

# Android

## WORKSHOP

Darja Tretjakova

[darja.tretjakova@chi.lv](mailto:darja.tretjakova@chi.lv)

@daria.tr

# Chili



@chililabs / chi.lv

# About me

- Android developer at @Chili
- 4 years
- Computer Science in RTU



Darja Tretjakova



@daria.tr



@darja.tretjakova



# Contents

- 5-6 Android, Java & Kotlin
- 7-12 Kotlin syntax
- 13-20 Android app structure
- **21** Android Studio shortcuts (!)
- 22-25 TODO app creation & configuration
- 26-44 TODO app tasks
- 45-46 TODO app extra tasks
- **47** Useful links (kotlin cheatsheet, project on github, Android docs)
- **48-56** TODO app classes full code

# Android

- Initial release date: 23.09.2008
- Open source
- Smartphones, tablets, watches, TV, game consoles etc.
- Until 10th version the names of new versions were desserts. Now Google ditched that & it's officially just Android 10. Bummer. Would have been letter Q
- Java until 2017, Kotlin since 2017



# Java & Kotlin

- 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.
- The new kid in town. Kotlin 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 😄



# Kotlin: Variables

- Kotlin doesn't have primitive types, everything is an object

`var x: Int = 1`

key word      name      type      value

# Kotlin: Variables

- Examples. Objects can be nullable or not nullable. To mark object as nullable, add “?” to type. If you try to set null to a non nullable variable, programme won’t compile.
- There are 2 types of variables: **val** and **var**. **val** can’t be changed after initialization, **var** can.

```
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
```

```
val number: Int = 4
number = 5 // won't compile
var decimal: Float = 2.5F
decimal = 3.2F // ok
```



# Kotlin: Functions

- Functions with / without parameters and return types

```
fun main() {  
    var mySum = sum(1, 2)  
    println("Sum: $mySum")  
}  
  
fun sum(a: Int, b: Int): Int {  
    var result = a + b  
    return result  
}
```

# Kotlin: Functions

- Comments look like this.

```
fun main() {  
    // this is a single line comment.  
    /* this  
       is a multiline  
       comment  
    */  
}
```

# Kotlin: Classes & objects

```
class Person {  
    public var name: String = "Daria"  
    private var age: Int = 22  
}  
  
class Car: Vechile {  
    override var color: String = "red"  
}  
  
class Car(val color: String): Vechile {  
    private var color: String = ""  
    init {  
        this.color = color  
    }  
}
```

# Kotlin: Practice

- Open `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.*

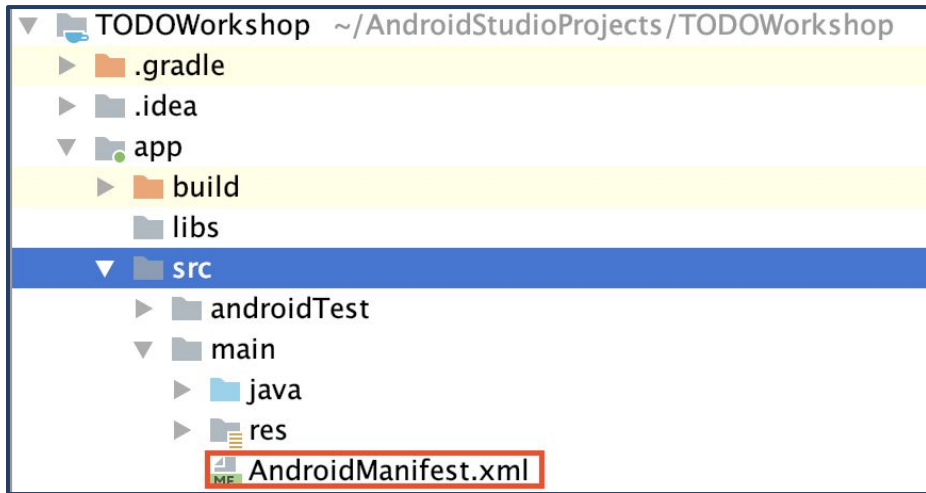
# Android: **Android Studio**

- Android Studio is an awesome IDE, which has many great tools for developing Android and Flutter apps. It is built on JetBrains's IntelliJ IDEA software.



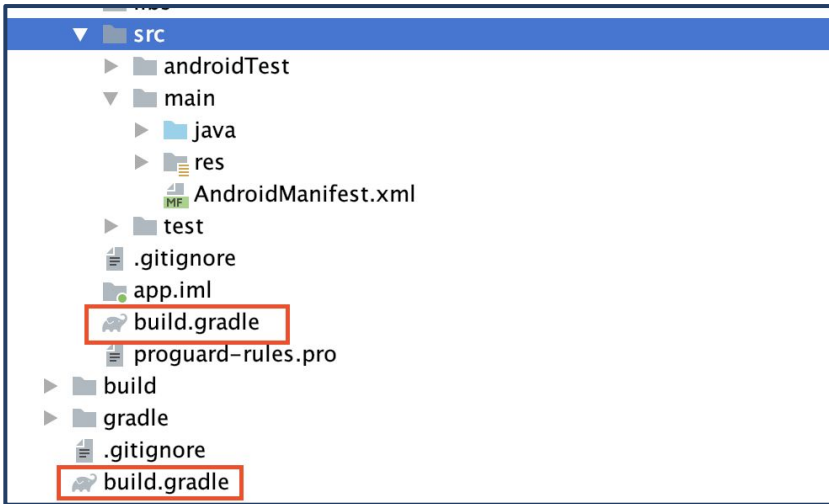
# Android app structure: **Android Manifest**

- Every app project must have an ***AndroidManifest.xml*** file (with precisely that name) at the root of the project source set. The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.
- App package name
- App components (activities, broadcast receivers, services, content providers)
- App permissions
- Hardware and software features the app requires



# Android app structure: Gradle

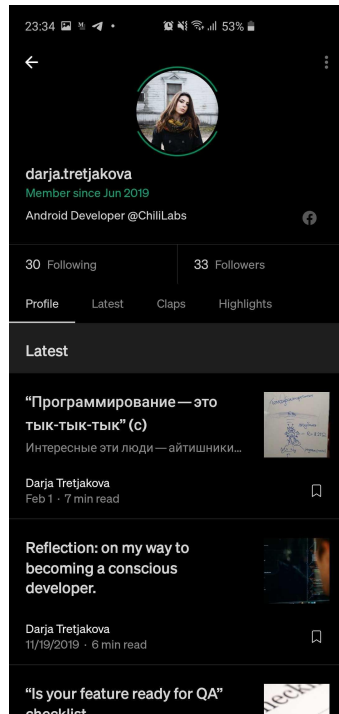
- Gradle is a build system that makes the task of building all .xml, .java, .kt & etc. files into an actual app file (.apk) easy for developers. It also takes care of all dependencies and libraries that your app needs.
- Top-level build.gradle
- Module-level build.gradle



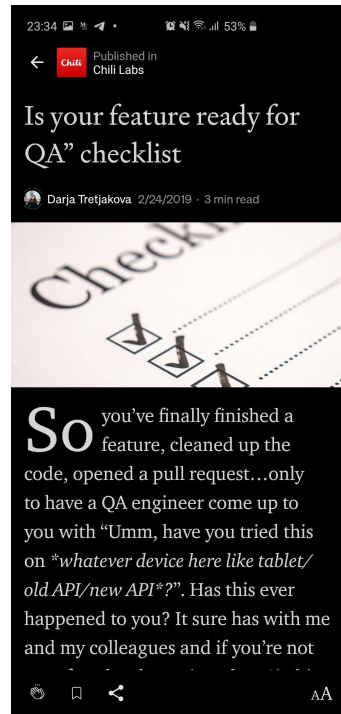
# Android app structure: Activity

- Activity is one of the main building blocks of an Android app. According to docs,

*An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI.*



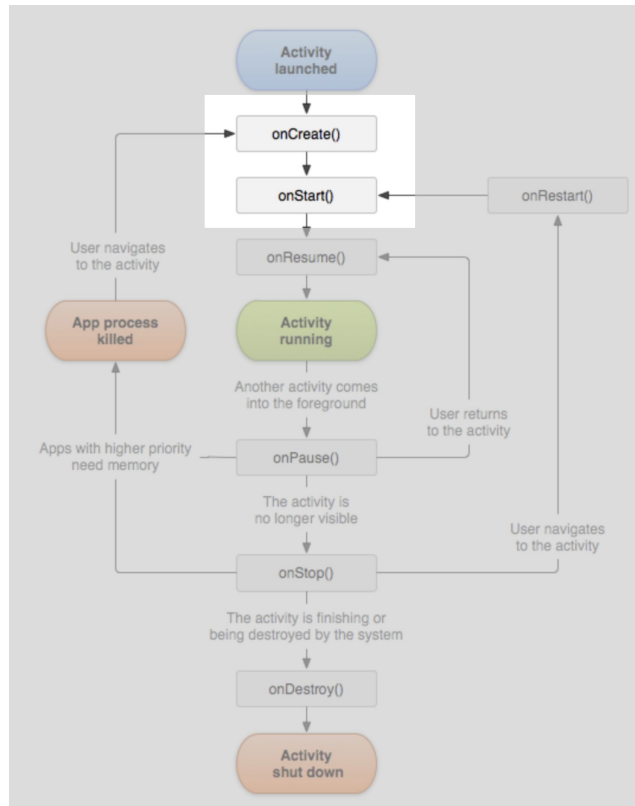
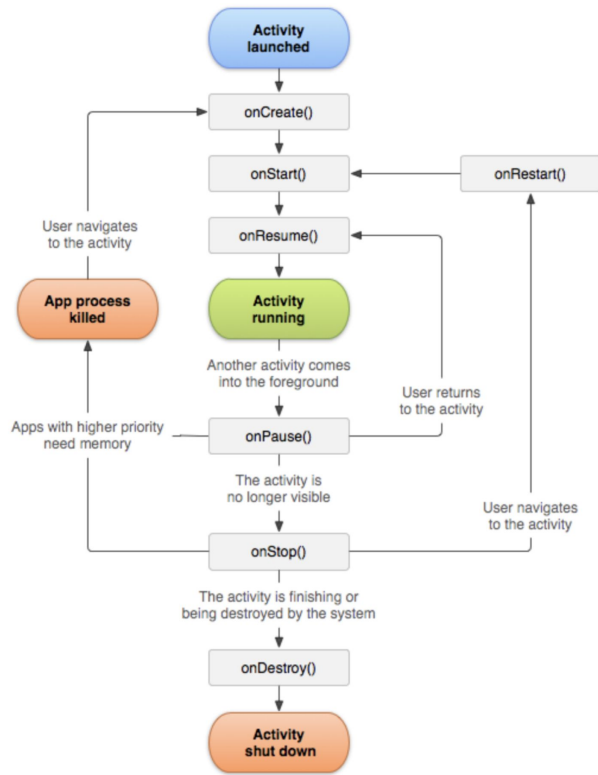
ProfileActivity



PostActivity



# Android app structure: Activity lifecycle



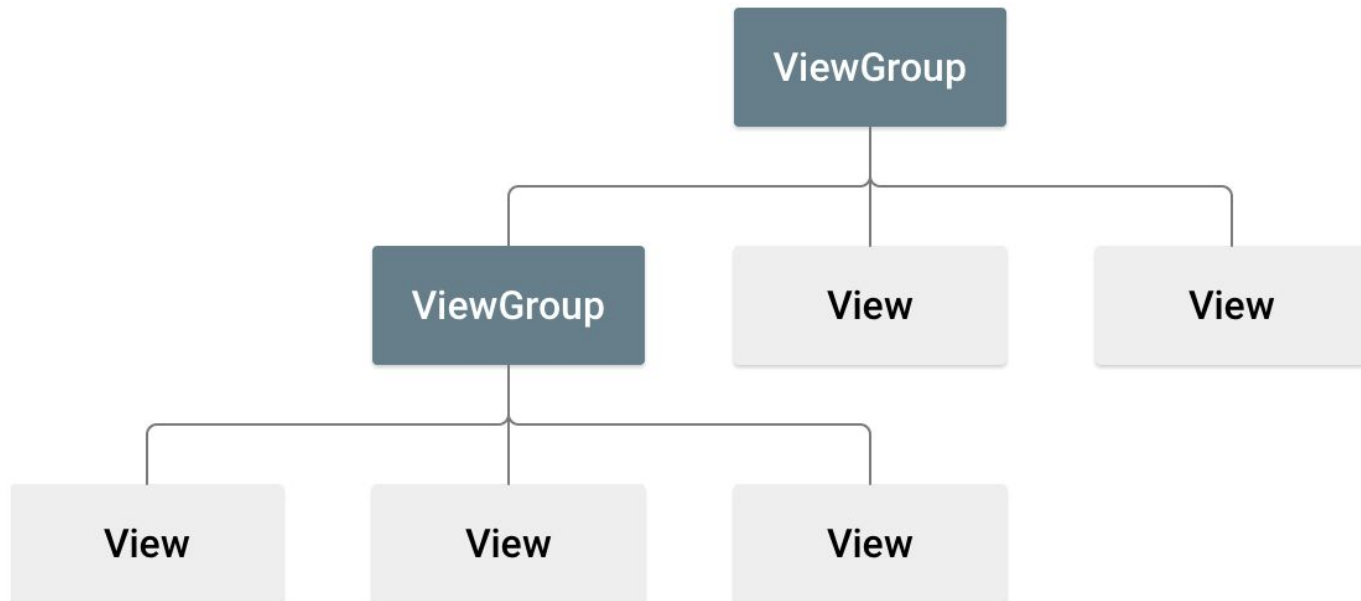
# Android app structure: Layouts / XML

A layout defines the structure for a user interface in your app, such as in an activity. A layout defines the structure for a user interface in your app, such as in an activity. All elements in the layout are built using a hierarchy of `View` and `ViewGroup` objects. A `View` usually draws something the user can see and interact with. Whereas a `ViewGroup` is an invisible container that defines the layout structure for `View` and other `ViewGroup` objects.

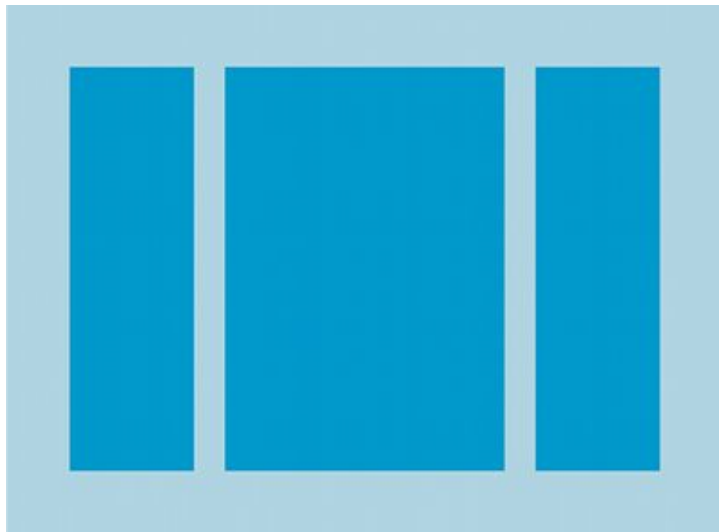
The `View` objects are usually called "widgets" and can be one of many subclasses, such as `Button` or `TextView`. The `ViewGroup` objects are usually called "layouts" can be one of many types that provide a different layout structure, such as `LinearLayout` or `ConstraintLayout`.

Declaring your UI in XML allows you to separate the presentation of your app from the code that controls its behavior. Using XML files also makes it easy to provide different layouts for different screen sizes and orientations (discussed further in Supporting Different Screen Sizes).

# Android app structure: **Layouts / XML**



# Android app structure: **Layouts / XML**



*LinearLayout*

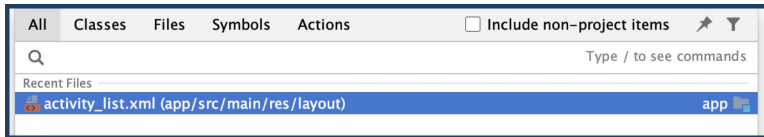


*ConstraintLayout*

# Android Studio: Shortcuts

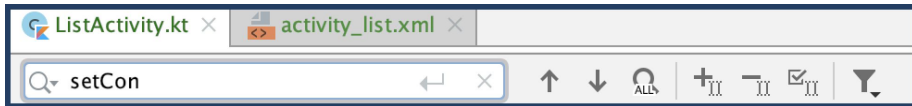
## 1. Search filename in all files

*Double SHIFT (Windows / Mac)*



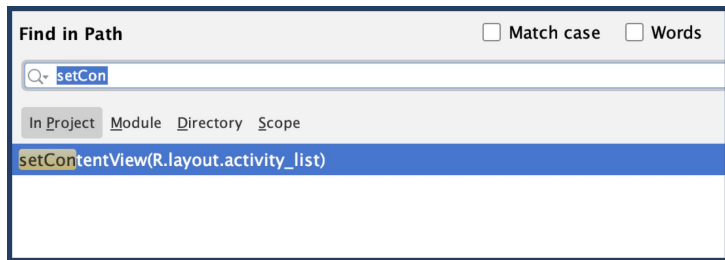
## 2. Search text in the currently opened file

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

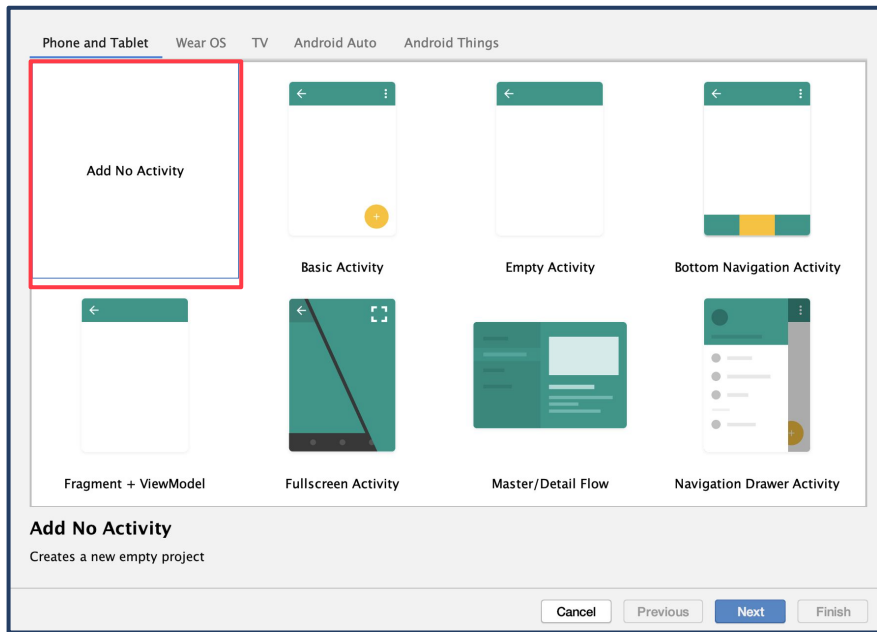
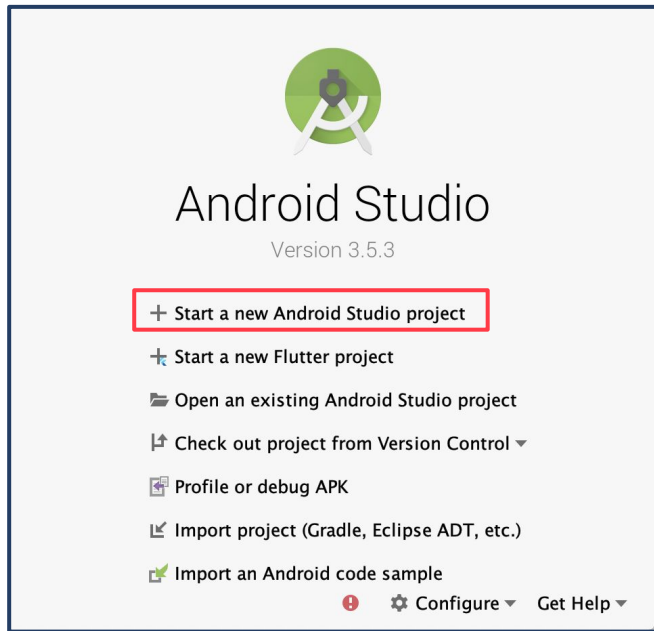


## 3. Search text in all files

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



# TODO App: Create project



# TODO App: Create project

Create New Project

Configure your project

Name  
TODO-workshop

Package name  
lt.todo.workshop

Save location  
/Users/darjatretjakova/AndroidStudioProjects/TODOWorkshop2

Language  
Kotlin

Minimum API level  
API 21: Android 5.0 (Lollipop)

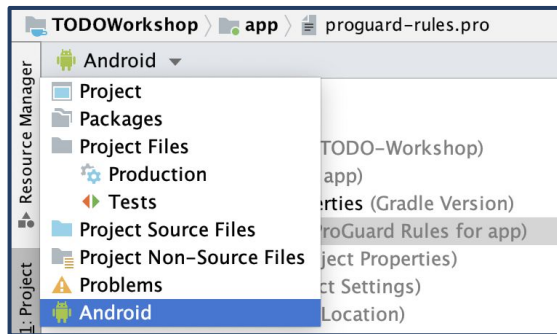
**i** Your app will run on approximately 85.0% of devices.  
[Help me choose](#)

☐ This project will support instant apps

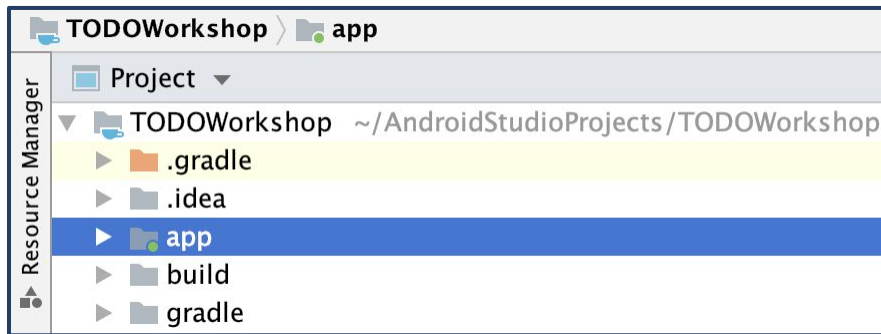
☒ Use androidx.\* artifacts

Cancel Previous Next **Finish**

# TODO App: Configure project



*Dropdown menu (select "Project")*

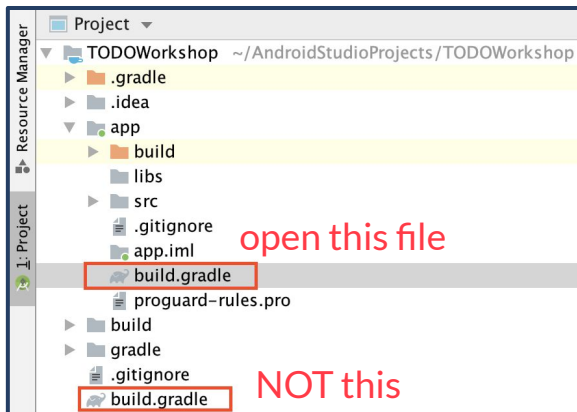


*How it should look like*



# TODO App: Add dependencies

1. Open build.gradle file



2. Add this line at the end of dependencies {}

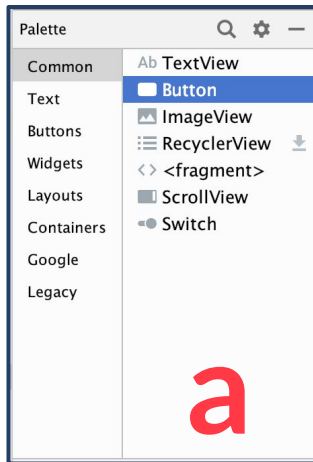
```
implementation "com.google.android.material:material:1.2.0-alpha04"
```

3. Click "sync now"

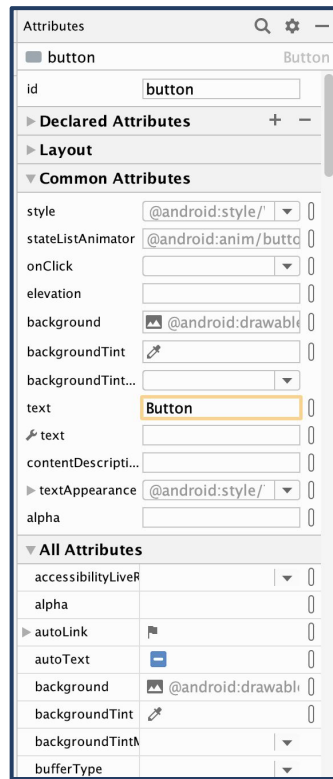
Sync Now

# TODO App: ListActivity

- Find “**Button**” in Palette, drop it. (a)
- Select it in Component Tree (b)
- Change **id** to “**add**” in Attributes (c)
- Change **layout\_gravity** to “**bottom**” and “**center\_horizontal**” in Attributes (c)
- Change text to “**add task**” in Attributes (c)
- See full code for `activity_list.xml` on slide X in case you're stuck



c



# TODO App: ListActivity

1. src / main / java -> It -> workshop -> todo -> right click -> new -> Kotlin file / class -> ListActivity
2. See full code for ListActivity on slide 48 in case you're stuck

```
class ListActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_list)  
    }  
}
```


! To import a class, set cursor on it's name and click alt + enter / option + enter on Windows / Mac

# TODO App: ListActivity

1. Open AndroidManifest.xml and replace code INSIDE **manifest** tag (slide 55)

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme" >
    <activity android:name=".ListActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

# TODO App: Launch

- On your phone, in settings (about phone/device) find your phone build number. Tap the build number 7 times to enable “developer options”
- In developer options enable “USB debugging”
- Connect your device to your laptop
- Find this icon  and press it
- Pick your phone name from the dialog
- Check your phone :)

# TODO App: **FormActivity**

1. Create a new layout, call it `activity_form`, change `LinearLayout` to `FrameLayout`
  - 1.1. Add `EditText`, change `id` to `task_text`, `hint` to “Task”
  - 1.2. You can also add paddings / margins. For that define numbers in dp, e.g. **16dp**
  - 1.3. Add `Button`, change `id` to `save`, `layout_gravity` to `bottom` and `end` (*DON'T mix up with just gravity*)
2. *See full code for `activity_form.xml` on slide 53 in case you're stuck*
3. Create new activity, call it `FormActivity`
4. *See full code for `FormActivity` on slide 49 in case you're stuck*

# TODO App: Open FormActivity

1. Set a click listener on add task button in ListActivity inside **onCreate**
2. Create an **Intent**, pass **this** as context and FormActivity::class.java as class
3. Call **startActivity()** function and pass created intent as argument
4. Run app

```
add.setOnClickListener {  
    var intent = Intent(this, FormActivity::class.java)  
    startActivity(intent)  
}
```

# TODO App: Database

1. Create a data class called Task with one String field called text

```
data class Task(val text: String)
```

2. Create an object called Database. It will have a list of tasks as a private field and two methods: get & add

```
object Database {  
    private var tasks: MutableList<Task> = mutableListOf()  
  
    fun add(task: Task) {  
        tasks.add(task)  
    }  
  
    fun get(): List<Task> {  
        return tasks.toList()  
    }  
}
```



# TODO App: Save note

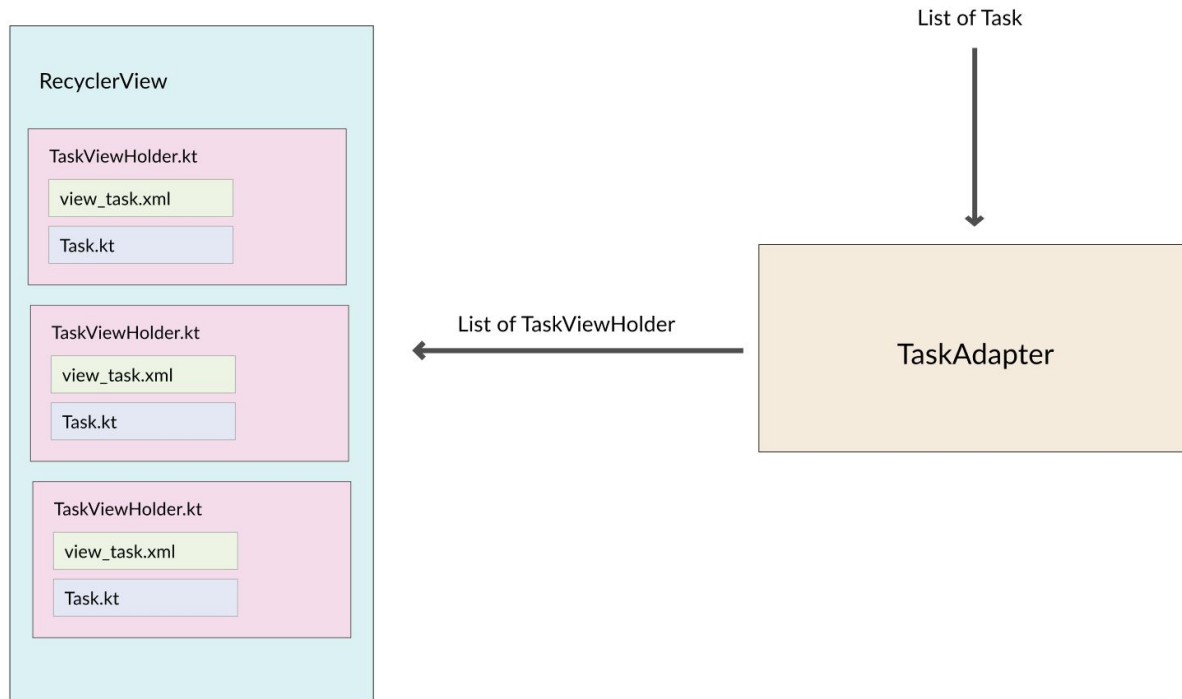
1. Set a click listener on save button in FormActivity onCreate.
2. Read text from EditText and create a Task.
3. Add this Task to Database.
4. Call finish() -> a method that closes current Activity
5. Run app

```
save.setOnClickListener {  
    var text = task.text.toString()  
    var newTask = Task(text)  
    Database.add(newTask)  
    finish()  
}
```

# TODO App: List logic

1. A view responsible for showing list on screen -> RecyclerView
2. View\_task.xml -> layout that describes how EACH list item will LOOK like
3. Task.kt -> class with tasks DATA (text, date, title etc.)
4. TaskViewHolder -> class that BINDS how item LOOKS like with DATA
5. TaskAdapter -> takes a list of Task.kt (pure data) and turns it into a list of TaskViewHolder

# TODO App: List logic



# TODO App: Add RecyclerView

1. Add a RecyclerView to activity\_list from Palette. Change **id** to **list**.

# TODO App: Create task view layout

1. Create a layout called `view_task`
2. Add `TextView` with id **`task_text`**
3. Change `TextView` height to `wrap_content`
4. Change (if you wish) text color and style
5. Add paddings and margins to `FrameLayout`
6. Add background to `FrameLayout`
7. Change height to `wrap_content`
8. *See full code on slide 53 in case you're stuck*

# TODO App: Create TaskViewHolder

1. Create a class called TaskViewHolder
2. Pass View as a constructor parameter
3. Inherit from RecyclerView.ViewHolder and pass view into constructor
4. Create a method called bind with a Task object parameter
5. Set tasks text to itemView.task\_text.text

```
class TaskViewHolder(view: View): RecyclerView.ViewHolder(view)
{
    fun bind(task: Task) {
        itemView.task_text.text = task.text
    }
}
```

# TODO App: Create TaskAdapter

1. Create a class called TaskAdapter
2. Inherit from RecyclerView.Adapter<TaskViewHolder>()
3. Override getItemCount(), onCreateViewHolder(), onBindViewHolder()

```
class TaskAdapter: RecyclerView.Adapter<TaskViewHolder>() {  
  
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TaskViewHolder {  
        TODO("not implemented") //To change body of created functions use File | Settings | File Templates.  
    }  
  
    override fun getItemCount(): Int {  
        TODO("not implemented") //To change body of created functions use File | Settings | File Templates.  
    }  
  
    override fun onBindViewHolder(holder: TaskViewHolder, position: Int) {  
        TODO("not implemented") //To change body of created functions use File | Settings | File Templates.  
    }  
  
}
```

# TODO App: Set tasks in adapter

1. Create a private field called **tasks** of type **List<Task>** and init it as *emptyList()*
2. Create a method **set()** with **List<Task>** as parameter
3. Set argument value to **this.tasks**
4. Call **notifyDataSetChanged()** to inform Adapter of data change
5. Return list size in *getItemCount()*

```
private var tasks: List<Task> = emptyList()

fun set(tasks: List<Task>) {
    this.tasks = tasks
    notifyDataSetChanged()
}

override fun getItemCount(): Int {
    return tasks.size
}
```



# TODO App: OnBindViewHolder

1. Get task from list at **position**
2. Call **bind()** method of holder and pass task

```
override fun onBindViewHolder(holder: TaskViewHolder, position: Int) {  
    var item = tasks[position]  
    holder.bind(item)  
}
```

# TODO App: **OnCreateViewHolder**

1. Create LayoutInflater and pass context of parent argument as context
2. Create a view by calling inflate method of LayoutInflater, pass R.layout.view\_task, parent and false as arguments
3. Create a viewholder from obtained view
4. Return viewholder

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TaskViewHolder {  
    var context = parent.context  
    var inflater = LayoutInflater.from(context)  
    var view = inflater.inflate(R.layout.view_task, parent, false)  
    var viewHolder = TaskViewHolder(view)  
    return viewHolder  
}
```

# TODO App: Bind adapter

1. Create a private field adapter of type TaskAdapter in ListActivity
2. In onCreate set this adapter to list
3. Set LinearLayoutManager() as list.layoutManager and pass this as context

*Inside class:*

```
var adapter = TaskAdapter()
```

*Inside onCreate:*

```
list.layoutManager = LinearLayoutManager(this)
```

```
list.adapter = adapter
```

# TODO App: Bind list data

1. Override onStart method
2. Pass Database list to adapter via adapter's set method
3. Run app

```
override fun onStart() {  
    super.onStart()  
  
    adapter.set(Database.get())  
}
```

# TODO App extra: Back button in FormActivity

1. Set up back button by calling `setDisplayHomeAsUpEnabled(true)` on `supportActionBar` in `onCreate`
2. Override `onOptionsItemSelected` and check item if. If it's `android.R.id.home`, then call `finish()`

```
supportActionBar?.setDisplayHomeAsUpEnabled(true)
```

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    val id = item.itemId  
    if (id == android.R.id.home) {  
        finish()  
        return true  
    } else return super.onOptionsItemSelected(item)  
}
```

# TODO App extra: Show toast

1. In save buttons onClick show Toast by calling Toast.makeText()...

```
Toast.makeText(this, "Saved", Toast.LENGTH_SHORT).show()
```

# Useful links

1. Full workshop code: <https://github.com/darijatretjakova/TODOWorkshop>
2. Kotlin cheatsheet: [https://kt.academy/static/Kotlin Cheat Sheet.pdf](https://kt.academy/static/Kotlin%20Cheat%20Sheet.pdf)
3. Android documentation: <https://developer.android.com/docs>

# TODO App: ListActivity.kt

```
package lt.workshop.todo

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import kotlinx.android.synthetic.main.activity_list.*

class ListActivity : AppCompatActivity() {

    private val adapter = TaskAdapter()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_list)

        add.setOnClickListener {
            val intent = Intent(this, FormActivity::class.java)
            startActivity(intent)
        }

        list.layoutManager = LinearLayoutManager(this)
        list.adapter = adapter
    }

    override fun onStart() {
        super.onStart()

        adapter.set(Database.get())
    }
}
```



# TODO App: **FormActivity.kt**

```
package lt.workshop.todo

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import kotlinx.android.synthetic.main.activity_form.*

class FormActivity: AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_form)

        save.setOnClickListener {
            val text = task.text.toString()
            val newTask = Task(text)
            Database.add(newTask)
            finish()
        }
    }
}
```

# TODO App: Task.kt, TaskViewHolder.kt

```
package lt.workshop.todo

data class Task(val text: String)
```

```
package lt.workshop.todo

import android.view.View
import androidx.recyclerview.widget.RecyclerView
import kotlinx.android.synthetic.main.view_task.view.*

class TaskViewHolder(view: View): RecyclerView.ViewHolder(view) {
    fun bind(task: Task) {
        itemView.task_text.text = task.text
    }
}
```

# TODO App: TaskAdapter.kt

```
package lt.workshop.todo

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView

class TaskAdapter: RecyclerView.Adapter<TaskViewHolder>() {

    private var tasks: List<Task> = emptyList()

    fun set(tasks: List<Task>) {
        this.tasks = tasks
        notifyDataSetChanged()
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TaskViewHolder {
        val context = parent.context
        val inflater = LayoutInflater.from(context)
        val view = inflater.inflate(R.layout.view_task, parent, false)
        return TaskViewHolder(view)
    }

    override fun getItemCount(): Int {
        return tasks.size
    }

    override fun onBindViewHolder(holder: TaskViewHolder, position: Int) {
        val item = tasks[position]
        holder.bind(item)
    }
}
```

# TODO App: activity\_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/add"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|center_horizontal"
        android:text="Add task" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

# TODO App: activity\_form.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/task"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:ems="10"
        android:hint="Task"
        android:inputType="textPersonName" />

    <Button
        android:id="@+id/save"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        android:text="Save" />
</FrameLayout>
```

# TODO App: `view_task.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/task"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:ems="10"
        android:hint="Task"
        android:inputType="textPersonName" />

    <Button
        android:id="@+id/save"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        android:text="Save" />
</FrameLayout>
```

# TODO App: **AndroidManifest.xml**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="lt.workshop.todo">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity android:name=".ListActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".FormActivity" />
    </application>
</manifest>
```

# TODO App:

## build.gradle

### (Module)

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"
    defaultConfig {
        applicationId "lt.workshop.todo"
        minSdkVersion 21
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.core:core-ktx:1.2.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
    implementation "com.google.android.material:material:1.2.0-alpha04"
}
```



# Chili



@chililabs / chi.lv

<https://bit.ly/2HK43gd>