

Федеральное агентство связи  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Сибирский государственный университет  
телекоммуникаций и информатики»

**Лабораторная работа №5**  
**«База Данных»**

Выполнили: студенты 3 курса  
ИВТ, гр. ИП-713  
Трусов К.В.  
Михеев Н.А.

Новосибирск, 2020 г.

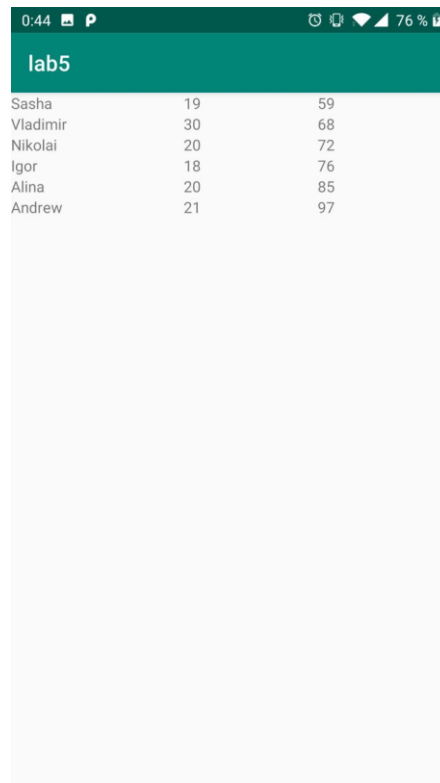
## Задание

Создать базу данных студентов (Имя, вес, рост, возраст - сгенерировать случайно). Вывести из базы данных все записи, отсортированные по возрасту, в таблицу (TableLayout).

## Решение поставленной задачи

Для решения поставленной задачи были реализованы три класса:

- Student – базовый класс для создания объектов студентов, в нем имеется конструктор с переменными: id, имени, возраста и веса
- DatabaseHandler – класс для управления БД, ее инициализация, обновление, функция добавления записи в базу
- MainActivity – класс для создания таблицы на layout, добавления в нее данных из базы и вывод на экран, так же членом этого класса является функция fillDB(), занимающаяся заполнением базы студентами



| Sasha    | 19 | 59 |
|----------|----|----|
| Vladimir | 30 | 68 |
| Nikolai  | 20 | 72 |
| Igor     | 18 | 76 |
| Alina    | 20 | 85 |
| Andrew   | 21 | 97 |

Рис.1 – демонстрация работы программы

## Листинг программы

Класс MainActivity:

```
class MainActivity : AppCompatActivity() {  
  
    var lst: MutableList<Student> = ArrayList()
```

```

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    filldb()

    val table = TableLayout(this)

    table.isStretchAllColumns = true
    table.isShrinkAllColumns = true

    val rows = Array(100, { TableRow(this) })

    val empty = TextView(this)

    for ((i, stud) in lst.withIndex()) {
        val disName = TextView(this)
        val disAge = TextView(this)
        val disWeight = TextView(this)
        disName.text = lst[i].name
        disAge.text = lst[i].age.toString()
        disWeight.text = lst[i].weight.toString()
        rows[i].addView(disName)
        rows[i].addView(disAge)
        rows[i].addView(disWeight)
        table.addView(rows[i])
    }

    setContentView(table)
}

private fun filldb()
{
    val student1 = Student("Igor", 18, (50..100).random())
    val student2 = Student("Nikolai", 20, (50..100).random())
    val student3 = Student("Andrew", 21, (50..100).random())
    val student4 = Student("Vladimir", 30, (50..100).random())
    val student5 = Student("Alina", 20, (50..100).random())
    val student6 = Student("Sasha", 19, (50..100).random())
}

```

```

        val db = DatabaseHandler(this)

        db.addStudent(student1)
        db.addStudent(student2)
        db.addStudent(student3)
        db.addStudent(student4)
        db.addStudent(student5)
        db.addStudent(student6)

        lst = db.readDB()
    }
}

```

## Класс Student:

```

class Student
{
    var age: Int = 0
    var name: String = ""
    var id: Int = 0
    var weight: Int = 0

    constructor(name: String, age: Int, weight: Int) {
        this.age = age
        this.name = name
        this.weight = weight
    }

    constructor()
}

```

## Класс DatabaseHandler:

```

class DatabaseHandler(var context: Context) :
    SQLiteOpenHelper(context, DB_NAME, null, DB_VERSION) {

    override fun onCreate(db: SQLiteDatabase?) {
        val CREATE_TABLE = "CREATE TABLE $TABLE_NAME " +
            "($ID Integer PRIMARY KEY, $NAME TEXT, $AGE Integer, $WEIGHT Integer)"
        val res = db?.execSQL(CREATE_TABLE)
    }
}

```

```
}
```

```
override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {  
    val res = db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")  
    onCreate(db)  
}
```

```
fun addStudent(student: Student): Boolean {  
    val db = this.writableDatabase  
    val values = ContentValues()  
    values.put(NAME, student.name)  
    values.put(AGE, student.age)  
    values.put(WEIGHT, student.weight)  
    val _success = db.insert(TABLE_NAME, null, values)  
    db.close()  
    Log.v("InsertedID", "$_success")  
    return (Integer.parseInt("_success") != -1)  
}
```

```
fun readDB() : MutableList<Student> {  
    var list : MutableList<Student> = ArrayList()  
    val db = this.readableDatabase  
  
    val query = "Select * from $TABLE_NAME"  
    val result = db.rawQuery(query, null)  
  
    if (result.moveToFirst()) {  
        do {  
            var stud = Student()  
            stud.id = result.getString(0).toInt()  
            stud.name = result.getString(1)  
            stud.age = result.getString(2).toInt()  
            stud.weight = result.getString(3).toInt()  
            list.add(stud)  
        } while (result.moveToNext())  
    }  
    list.sortBy { it.weight }  
    result.close()
```

```
        db.close()

        return list
    }

    companion object {
        private val DB_NAME = "MyDB"
        private val DB_VERSION = 2;
        private val TABLE_NAME = "Students"
        private val ID = "id"
        private val NAME = "Name"
        private val AGE = "Age"
        private val WEIGHT = "Weight"
    }
}
```