Chili

Android Workshop

Workbook

Chili

About me

My name is Darja Tretjakova. I'm an Android developer at @Chili. I've been developing apps for a little more than 4 years now. My developer career started the summer after school in my home town, Daugavpils. I also have a bachelor's degree in Computer Science from RTU.

Find me on Medium, LinkedIn or Instagram.
I would also love your feedback: you can mail me at darja.tretjakova@chi.lv

I hope you'll like my workshop :)

@daria.tr

Darja Tretjakova

@darja.tretjakova

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Android

Fun facts:

- Initial release date is 23.09.2008.
- Is open source (all source code is publicly accessbile.
- Until 10th version (the latest) all of the version names were called after desserts in alphabetical order. Google ditched that in 10th version. The letter would have been Q.
- Java used to be the only official language for developing Android apps. Now Kotlin is not only an official language, but also the preferred one.

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Glossary

Programming language

Fancy: a formal language, which comprises a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms (Wikipedia).

Simple: just like a natural language, but with very limited syntax. Under the hood it's all gets translated into 1 & 0 (on & off signals)

IDE (Integrated Development Environment)

Fancy: is a software application that provides comprehensive facilities to computer programmers for software development. (Wikipedia).

Simple: just a program to write programs. You can write programs in simple text files, but it's much harder. IDE just has some very cool tools to make developer's life easier.

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Glossary

API (Application Programming Interface)

Fancy: is an interface or communication protocol between a client and a server intended to simplify the building of client-side software. It has been described as a “contract” between the client and the server, such that if the client makes a request in a specific format, it will always get a response in a specific format or initiate a defined action.

Simple: “List of rules” that describe what you can & can’t do with some piece of software. API version in Android is essentially the same thing as just a new version of OS, but for developers. For example Android 7 is API 24, meaning that developers can write apps according to API 24 rules.

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Java

Java is one of the most of the popular programming languages. It has been around since 1995. An incredible tool, but it's quite hard for beginners because it has a lot of complex constructs. Used to be the only supported language for native Android apps until 2017.

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Kotlin

The new kid in town. Java with super powers. Essentially is a lot like Java, but with some redundant complexity removed & useful features added. Officially supported language for Android Development since 2017.

Once you try it after Java, you never go back 😅

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Programming

Variables

Variables are like atoms. You can't go smaller than that (well, you can, but let's not go there 🤓). And then there are chemical elements. In Kotlin there are 5 (not 118 as in the actual list of chemical elements 😅): Int (short for Integer), Float (short for floating point unit), Char (short for character), String (not short for anything) and Boolean.

- Integers are, well, integers.
- Floats are decimal numbers.
- Chars are single characters.
- Strings are lines of text. Or lines of chars.
- Boolean is a special type, it has only two values true or false. Nothing else. Just true or false.

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Programming

Variables

Ok, so if you would like to define a variable of some type in Kotlin, you would do it like this:

```
var number: Int = 4
var decimal: Float = 2.5F
var myNameInitial: Char = 'D'
var myName: String = "Daria"
var workshopIsAwesome: Boolean = true
var nullable: Int? = null
```

Now, there is one special case. Variable can be NULL. Null means that there is nothing inside this variable. It's empty. And we can't do anything with null values, otherwise we will have errors, because what can you do with something, that does not exist? That's why sometimes, before working with variables, we need to check, whether they really have something inside them. This is called a null check.

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Programming

Functions

Great. What good do I have from those so called variables? Well, you can do stuff with them. Perform all kinds of “functions” on them. For example, maths.

Let’s take a simple example c = a + b.

The result “c” will change depending on what a or b is.

Lets rename c to sum. So our example becomes

sum = a + b.

So if a = 3 & b = 4, then sum = 3 + 4 = 7.

And you can substitute those a & b with whatever numbers you want, the formula stays the same.

Let’s translate this to Kotlin:

```
fun sum(a: Int, b: Int): Int {
    var result = a + b
    return result
}
```

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Programming

Functions

Now. In Kotlin there is a special function. Every program written in Kotlin starts executing itself in a function called **main**. The computer, when you "tell" it to run a program written in Kotlin, looks for this function written (wherever) in code and starts from there. The code in the file is NOT linear.

```
fun main() {
    // program starts here. this is a comment.
}
```

"Comment" in code is some text, that is not translated by the computer into instructions. You can leave comments for yourself & fellow developers by "escaping" text with either // before it or inside /* your comment here*/. First is "single line comment", second one is "multiline comment".

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Programming

Functions

Function doesn't always have to return something or have arguments. For example **main**: it just executes what's inside it. Lets calculate sum & print in to console. Open "Kotlin Playground" - a website, that allows you to run Kotlin code in the browser.

```
play.kotlinlang.org
```

Also this website saves a lot of cookies, so if your code prints something odd or old, try to clear cookies for this browser. Even better: open it incognito for every task.

```
fun main() {
    var mySum = sum(1, 2)
    println("Sum: $mySum")
}

fun sum(a: Int, b: Int): Int {
    var result = a + b
    return result
}
```

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Programming

Conditionals

“If statement” is a logical construct for checking some condition. You can actually read it as a sentence: if some condition is true, do something. Otherwise do nothing. Or maybe otherwise do something else. The syntax is like this:

```
if (boolean condition) {
    // do something
} else if (another condition) {
    // do something else
} else {
    // do something if both conditions are false
}
```

Everything except if () {...} is optional.

To “check” something we use double equals sign: ==

Example: condition1 == condition2 is like a question “is condition1 the same thing as condition2?”

```
val sum = 3
if (sum == 3) {
    print(“Sum is 3”)
}
```

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Programming

Classes & objects

The hardest, but a very important part. Classes and object. Remember how I said that int, float etc. (further I'll call this whole group primitives) are like chemical elements? Well, objects are something that you can create by combining those chemical elements. If you combine H2 + O you'll get water. If you combine an int variable with name age and string variable with name firstName you can call it a Person. So you take some primitives, add some functions throw it into a mix with a keyword "class" (just like var) and here, you have a class.

```
class Person {
    var name: String = "Daria"
    var age: Int = 22
}
```

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Programming

Classes & objects

Now, you can go to **main** and create an OBJECT of CLASS Person. So class is basically just a description. When you create something according to this description, you create an object of this class.

```
var awesomePerson = Person()
```

And then you can change or look at the properties of this object. For example, let's print it.

```
fun main() {
    var awesomePerson = Person()
    // try to comment next line
    awesomePerson.name = "Alex"
    print("Person: ${awesomePerson.name}")
}

class Person {
    /* word public allows to use those fields
       in our main function */
    public var name: String = "Daria"
    public var age: Int = 22
}
```

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Programming

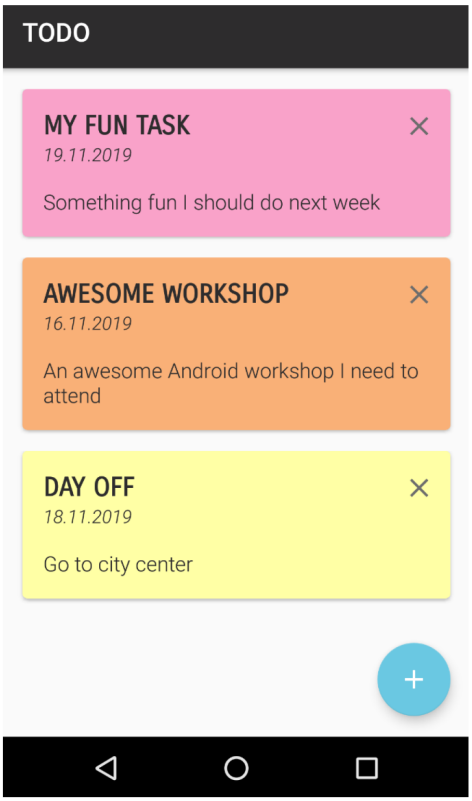
Ok, so this was a very brief and simplified introduction into programming. I hope you got the idea. Now it will escalate pretty fast, but keep in mind, that all of this stuff is breakable down to what we just discussed.

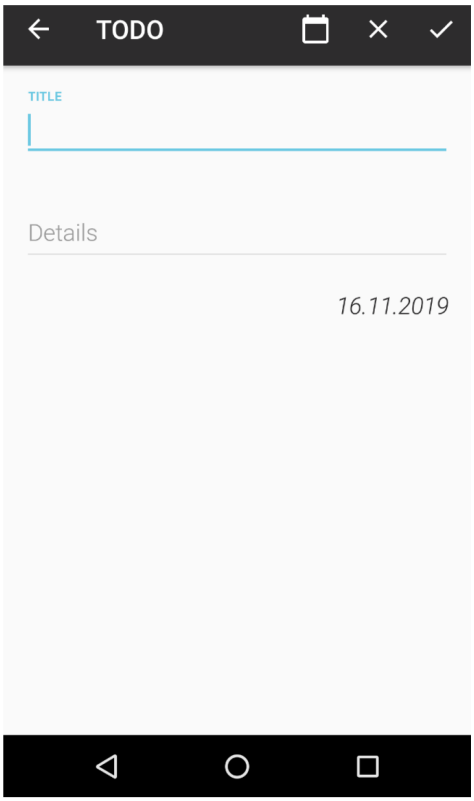
Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

TODO app

Description

1. First screen - list of TODO items with title, description and date. Button that opens a screen to add a new TODO item.
2. Second screen - form to create a TODO item with 2 input fields for title and details. Top menu with save, delete & date picker buttons.





Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Download project

Instructions

1. Download & install Android Studio

https://developer.android.com/studio

2. Download project from github

https://github.com/darjatretjakova/todo-workshop

3. Open Android Studio
4. File -> Open... -> Find downloaded project
5. On your phone, in settings (about phone/device) find your phone build number. Tap the build number 7 times to enable “developer options”
6. In developer options enable “USB debugging”
7. Connect your device to your laptop
8. Find this icon ▶ and press it
9. Pick your phone name from the dialog
10. Check your phone :)

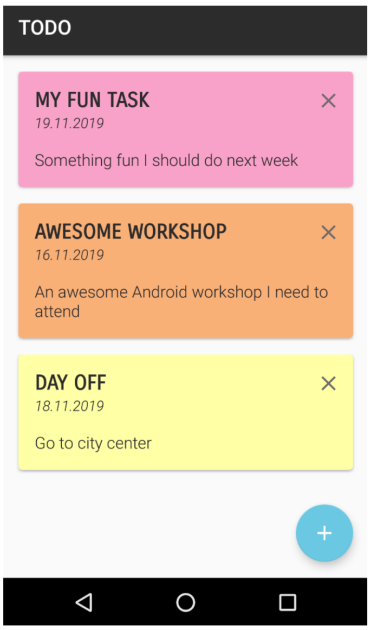
Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Activity

What is an Activity?

Activity = one screen. Screen with list of items = **ListActivity**. Screen with form for entering note = **FormActivity**. Activity is a special class, just like Person, but with it own functions and variables. It knows how to show buttons, what to do with them, etc. It does all of the logic. But how it actually looks is written in an xml file. Just like html for websites, but we have xml for Android. It's a markup language, not a programming language, just a special language to describe how the screen will look.

Remember how the program in Kotlin starts in a **main()** function? In Android it starts in an activity. In which activity you decide in a special file. In our project it will be **ListActivity.**

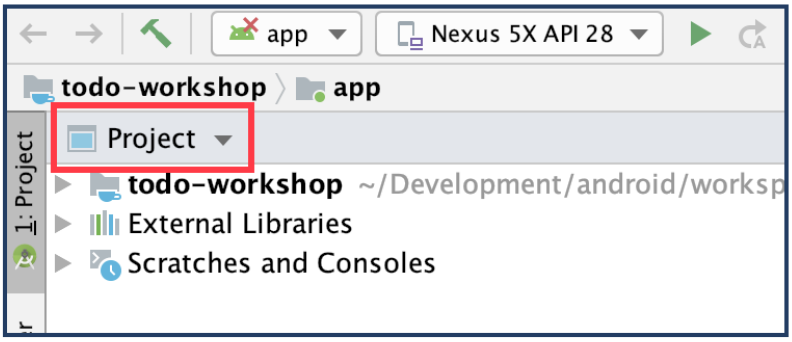


Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

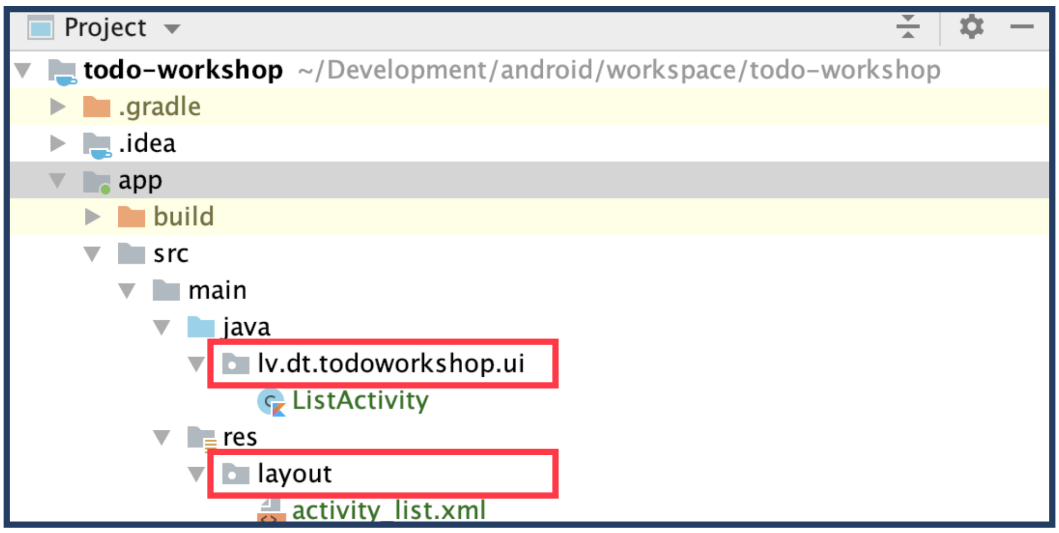
Android Studio

Structure of project

1. From dropdown menu select “Project”



2. In path *lv.dt.todoworkshop.ui* you will find all Activity and classes and in *res/layout/...* you will find all xml files.



Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Android Studio

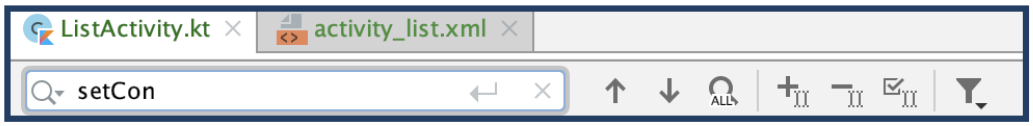
Useful shortcuts

1. Search filename in all files

Double SHIFT (Windows / Mac)

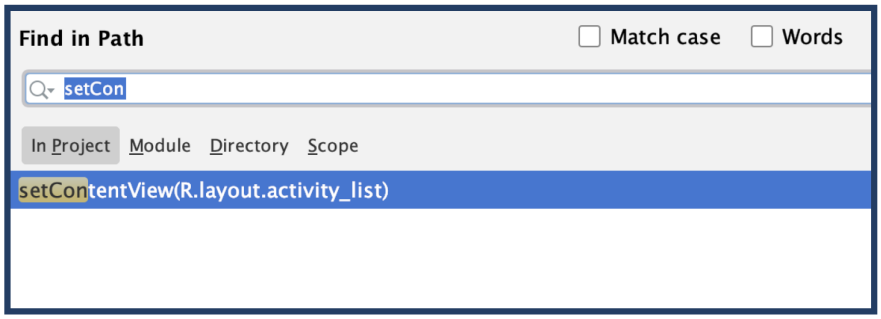
2. Search text in the currently opened file

CTRL + F / CMD + F (Windows / Mac)



3. Search text in all files

CTRL + SHIFT + F / CMD + SHIFT + F (Windows / Mac)



Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Task 1

Open FormActivity

Run the app. You should see an empty screen with header & button.

What happens, if we click on the button?

Nothing. So first what we need to do is react to the click of this button. And our reaction will be opening a new screen (new activity) with the form for our todo note.

So our note class will look like this:

```
// this is “pseudocode”, the field names are real
// but the real class in project is a bit
// more complex

class Note {
    var id: Long
    var title: String
    var details: String
    var date: String
}
```

We will write a function called `openFormActivity(noteId: Long?)`. We can use it for two cases: to save a new note, or update existing note. If we want to save a new note, we need to pass null to this function. If we want to update existing note, we will need do pass the id of this note. It will be called when “add” button is clicked.

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Task 1

Open FormActivity

Plan:

1. Set a click listener to “add” button. Find id of that button to set the listener.
2. On click of this button open FormActivity with null as noteId, because we are saving a completely new note.

Actions:

1. Our first activity is ListActivity. Let’s open it.
2. We see this: `setContentView(R.layout.activity_list)`, This means, that the xml that says how our activity looks like is called `activity_list.xml`. Let’s open it. (Open file by holding down CTRL (CMD for Mac) button and clicking on file name).
At the bottom you should see two tabs: “Design | Text”. Select Design.
3. At the bottom left in “ComponentTree” you should see a list of names. Click on “add”.
4. At the top right you should see an “Attributes” panel. Remember what’s written in the “id” field. To this id you will be setting the click listener.
5. Return to ListActivity. Find *“Task 1. Step 1.”* Set click listener and inside of it call `openFormActivity(null)`. If you’re stuck, highlight the *Cheat 1.1* on the last page of this file.
6. Find *“Task 1. Step 2.”* Create intent and call `startActivity` function. See *Cheat 1.2* in case you’re stuck.
7. Run the app, click the add button.

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Task 2

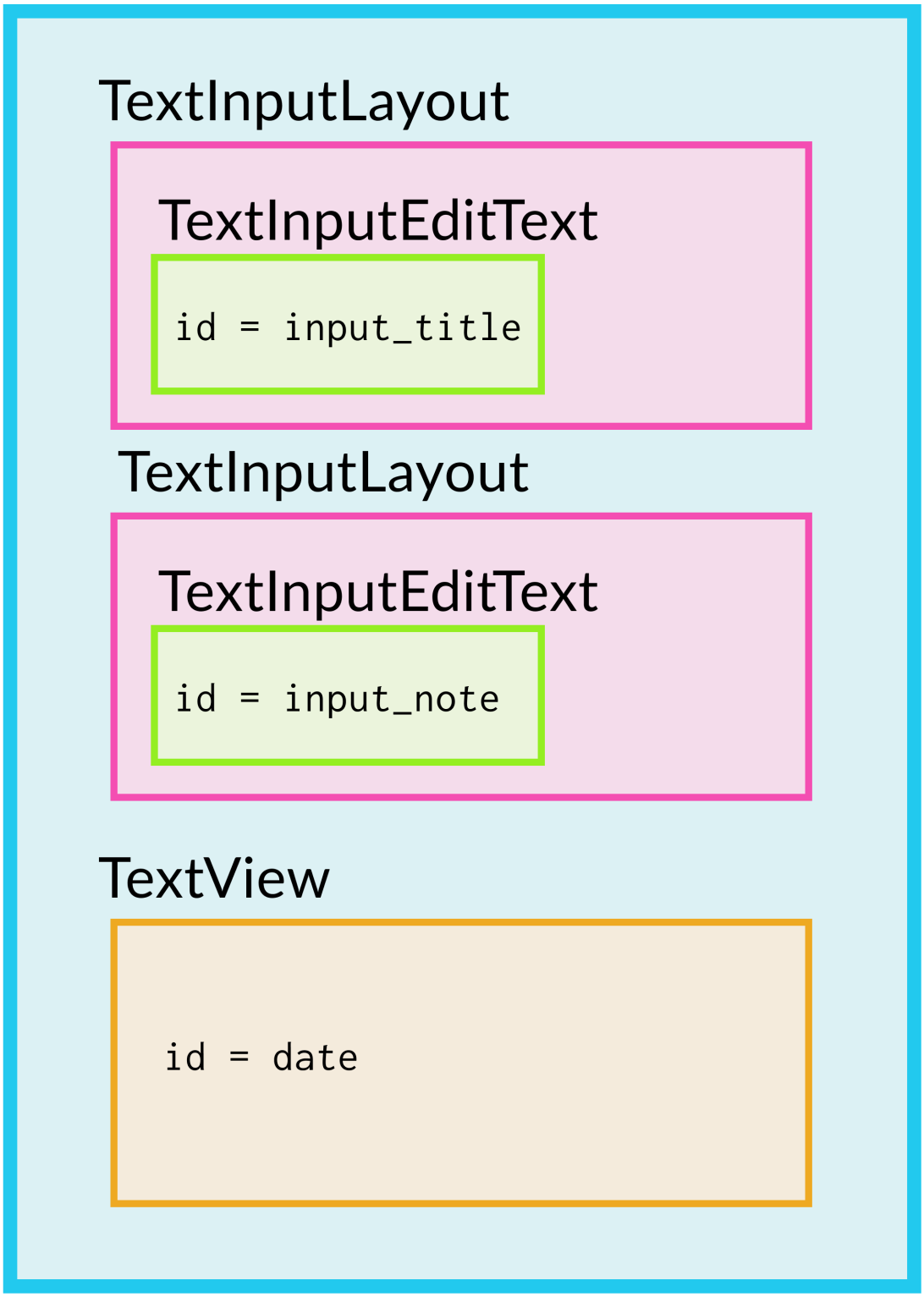
Create FormActivity layout

Building blocks of a screen are called Views. There is a TextView, Button, Image etc. They are all located inside of a Layout. Each view should have an id. All id's of views, menu items etc. that we create in our app are saved in a class called R. To access our id's we need to ask R class for them. R.layout.activity_list. Keyword "layout" here means that we are looking for a layout file with such name. R.id.add would mean that we are looking for a view somewhere in our app that has an id "add".
Layout is like a skeleton. It defines the geometry of the views inside it. There are a few types of layouts. We are going to use a LinearLayout. When you put views inside it, they all get stacked linearly, into one line, either horizontally or vertically, depending on attribute called "orientation".

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Task 2

LinearLayout



Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Task 2

Create FormActivity layout

Plan:

1. Create layout of activity_form.xml. It's a LinearLayout with vertical orientation.
2. Add 2 input fields for title and details. Add 1 text field for date.

Actions:

1. Open activity_form.xml.
2. From Design mode (Palette -> Text) add two views called TextInputLayout. Just click on it and drag to the screen. Set the hint of first TextInputLayout to "Title" and hint of second to "Details". You can find "hint" attribute in the attributes panel in top right.
3. IMPORTANT. TextInputLayout has a TextInputEditText inside of it. Set it's id to input_title. And the id of the TextInputEditText inside the second TextInputLayout to input_note.
4. Add a TextView. Set it's id to date.
5. You can also play around with different attributes: size, color, typeface, etc.
6. If you're stuck or want to have the exact same layout as in example, see *Cheat 2.1*.
7. Run the app, click the add button.

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Task 3

Add save & update functions

Plan:

1. When save button in the menu has been clicked, save note to database, if it's new, or update it in the database, if it's an existing note.

Actions:

1. Open FormActivity.
2. Find "Task 3. Step 1." or just open onSaveClick() function.
3. Check if currentNote is null. If true, call saveNote() function. Otherwise call updateNote() function. After check call finish() function to close screen and return to list. *Cheat 3.1.*
4. Find "Task 3. Step 2." or just open saveNote() function. Create a Note object. Pass currentDate as date, input_title.text.toString() as title and input_note.text.toString() as note. Call saveToDatabase() function and pass this note as param. *Cheat 3.2.*
5. Find "Task 3. Step 3." or just open updateNote() function. Create the same object as in 4th action, except also pass currentNote!!.id as id. Instead of saveToDatabase() , call updateInDatabase(). *Cheat 3.3.*

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Task 4

Fix FormActivity menu buttons

“When expression” is a construct a lot like if. You read it like this: When (some condition) is variant 1, do this, is variant 2, do that, variant 3, do something else etc. The syntax looks like this:

```
when (some condition) {
    variant1 -> {
        //do something
    }
    variant2 -> {
        //do something
    }
    variant3 -> {
        //do something
    }
    ...
}
```

We are going to check our menu button id’s. When menu button id is delete -> delete note, when save -> save note, when show date -> show date dialog, when back -> close screen.

Our menu items, just like views, have id’s. You can check them in file form_menu.xml. So we are going to react to them.

Android
Glossary
Java
Kotlin
Programming: Variables
Programming: Functions
Programming: Conditionals
Programming: Classes & objecs
TODO app
Download project
Activity
Android studio
Task 1
Task 2
Task 3
Task 4

Task 4

Fix FormActivity menu buttons

Plan:

1. Call correct functions according to menu button id's.

Actions:

1. When id is R.id.action_date, call showDateDialog(). Also return true after function call so that the system understands, that we handled the click and it doesn't need to do anything else. Do so after every action.
2. When id is R.id.action_delete, call deleteNote() function and finish() function to close screen. Don't forget to return true.
4. "Back" button is called "home" in Android. It's a system button, so it's id is in android.R. class, instead of just R. So when id is android.R.id.home, call finish() function to close screen. See Cheat 4.1 if you're stuck.

Cheats.

Cheat 1.1.

```
add.setOnClickListener {
        openFormActivity(null)
}
```

Cheat 1.2.

```
private fun openFormActivity(noteId: Long?) {
        val intent = FormActivity.createIntent(this, noteId)
        startActivity(intent)
}
```

Cheat 2.1.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/form_container"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:orientation="vertical">

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/layout_title"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:hint="title"
        android:layout_marginEnd="16dp"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/input_title"
            android:layout_height="match_parent"
            android:layout_width="match_parent"
            android:backgroundTint="#e2e2e2"
            android:fontFamily="sans-serif-smallcaps"
            android:singleLine="true"
            android:textAllCaps="true"
            android:textColor="@color/primary"
            android:textSize="20sp"
            android:textStyle="bold"
            android:typeface="normal" />
```

```
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/layout_note"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:hint="Details"
    android:layout_marginEnd="16dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="24dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/layout_title">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/input_note"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:backgroundTint="#e2e2e2"
        android:fontFamily="sans-serif-light"
        android:gravity="top"
        android:inputType="textMultiLine"
        android:singleLine="false"
        android:textColor="@color/primary"
        android:textSize="18sp" />
</com.google.android.material.textfield.TextInputLayout>

<TextView
    android:id="@+id/date"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:fontFamily="sans-serif-light"
    android:layout_gravity="bottom|end"
```

```
        android:layout_margin="16dp"
        android:layout_marginBottom="16dp"
        android:layout_marginStart="16dp"
        android:padding="2dp"
        android:textColor="@color/primary"
        android:textSize="18sp"
        android:textStyle="italic"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/layout_note" />
</LinearLayout>
```

Cheat 3.1.

```
private fun onSaveClick() {
    if (currentNote == null) {
        saveNote()
    } else {
        updateNote()
    }
    finish()
}
```

Cheat 3.2.

```
private fun saveNote() {
    val newNote = Note(
        date = currentDate,
        title = input_title.text.toString(),
        note = input_note.text.toString()
    )
    saveToDatabase(newNote)
}
```

Cheat 3.3.

```
private fun updateNote() {
    val newNote = Note(
        id = currentNote!!.id,
        date = currentDate,
        title = input_title.text.toString(),
        note = input_note.text.toString()
    )
    updateInDatabase(newNote)
}
```

Cheat 4.1.

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
        when(item.itemId) {
            R.id.action_save -> {
                onSaveClick()
                return true
            }
            R.id.action_date -> {
                showDateDialog()
                return true
            }
            R.id.action_delete -> {
                deleteNote()
                finish()
                return true
            }
            android.R.id.home -> {
                finish()
                return true
            }
        }
        return super.onOptionsItemSelected(item)
    }
```