

Министерство науки и высшего образования Российской Федерации

Калужский филиал  
федерального государственного бюджетного образовательного  
учреждения высшего образования  
**«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»**  
(КФ МГТУ им. Н.Э. Баумана)

**Ю.С. Белов, С.С. Гришунов**

**ИСПОЛЬЗОВАНИЕ БД В ANDROID ПРИЛОЖЕНИЯХ**  
Методические указания к выполнению лабораторной работы  
по курсу «Разработка мобильного программного обеспечения»

Калуга – 2019

УДК 004.42  
ББК 32.972.13  
Б435

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационных технологий».

Методические указания рассмотрены и одобрены:


- Кафедрой «Программного обеспечения ЭВМ, информационных технологий» (ИУ4-КФ) протокол № 51.4/9 от « 10 » апреля 2019 г.

Зав. кафедрой ИУ4-КФ

 к.т.н., доцент Ю.Е. Гагарин

- Методической комиссией факультета ИУ-КФ протокол № 11 от « 19 » апреля 2019 г.

Председатель методической  
комиссии факультета ИУ-КФ

 к.т.н., доцент М.Ю. Адкин

- Методической комиссией  
КФ МГТУ им.Н.Э. Баумана протокол № 7 от « 7 » 05 2019 г.

Председатель методической комиссии  
КФ МГТУ им.Н.Э. Баумана

 д.э.н., профессор О.Л. Перерва

Рецензент:

к.т.н., доцент кафедры ИУ6-КФ

 А.Б. Лачихина

Авторы

к.ф.-м.н., доцент кафедры ИУ4-КФ  
асс. кафедры ИУ4-КФ

 Ю.С. Белов  
 С.С. Гришунов

#### Аннотация

Методические указания по выполнению лабораторной работы по курсу «Разработка мобильного программного обеспечения» содержат описание работы с СУБД SQLite, способы сохранения результатов выполнения запросов к базе данных в списки, файлы и LogCat, а также особенности реализации Android-приложений с использованием списков и СУБД SQLite.

Предназначены для студентов 3-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ .....	5
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ .....	6
SQLite.....	7
НАЧАЛО РАБОТЫ С SQLite .....	8
СОХРАНЕНИЕ ДАННЫХ, ПОЛУЧЕННЫХ ИЗ БАЗЫ ДАННЫХ .....	12
ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ .....	15
ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ .....	15
ВАРИАНТЫ ЗАДАНИЙ .....	16
КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ .....	18
ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ .....	18
ОСНОВНАЯ ЛИТЕРАТУРА .....	19
ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА .....	19

## **ВВЕДЕНИЕ**

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Разработка мобильного программного обеспечения» на кафедре «Программное обеспечение ЭВМ, информационные технологии и прикладная математика» факультета фундаментальных наук Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 3-го курса направления подготовки 09.03.04 «Программная инженерия», содержат описание работы с СУБД SQLite, способы сохранения результатов выполнения запросов к базе данных в списки, файлы и LogCat, а также особенности реализации Android-приложений с использованием списков и СУБД SQLite и задание на выполнение лабораторной работы.

Для выполнения лабораторной работы студенту необходимы минимальные знания по программированию на высокоуровневом языке программирования Kotlin.

Требования к программному обеспечению:

- ОС Windows 7/8/10 / Ubuntu 16.04 (или старше)
- Android Studio

## **ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ**

Целью выполнения лабораторной работы является формирование практических навыков разработки приложений с использованием СУБД SQLite, списков и файлов при разработке Android-приложений с несколькими Activity.

Основными задачами выполнения лабораторной работы являются:

1. Научиться работать с СУБД SQLite.
2. Научиться сохранять результаты выполнения запросов к базе данных в списки, файлы и LogCat.
3. Понять особенности реализации Android-приложений с использованием списков и СУБД SQLite

Результатами работы являются:

- Подключенная СУБД SQLite к Android-приложению
- Программа с несколькими Activity, использующая СУБД SQLite, списки и файлы, для решения задачи согласно варианту задания
- Подготовленный отчет

## SQLITE

SQLite – это C- библиотека, реализующая движок базы данных SQL. Все данные хранятся в одном файле. Программы, использующие библиотеку SQLite, могут обращаться к базе данных с помощью языка SQL без работающего выделенного процесса СУБД. Это означает, что одновременные запросы (или параллельные пользователи) должны блокировать файл для безопасного изменения БД. Если в основном используется чтение данных, тогда никаких проблем нет, но если необходимо делать большое количество одновременных обновлений, то приложение будет тратить больше времени на синхронизацию блокировки файлов, чем делать настоящую работу.

### Типы данных SQLite

В SQLite применяется следующая система типов данных:

- INTEGER: представляет целое число, аналог типу Int в kotlin
- REAL: представляет число с плавающей точкой, аналог float и Double в kotlin
- TEXT: представляет набор символов, аналог String в kotlin
- BLOB: представляет массив бинарных данных, например, изображение, аналог типу Int в kotlin

SQLite не имеет классов, предназначенных для хранения дат и/или времени. Вместо этого, встроенные функции даты и времени в SQLite способны работать с датами и временем, сохраненными в виде значений TEXT, REAL и INTEGER в следующих форматах:

- TEXT как строка формата ISO8601 ("YYYY-MM-DD HH:MM:SS.SSS").
- REAL как числа юлианского календаря. То есть число дней с полудня 24 ноября 4714 г. до н.э. по Гринвичу в соответствии с ранним григорианским календарём.
- INTEGER как время Unix, — количество секунд с 1970-01-01 00:00:00 UTC.

В приложениях следует выбирать, в каком из этих форматов хранить даты и время, а затем можно свободно конвертировать из одного формата в другой с помощью встроенных функций даты и времени.

## НАЧАЛО РАБОТЫ С SQLITE

В Android имеется встроенная поддержка одной из распространенных систем управления базами данных - [SQLite](#). Для этого в пакете `android.database.sqlite` определен набор классов, которые позволяют работать с базами данных SQLite. И каждое приложение может создать свою базу данных.

Чтобы использовать SQLite в Android, надо создать базу данных с помощью выражение на языке SQL. После этого база данных будет храниться в каталоге приложения по [пути](#):

```
DATA/data/[Название_приложения]/databases/[Название_файла_
базы_данных]
```

Основную функциональность по работе с базами данных предоставляет пакет `android.database`. Функциональность непосредственно для работы с SQLite находится в пакете `android.database.sqlite`. База данных в SQLite представлена классом `android.database.sqlite.SQLiteDatabase`. Он позволяет выполнять запросы к БД, выполнять с ней различные манипуляции.

Класс `android.database.sqlite.SQLiteCursor` предоставляет запрос и позволяет возвращать набор строк, которые соответствуют этому запросу.

Класс `android.database.sqlite.SQLiteQueryBuilder` позволяет создавать SQL-запросы.

Сами sql-выражения представлены классом `android.database.sqlite.SQLiteStatement`, которые позволяют с помощью плейсхолдеров вставлять в выражения динамические данные.

Класс `android.database.sqlite.SQLiteOpenHelper` позволяет создать базу данных со всеми таблицами, если их еще не существует.

Сохраняемые [данные](#) должны представлять соответствующие типы в kotlin.

## Создание и открытие базы данных

Для создания или открытия новой базы данных из кода Activity в Android можно вызвать метод `openOrCreateDatabase()`. Этот метод может принимать три параметра:

- название для базы данных;
- числовое значение, которое определяет режим работы (как правило, в виде константы `MODE_PRIVATE`);
- необязательный параметр в виде объекта `SQLiteDatabase.CursorFactory`, который представляет фабрику создания курсора для работы с БД.

Например, создание базы данных `app.db`:

```
var db = baseContext.openOrCreateDatabase("app.db",  
Context.MODE_PRIVATE, null)
```

Для выполнения запроса к базе данных можно использовать метод `execSQL` класса `SQLiteDatabase`. В этот метод передается SQL-выражение. Например, создание в базе данных таблицы `users`:

```
var db = baseContext.openOrCreateDatabase("app.db",  
Context.MODE_PRIVATE, null)  
db.execSQL("CREATE TABLE IF NOT EXISTS users (name TEXT,  
age INTEGER)");
```

Если надо не просто выполнить выражение, но и получить из БД какие-либо данные, то используется метод `rawQuery()`. Этот метод в качестве параметра принимает SQL-выражение, а также набор значений для выражения `sql`. Например, получение всех объектов из базы данных:

```
var db = baseContext.openOrCreateDatabase("app.db",  
Context.MODE_PRIVATE, null)  
db.execSQL("CREATE TABLE IF NOT EXISTS users (name  
TEXT, age INTEGER)");  
var query = db.rawQuery("SELECT * FROM users;", null);
```



```

if(query.moveToFirst()){
    var name = query.getString(0);
    var age = query.getInt(1);
}

```

Метод `db.rawQuery()` возвращает объект `Cursor`, с помощью которого возможно извлечь полученные данные.

Возможна ситуация, когда в базе данных не будет объектов, и для этого методом `query.moveToFirst()` пытаемся переместиться к первому объекту, полученному из БД. Если этот метод возвратит значение `false`, значит запрос не получил никаких данных из БД.

Другим способом выполнения запроса к базе является вызов метода `db.query()`. Метод `db.query` имеет следующий набор параметров:

- `String Table Name`: Имя таблицы, из которой осуществляется выборка.
- `String [ ] columns`: список столбцов, которые войдут в результат.
- `String WHERE clause`: шаблон where-условия или `null`.
- `String [ ] selection args`: массив с аргументами where-условия.
- `String Group by`: условие группировки.
- `String Having`: условие HAVING.
- `String Order by`: порядок сортировки.

Объект `Cursor`, возвращаемый методом `query()`, обеспечивает доступ к набору записей результирующей выборки. Для обработки возвращаемых данных объект `Cursor` имеет набор методов для чтения каждого типа данных — `getString()`, `getInt()` и `getFloat()`.

## Основные операции с данными таблицы

К основным операциям с данными относятся вставка, обновление и удаление данных. Для выполнения этих `SQLiteDatabase` имеет методы `insert()`, `update()` и `delete()`.

**Метод `insert()`:** `long insert (String table, String nullColumnHack, ContentValues values);`

В метод `insert()` необходимо передать три параметра:

- `table` — имя таблицы, в которую будет вставлена запись;
- `nullColumnHack` — в базе данных SQLite не разрешается вставлять полностью пустую строку, и если строка, полученная от клиента контент-провайдера, будет пустой, то только этому столбцу явно будет назначено значение `null`;
- `values` — карта отображений (класс `Map` и его наследники), передаваемая клиентом контент-провайдера, которая содержит пары ключ-значение. Ключи в карте должны быть названиями столбцов таблицы, значения — вставляемыми данными.

Метод `insert()` возвращает идентификатор `_id` вставленной строки или `-1` в случае ошибки.

**Метод `update()`:** `int update (String table, ContentValues values, String whereClause, String[] whereArgs)`

Обновление строк также происходит с помощью класса `ContentValues`. В третьем параметре указывается условие для обновления. Последний параметр является массивом и позволяет использовать множественные условия для запроса.

**Метод `delete()`:** `int delete (String table, String whereClause, String[] whereArgs)`

Чтобы удалить строку, необходимо вызвать метод `delete()` в контексте базы данных, указав имя таблицы и оператор `WHERE`. Метод практически совпадает с методом `update()`.

## СОХРАНЕНИЕ ДАННЫХ, ПОЛУЧЕННЫХ ИЗ БАЗЫ ДАННЫХ

### Работа со списками

Android представляет широкую палитру элементов, которые представляют списки. Все они являются наследниками класса `android.widget.AdapterView`. Это такие виджеты как `ListView`, `GridView`, `Spinner`. Они могут выступать контейнерами для других элементов управления.

При работе со списками имеют дело с тремя компонентами. Во-первых, это сами элементы списков (`ListView`, `GridView`), которые отображают данные. Во-вторых, это источник данных - массив, объект `ArrayList`, база данных и т.д., в котором находятся сами отображаемые данные. И в-третьих, это адаптеры - специальные компоненты, которые связывают источник данных с элементом списка.

Рассмотрим связь элемента `ListView` с источником данных с помощью одного из таких адаптеров - класса `ArrayAdapter`.

Класс `ArrayAdapter` представляет собой простейший адаптер, который связывает массив данных с набором элементов `TextView`, из которых, к примеру, может состоять `ListView`. То есть в данном случае источником данных выступает массив объектов. `ArrayAdapter` вызывает у каждого объекта метод `toString()` для приведения к строковому виду и полученную строку устанавливает в элемент `TextView`.

Для создания адаптера используется следующий конструктор

```
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,  
Array<String>),
```

где `this` - текущий объект `activity`,

`android.R.layout.simple_list_item_1` - файл разметки списка, который фреймворк представляет по умолчанию. Он находится в папке `Android SDK` по пути `platforms/[android-номер_версии]/data/res/layout`,

`Array<String>` - массив данных. Здесь необязательно указывать именно массив, это может быть список `ArrayList<T>`.

В конце необходимо установить для ListView адаптер с помощью метода `setAdapter()`.

### **Работа с файловой системой**

Работа с настройками уровня `activity` и приложения позволяет сохранить небольшие данные отдельных типов (`string`, `int`), но для работы с большими массивами данных, такими как графически файлы, файлы мультимедиа и т.д., придется обращаться к файловой системе.

ОС Android построена на основе Linux. Этот факт находит свое отражение в работе с файлами. Так, в путях к файлам в качестве разграничителя в Linux использует слеш `"/"`, а не обратный слеш `"\"` (как в Windows). А все названия файлов и каталогов являются регистрозависимыми, то есть `"data"` это не то же самое, что и `"Data"`.

Приложение Android сохраняет свои данные в каталоге `/data/data/<название_пакета>/` и, как правило, относительно этого каталога будет идти работа.

Для работы с файлами абстрактный класс `android.content.Context` определяет ряд методов:

- `deleteFile(String name)`: удаляет определенный файл
- `fileList()`: получает все файлы, которые содержатся в подкаталоге `/files` в каталоге приложения
- `getCacheDir()`: получает ссылку на подкаталог `cache` в каталоге приложения
- `getDir(String dirName, int mode)`: получает ссылку на подкаталог в каталоге приложения, если такого подкаталога нет, то он создается
- `getExternalCacheDir()`: получает ссылку на папку `/cache` внешней файловой системы устройства
- `getExternalFilesDir()`: получает ссылку на каталог `/files` внешней файловой системы устройства
- `getFilePath(String filename)`: возвращает абсолютный путь к файлу в файловой системе
- `openFileInput(String filename)`: открывает файл для чтения
- `openFileOutput (String name, int mode)`: открывает файл для записи

Все файлы, которые создаются и редактируются в приложении, как правило, хранятся в подкаталоге /files в каталоге приложения.

Для непосредственного чтения и записи файлов применяются также стандартные классы `kotlin` из пакета `java.io`.

При создании файла можно оперировать двумя разными режимами:

- `MODE_PRIVATE`: файлы могут быть доступны только владельцу приложения (режим по умолчанию)
- `MODE_APPEND`: данные могут быть добавлены в конец файла

Если же надо было дописать файл, тогда надо было бы использовать режим `MODE_APPEND`:

```
var fos = openFileOutput(FILE_NAME,  
    MODE_APPEND);
```

Для чтения файла применяется поток ввода `FileInputStream`:

```
var fin = openFileInput(FILE_NAME);
```

## LogCat

В Android SDK входит набор инструментов, предназначенных для отладки. Самый важный инструмент при отладке - это LogCat (можно перевести как Логичный Кот). Он отображает сообщения логов (журнал логов), рассылаемые при помощи различных методов.

В Android есть специальный класс `android.util.Log` для подобных случаев. Он позволяет разбивать сообщения по категориям в зависимости от важности. Для разбивки по категориям используются специальные методы, которые легко запомнить по первым буквам, указывающие на категорию:

- `Log.e()` - ошибки (error)
- `Log.w()` - предупреждения (warning)
- `Log.i()` - информация (info)
- `Log.d()` - отладка (debug)
- `Log.v()` - подробности (verbose)

## ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Для всех вариантов создать базу данных согласно варианту задания. К таблице добавить поле id. Сформировать следующие запросы: 1) произвести сортировку всех полей таблицы по одному из числовых полей; 2) сгруппировать данные по нескольким одинаковым полям; 3) вычислить сумму значений одного из числовых полей; 4) вычислить средние значения по сгруппированным полям; 5) отобразить поле с максимальным значением числовой величины; 6) отобразить поля таблицы, в которых числовые величины больше заданной; 7) отобразить поля таблицы, в которых числовые величины меньше средней; 8) отобразить значения только одного из полей, для которых числовое значение больше заданного.

В таблице должно быть не менее 15 записей.

Результаты запроса представить в следующем виде согласно таблице:

№ запроса	Список	Лог	Файл
1		+	+
2	+	+	
3		+	+
4	+	+	+
5		+	
6	+	+	
7	+	+	
8		+	

При выполнении лабораторной работы использовать intent-объекты. Реализацию всех запросов выполнить в рамках одного приложения.

## ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ

Программа может быть реализована на языке высокого уровня Kotlin.

## ВАРИАНТЫ ЗАДАНИЙ

1. Автомобиль1: марка, модель, цвет, максимальная скорость, мощность двигателя.
2. Автомобиль2: тип (седан, кабриолет и пр.), производитель, марка, вместимость багажника, наличие ABS, количество подушек безопасности, средний расход топлива.
3. Автобус: производитель, цвет, количество пассажиров, мощность двигателя, объем топливного бака.
4. Книга: тип (словарь, учебник, художественная литература и пр.), издательство, год издания, количество страниц, тип обложки.
5. Компьютер1: тактовая частота процессора, объем жесткого диска, модель видеокарты, объем оперативной памяти, мощность блока питания.
6. Компьютер2: объем жесткого диска, объем видеопамати, производитель видеокарты, производитель материнской платы, количество usb разъемов.
7. Компьютер3: производитель материнской платы, производитель видеокарты, объем оперативной памяти, год изготовления, производитель монитора.
8. Ноутбук1: название процессора, диагональ экрана, наличие дискретной видеокарты, объем жесткого диска, операционная система, цена.
9. Ноутбук2: производитель, объем жесткого диска, наличие SSD, объем оперативной памяти, наличие Full HD разрешения экрана, время автономной работы
10. Ноутбук3: операционная система, масса, наличие 3G модема, диагональ экрана, объем оперативной памяти, время автономной работы.
11. Планшет1: производитель, производитель процессора, диагональ экрана, наличие 3G, операционная система, объем оперативной памяти
12. Планшет2: производитель, наличие встроенного ПЗУ, операционная система, объем оперативной памяти, цена.
13. Принтер: производитель, тип, скорость печати, емкость лотка для бумажки, цветность, наличие сетевого интерфейса.

14. Аудитория: материально-ответственное лицо (МОЛ), площадь, количество окон, наличие сигнализации, количество столов, количество компьютеров.

15. Жилой дом: количество этажей, наличие лифта, количество квартир, количество квартир на этаже, наличие интернета.

16. Магазин: тип (продовольственный, хозяйственный и пр.), торговая площадь, число продавцов, ассортимент товаров, количество этажей.

17. Менеджер: пол, возраст, образование, должность, заработная плата, стаж.

18. Город: название, население, площадь, наличие метро, количество районов, категория (федерального значения, столица региона, столица района и пр.)

19. Библиотека: название, страна местонахождения, год основания, количество книг, количество постоянных читателей



## **КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ**

1. Перечислите типы данных использующиеся в SQLite.
2. Опишите процесс создания базы данных SQLite.
3. Опишите процесс открытия базы данных SQLite.
4. Перечислите операции с данными таблицы.
5. Опишите механизм работы со списками.
6. Опишите механизм работы с файлами?
7. Раскройте понятие LogCat.
8. Перечислите категории сообщений LogCat.
9. Раскройте понятие ArrayAdapter.

## **ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ**

На выполнение лабораторной работы отводится 1,5 занятия (3 академические часа: 2 часа на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета).

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания (вариант), листинг (xml код разметки экрана, графическое представление, соответствующее разметке экрана, код программы и при необходимости наличие кода дополнительных классов), результаты выполнения работы (графические изображения примеров работы приложения), выводы).

## ОСНОВНАЯ ЛИТЕРАТУРА

1. Семакова, А. Введение в разработку приложений для смартфонов на ОС Android / А. Семакова. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 103 с. : ил. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=429181>
2. Введение в разработку приложений для ОС Android / Ю.В. Березовская, О.А. Юфрякова, В.Г. Вологодина и др. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 434 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=428937>
3. Разработка приложений для смартфонов на ОС Android / Е.А. Латухина, О.А. Юфрякова, Ю.В. Березовская, К.А. Носов. - 2-е изд., исправ. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 252 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=428807>

## ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

4. Дэвид, Х. Разработка приложений Java EE 6 в NetBeans 7. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2013. — 330 с. — Режим доступа: <http://e.lanbook.com/book/58693> — Загл. с экрана.
5. Сильвен, Р. Android NDK. Разработка приложений под Android на C/C++. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2012. — 496 с. — Режим доступа: <http://e.lanbook.com/book/9126> — Загл. с экрана.
6. Ретабоуил, С. Android NDK: руководство для начинающих. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2016. — 518 с. — Режим доступа: <http://e.lanbook.com/book/82810> — Загл. с экрана.

7. Соколова, В.В. Разработка мобильных приложений: учебное пособие / В.В. Соколова ; Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Томский государственный университет», Министерство образования и науки Российской Федерации. - Томск: Издательство Томского политехнического университета, 2015. - 176 с.: ил., табл., схем. - Библиогр. в кн.. - ISBN 978-5-4387-0369-3; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=442808> (15.06.2017).
8. Баженова, И.Ю. Язык программирования Java / И.Ю. Баженова. - М.: Диалог-МИФИ, 2008. - 254 с. : табл., ил. - ISBN 5-86404-091-6 ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=54745> (15.06.2017).
9. Джошуа Блох Java. Эффективное программирование [Электронный ресурс]/ Джошуа Блох— Электрон. текстовые данные. — Саратов: Профобразование, 2017.— 310 с.— Режим доступа: <http://www.iprbookshop.ru/64057.html> .— ЭБС «IPRbooks»

### Электронные ресурсы:

10. Научная электронная библиотека <http://eLIBRARY.RU>
11. Электронно-библиотечная система <http://e.lanbook.com>
12. Электронно-библиотечная система «Университетская библиотека онлайн» <http://biblioclub.ru>
13. Электронно-библиотечная система IPRBook <http://www.iprbookshop.ru>
14. Базовый сайт о системе Android - [https://www.android.com/intl/ru\\_ru](https://www.android.com/intl/ru_ru)
15. Разработка приложения на базе Android - <https://developer.android.com/index.html>
16. <http://sqlite.org> (англ)