**Практическая работа №28. Android SQLite на Kotlin**

*Цель работы:* создать мобильное приложение с базой данных Android SQLite на Kotlin.

**Теоретический материал**

В Android имеется встроенная поддержка одной из распространённых систем управления базами данных — SQLite. Для работы с базами данных SQLite в Android можно использовать следующие классы:

1. android.database.sqlite.SQLiteDatabase — позволяет выполнять запросы к базе данных и выполнять с ней различные манипуляции.
2. android.database.sqlite.SQLiteCursor — предоставляет запрос и позволяет возвращать набор строк, которые соответствуют этому запросу.
3. android.database.sqlite.SQLiteQueryBuilder — позволяет создавать SQL-запросы.
4. android.database.sqlite.SQLiteStatement — позволяет с помощью плейсхолдеров вставлять в выражения динамические данные.
5. android.database.sqlite.SQLiteOpenHelper — позволяет создать базу данных со всеми таблицами, если их ещё не существует.

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

1. Название для базы данных.
2. Числовое значение, которое определяет режим работы (как правило, в виде константы MODE\_PRIVATE).
3. Необязательный параметр в виде объекта SQLiteDatabase.CursorFactory, который представляет фабрику создания курсора для работы с базой данных.

**Задание**

*Разработать мобильное приложение, которое позволяет управлять списками покупок.*

*На главном экране приложения отображается текущий список покупок. Приложение должно позволять создавать, удалять товары из списка покупок. Пользователь может добавить новый элемент к списку покупок, удалить элемент из списка, пометить элемент как купленный (поставить галочку о покупке).*

*Каждый элемент списка имеет следующие параметры: наименование и объем покупки с единицами измерения. Наименование и единица измерения выбираются из списка.*

**Пример работы базой данных Android SQLite на Kotlin**

Шаг 1: Создайте новый проект

Шаг 2: Предоставление разрешения на доступ к хранилищу в файле AndroidManifest.xml

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

Шаг 3: Работа с файлом activity\_main.xml

Перейдите к приложению > разрешения > макета > activity\_main.xml. Добавьте приведенный ниже код в свой файл. Ниже приведен код для activity\_main.xml.

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
 <!-- Поле для ввода имени -->  
 <EditText  
 android:id="@+id/enterName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/enter\_name"  
 android:textSize="22sp"  
 android:layout\_margin="20sp"/>  
  
 <!-- Поле для ввода возраста -->  
 <EditText  
 android:id="@+id/enterAge"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20sp"  
 android:textSize="22sp"  
 android:hint="Enter Age"/>  
  
 <!-- Кнопка для добавления имени -->  
 <Button  
 android:id="@+id/addName"  
 android:layout\_width="150sp"  
 android:layout\_gravity="center"  
 android:background="@color/colorPrimary"  
 android:text="Add Name"  
 android:textColor="#ffffff"  
 android:textSize="20sp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20sp"/>  
  
 <!-- Кнопка для печати имени -->  
 <Button  
 android:id="@+id/printName"  
 android:layout\_width="150sp"  
 android:layout\_gravity="center"  
 android:background="@color/colorPrimary"  
 android:text="Print Name"  
 android:textColor="#ffffff"  
 android:textSize="20sp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20sp"/>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <!-- Просмотр текста, чтобы получить все имена -->  
 <TextView  
 android:id="@+id/Name"  
 android:textAlignment="center"  
 android:layout\_weight="1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20sp"  
 android:text="Name\n\n"  
 android:textSize="22sp"  
 android:padding="10sp"  
 android:textColor="#000000"/>  
  
 <!-- Просмотр всех возрастов -->  
 <TextView  
 android:layout\_weight="1"  
 android:id="@+id/Age"  
 android:textAlignment="center"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20sp"  
 android:text="Age\n\n"  
 android:textSize="22sp"  
 android:padding="10sp"  
 android:textColor="#000000"/>  
  
 </LinearLayout>  
  
</LinearLayout>

Шаг 4: Создание нового класса для операций SQLite

Перейдите к приложению > java > имя пакета вашего проекта > Щелкните его правой кнопкой мыши > New > Kotlin class и назовите его DBHelper и добавьте к нему приведенный ниже код. Чтобы сделать код более понятным, добавлены комментарии.

package com.example.myapplication  
  
import android.content.ContentValues  
import android.content.Context  
import android.database.Cursor  
import android.database.sqlite.SQLiteDatabase  
import android.database.sqlite.SQLiteOpenHelper  
  
class DBHelper(context: Context, factory: SQLiteDatabase.CursorFactory?) :  
 SQLiteOpenHelper(context, DATABASE\_NAME, factory, DATABASE\_VERSION) {  
  
 // ниже приведен метод создания базы данных с помощью запроса sqlite  
 override fun onCreate(db: SQLiteDatabase) {  
 // ниже приведен запрос sqlite, где имена столбцовм  
 // вместе с их типами данных  
 val query = ("CREATE TABLE " + TABLE\_NAME + " ("  
 + ID\_COL + " INTEGER PRIMARY KEY, " +  
 NAME\_COl + " TEXT," +  
 AGE\_COL + " TEXT" + ")")  
  
 // мы вызываем sqlite  
 // метод выполнения нашего запроса  
 db.execSQL(query)  
 }  
  
 override fun onUpgrade(db: SQLiteDatabase, p1: Int, p2: Int) {  
 // этот метод предназначен для проверки того, существует ли таблица уже  
 db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME)  
 onCreate(db)  
 }  
  
 // Этот метод предназначен для добавления данных в нашу базу данных  
 fun addName(name : String, age : String ){  
  
 // ниже мы создаем  
 // переменную значений содержимого  
 val values = ContentValues()  
  
 // мы вставляем наши значения  
 // в виде пары ключ-значение  
 values.put(NAME\_COl, name)  
 values.put(AGE\_COL, age)  
  
 // здесь мы создаем  
 // доступную для записи переменную  
 // нашей базы данных, поскольку мы хотим  
 // вставить значение в нашу базу данных  
 val db = this.*writableDatabase* // все значения вставляются в базу данныхe  
 db.insert(TABLE\_NAME, null, values)  
  
 // наконец-то мы  
 // закрываем нашу базу данных  
 db.close()  
 }  
  
  
 // приведенный ниже способ заключается в получении  
 // всех данных из нашей базы данных  
 fun getName(): Cursor? {  
  
 // здесь мы создаем читаемую  
 // переменную нашей базыданных  
 // поскольку мы хотим считывать из нее значение  
 val db = this.*readableDatabase* // приведенный ниже код возвращает курсор для  
 // чтения данных из базы данных  
 return db.rawQuery("SELECT \* FROM " + TABLE\_NAME, null)  
  
 }  
  
 companion object{  
 // здесь мы определили переменные для нашей базы данных  
  
 // below is variable for database name  
 private val DATABASE\_NAME = "GEEKS\_FOR\_GEEKS"  
  
 // ниже приведена переменная для имени базы данных  
 private val DATABASE\_VERSION = 1  
  
 // ниже приведена переменная для имени таблицы  
 val TABLE\_NAME = "gfg\_table"  
  
 // ниже приведена переменная для столбца id  
 val ID\_COL = "id"  
  
 // ниже приведена переменная для столбца name  
 val NAME\_COl = "name"  
  
 // ниже приведена переменная для столбца age  
 val AGE\_COL = "age"  
 }  
}

Шаг 5: Работа с файлом MainActivity.kt

Перейдите к файлу MainActivity.kt и обратитесь к следующему коду. Ниже приведен код файла MainActivity.kt. Внутри кода добавляются комментарии для более детального понимания кода.

package com.example.myapplication  
  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Toast  
import kotlinx.android.synthetic.main.activity\_main.\*  
  
class MainActivity : AppCompatActivity() {  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 // приведенный ниже код предназначен для добавления по щелчку  
 // прослушиватель к нашей кнопке добавить имя  
 addName.setOnClickListener**{** // ниже мы создали  
 // новый класс DBHelper,  
 // и передали ему контекст  
 val db = DBHelper(this, null)  
  
 // создаем переменные для значений  
 // в текстах редактирования имени и возраста  
 val name = enterName.text.toString()  
 val age = enterAge.text.toString()  
  
 // вызывающий метод для добавления  
 // имени в нашу базу данных  
 db.addName(name, age)  
  
 // Всплывающее сообщение на экране  
 Toast.makeText(this, name + " added to database", Toast.*LENGTH\_LONG*).show()  
  
 // наконец, очистка редактируемых текстов  
 enterName.text.clear()  
 enterAge.text.clear()  
 **}** // приведенный ниже код предназначен для добавления при нажатии кнопки  
 // прослушиватель к нашей кнопке печати имени  
 printName.setOnClickListener**{** // создаем класс DBHelper  
 // и передаем ему контекстt  
 val db = DBHelper(this, null)  
  
 // ниже указана переменная для курсора  
 // мы вызвали метод, чтобы получить  
 // все имена из нашей базыданных  
 // и добавить в текстовое представление имени  
 val cursor = db.getName()  
  
 // перемещение курсора в первую позицию и  
 // добавление значения в текстовом представленииw  
 cursor!!.moveToFirst()  
 Name.append(cursor.getString(cursor.getColumnIndex(DBHelper.NAME\_COl)) + "\n")  
 Age.append(cursor.getString(cursor.getColumnIndex(DBHelper.AGE\_COL)) + "\n")  
  
 // перемещаем курсор в следующую  
 // позицию и добавляем значения  
 while(cursor.moveToNext()){  
 Name.append(cursor.getString(cursor.getColumnIndex(DBHelper.NAME\_COl)) + "\n")  
 Age.append(cursor.getString(cursor.getColumnIndex(DBHelper.AGE\_COL)) + "\n")  
 }  
  
 // наконец-то мы закрываем наш курсор  
 cursor.close()  
 **}** }  
}

Теперь запустите приложение и посмотрите на результат.