МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«25» 02 2022 г.

Разработка мобильных приложений

Методические указания к практическим работам
для студентов направления подготовки 09.03.01

Курск 2022

УДК 004

Составители: А.В. Киселев, О.О. Яночкина

Рецензент
Кандидат технических наук, доцент *Ю.А. Халин*

Разработка мобильных приложений: методические указания к практическим работам для студентов направлений подготовки 09.03.01/ Юго-Зап. гос. ун-т; сост.; А.В. Киселев, О.О. Яночкина. – Курск, 2022. - 28 с.: - ил. 8 , табл. 1.– Библиогр.: с. 28

Содержат сведения по вопросам разработки мобильных приложений на базе операционной системы Android.

Предназначены для студентов направления подготовки 09.03.01 очной формы обучения.

Методические указания соответствуют рабочей программе дисциплины «Разработка мобильных приложений».

Текст печатается в авторской редакции

Подписано в печать . Формат 60*84 1/16.
Усл. печ. л.___. Уч.-изд. л.1,3 . Тираж 50 экз. Заказ 902 . Бесплатно.
Юго-Западный государственный университет.
305040 Курск, ул. 50 лет Октября, 94.

Содержание

Практическая работа №1. Взаимодействие с сервером.

Цель работы:

Изучить работу с потоками. Научиться работать с мультимедиа файлами. Изучить работу с классом AsyncTask

Теоретические сведения

Главный и обычный потоки

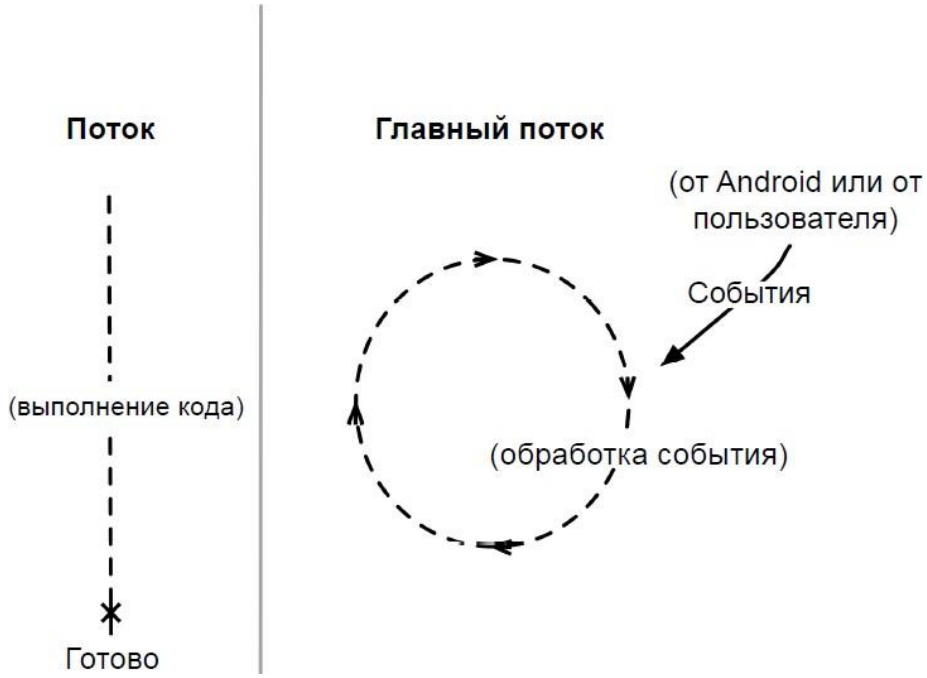

Рисунок 1.1 – Графическое представление потоков

Асинхронные потоки в Android

AsyncTask-это специальный абстрактный класс, предоставляющий набор методов для реализации:

- onPreExecute-для размещения инициализирующего кода (UI поток)
- doInBackground-для размещения тяжелого кода, который будет выполняться в другом потоке
- onProgressUpdate-для информирования о прогрессе (UI поток)
- onPostExecute-для обработки результата, возвращенного doInBackground(UI поток) и вспомогательных методов
- isCancelled-для получения информации об отмене задачи

- publishProgress-для перевода сообщения о прогрессе в UI поток с последующим вызовом onProgressUpdate

Последовательность выполнения методов AsyncTask

onPreExecute

doInBackground

onPostExecute

publishProgress

onProgressUpdate

Правила использования AsyncTask:

Объект AsyncTaskдолжен быть создан в UI-потоке

- Метод execute должен быть вызван в UI-потоке
- Метод executeможет быть запущен только один раз
- Не вызывайте методы onPreExecute, doInBackground, onPostExecuteи onProgressUpdate

Передача данных в AsyncTask

Объявляем класс

Class MyAsyncTaskextends AsyncTask<String, Integer, Long>{

…

}

- Первый параметр используется методом doInBackground protectedLong doInBackground(String... urls)
- Второй параметр используется методом onProgressUpdate protectedvoidonProgressUpdate(Integer… progress)
- Третий параметр используется методом onPostExecute protectedvoidonPostExecute(Longresult)

Промежуточные данные

Последовательность действий для передачи промежуточных данных в основной поток программы:

- В методе doInBackgroundвызываем метод publishProgress

- В методе onProgressUpdate обрабатываем переданный в publishProgress параметр и выводим прогресс

Метод get

- Возвращает результат выполнения метода doInBackground
- Вызывается из UI потока

```
MyAsyncTaskat = newMyAsyncTask();
…
result= at.get();
```

Задание на лабораторную работу Задание 1.

Рассмотрите пример передачи данных.

```
MyAsyncTaskat = newMyAsyncTask();
at.execute(“url1”, “url2”);

doInBackground(String... urls)
```

Задание 2. Рассмотрите пример вывода промежуточных данных.

```
@Override
Protected void doInBackground(String... urls)
        {try{
                int cnt= 0;
                for(Stringurl: urls)
                {
                        // обрабатываем первый параметр
                                …
                        // выводим промежуточные результаты
                        cnt++;
                        publishProgress(cnt);
        }
        TimeUnit.SECONDS.sleep(1);
} catch(InterruptedExceptione)
```

```
        {e.printStackTrace();
        }
        return null;
    }
    @Override
        protected void onProgressUpdate(Integer... values) {
            super.onProgressUpdate(values);
            tv.setText("обработано " + values[0] + " параметров");
    }
```

Задание 3. Проверьте пример создания простой асинхронной задачи

```
public class MainActivity extends
Activity{MyAsyncTask at;
TextView tv;

public void onCreate(Bundle savedInstance State) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        tv= (TextView)
        findViewById(R.id.tv); MyAsyncTask
        at = new MyAsyncTask();at.execute();
}
}
```

Задание 4. На основании изученных примеров разработать приложение сохраняющее статистику проигрываемых песен на радио Мегабайт. Для сохранения песни и названия необходимо создать базу данных, содержащую таблицу со следующими полями:

1) ID
2) Исполнитель

3) Название трека
4) Время внесения записи

При включении приложения необходимо производить проверку подключения к Интернету. В случае если подключение отсутствует – выводить всплывающее сообщение (Toast) с предупреждением о запуске в автономном режиме (доступен только просмотр внесенных ранее записей).

После включения приложение должно производить асинхронный опрос сервера с интервалом 20 секунд. Если название трека не совпадает с последней записью в таблице необходимо произвести запись в БД.

URL адрес, по которому можно получить информацию о текущем треке и исполнителе:

http://media.ifmo.ru/api_get_current_song.php

Формат возвращаемых данных – JSON.

В случае успешного выполнения запроса результат будет иметь вид:

{“result”: “success”, “info” : “*Исполнитель – Название трека*” }

В случае ошибки API вернет следующую строку:

{“result”: “error”, “info” : “*Информация об ошибке*” }

Для успешного взаимодействия с API при обращении к странице необходимо передавать логин (login) и пароль(password) как POST-параметры.

login**: 4707login**

password**: 4707pass**

Приложение должно содержать активити, позволяющее просматривать внесенные в базу данных записи.

Практическая работа №2. Хранение данных. Настройки ивнешние файлы.

Цель работы

Изучить инструменты хранения данных, а также работу с внешними файлами.

Теоретические сведения

Варианты взаимодействия с сервером

1. Сокеты

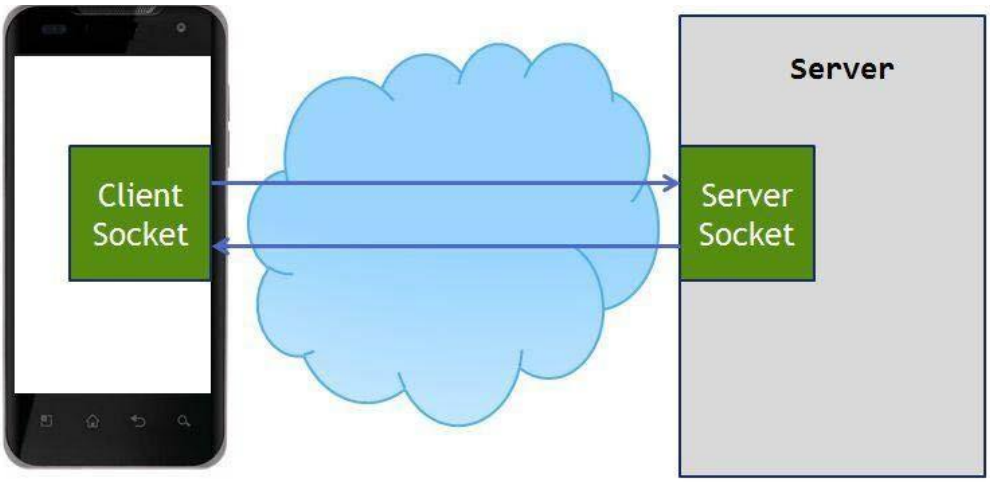

Рисунок 2.1 – Схема взаимодействия клиентского сокета с серверным

Используется в приложениях, где важна скорость доставки сообщения, важен порядок доставки сообщений и необходимо держать стабильное соединение с сервером. Такой способ зачастую реализуется в мессенджерах и играх.

2. Частые опросы

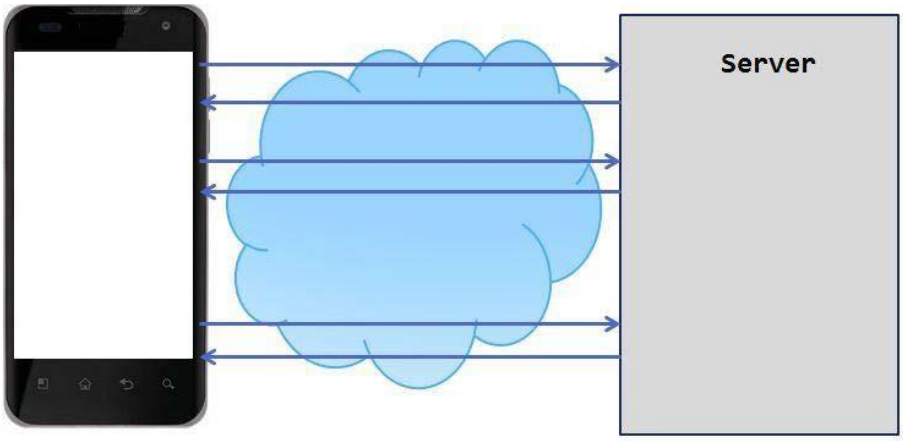

Рисунок 2.2 – Схема реализации частых опросов

Клиент посылает запрос на сервер и запрашивает новые данные. Сервер отвечает на запрос клиента и отдает все, что у него накопилось к этому моменту.

3. Длинные опросы

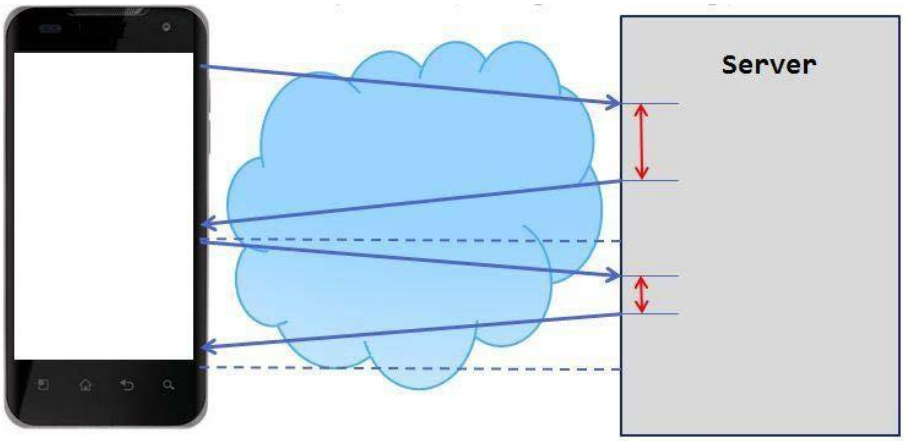

Рисунок 2.3 – Схема реализации длинных опросов

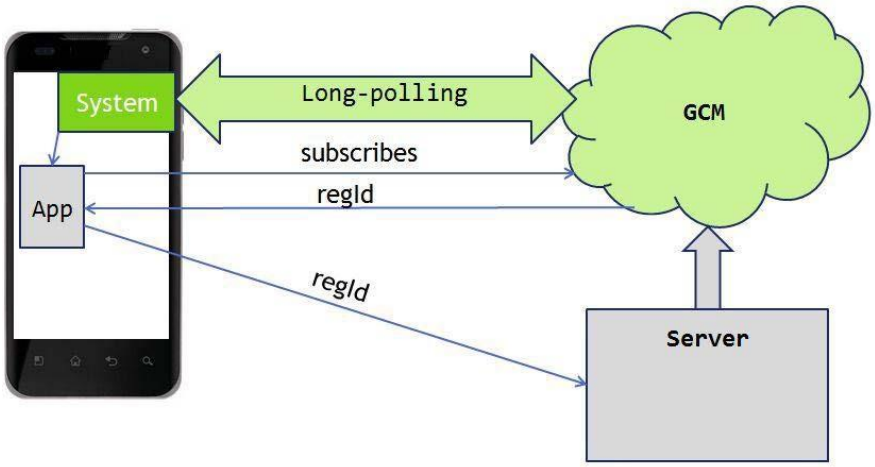

Рисунок 2.4 – Механизм реализации длинных опросовМетод длинных опросов

- Реализация этого подхода достаточно сложна на мобильном клиенте в первую очередь из-за нестабильности мобильного соединения.
- При данном подходе расходуется меньше трафика, чем при обычном polling’е, т.к. сокращается количество установок соединений с сервером.
- Механизм long polling, или пуш-уведомлений (push notifications), реализован в самой платформе Android. (ваше приложение подписывается у сервиса Google Cloud Messaging (GCM) на получение пуш-уведомлений)

Таблица 2.1 – Библиотеки для работы с сетью

java.net	Содержит классы, связанные с сетевыми функциями, в том числе сокеты потоков и датаграмм, протокол IP, а также общие средства для работы с HTTP. Это многоцелевой ресурс для работы с сетями.
java.io	Пакет не относится непосредственно к сетям. Его классы используются сокетами и соединениями, содержащимися в других пакетах Java. Они используются также для обмена с локальными файлами (что часто происходит при взаимодействии с сетью).
java.nio	Содержит классы, которые служат буфером для определенных типов данных. Удобен для организации сетевой связи между двумя конечными точками средствами Java.
org.apache.*	Набор пакетов, которые обеспечивают точный контроль и функции для HTTP-коммуникаций на основе Apache - популярного веб-сервера с открытым исходным кодом.
android.net	Содержит дополнительные сокеты доступа к сети в дополнение к основным классам java.net.*. Этот пакет включает в себя класс URI, который часто используется в разработке приложений Android, не связанных с сетью.
android.net.http	Содержит классы для работы с сертификатами SSL.
android.net.wifi	Содержит классы для реализации всех аспектов WiFi (802.11 Wireless Ethernet) на платформе Android.

Разрешения

- android.permission.INTERNET
- android.permission.ACCESS_NETWORK_STATE

Задание на лабораторную работу

Задание 1. Изучите пример подключения к

сети.public void myClickHandler(View view) {

...

```
ConnectivityManager connMgr = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
 NetworkInfo networkInfo =
connMgr.getActiveNetworkInfo();if (networkInfo != null &&
networkInfo.isConnected()) {
        // fetch data
} else {
        // display error
        }
        ...
}
```

Задание 2. Изучите код приложений

HttpURLConnection

```
private String downloadUrl(String myurl) throws IOException
        {InputStream is = null;
        // Only display the first 500 characters of the retrieved
        // web page content.
        int len = 500;
        try {
                URL url = new URL(myurl);
                HttpURLConnection       conn        =
                                        (HttpURLConnection)
        url.openConnection();
                conn.setReadTimeout(10000 /* milliseconds */);
                conn.setConnectTimeout(15000 /* milliseconds */);
                conn.setRequestMethod("GET"); conn.setDoInput(true);
                // Starts the query conn.connect();
                int response = conn.getResponseCode();
```

```
            Log.d(DEBUG_TAG, "The response is: " + response); is
            =conn.getInputStream();
            // Convert the InputStream into a string
            String contentAsString = readIt(is, len);
            return contentAsString;
        // Makes sure that the InputStream is closed after the app is
        // finished using it.
        } finally {
            if (is != null) {
            is.close();
            }
        }
}
```

Преобразование полученной информации к типу Srting

```
public String readIt (InputStream stream, int len) throws IOException,
UnsupportedEncodingException {
        Reader reader = null;
        reader = new InputStreamReader(stream, "UTF-8");
        char[] buffer = new char[len];
        reader.read(buffer); return
        new String(buffer);
        }
```

Http GET запрос

- Создаем HttpClient

  ```
  HttpClient client = new DefaultHttpClient();
  ```

- Создаем объект HttpGet

  ```
  HttpGet request = new HttpGet("http://www.example.com");
  ```

- Выполняем HTTP запрос

```
        HttpResponse response;
try {
        response = client.execute(request);
        Log.d("Response of GET request", response.toString());
        } catch (ClientProtocolException e) {
                // TODO Auto-generated catch block
        e.printStackTrace();
        } catch (IOException e) {
                // TODO Auto-generated catch block
        e.printStackTrace();
        }
```

Взаимодействие с сервером через сокеты

```
public class Requester extends Thread {
        Socket requestSocket;
        String message;
        StringBuilder returnStringBuffer = new
        StringBuilder();Message lmsg;
        int ch;
        @Override public void run()
                {try {
                this.requestSocket = new Socket("remote.servername.com",
        13);
                InputStreamReader isr = new
        InputStreamReader(this.requestSocket. getInputStream(), "ISO-8859-
        1");
                while ((this.ch = isr.read()) != -1) {
                this.returnStringBuffer.append((char) this.ch);
                        }
                this.message = this.returnStringBuffer.toString();
                this.lmsg = new Message();
```

```
                this.lmsg.obj = this.message;
                this.lmsg.what = 0;
                h.sendMessage(this.lmsg);
                this.requestSocket.close();
        }
        catch (Exception ee) {
                Log.d("sample application", "failed to read data"
                                +ee.getMessage());
                }
        }
}
```

Задание 3. Работа с внешними файлами.

Разработать мобильное приложение, позволяющее пользователю асинхронно скачивать файлы журнала Научно-технический вестник. Файлы хранятся на сервере в формате PDF и расположены по адресу:

http://ntv.ifmo.ru/file/journal/идентификатор_журнала.pdf

Не для всех ID имеются журналы, поэтому необходимо предусмотреть сообщение об отсутствии файла. В случае если файл не найден, ответ от сервера будет содержать главную страницу сайта.

Определить существует ли файл можно по возвращаемому сервером заголовку (параметр content-type).

Примеры ссылок:

http://ntv.ifmo.ru/file/journal/1.pdf – возвращен PDF файл

http://ntv.ifmo.ru/file/journal/2.pdf – файл не найден, возвращена главная страница сайта

Файлы должны храниться на устройстве в папке, создаваемой при первом запуске приложения (путь до папки и ее название определите самостоятельно).

После окончания загрузки файла должна становиться доступнойкнопка «Смотреть» и кнопка «Удалить».

При нажатии на кнопку «Смотреть» должно происходить открытие сохраненного на устройстве файла. Предусмотреть ошибку, если на устройстве не установлено приложение, открывающее PDF файлы.

При нажатии на кнопку «Удалить» загруженный файл должен удаляться с устройства.

Рисунок 2.5 – Интерфейс приложения

Задание 4. Хранение и чтение настроек.

При запуске приложения пользователю должно выводиться всплывающее полупрозрачное уведомление (popupWindow), с краткой инструкцией по использованию приложения (можете написать случайный текст), чекбоксом «Больше не показывать» и кнопкой «ОК».

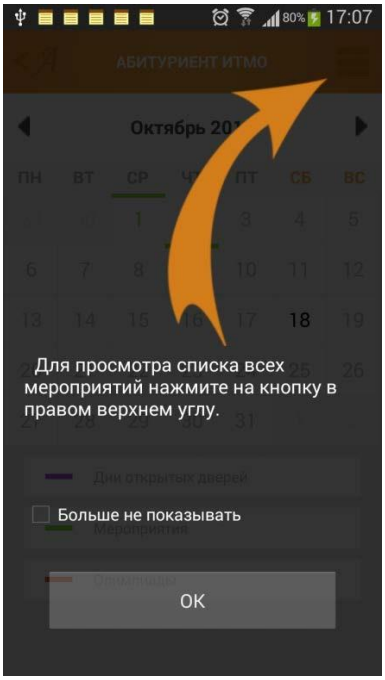

Рисунок 2.6. – Результат работы приложения

Если чекбокс был отмечен и нажата кнопка ОК, необходимо произвести сохранение данного параметра используя SharedPreferences. При следующем запуске приложения производить проверку параметра, и не выводить всплывающее сообщение, если чекбокс был отмечен.

Лабораторная работа №3. Уведомления.

Цель работы

Изучить инструменты хранения данных, а также работу с внешними файлами.

Теоретические сведения

Варианты хранения данных на устройстве

- Preferences (аналог INI-файлам в ОС Windows)
- SQLite - база данных, таблицы
- обычные файлы - внутренние и внешние (на SD карте)

Preferences

Значения сохраняются в виде пары: имя, значение

- Для хранения используется файл формата XML
- Путь к сохраняемым настройкам /data/data/YOUR_PACKAGE_NAME/shared_prefs/YOUR_PREFS_NAME.xml

Отличие методов getPreferences и getSharedPreferences

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

Метод getSharedPreferences

public abstract SharedPreferences getSharedPreferences (String name, int mode)

- Name – имя файла с настройками
- Mode – режим

  MODE_PRIVATE – режим по умолчанию. Файл доступен из только из текущего приложения.

  MODE_WORLD_READABLE (deprecated in API level 17) – разрешеночтение всеми приложениями.

  MODE_WORLD_WRITEABLE - (deprecated in API level 17) –

разрешена запись всеми приложениями.

MODE_MULTI_PROCESS – режим для взаимодействия с файлом из нескольких процессов

Создание экрана настроек

pref.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
        <CheckBoxPreference
                android:key="notif"
                android:summary="Enable notifications"
                android:title="Notifications">
        </CheckBoxPreference>
        <EditTextPreference
                android:key="address"
                android:summary="Address for notifications"
                android:title="Address">
    </EditTextPreference>
</PreferenceScreen>
```

Создание экрана настроек

PrefActivity.java

```
public class PrefActivity extends PreferenceActivity
{@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.pref);
        }
}
```

Создание экрана настроек

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">
        <TextView
                android:id="@+id/tvInfo"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="">
        </TextView>
</LinearLayout>
```

Создание экрана настроек

MainActivity.java

```
public class MainActivity extends Activity
        {TextView tvInfo;
        SharedPreferences sp;
        /** Called when the activity is first created. */
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                tvInfo = (TextView) findViewById(R.id.tvInfo);
                sp = PreferenceManager.getDefaultSharedPreferences(this);
        // полная очистка настроек
        // sp.edit().clear().commit();
}
protected void onResume() {
        Boolean notif = sp.getBoolean("notif", false);
```

```
        String address = sp.getString("address", "");
        String text = "Notifications are " + ((notif) ? "enabled, address = " +
address : "disabled");
        tvInfo.setText(text);
        super.onResume();
        }
        public boolean onCreateOptionsMenu(Menu menu) {
                MenuItem mi = menu.add(0, 1, 0, "Preferences");
        mi.setIntent(new Intent(this, PrefActivity.class));
                return super.onCreateOptionsMenu(menu);
        }
}
```

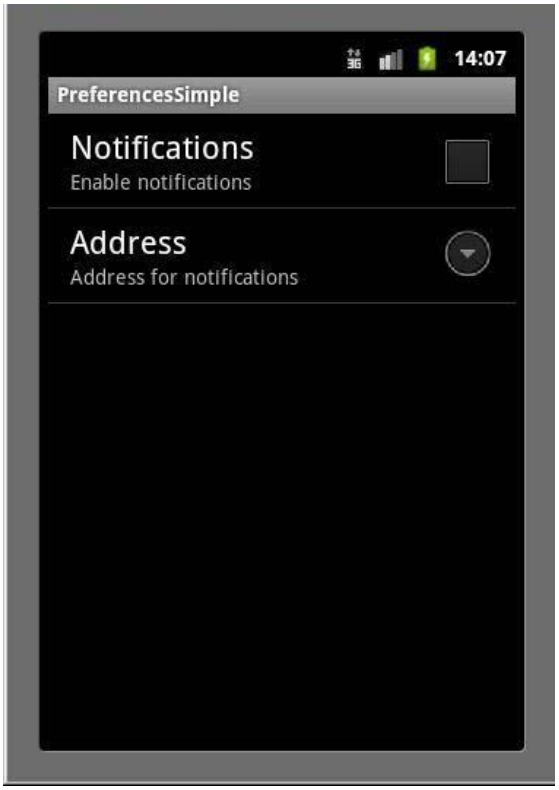

Рисунок 3.1 – Интерфейс приложения

Работа с файлами

Основные используемые классы

- BufferedReader, BufferedWriter
- InputStreamReader, OutputStreamWriter

- openFileInput, openFileOutput\
- FileReader, FileWriter
- File

Пример

```
public class MainActivity extends Activity
        {final String LOG_TAG =
        "myLogs"; final String FILENAME
        = "file";
        final String DIR_SD = "MyFiles";
        final String FILENAME_SD = "fileSD";
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }
        public void onclick(View v)
                {switch (v.getId()) {
                case R.id.btnWrite:
                        writeFile()
                        ;break;
                case R.id.btnRead:
                        readFile()
                        ;break;
                case
                        R.id.btnWriteS
                        D:
                        writeFileSD();
                        break;
```

```
            case
                R.id.btnReadS
                D:
                readFileSD();
                break;
        }
}
```

Запись файла в память устройства

```
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(); }
        atch (IOException e) {
        e.printStackTrace();
        }
}
```

Чтение файла из памяти устройства

```
void readFile()
{try {
// открываем поток для чтения BufferedReader br
= new BufferedReader(new InputStreamReader(
openFileInput(FILENAME)));
```

```
String str = "";
// читаем содержимое
while ((str = br.readLine()) != null)
        {Log.d(LOG_TAG, str);
}
} catch (FileNotFoundException e)
        {e.printStackTrace();
}
catch (IOException e) {
                e.printStackTrace();
        }
}
```

Запись на SD карту

```
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);
// создаем каталог sdPath.mkdirs();
// 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();
}
}
```

Чтение с SD карты

```
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();
}
}
```

Разрешение на работу с файлами на SD карте

- android.permission.READ_EXTERNAL_STORAGE
- android.permission.WRITE_EXTERNAL_STORAGE

Задание на лабораторную работу

Задание 1. Разработать мобильное приложение, позволяющее устанавливать напоминания.

Требования к приложению

1) Создание напоминаний и сохранение их в базу данных. В БД должны храниться следующие значения (Заголовок, Текст уведомления, дата уведомления)
2) Просмотр установленных уведомлений
3) Удаление уведомлений
4) Дата напоминания должна устанавливаться с помощью TimePickerDialog и DatePickerDialog
5) Стилизовать напоминание в Notification Center и Status bar (установить собственный лого)
6) При нажатии на уведомление в Notification Center переходить в активити приложения с полным текстом уведомления.

Классы для создания приложения: Notification, NotificationManager, PendingIntent, BroadcastReceiver, AlarmManager

Список литературы

1. Куркин, А.В. Программирование под платформу Android [Текст]: учебное пособие / Куркин А.В. – СПб.: Университет ИТМО, 2015. - 35 с.
2. Григорьева, И.И. Разработка мобильных приложений [Текст]: учебное пособие / И.И. Григорьева, С.С. Самборецкий. – Тюмень, 2015. - 125 с.
3. Дэрси, Л. Android за 24 часа. Программирование приложений под операционную систему Google [Текст] / Л.Дэрси, Ш.Кондер. - ; М.: Рид Групп, 2011. - 464 с.
4. Колисниченко, Д. Программирование для Android [Текст]: самоучитель / Д. Колисниченко. – М.: БХВ-Петербург - Москва, 2012. - 272 с.
5. Майер, Р. Android 4. Программирование приложений для планшетных компьютеров и смартфонов [Текст] / Р. Майер. – М.: Эксмо, 2013. - 816 с.
6. Медникс, З. Программирование под Android [Текст] / З.Медникс, Л.Дорнин, Б.Мик, М.Накамура. – СПб.: Питер, 2013. – 560 с.
7. Ретабоуил, С. Android NDK. Разработка приложений под Android на C/C++ [Текст] / С. Ретабоуил. – М.: ДМК Пресс, 2012. - 496 с.
8. Цехнер, М. Программирование игр под Android [Текст] / М. Цехнер. – СПб.: Питер, 2013. – 688 с.