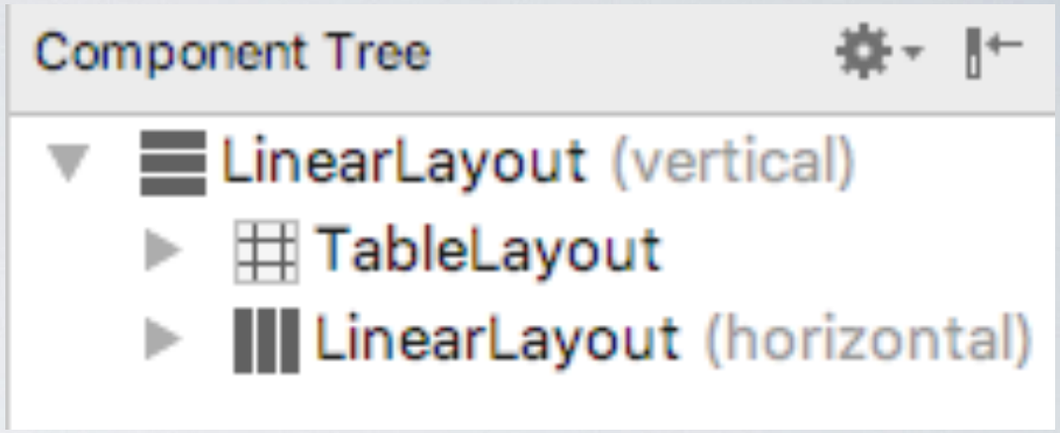
WSTĘP DO ANDROIDA

Laboratorium 4
Systemy i aplikacje bez granic



- Tworzymy nowy projekt typu Empty Activity o nazwie DatabaseExample

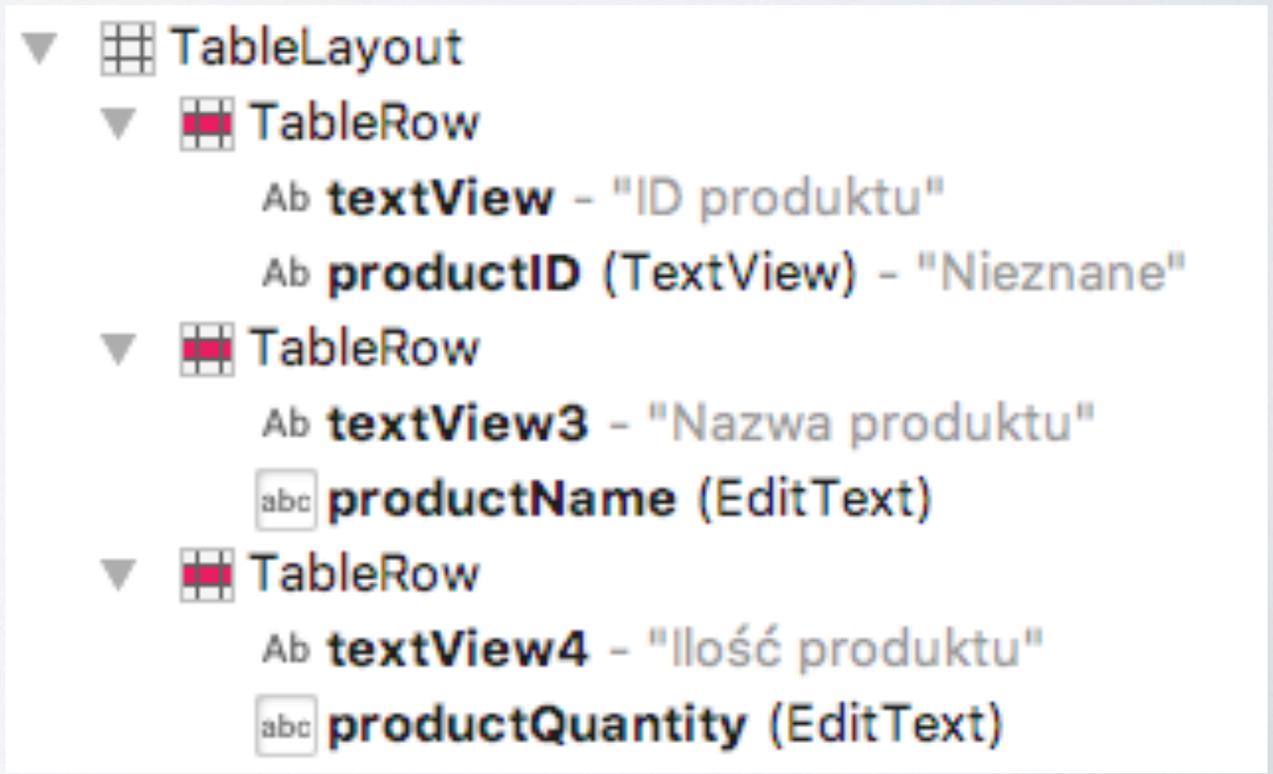
- Usuwamy wszystko z projektu interfejsu użytkownika i jako korzeń umieszczamy LinearLayout (vertical)

- Na drugim poziomie umieszczamy TableLayout, a poniżej LinearLayout (horizontal)

III

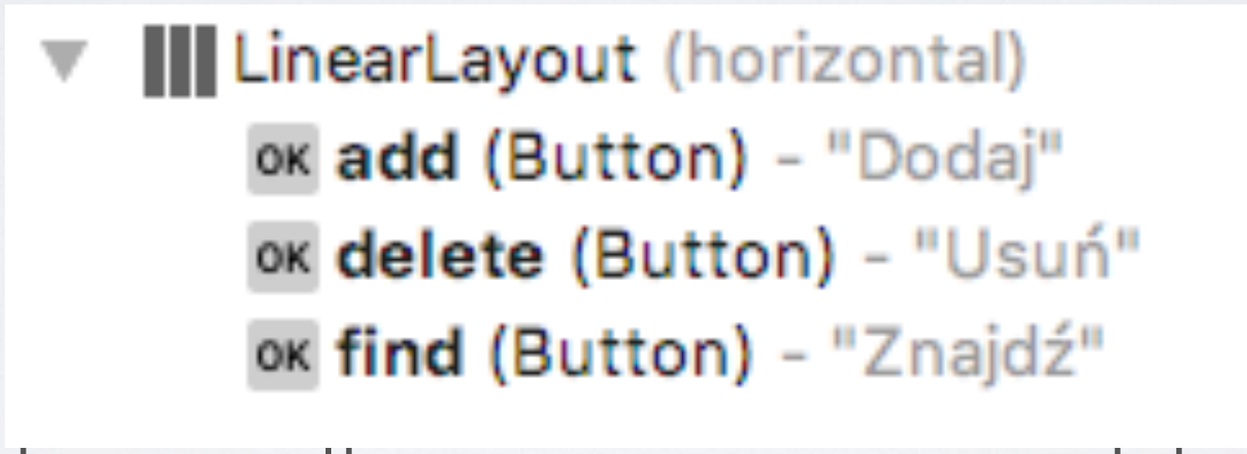
- W TableLayout mamy mieć 3 wiersze TableRow
- W pierwszym wierszu mają znaleźć się 2 TextView obok siebie
- W drugim wierszu TextView i PlainText
- Podobnie w trzecim

ID produktu	Nieznane	
Nazwa produktu	_____	
Ilość produktu	_____	



III

- W dolnym LinearLayout umieszczamy 3 przyciski Button



- Nadajemy kontrolką nazwy, a przyciskom metody obsługi onClick: newProduct, removeProduct, lookupProduct

8:00

Database

ID produktu	Nieznane	
Nazwa produktu		
Ilość produktu		

DODAJ | USUŃ | ZNAJDŹ

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginHorizontal="10dp"
            android:layout_marginVertical="10dp">

            <TextView...>

            <TextView...>
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginHorizontal="10dp"
        android:layout_marginVertical="10dp">

        <TextView...>

        <EditText
            android:id="@+id/productName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="none"
            android:textSize="18sp" />
    </TableRow>
```

```
    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginHorizontal="10dp"
        android:layout_marginVertical="10dp">

        <TextView...>

        <EditText
            android:id="@+id/productQuantity"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="numberDecimal"
            android:textSize="18sp" />
    </TableRow>

</TableLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal"
    android:orientation="horizontal">

    <Button
        android:id="@+id/add"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:onClick="newProduct"
        android:text="Dodaj" />
```

|||

```
<Button
    android:id="@+id/delete"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:onClick="removeProduct"
    android:text="Usuń" />

    <Button
        android:id="@+id/find"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:onClick="lookupProduct"
        android:text="Znajdź" />

</LinearLayout>
</LinearLayout>
```

III

- Dodajemy do projektu klasę Product

```
class Product {
    var id: Int = 0
    var productName: String? = null
    var quantity: Int = 0

    constructor(id: Int, productname: String, quantity: Int) {
        this.id = id
        this.productName = productname
        this.quantity = quantity
    }

    constructor(productname: String, quantity: Int) {
        this.productName = productname
        this.quantity = quantity
    }
}
```

III

- Dodajemy kolejną klasę, dziedziczącą po klasie SQLiteOpenHelper

```
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import android.content.Context
import android.content.ContentValues

class MyDBHandler(context: Context, name: String?,
                  factory: SQLiteDatabase.CursorFactory?, version: Int) : SQLiteOpenHelper(context,
        DATABASE_NAME, factory, DATABASE_VERSION) {
```

- Dodajemy wewnątrz companion object

```
companion object {
    private val DATABASE_VERSION = 1
    private val DATABASE_NAME = "productDB.db"
    val TABLE_PRODUCTS = "products"
    val COLUMN_ID = "_id"
    val COLUMN_PRODUCTNAME = "productname"
    val COLUMN_QUANTITY = "quantity"
}
```

III

- Dodajemy metody do klasy

```
override fun onCreate(db: SQLiteDatabase) {
    val CREATE_PRODUCTS_TABLE = ("CREATE TABLE " +
            TABLE_PRODUCTS + "("
            + COLUMN_ID + " INTEGER PRIMARY KEY," +
            COLUMN_PRODUCTNAME
            + " TEXT," + COLUMN_QUANTITY + " INTEGER" + ")")
    db.execSQL(CREATE_PRODUCTS_TABLE)
}

override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int,
                       newVersion: Int) {
    db.execSQL( sql: "DROP TABLE IF EXISTS " + TABLE_PRODUCTS)
    onCreate(db)
}
```

|||

```
fun addProduct(product: Product) {
    val values = ContentValues()
    values.put(COLUMN_PRODUCTNAME, product.productName)
    values.put(COLUMN_QUANTITY, product.quantity)
    val db = this.writableDatabase
    db.insert(TABLE_PRODUCTS,  nullColumnHack: null, values)
    db.close()
}

fun findProduct(productname: String): Product? {
    val query = "SELECT * FROM $TABLE_PRODUCTS WHERE $COLUMN_PRODUCTNAME LIKE \"$productname\""
    val db = this.writableDatabase
    val cursor = db.rawQuery(query,  selectionArgs: null)
    var product: Product? = null

    if (cursor.moveToFirst()) {
        //cursor.moveToFirst()

        val id = Integer.parseInt((cursor.getString( columnIndex: 0)))
        val name = cursor.getString( columnIndex: 1)
        val quantity = Integer.parseInt(cursor.getString( columnIndex: 2))
        product = Product(id, name, quantity)
        cursor.close()
    }
    db.close()
    return product
}
```

|||

```
fun deleteProduct(productname: String): Boolean {
    var result = false
    val query = "SELECT * FROM $TABLE_PRODUCTS WHERE $COLUMN_PRODUCTNAME = \"$productname\""

    val db = this.writableDatabase
    val cursor = db.rawQuery(query,  selectionArgs: null)
    if (cursor.moveToFirst()) {
        val id = Integer.parseInt(cursor.getString( columnIndex: 0))
        db.delete(TABLE_PRODUCTS,  whereClause: COLUMN_ID + " = ?", arrayOf(id.toString()))
        cursor.close()
        result = true
    }
    db.close()
    return result
}
```

III

- W klasie aktywności dodajemy metody obsługi przycisków

```
fun newProduct(view: View) {
    val dbHandler = MyDBHandler( context: this,  name: null,  factory: null,  version: 1)

    val quantity = Integer.parseInt(productQuantity.text.toString())

    val product = Product(productName.text.toString(), quantity)

    dbHandler.addProduct(product)
    productName.setText("")
    productQuantity.setText("")
}
```

|||

```
fun lookupProduct(view: View) {
    val dbHandler = MyDBHandler( context: this,  name: null,  factory: null,  version: 1)

    val product = dbHandler.findProduct(
            productName.text.toString())

    if (product != null) {
        productID.text = product.id.toString()

        productQuantity.setText(
                product.quantity.toString())
    } else {
        productID.text = "Nie znaleziono"
    }
}
```

|||

```
fun removeProduct(view: View) {
    val dbHandler = MyDBHandler( context: this,  name: null,  factory: null,  version: 1)

    val result = dbHandler.deleteProduct(
            productName.text.toString())

    if (result) {
        productID.text = "Produkt usunięty"
        productName.setText("")
        productQuantity.setText("")
    } else
        productID.text = "Nie znaleziono"
}
```

- Uruchamiamy