDM\_SAVE, Menu.NONE, "Save")

.setIcon(R.drawable.ic\_menu\_save)

.setAlphabeticShortcut('s');

menu.add(Menu.NONE, IDM\_EXIT, Menu.NONE, "Exit")

.setIcon(R.drawable.ic\_menu\_exit)

.setAlphabeticShortcut('x');

return(super.onCreateOptionsMenu(menu));

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case IDM\_OPEN:

openFile(FILENAME);

break;

case IDM\_SAVE:

saveFile(FILENAME);

break;

case IDM\_EXIT:

finish();

break;

default:

return false;

}

return true;

}

private void openFile(String fileName) {

try {

InputStream inStream = openFileInput(FILENAME);

if (inStream != null) {

InputStreamReader tmp =

new InputStreamReader(inStream);

BufferedReader reader = new BufferedReader(tmp);

String str;

StringBuffer buffer = new StringBuffer();

while ((str = reader.readLine()) != null) {

buffer.append(str + "\n");

}

inStream.close();

mEdit.setText(buffer.toString());

}

}

catch (Throwable t) {

Toast.makeText(getApplicationContext(),

"Exception: " + t.toString(), Toast.LENGTH\_LONG)

.show();

}

}

private void saveFile(String FileName) {

try {

OutputStreamWriter outStream =

new OutputStreamWriter(openFileOutput(FILENAME, 0));

outStream.write(mEdit.getText().toString());

outStream.close();

}

catch (Throwable t) {

Toast.makeText(getApplicationContext(),

"Exception: " + t.toString(), Toast.LENGTH\_LONG)

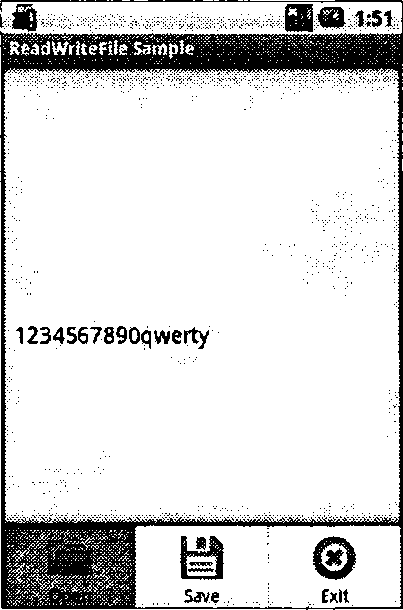
.show();

}

}

}

Внешний вид приложения, позволяющего читать и записывать текст в файл:



*Текстовый редактор с возможностью сохранения содержимого в файле*

В результате мы получили простую записную книжку, сохраняющую записи сколь угодно долгое время в файле. В качестве упражнения попробуйте усовершенствовать приложение, добавив возможность создания новых файлов, их открытия и сохранения, а также в случае необходимости удаления файлов.