Оглавление

[1. Обобщения 2](#_Toc94022046)

[Дженерики 2](#_Toc94022047)

[Обобщенные функции 3](#_Toc94022048)

[Ограничение типа Bounded generics 4](#_Toc94022049)

[2. Вариантность и инвариантность 4](#_Toc94022050)

[Типы и подтипы 4](#_Toc94022051)

[Контрвариантные параметры типов 8](#_Toc94022052)

[Класс может быть ковариантным по одному параметру и контрвариантным по второму 9](#_Toc94022053)

[3. Лямбда выражения, функции высшего порядка, inline функции 10](#_Toc94022054)

[Лямбда функция для коллекции 11](#_Toc94022055)

[4. Разметка 11](#_Toc94022056)

[Позиционирование внутри родителя 12](#_Toc94022057)

[Видимость элементов 13](#_Toc94022058)

[код 13](#_Toc94022059)

[5. Виды UI элементов 13](#_Toc94022060)

[TextView 14](#_Toc94022061)

[EditText поле ввода 15](#_Toc94022062)

[Button кнопка 16](#_Toc94022063)

[ImageView Изображения 17](#_Toc94022064)

[ProgressBar 17](#_Toc94022065)

[Checkbox флажок 18](#_Toc94022066)

[Swith как флажок, только выполняется моментально 18](#_Toc94022067)

[RadioGroup радиокнопка 19](#_Toc94022068)

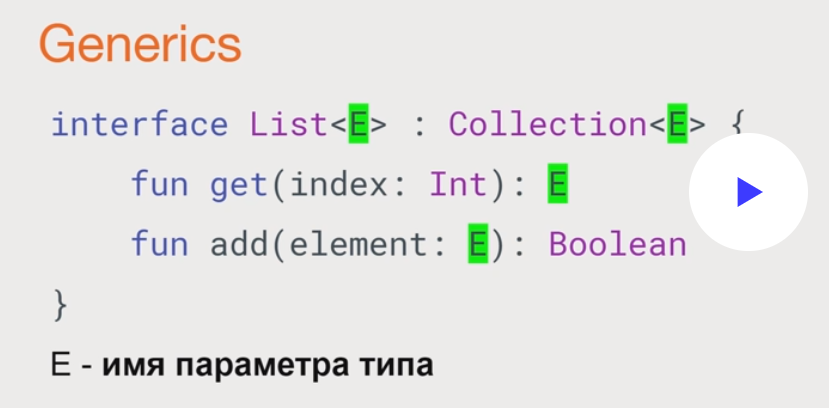
[6. ViewGroups 19](#_Toc94022069)

[FrameLayout 19](#_Toc94022070)

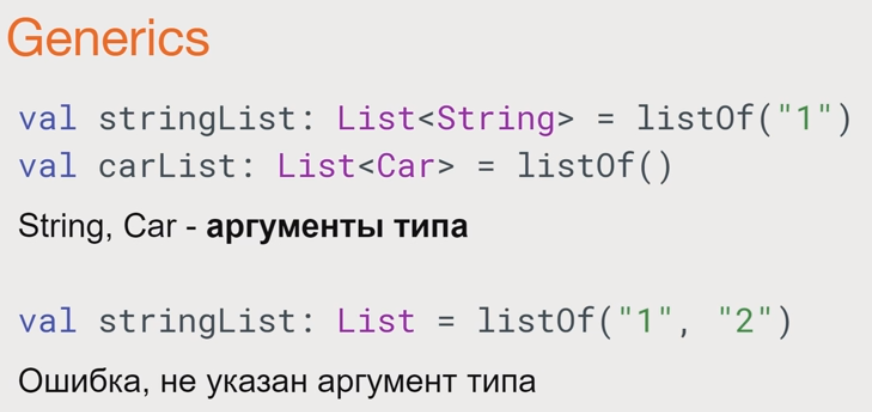
# Обобщения

## Дженерики

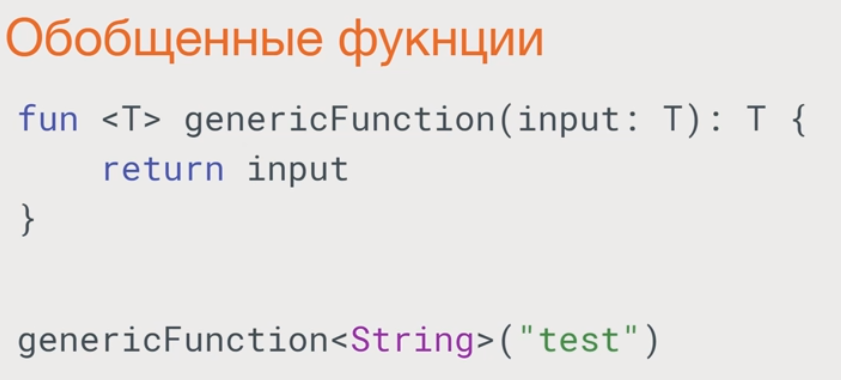
Дженерики позволяют указать, что код будет работать не с конкретным типом данных, а с любым типом, который указан в момент использования.



Параметры типа указываются в укгловых скобках.



## Обобщенные функции



package com.application.balticactivity

fun main() {

val genericObject = GenericClass<String?, Int>(defaultValue = "String")

genericObject.updateItem("dsg", 341)

printGenericObject<String?>("dsg")

}

class GenericClass<T, R>(defaultValue: T) {

var item: T? = null

private set

fun updateItem(newItem: T, returnItem: R):R {

item = newItem

//item is String доступна проверка на тип

return returnItem

}

}

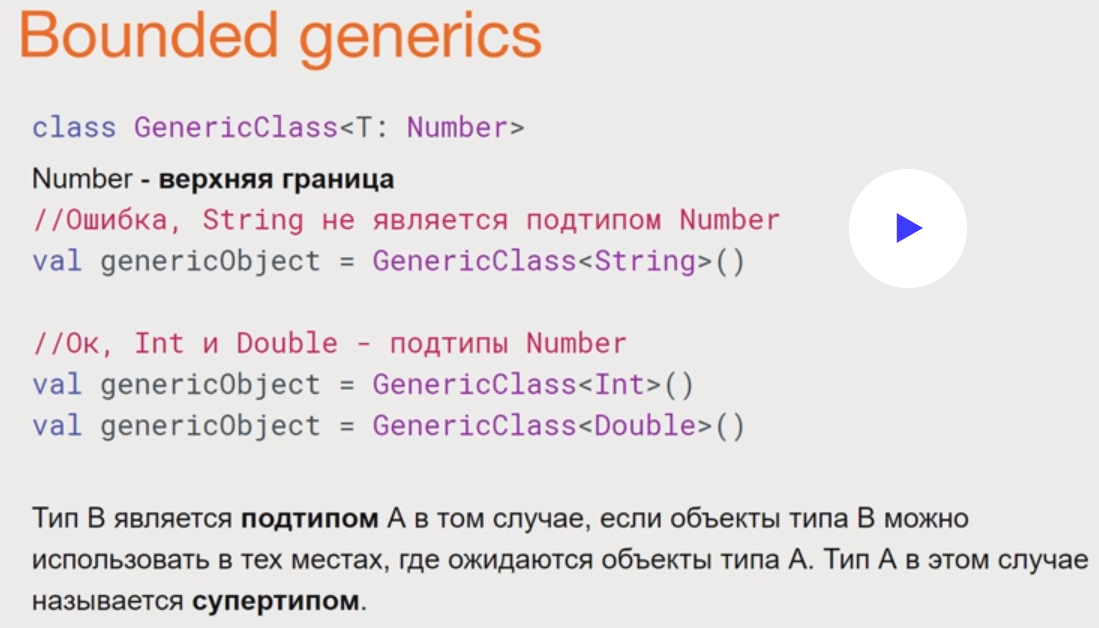
//обобщенная функция

fun <T> printGenericObject(item: T) {

println(item.toString())

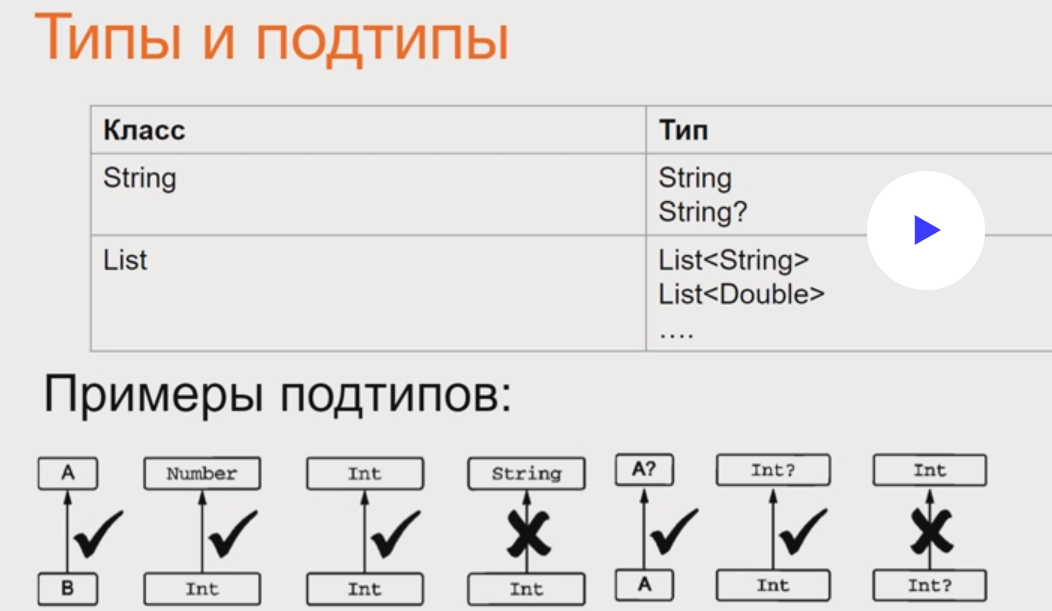
}

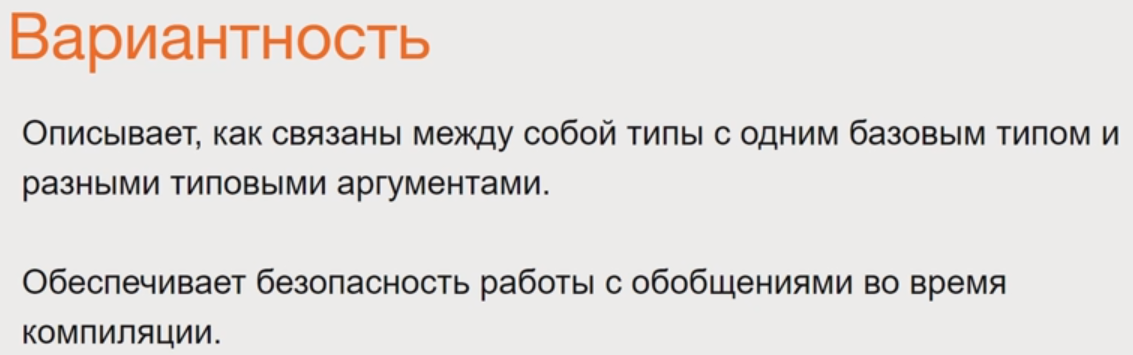
## Ограничение типа Bounded generics

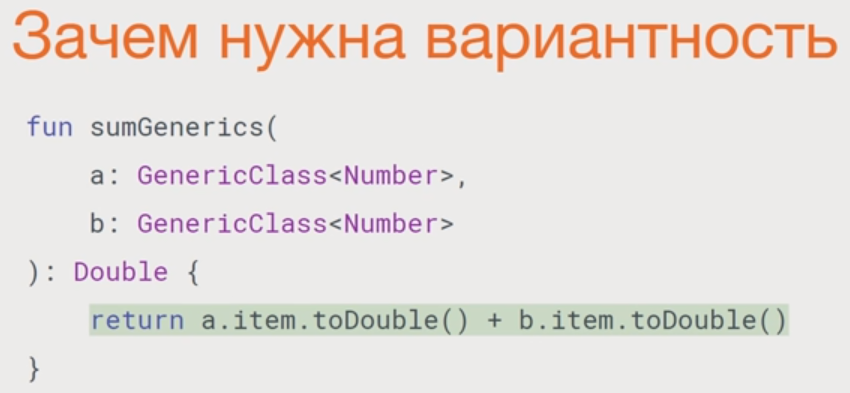


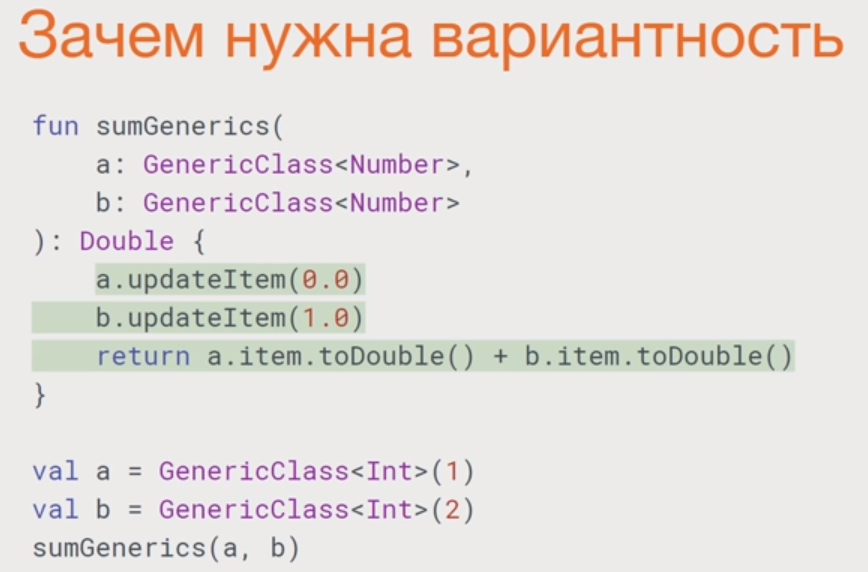
# Вариантность и инвариантность

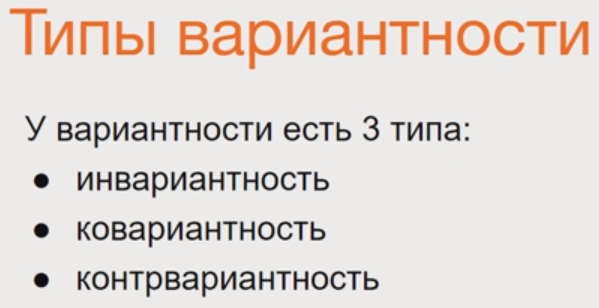
## Типы и подтипы



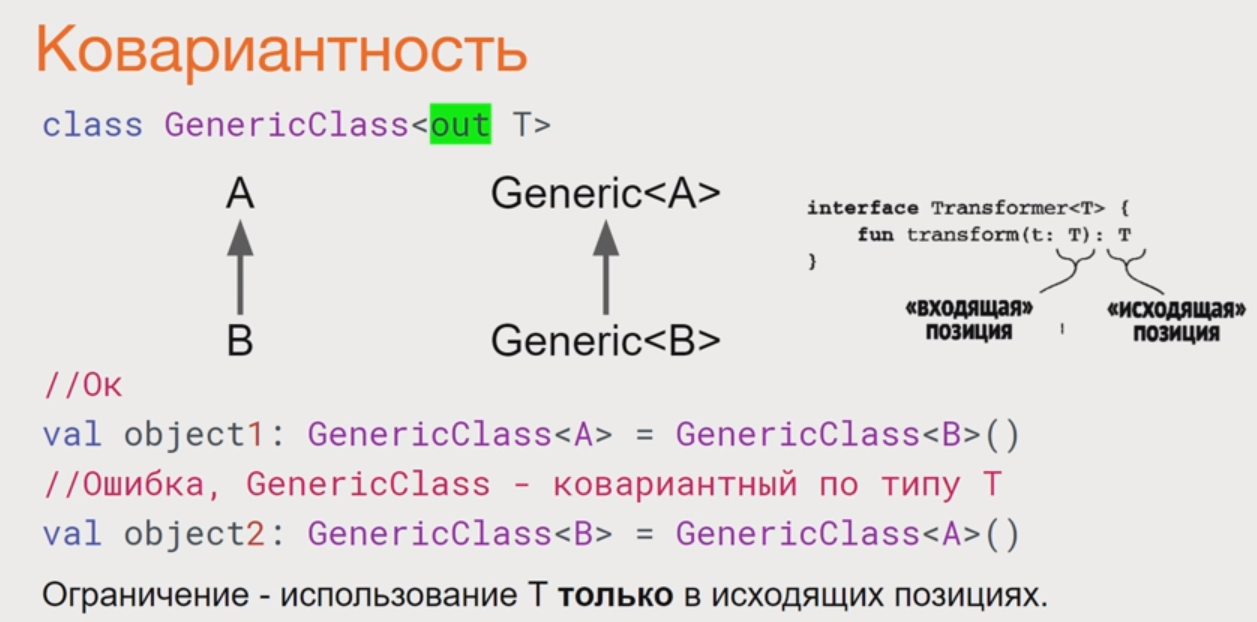


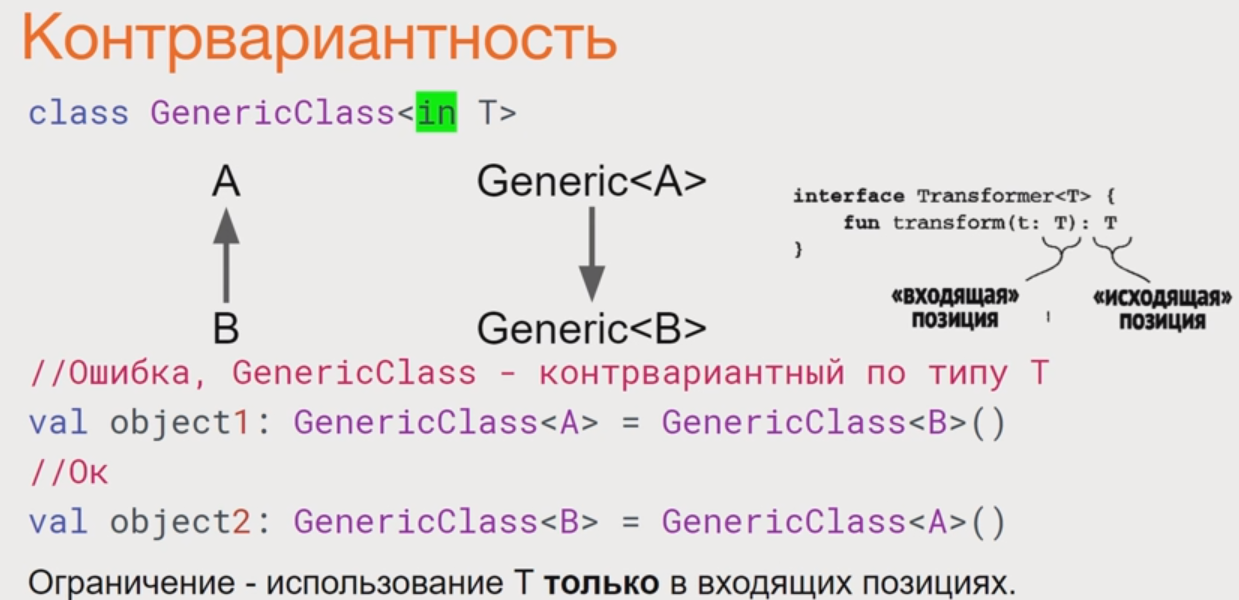












**package** com.application.balticactivity  
  
**fun** main() {  
 **val** genericObject1 = GenericClass<Double>(defaultValue = 2.3)  
  
 **val** genericObject2 = GenericClass<Double>(5.0)  
 **val** genericObject3 = GenericClass<Double>(0.75)  
 **val** genericObject4 = GenericClass<Double>(8.1)  
 *println*(*sumGeneric*(genericObject1, genericObject2))  
 *println*(*sumGeneric*(genericObject3, genericObject4))  
  
 *printGenericObject*<String?>(**"dsg"**)  
}  
  
**class** GenericClass<**out** T: Number>(defaultValue: T) {  
 */\*var item: T = defaultValue//это свойство в инвариантной позиции  
 private set\*/* **private var item**: T = defaultValue*//это свойство в ковариантной позиции* **private set  
  
 fun** getItem(): T {*//тип T в исходящей позиции* **return item** }  
  
 */\*fun updateItem(newItem: T) {  
 item = newItem  
 //item is String доступна проверка на тип  
 item?.toFloat()  
 item?.toInt()  
 }\*/*}  
  
*//функция принимает два Generic класса и вычисляет сумму их свойст item***fun** sumGeneric(a: GenericClass<Number>, b: GenericClass<Number>): Int {  
 **return** a.getItem().toInt() + b.getItem().toInt()  
}  
  
*//обобщенная функция***fun** <T> printGenericObject(item: T) {  
 *println*(item.*toString*())  
  
}

## Контрвариантные параметры типов

**package** com.application.balticactivity  
  
**fun** main() {  
 **val** genericObject1 = GenericClass<Double>(defaultValue = 2.3)  
  
 **val** genericObject2 = GenericClass<Double>(5.0)  
 **val** genericObject3 = GenericClass<Double>(0.75)  
 **val** genericObject4 = GenericClass<Double>(8.1)  
 *println*(*sumGeneric*(genericObject1, genericObject2))  
 *println*(*sumGeneric*(genericObject3, genericObject4))  
 *//printGenericObject<String?>("dsg")  
  
 updateContrvariant*(contrvariantClass<Any> (defaultValue = 234))  
}  
  
**class** GenericClass<**out** T: Number>(defaultValue: T) {  
 */\*var item: T = defaultValue//это свойство в инвариантной позиции  
 private set\*/* **private var item**: T = defaultValue*//это свойство в ковариантной позиции* **private set  
  
 fun** getItem(): T {*//тип T в исходящей позиции* **return item** }  
  
 */\*fun updateItem(newItem: T) {  
 item = newItem  
 //item is String доступна проверка на тип  
  
 item?.toFloat()  
 item?.toInt()  
  
 }\*/*}  
  
*//функция принимает два Generic класса и вычисляет сумму их свойст item***fun** sumGeneric(a: GenericClass<Number>, b: GenericClass<Number>): Int {  
 **return** a.getItem().toInt() + b.getItem().toInt()  
}  
  
*//обобщенная функция***fun** <T> printGenericObject(item: T) {  
 *println*(item.*toString*())  
  
}  
  
**class** contrvariantClass<**in** T>(defaultValue: T) {*//класс контрвариантный по T* **private var item**: T = defaultValue  
  
 **fun** setItem(newItem:T){  
 **item** = newItem  
 }  
}  
  
**fun** updateContrvariant(input: contrvariantClass<Number>) {  
 input.setItem(2.2)  
}

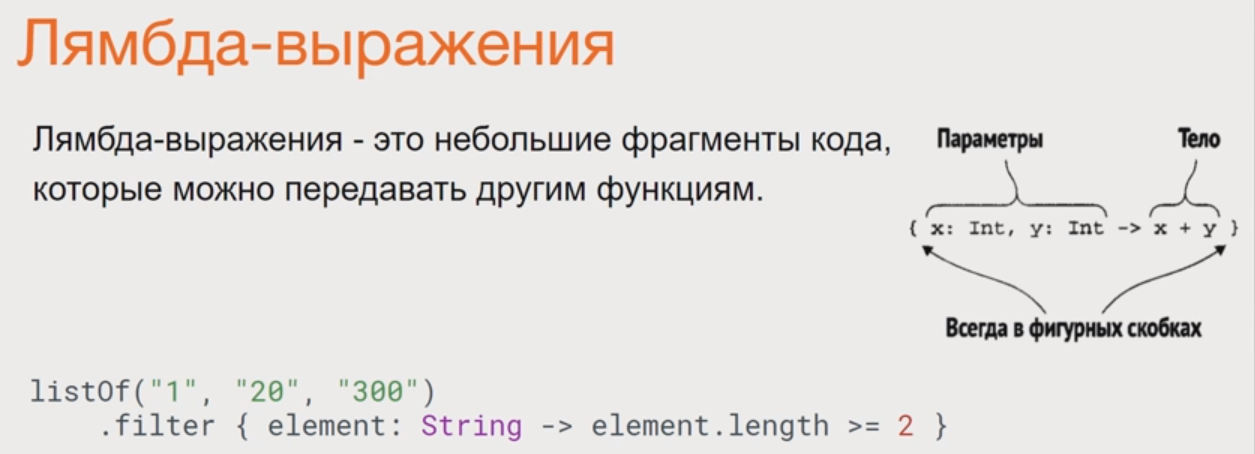
Функция установила значение Double для объекта у которого в свойстве хранится значение Any.

## Класс может быть ковариантным по одному параметру и контрвариантным по второму

**package** com.application.balticactivity  
  
**fun** main() {  
 **val** genericObject1 = GenericClass<Double>(defaultValue = 2.3)  
  
 **val** genericObject2 = GenericClass<Double>(5.0)  
 **val** genericObject3 = GenericClass<Double>(0.75)  
 **val** genericObject4 = GenericClass<Double>(8.1)  
 *println*(*sumGeneric*(genericObject1, genericObject2))  
 *println*(*sumGeneric*(genericObject3, genericObject4))  
 *//printGenericObject<String?>("dsg")  
  
 updateContrvariant*(contrvariantClass<Any> (defaultValue = 234))  
}  
  
**class** GenericClass<**out** T: Number>(defaultValue: T) {  
 */\*var item: T = defaultValue//это свойство в инвариантной позиции  
 private set\*/* **private var item**: T = defaultValue*//это свойство в ковариантной позиции* **private set  
  
 fun** getItem(): T {*//тип T в исходящей позиции* **return item** }  
  
 */\*fun updateItem(newItem: T) {  
 item = newItem  
 //item is String доступна проверка на тип  
  
 item?.toFloat()  
 item?.toInt()  
  
 }\*/*}  
  
*//функция принимает два Generic класса и вычисляет сумму их свойст item***fun** sumGeneric(a: GenericClass<Number>, b: GenericClass<Number>): Int {  
 **return** a.getItem().toInt() + b.getItem().toInt()  
}  
  
*//обобщенная функция***fun** <T> printGenericObject(item: T) {  
 *println*(item.*toString*())  
  
}  
  
**class** contrvariantClass<**in** T>(defaultValue: T) {*//класс контрвариантный по T* **private var item**: T = defaultValue  
  
 **fun** setItem(newItem:T){  
 **item** = newItem  
 }  
}  
  
**fun** updateContrvariant(input: contrvariantClass<Number>) {  
 input.setItem(2.2)  
}

Обобщения - глава 9 из книги Kotlin in action

# Лямбда выражения, функции высшего порядка, inline функции



**package** com.application.balticactivity  
  
**fun** main() {  
 **val** lambda = **{** *println*(**"Lambda"**)**}** lambda()  
  
 **val** lambdaWithParams = **{**x: Int **->** *println*(**"X from lambda: $**x**"**)**}** lambdaWithParams(10)  
  
 **var** x = 15  
 **var** y = 17  
 **val** sumLambda = **{**x:Int, y:Int **->** x + y**}** *println*(**"Sum $**x **and $**y **is ${**sumLambda(x, y)**}"**)  
}

Среди списка пользователей найти пользователя с максимальным возрастом

**package** com.application.balticactivity  
  
**data class** User(  
 **val name**:String,  
 **val lastName**:String,  
 **val age**: Int = 0  
) {  
 **var innerState**:String = **""**}

**package** com.application.balticactivity  
  
**fun** main() {  
 **val** users = *listOf*<User>(  
 User(**"Vasya"**, **"Sergeev"**, 35),  
 User(**"Ivan"**, **"Sergeev"**, 44),  
 User(**"Pavel"**, **"Kotin"**, 18),  
 User(**"Igar"**, **"nikitin"**, 22)  
 )  
  
 **var** oldestUser:User? = **null  
 for** (user **in** users){  
 **val** currentAge = user.**age  
 val** maxAge = oldestUser?.**age**?: Int.**MIN\_VALUE  
  
 if** (currentAge > maxAge){  
 oldestUser = user  
 }  
 }  
 *println*(oldestUser)  
}

## Лямбда функция для коллекции

Найти самого старого сотрудника

**val** oldestUserMaxBy = users.*maxBy***{**user: User **->** user.**age}**

Если один параметр, то его можно не именовать

**val** oldestUserMaxBy = users.*maxBy***{**user: User **->** user.**age.let{it + 1}}**

Работа с внешним окружением из лямбды

**val** user1 = users[0]  
**val** oldestUserMaxBy = users.*maxBy***{**user: User **->** user1.**age}**

# Разметка

Форматирование XML файла. Атрибуты должны быть с отступом.



**android:layout\_width="200px"   
android:layout\_height="200px"**

**android:layout\_width="wrap\_content" //принять размер как размер содержимого (текста)**

**android:layout\_width="match\_parent" // принять размер родителя**

настройка ширины и высоты элемента.

View – это базовый класс для элементов

ViewGroup – базовый класс для группы элементов

**android:background="@color/colorAccent"**

Отступы и поля:

Margin (padding) : start, top, end, bottom

**android:layout\_margin="20dp"  
android:padding="10dp"**





Размер вьюшек указывается в dp, а размер текста в sp (он должен масштабироваться).

## Позиционирование внутри родителя

Как элемент будет расположен внутри объекта

**android:layout\_gravity="center"**

Как элементы будут расположены внутри текущего контейнера

**android:gravity="center"**

Можно использовать сразу несколько определений



## Видимость элементов

**android:visibility="visible"**

**android:visibility=" invisible" невидим но занимает пространство**

**android:visibility="gone" невидим и не занимает пространства**

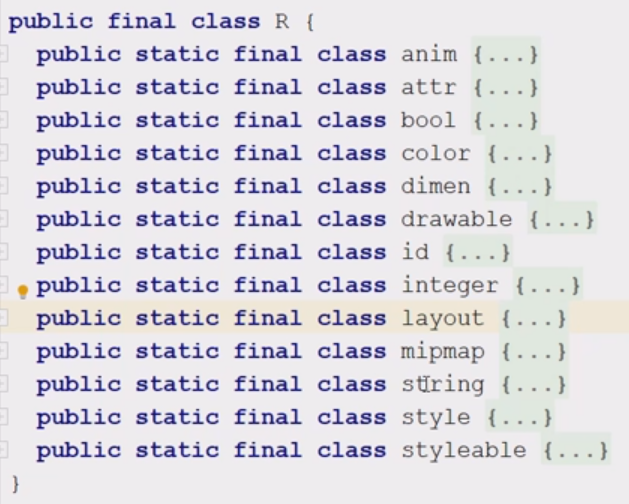
## код

*<?***xml version="1.0" encoding="utf-8"***?>*<**TextView xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="200dp"  
 android:layout\_height="200dp"  
 android:background="@color/colorAccent"  
 android:text="@string/text\_view"  
 android:textSize="20sp"  
  
 android:layout\_margin="20dp"  
 android:padding="10dp"  
  
 android:layout\_gravity="center"  
 android:gravity="center\_horizontal|bottom"  
  
 android:visibility="invisible"** />

# Виды UI элементов

**override fun** onCreate(savedInstanceState: Bundle?) {*//вызывается при открытии приложения* **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)*//передача ресурса разметки*

Класс R



*//поиск элемента типа TextView который имеет айди textView*findViewById<TextView>(R.id.*textView*)

Но вместо того, чтобы искать вью по его айди, если подключены в build.gradle

apply **plugin**: **'kotlin-android-extensions'**

можно обращаться сразу по айди к элементам

clearButton.setOnClickListener()

Установить текст из ресурса

textView.setText(R.string.*app\_name*)

Вернуть то, что установлено в ресурсе строки элемента *text\_view*

**val** text = *resources*.getString(R.string.*text\_view*)

## TextView

**android:text="@string/app\_name"**

**android:textSize="20sp"**

//это чтобы показать на экране информацию о собранном приложении

val textView = findViewById<TextView>(R.id.textView)

textView.text = """

BuildType=${BuildConfig.BUILD\_TYPE}

flavor=${BuildConfig.FLAVOR}

VersionCode = ${BuildConfig.VERSION\_CODE}

VersionName = ${BuildConfig.VERSION\_NAME}

ApplicationId = ${BuildConfig.APPLICATION\_ID}

"""

//поиск элемента типа TextView который имеет айди textView

val textView = findViewById<TextView>(R.id.textView)

textView.text = "new Text"

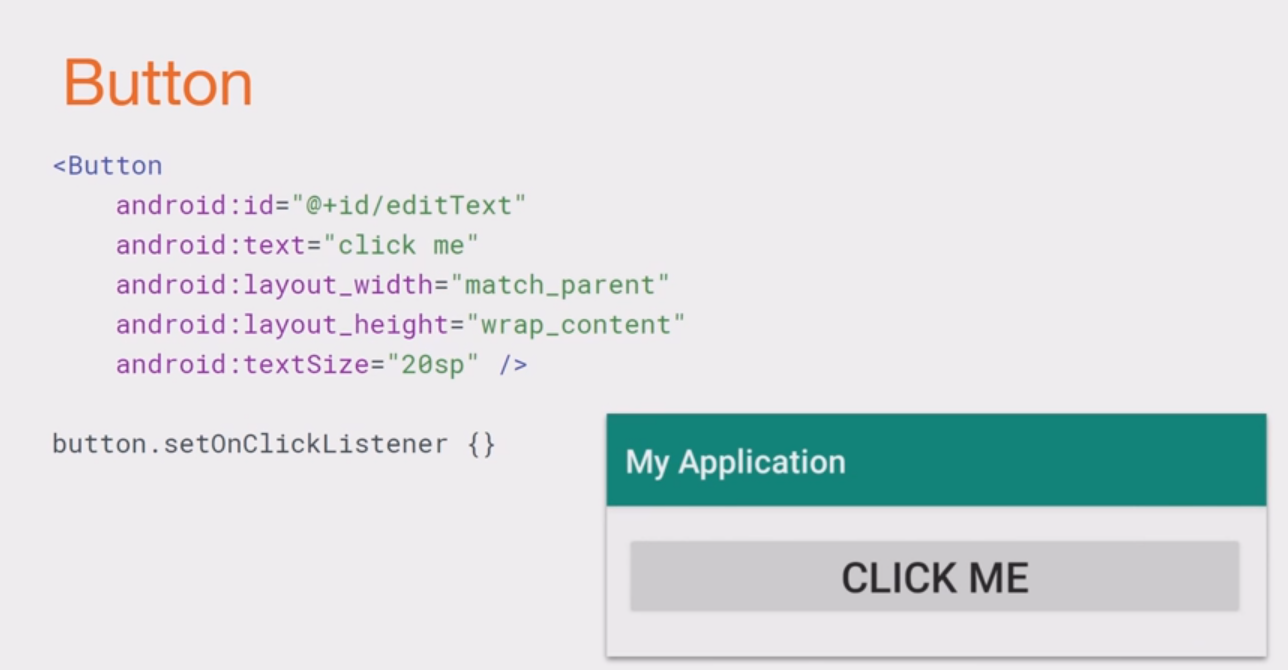
textView.setText(R.string.app\_name)

val text = resources.getString(R.string.text\_view)

## EditText поле ввода



## Button кнопка



Повесить листенер на кнопку:

clearButton.setOnClickListener(**object** : View.OnClickListener{  
 **override fun** onClick(p0: View?) {  
 nameInput.setText(**""**)  
 }  
})

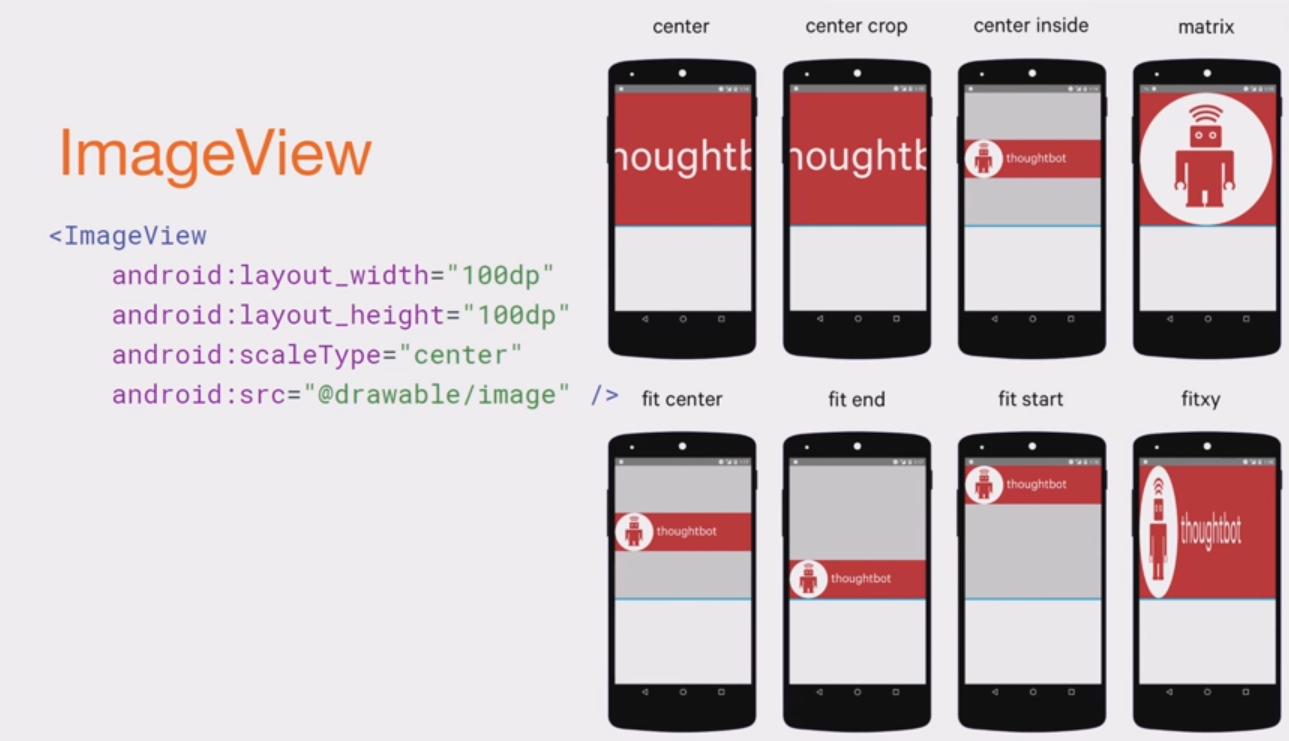
и это можно перевести в лямбду вот так:

clearButton.setOnClickListener **{** nameInput.setText(**""**) **}**

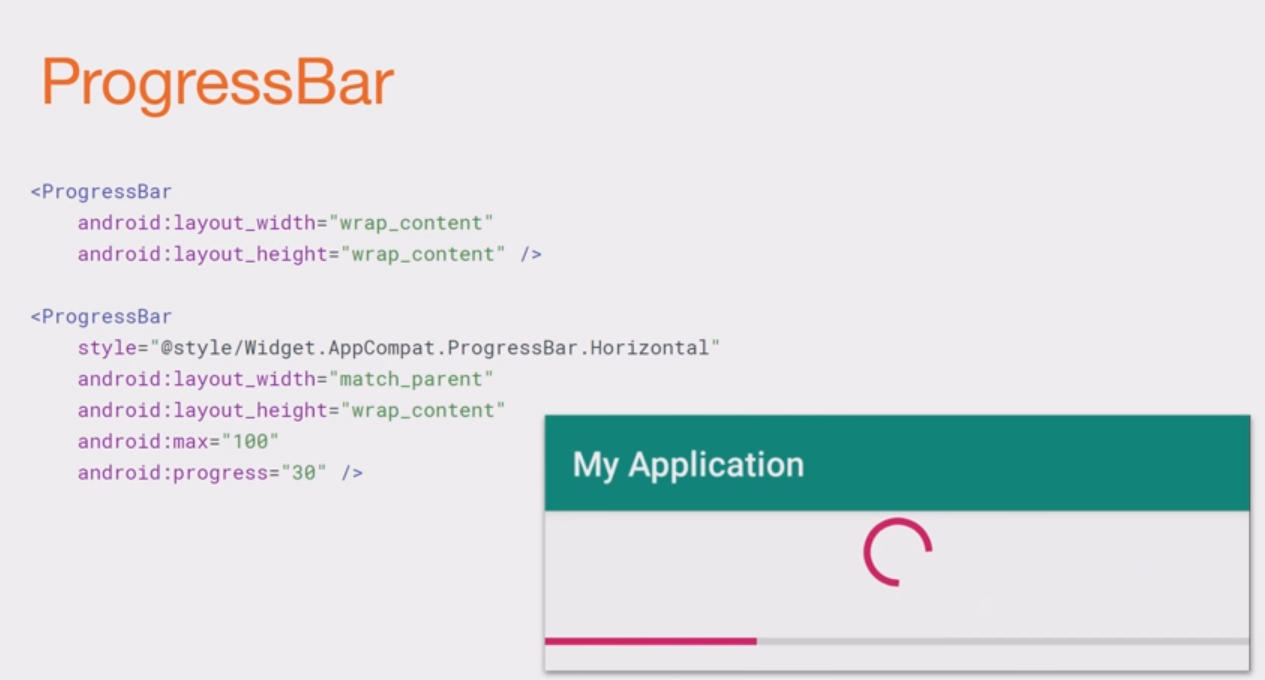
при нажатии кнопки показывать облачко с подсказкой, что произошло какое-то действие

clearButton.setOnClickListener **{** nameInput.setText(**""**)  
 Toast.makeText(**this**, R.string.*cleared\_text*, Toast.*LENGTH\_SHORT*)  
**}**

## ImageView Изображения

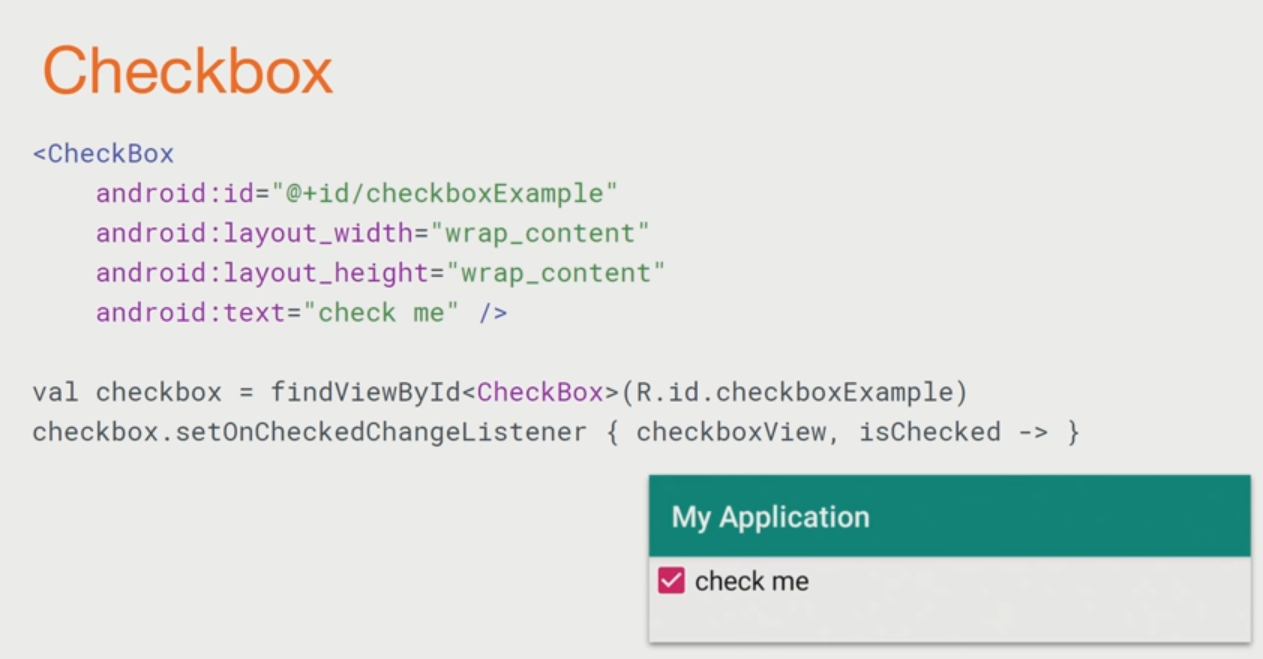


## ProgressBar

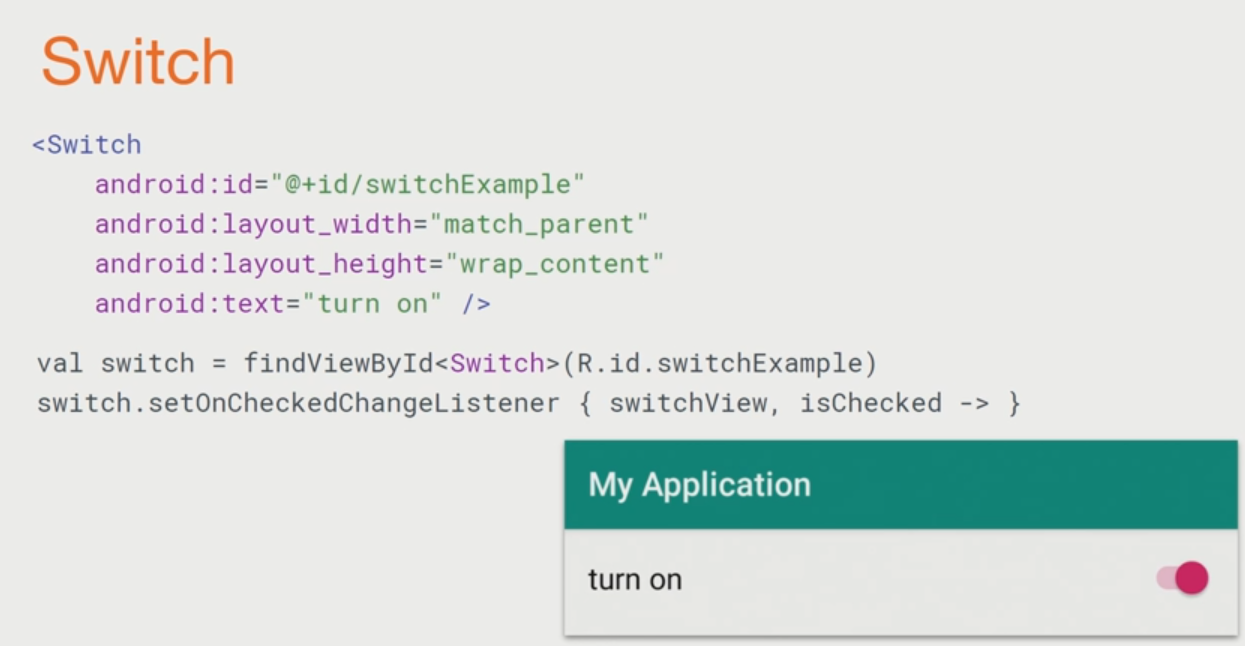


//после нажатия на кнопку будет появляться прогрес бар, который будет крутиться 2 секунды, а потом будет показываться оповещение, что данные загружены

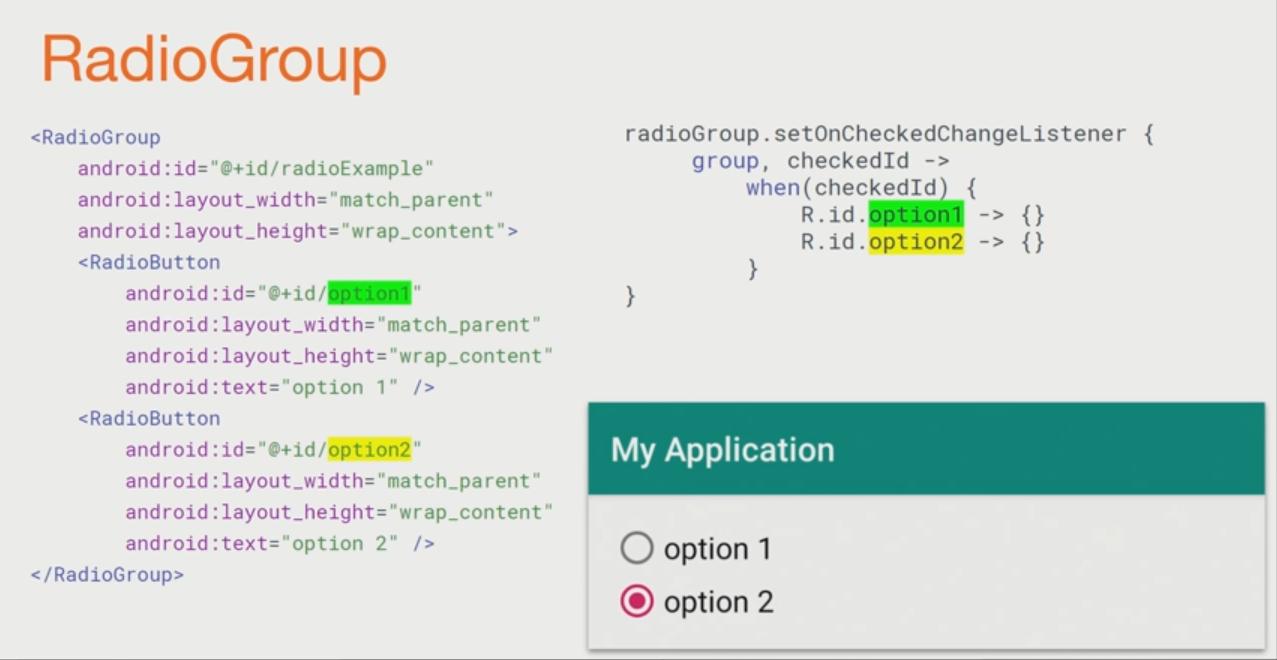
## Checkbox флажок



## Swith как флажок, только выполняется моментально

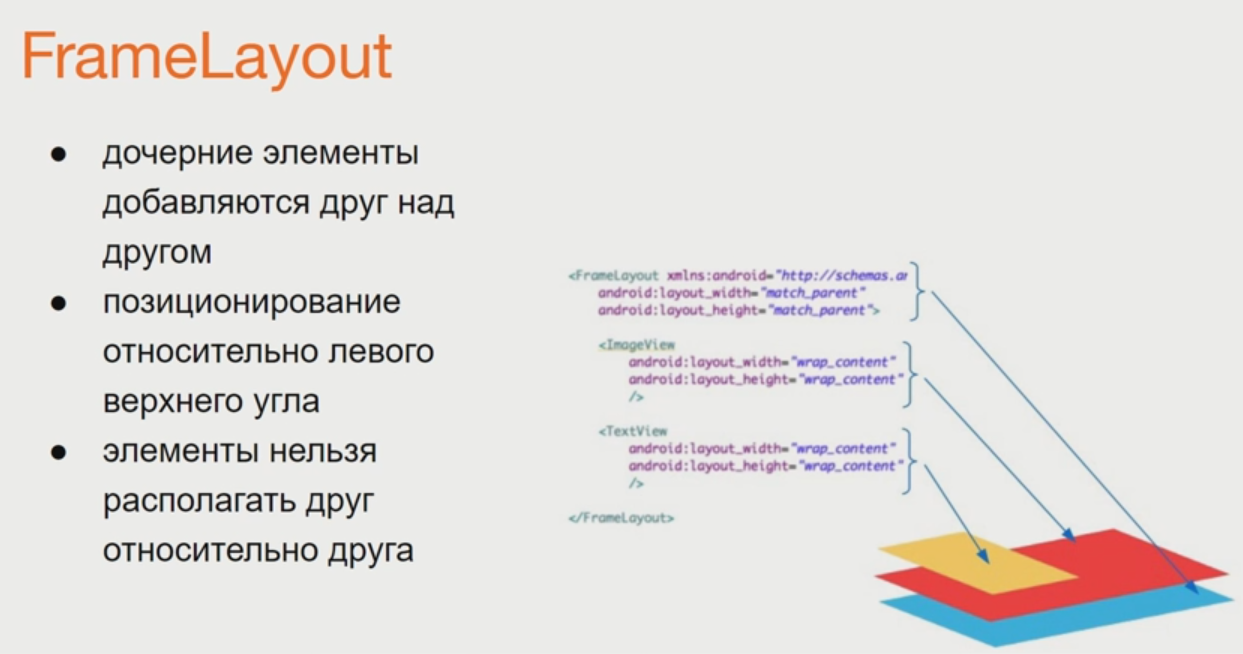


## RadioGroup радиокнопка

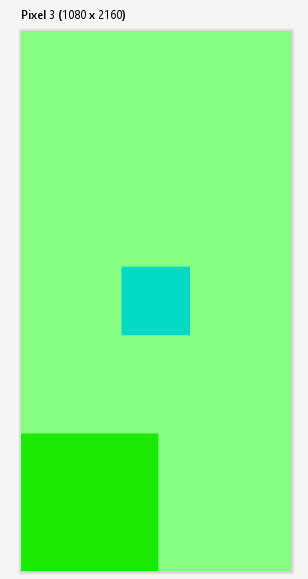


# ViewGroups

## FrameLayout

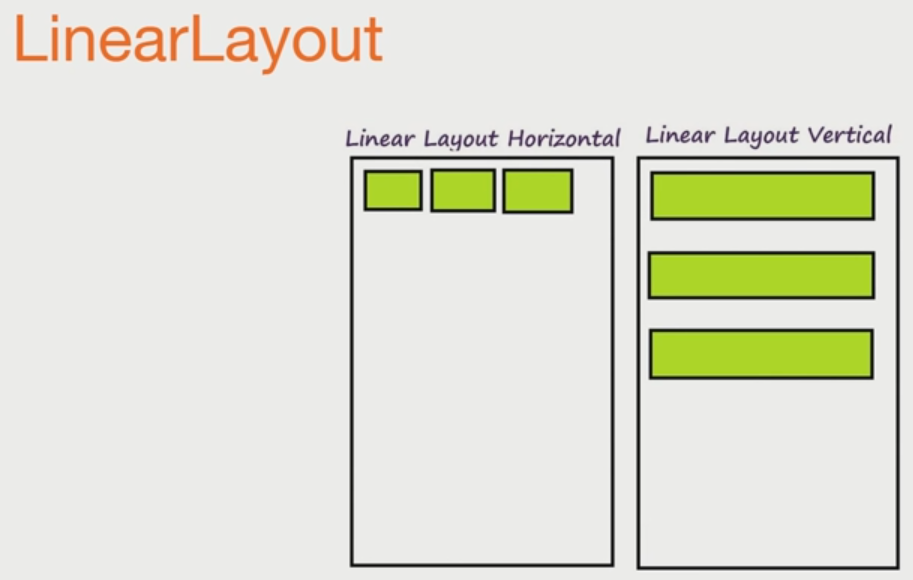


*<?***xml version="1.0" encoding="utf-8"***?>*<**FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**View  
 android:layout\_width="200dp"  
 android:layout\_height="200dp"  
 android:layout\_gravity="bottom"  
 android:background="@color/colorPrimary"**/>  
  
 <**View  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:layout\_gravity="center"  
 android:background="@color/colorAccent"**/>  
   
</**FrameLayout**>

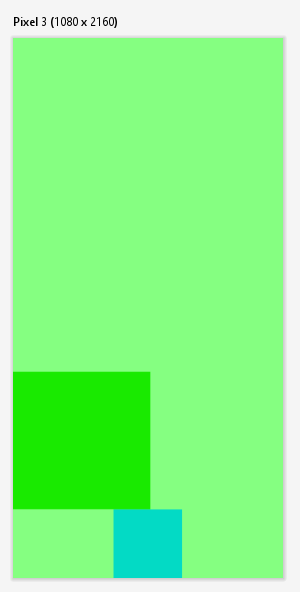


## LinearLayout

**android:orientation="vertical"**



*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:gravity="center|bottom"** >  
  
 <**View  
 android:layout\_width="200dp"  
 android:layout\_height="200dp"  
 android:layout\_gravity="bottom"  
 android:background="@color/colorPrimary"**/>  
  
 <**View  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:layout\_gravity="center"  
 android:background="@color/colorAccent"**/>  
  
</**LinearLayout**>



Указание размера по весам

Указать сколько всего частей весов есть в разметке контейнера:

**android:weightSum="10"**

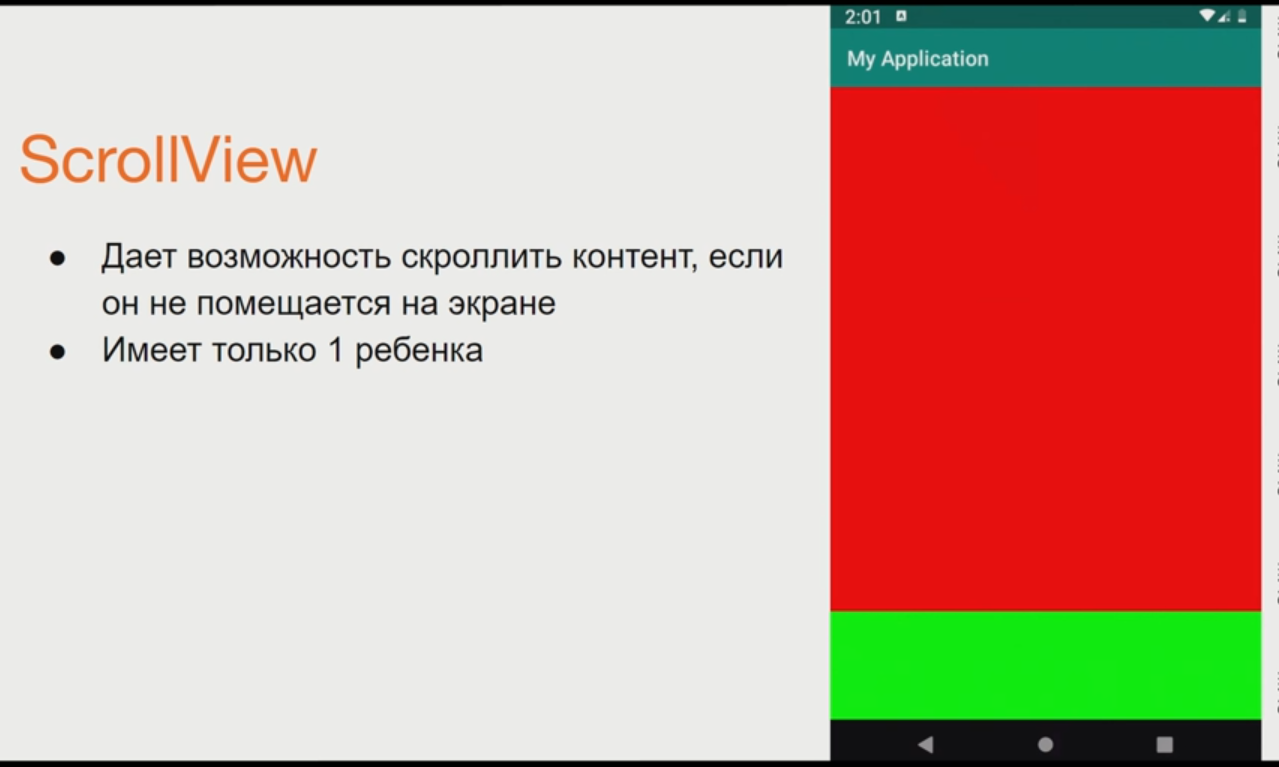
Указать какой вес занимает конкретная вью:

**android:layout\_weight="2"**

Пример кода:

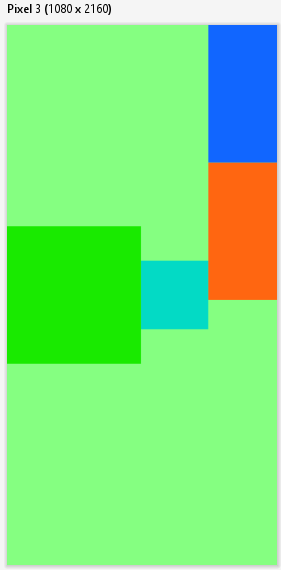
*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:gravity="center|bottom"  
 android:weightSum="10"**>  
  
 <**View  
 android:layout\_weight="2"  
 android:layout\_width="0dp"  
 android:layout\_height="200dp"  
 android:layout\_gravity="center"  
 android:background="@color/colorPrimary"**/>  
  
 <**View  
 android:layout\_weight="1"  
 android:layout\_width="0dp"  
 android:layout\_height="100dp"  
 android:layout\_gravity="center"  
 android:background="@color/colorAccent"**/>  
  
</**LinearLayout**>

## ScrollView контейнер



## Вложенные контейнеры

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**View  
 android:layout\_weight="2"  
 android:layout\_width="0dp"  
 android:layout\_height="200dp"  
 android:layout\_gravity="center"  
 android:background="@color/colorPrimary"**/>  
  
 <**View  
 android:layout\_weight="1"  
 android:layout\_width="0dp"  
 android:layout\_height="100dp"  
 android:layout\_gravity="center"  
 android:background="@color/colorAccent"**/>  
  
 <**LinearLayout  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"** >  
  
 <**View  
 android:layout\_width="wrap\_content"  
 android:layout\_height="200dp"  
 android:layout\_gravity="center"  
 android:background="#1166ff"**/>  
  
 <**View  
 android:layout\_width="wrap\_content"  
 android:layout\_height="200dp"  
 android:layout\_gravity="center"  
 android:background="#ff6611"**/>  
  
 </**LinearLayout**>  
</**LinearLayout**>



## Динамическое добавление элементов UI

1. Создаем новый активити (DinamicActivity), где переопределяем метод onCreate

**package** com.application.balticactivity  
  
**import** android.os.Bundle  
**import** androidx.appcompat.app.AppCompatActivity  
  
**class** DynamicActivity:AppCompatActivity() {  
 **override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_dynamic*)  
 }  
}

1. Регистрируем в манифесте новый лаяут. Скопировали описание MainActivity, убрали из него intent-filter, вставили копию и указали лаяут DynamicActivity:

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.application.balticactivity"**>  
  
 <**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=".MainActivity"**></**activity**>  
  
 <**activity android:name=".DynamicActivity"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
 </**application**>  
  
</**manifest**>

1. Добавляем в разметку необходимые вью

Пользователь вводит текст. После нажатия кнопки введенный текст помещается в контейнер в виде вью.

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:padding="16dp"  
 android:orientation="vertical"** >  
  
 <**EditText  
 android:id="@+id/textInput"  
 android:hint="@string/enter\_text"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"** />  
  
 <**Button  
 android:layout\_marginTop="8dp"  
 android:id="@+id/addTextButton"  
 android:text="@string/add\_text\_button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"**/>  
  
</**LinearLayout**>

1. Пишем код в активити, который будет ловить нажатия на кнопку и создавать новые вью.

**package** com.application.balticactivity  
  
**import** android.os.Bundle  
**import** android.widget.TextView  
**import** androidx.appcompat.app.AppCompatActivity  
**import** kotlinx.android.synthetic.main.activity\_dynamic.\*  
  
**class** DynamicActivity:AppCompatActivity() {  
 **override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_dynamic*)  
  
 addTextButton.setOnClickListener()**{  
 val** textToAdd = textInput.*text*.toString()  
 TextView(**this**).*apply* **{** *text* = textToAdd   
 **}  
 }** }  
}

1. Для того, чтобы задавать новым вью параметры размеры, создадим класс лаяут параметров

addTextButton.setOnClickListener()**{  
 val** textToAdd = textInput.*text*.toString()  
 TextView(**this**).*apply* **{** *text* = textToAdd  
 LinearLayout.LayoutParams(  
 LinearLayout.LayoutParams.*WRAP\_CONTENT*,  
 LinearLayout.LayoutParams.*WRAP\_CONTENT* ).*apply* **{  
 gravity** = **when**(Random.nextInt(3)) {  
 0 -> Gravity.*CENTER* 1 -> Gravity.*END* **else** -> Gravity.*START* }   
 **}  
 }  
}**

1. Использование лаяут параметров и пример кода

**package** com.application.balticactivity  
  
**import** android.os.Bundle  
**import** android.view.Gravity  
**import** android.widget.LinearLayout  
**import** android.widget.TextView  
**import** androidx.appcompat.app.AppCompatActivity  
**import** kotlinx.android.synthetic.main.activity\_dynamic.\*  
**import** kotlin.random.Random  
  
**class** DynamicActivity:AppCompatActivity() {  
 **override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_dynamic*)  
  
 addTextButton.setOnClickListener()**{  
 val** textToAdd = textInput.*text*.toString()  
 **val** textViewToAdd = TextView(**this**).*apply* **{** *text* = textToAdd  
 *layoutParams* = LinearLayout.LayoutParams(  
 LinearLayout.LayoutParams.*WRAP\_CONTENT*,  
 LinearLayout.LayoutParams.*WRAP\_CONTENT* ).*apply* **{  
 gravity** = **when**(Random.nextInt(3)) {  
 0 -> Gravity.*CENTER* 1 -> Gravity.*END* **else** -> Gravity.*START* }  
 **}  
 }** dynamicContainer.addView(textViewToAdd)  
 **}** }  
}

1. Чтобы иметь возможность прокрутки экрана, добавим LinearLayout внутрь ScrollView.

*<?***xml version="1.0" encoding="utf-8"***?>*<**ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**LinearLayout  
 android:id="@+id/dynamicContainer"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp"**>  
  
 <**EditText  
 android:id="@+id/textInput"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/enter\_text"** />  
  
 <**Button  
 android:id="@+id/addTextButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:text="@string/add\_text\_button"** />  
  
 </**LinearLayout**>  
</**ScrollView**>

## Добавить из кода элемент, который подгружается из ресурсов

Добавим в лаяут текст вью и кнопку, по нажатию которой элемент будет удаляться из экрана.

Здесь добавлено пространство имен tools, которое содержит опции, которые исполняются только в режиме превью экрана и не будут видны после запуска настоящего приложения.

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"**>  
  
 <**TextView  
 android:id="@+id/textView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:textSize="20sp"  
 android:textColor="@android:color/black"  
 tools:text="text 1 2 3"  
 tools:background="#ff0000"** />  
  
 <**Button  
 android:id="@+id/deleteButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="X"** />  
  
</**LinearLayout**>

Получить вью из лаяут ресурса. Инфлейтить разметку. Превращает разметку в объект вью.

*//создает вью из ресурса разметки***val** view = *layoutInflater*.inflate(R.layout.*item\_text*, dynamicContainer, **false**)

Если передать false, то будет получен LinearLayout, если передать true, то будет возвращен контейнер, в который был добавлен только что созданный лаяут.

Будем использовать false, т.к. во вьюшку еще надо добавить текст.

**package** com.application.balticactivity  
  
**import** android.os.Bundle  
**import** androidx.appcompat.app.AppCompatActivity  
**import** kotlinx.android.synthetic.main.activity\_dynamic.\*  
**import** kotlinx.android.synthetic.main.item\_text.view.\*  
  
**class** DynamicActivity:AppCompatActivity() {  
 **override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_dynamic*)  
  
 addTextButton.setOnClickListener()**{  
 val** textToAdd = textInput.*text*.toString()  
  
 *//создает вью из ресурса разметки* **val** view = *layoutInflater*.inflate(R.layout.*item\_text*, dynamicContainer, **false**)  
 view.*apply* **{** textView.*text* = textToAdd  
  
 deleteButton.setOnClickListener()**{** dynamicContainer.removeView(**this**)  
 **}  
 }** dynamicContainer.addView(view)  
  
 **}** }  
}

Вот результат:



Про метод setContentView

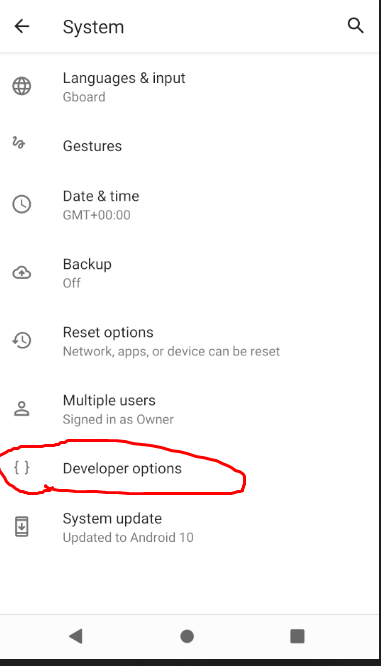
setContentView(R.layout.*activity\_dynamic*)

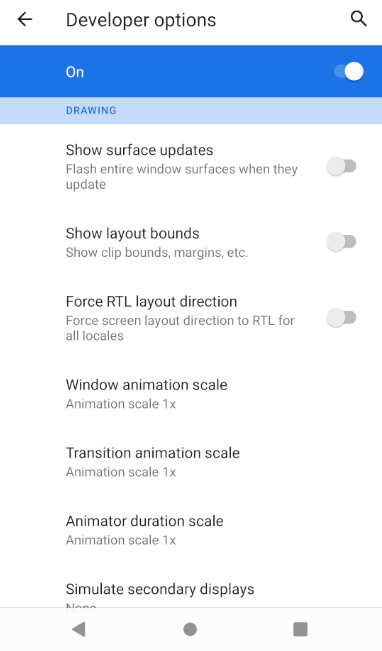
в него можно передавать не только ресурс лаяута, но и вьюшку.

При таком решении мы отображаем вообще все элементы на экране постоянно, и это может отразиться на производительности, если элементов много.

Для повышения производительности используется RecyclerView и ListView.

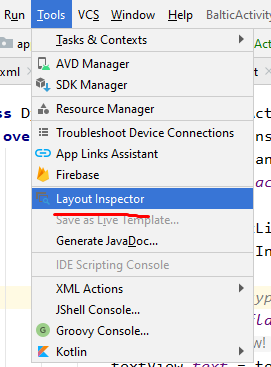
## Настройки разработчика для реализации верстки





Show layout bounds – показывать разметку, границы вьюшек.

Также в андроид студии tools-> layout ins:



<https://developer.android.com/studio/debug/layout-inspector>

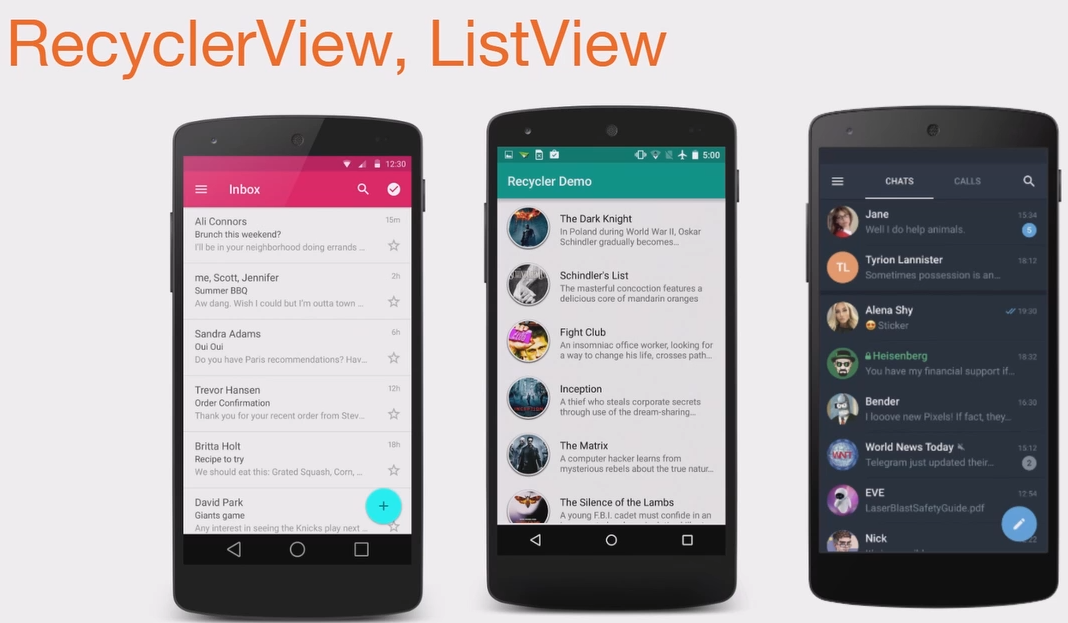
<https://developer.android.com/topic/performance/rendering/inspect-gpu-rendering>

<https://developer.android.com/topic/performance/rendering/overdraw>

<https://play.google.com/store/apps/details?id=com.dziemia.w.window&hl=ru>  
<https://play.google.com/store/apps/details?id=com.appsisle.developerassistant&hl=ru>

# Сложная верстка

## RecyclerView и ListView



## ConstraintLayout

