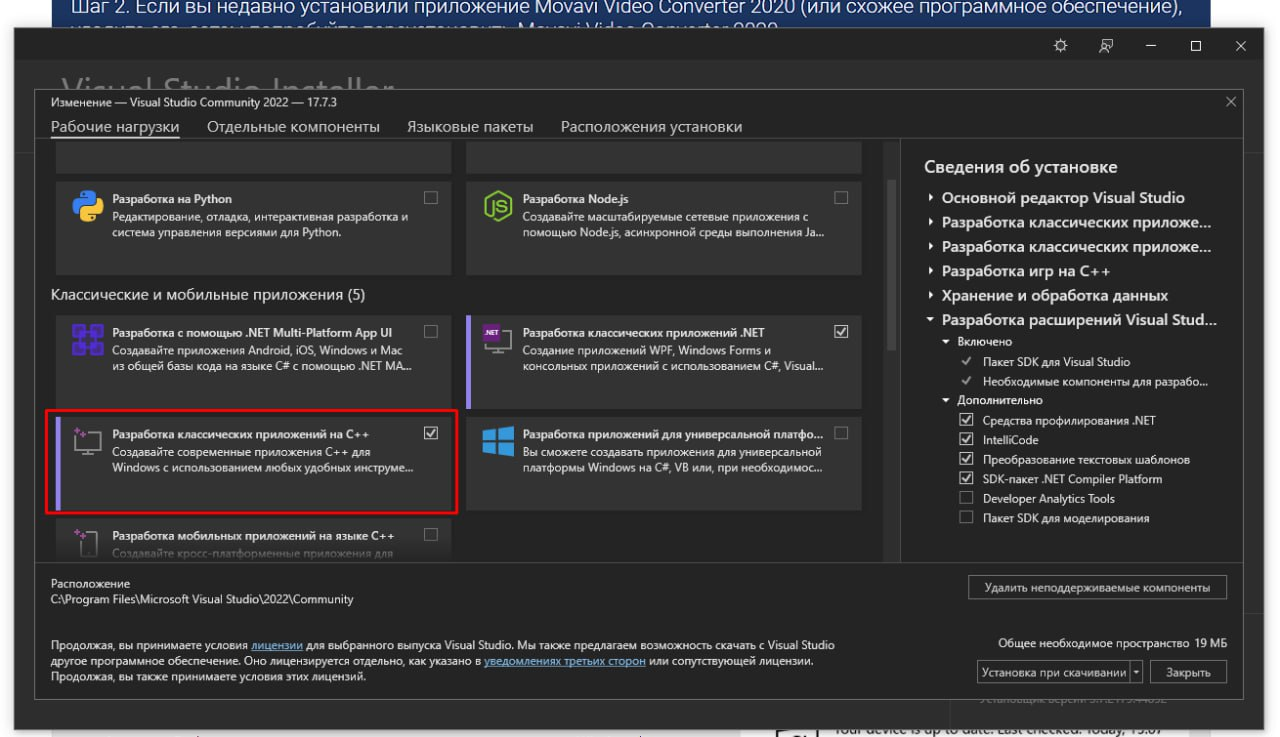
Открываем папку servert, в нем запускаем Start.exe он должен открыться и сразу же закрыться, если вылетают ошибки что нет каких-то DLL то надо скачать средство разработки на с++ в Visual Studio. Для этого надо открыть Visual Studio installer, далее нажать кнопку изменить и скачать “разработка классических приложений на с++”



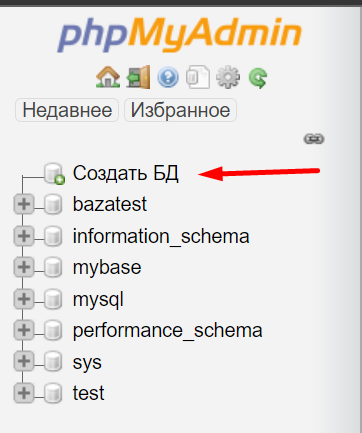
После чего опять запускаем Start.exe, после запускаем сначала sql.bat и php.bat.

Заходим <http://127.0.0.1:8080/phpmyadmin/index.php>

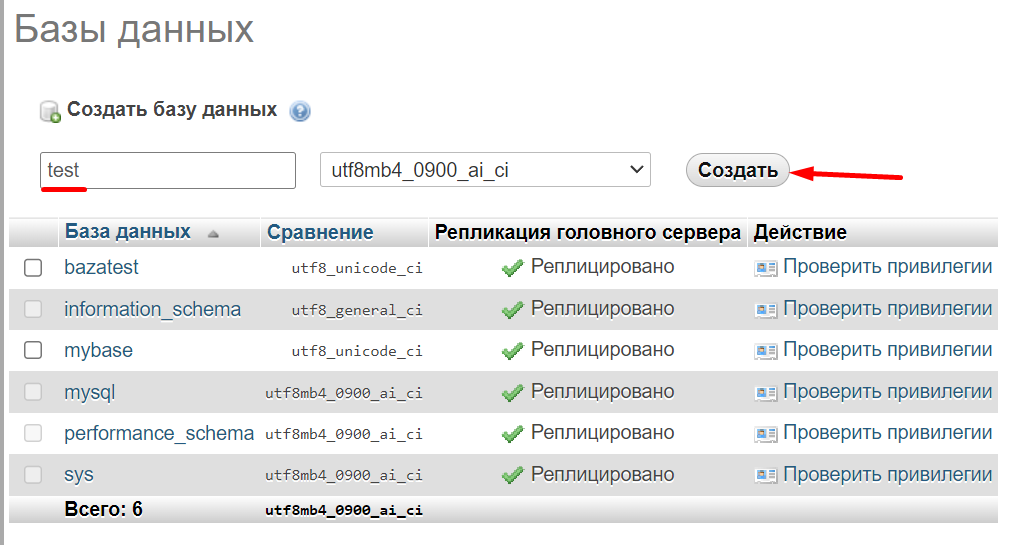
имя пользователя: root

пароль:

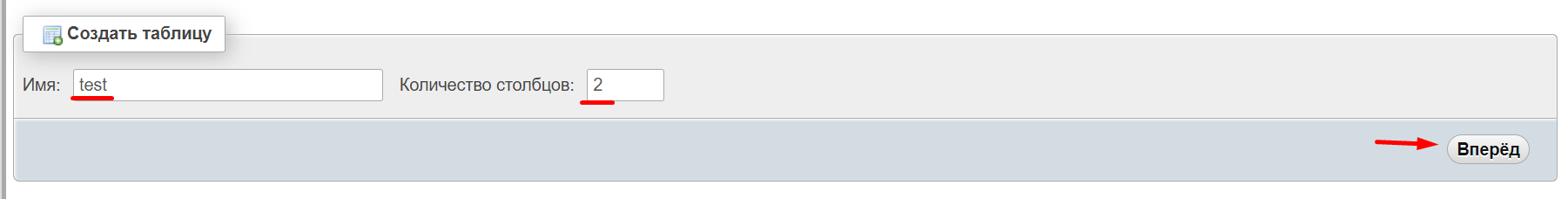
создаем БД



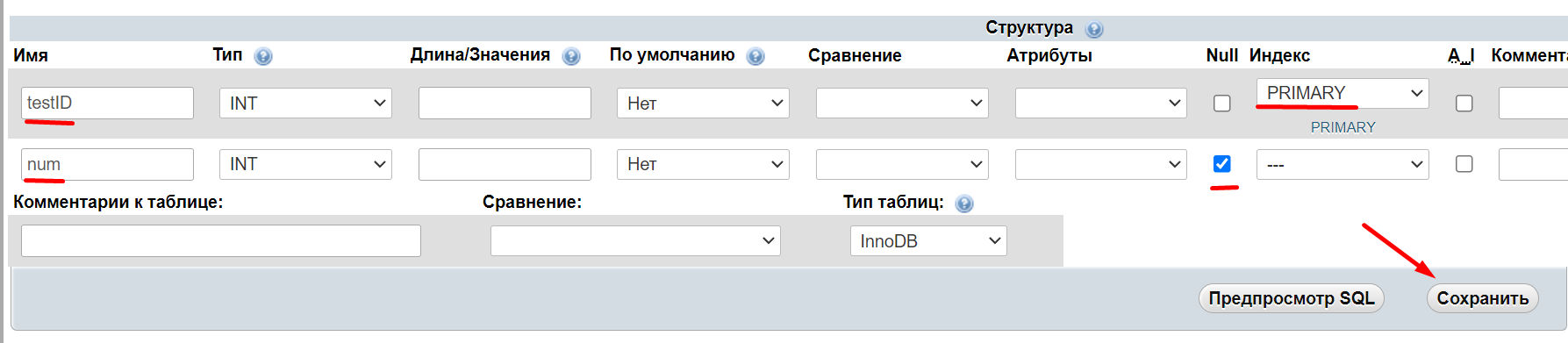
Называем БД test и создаем

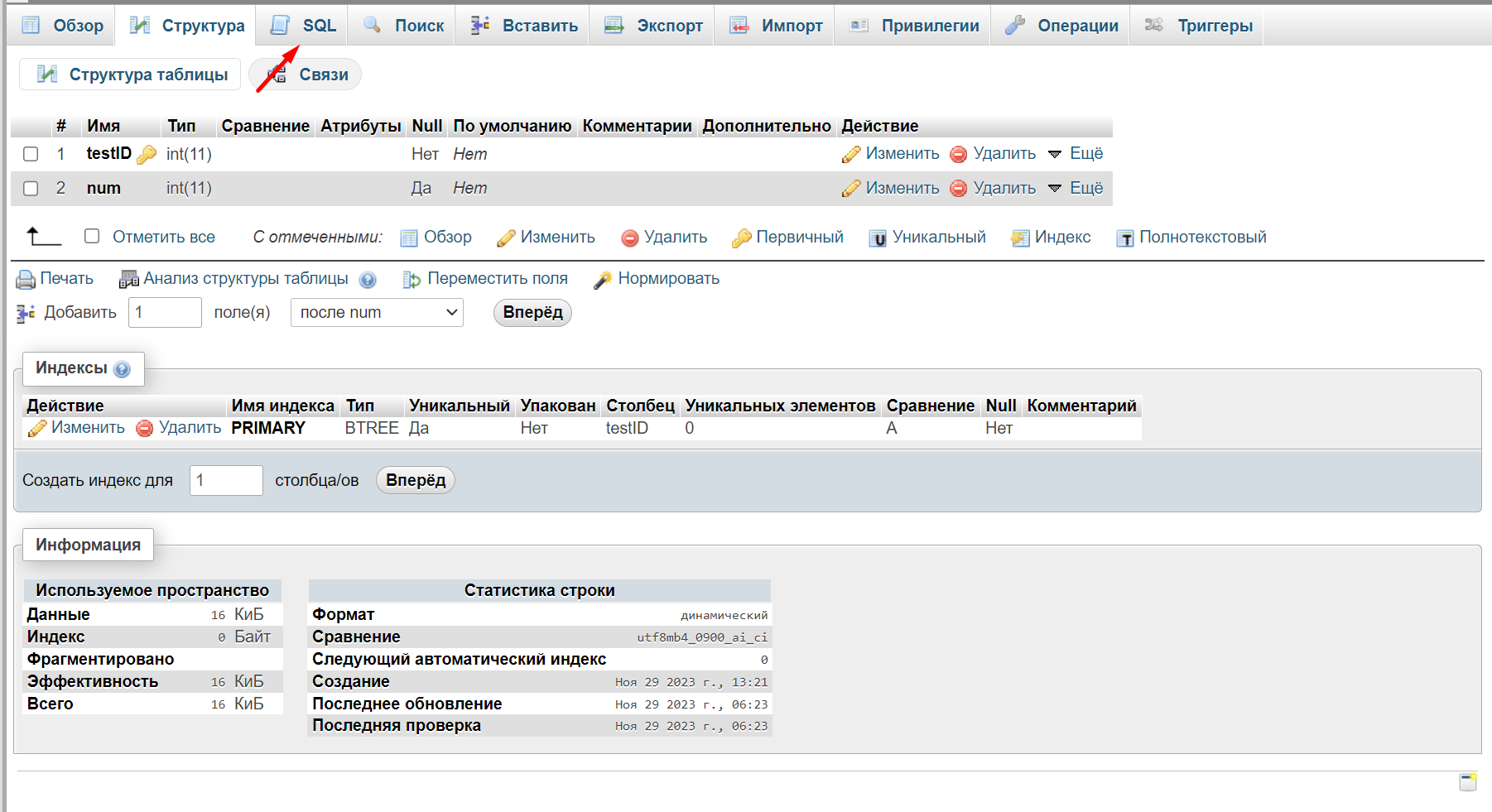
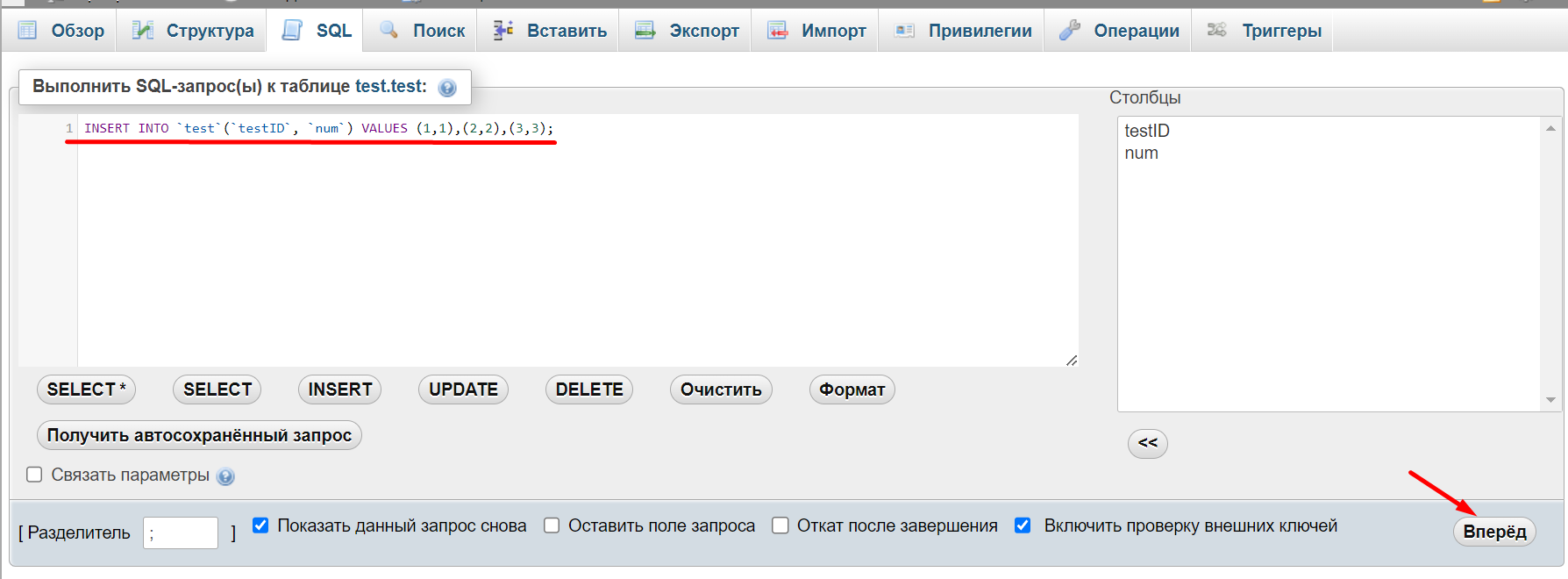


Далее создаем таблицу test, делаем 2 поля и вперед



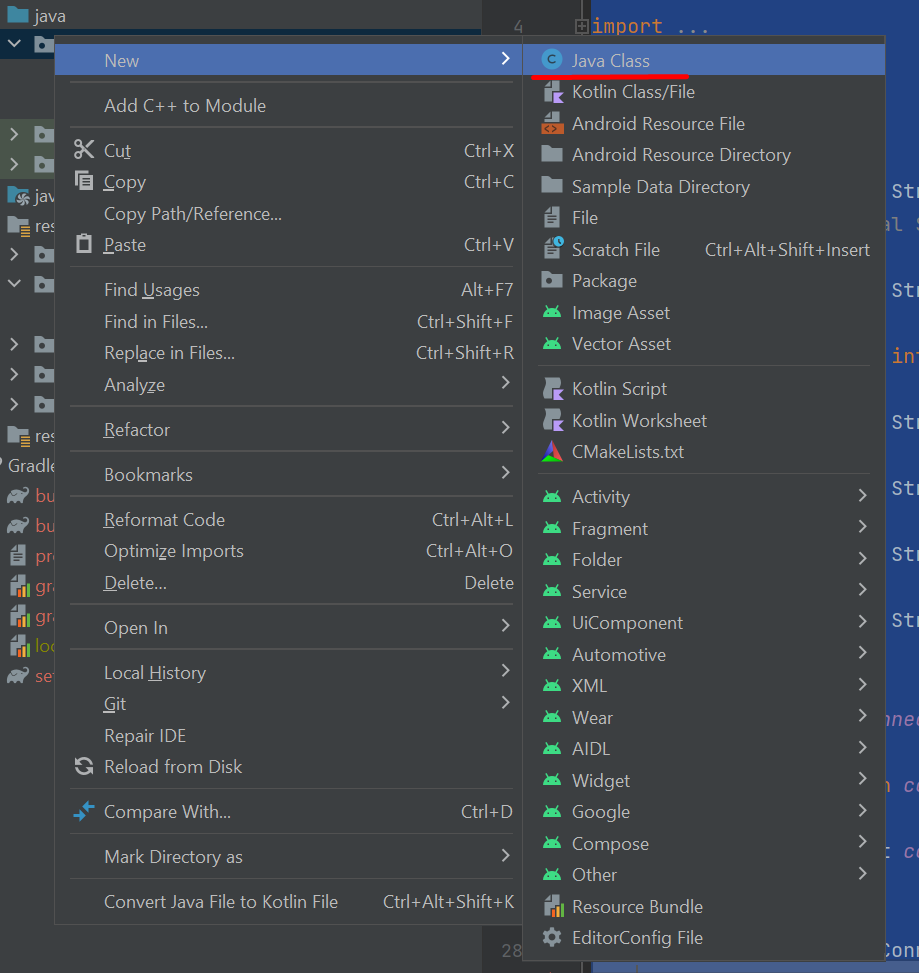
Называем два поля как testID(PRIMARY KEY) и num(NULL) и сохраняем



  
вставляем   
INSERT INTO `test`(`testID`, `num`) VALUES (1,1),(2,2),(3,3);  
и выполняем   


Переходим в Android studio и создаем empty activity.

Рядом с MainActivity создаем Java Class и называем его DB



Вставляем код в два файла (примечание, первую строку в файлах трогать не надо все остальное меняем)

Activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout 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"  
 tools:context=".MainActivity">  
  
 <TextView  
 android:id="@+id/texts"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Hello World!"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

MainActivity.java

package com.example.db\_test;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.util.Log;  
import android.widget.TextView;  
  
public class MainActivity extends AppCompatActivity {  
  
 TextView texts;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 DB.*connectionToDataBase*.start();  
 try {  
 DB.*connectionToDataBase*.join();  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 DB.*getNum*.start();  
 try {  
 DB.*getNum*.join();  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 String s = "";  
 for (int i : DB.*nums*) {  
 s += " " + String.*valueOf*(i);  
 }  
 texts = findViewById(R.id.*texts*);  
 texts.setText(s);  
  
 DB.*closeConnectionToDataBase*.start();  
 try {  
 DB.*closeConnectionToDataBase*.join();  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 }  
}

DB.java

import android.util.Log;  
  
import java.sql.DriverManager;  
import java.sql.Connection;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.ArrayList;  
import java.util.List;  
  
public class DB {  
 static private final String *server* = "10.0.2.2";  
 static private final String *database* = "test";  
 static private final int *port* = 3306;  
 static private final String *user* = "root";  
 static private final String *pass* = "";  
 static private final String *driver* = "com.mysql.jdbc.Driver";  
 static private final String *url* = String.*format*("jdbc:mysql://%s:%d/%s", *server*, *port*, *database*);  
  
 static Connection *connection* = null;  
 static public boolean *conStatus*;  
 static public Connect *connectionToDataBase* = new Connect();  
  
 public static class Connect extends Thread {  
 public void run() {  
 try {  
 Class.*forName*(*driver*);  
 Log.*i*("DB", "con" + *url*);  
 *connection* = DriverManager.*getConnection*(*url*, *user*, *pass*);  
 *conStatus* = true;  
 Log.*i*("DB", "connectionToDataBase: connection true" + *url*);  
 } catch (SQLException e) {  
 Log.*e*("DB", "connectionToDataBase: connection false" + *url*);  
 e.printStackTrace();  
 *conStatus* = false;  
 } catch (ClassNotFoundException e) {  
 Log.*e*("DB", "connectionToDataBase: connection false" + *url*);  
 e.printStackTrace();  
 *conStatus* = false;  
 }  
 *connectionToDataBase* = new Connect();  
 }  
 }  
  
 public static abstract class Query extends Thread{  
 protected String sql = null;  
 protected ResultSet rs;  
 protected abstract void logic() throws SQLException;  
 public void run() {  
 if (!*conStatus*) {  
 return;  
 }  
 if (sql == null){  
 Log.*e*("DB", "run: sql query is null");  
 return;  
 }  
 try(Statement stmt = *connection*.createStatement() ) {  
 Log.*i*("DB", "run: \n" + sql);  
 rs = stmt.executeQuery(sql);  
 this.logic();  
 } catch (SQLException e) {  
 Log.*e*("DB", "query: \n" + e);  
 }  
 }  
 }  
  
 public static ArrayList<Integer> *nums* = new ArrayList<>();  
 public static GetNum *getNum* = new GetNum();  
 public static class GetNum extends Query {  
 public GetNum(){  
 sql = "SELECT \* FROM test";  
 }  
 @Override  
 protected void logic() throws SQLException {  
 while (rs.next()) {  
 *nums*.add(rs.getInt("num"));  
 }  
 }  
 }  
  
  
 static public Close *closeConnectionToDataBase* = new Close();  
  
 public static class Close extends Thread {  
 public void run() {  
 try {  
 *connection*.close();  
 } catch (SQLException e) {  
 Log.*e*("BD", "close: \n" + e);  
 }  
 *closeConnectionToDataBase* = new Close();  
 }  
 }  
  
}

в build.gradle (Module :app) в dependencies добавляем строку

implementation 'mysql:mysql-connector-java:5.1.49'

в AndroidManifest добавляем

<uses-permission android:name="android.permission.INTERNET" />

static private final String *server* = "10.0.2.2";  
static private final String *database* = "test";  
static private final int *port* = 3306;  
static private final String *user* = "root";  
static private final String *pass* = "";  
static private final String *driver* = "com.mysql.jdbc.Driver";  
static private final String *url* = String.*format*("jdbc:mysql://%s:%d/%s", *server*, *port*, *database*);

это надо для подключения к бд в конце создается url для подключения.

static Connection *connection* = null;  
static public boolean *conStatus*;  
static public Connect *connectionToDataBase* = new Connect();

переменные для создания связи между приложением и БД

public static class Connect extends Thread {  
 public void run() {  
 try {  
 Class.*forName*(*driver*);  
 Log.*i*("DB", "con" + *url*);  
 *connection* = DriverManager.*getConnection*(*url*, *user*, *pass*);  
 *conStatus* = true;  
 Log.*i*("DB", "connectionToDataBase: connection true" + *url*);  
 } catch (SQLException e) {  
 Log.*e*("DB", "connectionToDataBase: connection false" + *url*);  
 e.printStackTrace();  
 *conStatus* = false;  
 } catch (ClassNotFoundException e) {  
 Log.*e*("DB", "connectionToDataBase: connection false" + *url*);  
 e.printStackTrace();  
 *conStatus* = false;  
 }  
 *connectionToDataBase* = new Connect();  
 }  
}

класс для подключения к БД сделан через Thread чтоб подключение можно было делать в отдельном потоке

public static abstract class Query extends Thread{  
 protected String sql = null;  
 protected ResultSet rs;  
 protected abstract void logic() throws SQLException;  
 public void run() {  
 if (!*conStatus*) {  
 return;  
 }  
 if (sql == null){  
 Log.*e*("DB", "run: sql query is null");  
 return;  
 }  
 try(Statement stmt = *connection*.createStatement() ) {  
 Log.*i*("DB", "run: \n" + sql);  
 rs = stmt.executeQuery(sql);  
 this.logic();  
 } catch (SQLException e) {  
 Log.*e*("DB", "query: \n" + e);  
 }  
 }  
}

абстрактный класс для создания запросов к бд тоже наследуется от Thread чтоб запрос происходил в другом потоке и не тормозил приложение

здесь поля

* sql отвечает за строку запроса в БД
* rs отвечает за полученную таблицу из БД

абстрактные методы(для переопределения)

* logic он нужен для выполнения какого то кода после получения таблицы из базы данных

public static ArrayList<Integer> *nums* = new ArrayList<>();  
public static GetNum *getNum* = new GetNum();  
public static class GetNum extends Query {  
 public GetNum(){  
 sql = "SELECT \* FROM test";  
 }  
 @Override  
 protected void logic() throws SQLException {  
 while (rs.next()) {  
 *nums*.add(rs.getInt("num"));  
 }  
 }  
}

вот так создается запрос

создаем две переменные

nums(ArrayList<Integer>) в котором будут хранится данные которые мы получаем из БД (вместо Integer в ArrayList<Integer> можно использовать другие типы данных или свои классы для хранения полученных данных)

getNum это экземпляр класса запроса через эту переменную будет производится запрос в бд

класс GetNum наследуется от класса Query в конструкторе (GetNum) в sql кладем строку с запросом. Далее переопределяем метод Logic в котором реализуем добавление данных из полученной таблицы(rs) в массив(nums)

static public Close *closeConnectionToDataBase* = new Close();  
  
public static class Close extends Thread {  
 public void run() {  
 try {  
 *connection*.close();  
 } catch (SQLException e) {  
 Log.*e*("BD", "close: \n" + e);  
 }  
 *closeConnectionToDataBase* = new Close();  
 }  
}

класс для закрытия соединения с БД

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 DB.*connectionToDataBase*.start();  
 try {  
 DB.*connectionToDataBase*.join();  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 DB.*getNum*.start();  
 try {  
 DB.*getNum*.join();  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 String s = "";  
 for (int i : DB.*nums*) {  
 s += " " + String.*valueOf*(i);  
 }  
 texts = findViewById(R.id.*texts*);  
 texts.setText(s);  
 DB.*closeConnectionToDataBase*.start();  
 try {  
 DB.*closeConnectionToDataBase*.join();  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
}

Здесь происходит подключение к БД, потом запрос, вывод результата и закрытие подключения