Содержание

Введение

Существует множество различных языков программирования, таких как: Java, C++, C и так далее.

Однако большой интерес представляют относительно молодые языки, так как их синтаксис подстроен под современные реалии программного обеспечения и новых технологий в сфере мобильных приложений. Одним из таких языков программирования является Kotlin.

Kotlin — это статически типизированный язык программирования (тип переменной известен во время компиляции, то есть еще до запуска программы). В отличие от Java, где программы строятся на классах, основным строительным блоком программы на Kotlin является функция. Однако Kotlin также поддерживает объектно-ориентированный подход к программированию.

Целью данной работы является изучение теоретических основ языка программирования Kotlin и их практическая реализация. Для достижения поставленной цели требуется решить следующие задачи:

— изучить основные компоненты языка и его синтаксис;

— разработать приложение "Галерея"на основе языка Kotlin в среде Android Studio.

1 Язык программирования Kotlin

1.1 Переменные и типы данных

На начальном этапе обучения были рассмотрены основы языка Kotlin, а именно переменные и типы данных. Для создания переменной используется одно из двух ключевых слов: val (значение переменной изменять нельзя) или var (значение переменной можно изменять), после которых ставится название переменной и ее тип. Конструкция такого построения выглядит следующим образом:

```
fun main() {
        var a: Int
}
```

В случае постоянного значения - константы применяется обозначение const. К таким переменным выдвигаются некоторые требования, как нахождение на верхнем уровне, то есть вне класса, а также соответствие одному из примитивных типов данных.

Данные подразделяются на четыре типа: числовые (Byte, Short, Int, Long, Float, Double), логические, символьные, строковые, а также на геттеры и сеттеры.

Boolean - это логический тип данных, принимающий два значения: true или false:

```
fun main() {
        val a: Boolean = true
}
```

Char - это символьный тип данных, принимающий только один символ, который нужно заключать в одинарные кавычки:

```
fun main() {
        val a: Char = 'b'
}
```

String - это строковый тип, принимающий значения «слов», которые нужно заключать в двойные кавычки:

```
fun main() {
        val a: String = "hello world"
}
```

Геттеры управляют получением значения свойства и определяются с помощью ключевого слова get. А сеттеры определяют логику установки значения переменной и определяются ключевым словом set.

1.2 Функции и операторы проверок на null

Основой языка программирования Kotlin являются функции. Для того, чтобы объявить функцию используется ключевое слово fun:

```
fun doSomething(firstParameter: String): Int {
}
```

В круглых скобках указываются параметры функции, а после двоеточия тип возвращаемого значения. В случае, когда функция ничего не возвращает, тогда тип возвращаемого значения указывается, как Unit.

Также Kotlin отличается от всех остальных языков тем, что защищен от такой ошибки, как NullPointerException. Поэтому в случае ее возникновения синтаксис языка предусматривает решение данной проблемы при помощи нескольких операторов.

Если переменная может быть null, то к типу переменной добавляется знак "?":

```
var name: String? = "World"
name = null
```

Иначе можно воспользоваться оператором безопасного вызова "?.":

```
print("Строка длиной ${name?.length}")
```

Оператор !! используется в случае, когда ошибку нужно вызвать самому пользователю путем ее явного указания, то есть:

```
        print("Строка длиной ${name!!.length}")
```

Альтернативой всех выше перечисленных операторов является Элвис-оператор, который выступает в роли сокращения от выражения "если не null, иначе":

```
        name?.length ?: -1
```

1.3 Основные классы и их свойства

Для объявления класса используется ключевое слово class:

```
        class Person(name: String, age: Int) {
                ...
        }
```

В круглых скобках расположен основной конструктор класса. Если у конструктора есть аннотации или модификаторы видимости, ключевое слово constructor используется обязательно, а модификаторы идут перед ним:

```
        class Person @Inject constructor(val name: String)
            {
                ...
        }
```

Свойства класса могут быть включены в объявление класса при инициализации основного конструктора, то есть:

```
        class Person (val firstname: String, val lastname:
            String, var isHasCar: Boolean = false)
```

Аналогом алгоритма инициализации свойств выступает алгоритм реализации блока init. При создании экземпляра класса блоки инициализации выполняются в том порядке, в котором они идут в теле класса, чередуясь с инициализацией свойств. Параметры основного конструктора могут быть использованы в инициализирующем блоке:

```
        class InitOrderDemo(name: String) {
                val firstProperty = "Первое свойство:
                    $name".also(::println)
```

```
        init {
            println("Первый блок инициализации
                : ${name}" )
        }
        val secondProperty = "Второе свойство: ${
            name. length}".also(::println)
        init {
            println("Второй блок инициализации
                : ${name.length}" )
        }
    }
```

В классах также могут быть объявлены дополнительные конструкторы (secondary constructors), перед которыми используется ключевое слово constructor. Если у класса есть основной конструктор, каждый дополнительный конструктор должен прямо или косвенно ссылаться на основной при помощи ключевого слова this, например:

```
    class Person(val name: String) {

        constructor(name: String, age: Person):
            this(name) {
                // Тело дополнительного
                конструктора
        }
    }
```

Отличительной особенностью классов во многих программных языках является один из принципов объектно-ориентированного программирования, включенных в свойства класса, который носит название - наследование. Kotlin не является исключением в случае рассмотрения наследования как принципа ООП. В синтаксисе языка программирования Kotlin для всех классов родительским суперклассом является класс Any:

```
    class Example
```

В данном случае представленный класс неявно наследуется от суперкласса Any.

```
open class Base // Класс открыт для наследования
```

По умолчанию все классы в Kotlin имеют статус final. При объявлении суперкласса имя находится за знаком двоеточия в оглавлении класса, то есть:

```
open class Base(p: Int)
class Derived(p: Int) : Base(p)
```

```
class MyView : View {
        constructor(ctx: Context) : super(ctx)
        constructor(ctx: Context, attrs:
           Attributes) : super(ctx, attrs)
}
```

Еще одной важной особенностью теории языка Kotlin является переопределение методов класса. Для данного процесса переопределения используется модификатор override. Член класса помеченный override является open, поэтому при необходимости нужно использовать final, чтобы запретить переопределение, следующим образом:

```
open class Rectangle() : Shape() {
        final override fun draw() { /*...*/ }
}
```

Переопределение свойств выражается в программном коде следующего вида:

```
open class Shape {
        open val vertexCount: Int = 0
}
class Rectangle : Shape() {
        override val vertexCount = 4
}
```

Усложненной формой класса являются абстрактные классы с определенным рядом индивидуальных свойств. Абстрактный класс определяется

ключевым словом abstract. Класс или объект могут наследоваться только от одного абстрактного класса, например:

```
abstract class Polygon {
        abstract fun draw()
}
class Rectangle : Polygon() {
        override fun draw() {
                // рисование прямоугольника
        }
}
```

Для хранения данных в Kotlin используются data-классы. Они определяются с помощью модификатора data. У таких классов появляется метод copy(). Реализация data-класса выглядит следующим образом:

```
data class User(val name: String, val age: Int)
```

В свою очередь основные требования к реализации достаточно просты, а именно:

— основной конструктор должен иметь как минимум один параметр;

— все параметры основного конструктора должны быть отмечены, как val или var;

— data - классы не могут быть абстрактными, open, sealed или inner.

Для объявления перечислений используется ключевое слово enum. Каждая enum-константа является объектом и экземпляром enum-класса, которые могут быть инициализированы. Поэтому такие константы могут объявлять свои собственные анонимные классы, то есть, например следующие:

```
enum class ProtocoState {
        WAITING {
                override fun signal() = TALKING
        },
        TALKING {
```

```
                override fun signal() = WAITING
            };
            abstract fun signal(): ProtocolState
        }
```

Для объявления изолированного класса используется модификатор sealed и такие классы называют sealed-классами. Они схожи с анонимными enum-классами, поэтому наследник изолированного класса может иметь несколько экземпляров. Данный класс отличается от остальных тем, что является абстрактным, он не может быть создан напрямую и может иметь абстрактные компоненты. Пример такого класса:

```
        fun log(e: Error) = when(e) {
            is FileReadError -> { println("Error while
                reading file ${e.file}") }
            is DatabaseError -> { println("Error while
                reading from database ${e.source}") }
            RuntimeError -> {println("Runtime error")
               }
            // оператор 'else' не требуется, потому
               что мы покрыли все возможные случаи
        }
```

Рассмотрение представленных классов достаточно для понимания структуры языка программирования Kotlin.

2 Среда разработки Android Studio

Android Studio — интегрированная среда разработки производства Google, с помощью которой разработчикам становятся доступны инструменты для создания приложений на платформе Android OS. Android Studio можно установить на Windows, Mac и Linux. Android Studio создавалась на базе IntelliJ IDEA.

В IDE присутствуют макеты для создания UI, с чего обычно начинается работа над приложением. В Studio содержатся инструменты для разработки

решений для смартфонов и планшетов, а также новые технологические решения для Android TV, Android Wear, Android Auto, Glass и дополнительные контекстные модули. Главными преимуществами данной платформы являются следующие критерии:

— среда разработки поддерживает работу с несколькими языками программирования, к которым относятся самые популярные – C/C++, Java, Kotlin;

— тестирование корректности работы новых игр, утилит, их производительности на той или иной системе, происходит непосредственно в эмуляторе;

— достаточно большая библиотека с готовыми шаблонами и компонентами для разработки ПО;

— разработка приложения для Android N – самой последней версии операционной системы;

— предварительная проверка уже созданного приложения на предмет ошибок в нем;

— большой набор средств инструментов для тестирования каждого элемента приложения, игры;

2.1 Компоненты интерфейса

В своем узком понимании компоненты интерфейса в среде разработки Android Studio — это то, на чем строятся приложения. Основные составляющие такого интерфейса подразделяются на четыре категории:

— Activity;

— Service;

— Broadcast Receiver;

— Content Provider.

Возможности первого компонента Activity представлены в виде открытия новых экранов, запуске фоновой работы, проигрывании медиа-контента и передаче данных в другие приложения.

Возможности второго компонента Service предусматривают создание фоновых задач, например, музыки в плеере, а также загрузку или обработку данных и трансляцию геолокации. Для доступа компонентов к информации внутри в помощь выступает составляющая компонента интерфейса - Bound Service, которая также может являться сервисом. У компонента Service есть два типа: Background - задача не видна и Foreground - задача видна.

Главным компонентом по обработке и приему сообщений системы является компонент интерфейса - Broadcast Receiver. Система посылает события (перезагрузка смартфона, например), а компонент, в свою очередь, принимает их и выполняет задачу.

Content Provider — это компонент для доступа к данным. Он предоставляет другому данные одного приложения операциями CRUD, а сами запросы обрабатывает уже ContentResolver.

2.2 Макеты интерфейса

В Android SDK предусмотрено множество виджетов, которые можно настраивать для получения нужного оформления и поведения. Каждый виджет является экземпляром класса View или одного из его подклассов (например, TextView или Button).

Объект ViewGroup — это такой подвид View, который содержит и упорядочивает другие виджеты. При этом сам ViewGroup контент не отображает, он распоряжается тем, где отображается содержимое других View. Объекты ViewGroup еще часто называют макетами. Все классы компонентов UI можно разделить на две группы: видимые элементы (потомки классы View: TextView, ImageView, Button) и контейнеры для элементов (потомки класса ViewGroup).

Далее происходит последовательное перечисление самых востребованных макетов интерфейса и их непосредственная работа.

Виджеты входят в иерархию объектов View, называемую иерархией представления. Корневым элементом иерархии представления является элемент Layout, который и является макетом пользовательского интерфейса.

FrameLayout предназначен для блокировки области на экране для отображения одного элемента. Размер FrameLayout - это размер его самого большого дочернего элемента.

LinearLayout располагает элементы внутри себя друг за другом вертикально или горизонтально. Направление задается через orientation в верстке. Распределяет свободное пространство по длине или ширине через веса.

ConstraintLayout позволяет создавать большие и сложные макеты с плоской иерархией (без вложений). Он похож на RelativeLayout в том, что все view раскладываются в соответствии с отношениями между родственными view и родительским макетом, но он более гибок, чем RelativeLayout, и проще в использовании с редактором макетов Android Studio.

CoordinatorLayout, часть библиотеки поддержки дизайна Android, является подклассом FrameLayout и поэтому наследует его использование layout_gravity для позиционирования дочерних элементов, но также включает в себя концепцию поведения. Прикрепление поведения к представлению либо с помощью атрибута layout_behavior, либо с помощью функции setBehavior() позволяет этому поведению перехватывать практически все, что находится перед базовым представлением: измерение, компоновку, вложенную прокрутку, события касания, изменения в указанных зависимых представлениях и вставки окон.

3 Программная реализация приложения "Галерея"на языке программирования Kotlin

Программный код первого окна, то есть сплеш экрана, реализует экран с логтипом компании Surf с удерживающей паузой этого окна, составляющей

500 миллисекунд. Пауза реализуется с помощью handler и Looper, задерживающие основной поток на указанный промежуток времени. Затем используется Intent, в котором происходит явное указание перехода с одного основного компонента Activity (окна) на другое окно приложения.

```
package ru.summer.test

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.view.View
import ru.summer.test.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

        private lateinit var bindingClass :
           ActivityMainBinding
        private var handler = Handler()

        override fun onCreate(savedInstanceState: Bundle?)
            {
                super.onCreate(savedInstanceState)
                bindingClass = ActivityMainBinding.inflate
                   (layoutInflater)
                setContentView(bindingClass.root)


                var intent = Intent(this, second_activity
                   ::class.java)

                handler = Handler(Looper.getMainLooper())
                handler.postDelayed({
                        startActivity(intent)
```

```
                }, 500)
        }


}
```

Программная реализация дизайна сплеш экрана выглядит следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:
    android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto
            "
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">



        <ImageView
                android:id="@+id/imageView"

                android:layout_width="91dp"
                android:layout_height="104.6dp"
                app:layout_constraintBottom_toBottomOf="
                    parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="
                    parent"
                app:layout_constraintTop_toTopOf="parent"
                app:srcCompat="@drawable/img" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Далее рассматривается второй компонент Activity, то есть новый экран - экран Авторизации. В программном коде используется Binding class. На экране появляются два поля Edit Text для логина, т.е. телефона с фиксированным маркером (+7) при нажатии, аналогично и для пароля. Для личных данных формируется скрытый ввод, который возможно отключить, нажав на кнопку "глаза". После нажатия на кнопку Войти происходит проверка введенных данных через условный оператор when с константными номерами, вынесенными в отдельный класс. Также предусмотрена проверка в случае неверного ввода данных вход не будет совершен, а на экране появится ошибка "неверный логин или пароль". Через Intent и метод putExtra происходит переход в новый Activity с данными о флаге с пометкой TOTAL.COUNT.

```
package ru.summer.test

object Constance {

        const val user_1 = "+71234567890"
        const val user_2 = "+79876543219"
        const val user_3 = "+78005003030"

        const val user_1_pass = "qwerty"
        const val user_2_pass = "qwerty"
        const val user_3_pass = "qwerty"
}
```

Программная реализация экрана Авторизации:

```
package ru.summer.test

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.text.method.HideReturnsTransformationMethod
import android.text.method.PasswordTransformationMethod
import android.view.View
```

```
import ru.summer.test.databinding.ActivityMainBinding
import ru.summer.test.databinding.SecondActivityBinding

class second_activity : AppCompatActivity() {

        private lateinit var bindingClass :
           SecondActivityBinding
        private var i = 0
        private var clicc = 0

        override fun onCreate(savedInstanceState: Bundle?)
            {
                super.onCreate(savedInstanceState)
                bindingClass = SecondActivityBinding.
                   inflate(layoutInflater)
                setContentView(bindingClass.root)
                bindingClass.edLogin.setOnClickListener {
                        bindingClass.edLogin.setText("+7")
                        bindingClass.edLogin.setSelection
                           (2)
                }

                bindingClass.btSeePass.setOnClickListener
                   {
                        if (clicc == 0) {
                                bindingClass.edPass.
                                   transformationMethod =
                                HideReturnsTransformationMethod
                                   .getInstance()
                                bindingClass.edPass.
                                   setSelection((
                                   bindingClass.edPass.
                                   text.toString()).length
                                   )
                                clicc ++
```

```
                }
                else {
                        bindingClass.edPass.
                           transformationMethod =
                        PasswordTransformationMethod
                           .getInstance()
                        bindingClass.edPass.
                           setSelection((
                           bindingClass.edPass.
                           text.toString()).length
                           )
                        clicc = 0
                }
        }

        bindingClass.bt.setOnClickListener {
                val login = bindingClass.edLogin.
                   text.toString()

                when (login) {
                        Constance.user_1 -> {
                                if(bindingClass.
                                   edPass.text.
                                   toString() ==
                                   Constance.
                                   user_1_pass){
                                        i = 1
                                        openAct3()
                                }
                                else
                                bindingClass.tverr
                                   .text = "
                                   Неверный логин
                                   или пароль"
                        }
```

```
Constance.user_2 -> {
        if(bindingClass.
           edPass.text.
           toString() ==
           Constance.
           user_2_pass) {
                i = 2
                openAct3()
        }
        else
        bindingClass.tverr
           .text = "
           Неверный логин
           или пароль"
}

Constance.user_3 -> {
        if(bindingClass.
           edPass.text.
           toString() ==
           Constance.
           user_3_pass){
                i = 3
                openAct3()
        }
        else
        bindingClass.tverr
           .text = "
           Неверный логин
           или пароль"
}

else -> {
        bindingClass.tverr
```

```
                                                .visibility =
                                                View.VISIBLE
                                            bindingClass.tverr
                                                .text = "
                                                Неверный логин
                                                или пароль"
                                        }
                                }
                        }
                }

                fun openAct3 () {
                        val openIntent = Intent (this,
                            third_activity::class.java)
                        openIntent.putExtra(third_activity.
                            TOTAL_COUNT, i)
                        startActivity(openIntent)
                }
}
```

Программная реализация дизайна экрана Авторизации:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:
    android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto
            "
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".second_activity">

        <TextView
                android:id="@+id/tv1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

```
            android:layout_marginStart="17dp"
            android:layout_marginTop="38dp"
            android:text="Вход"
            android:textColor="@color/black"
            android:textSize="24sp"
            app:layout_constraintStart_toStartOf="
               parent"
            app:layout_constraintTop_toTopOf="parent"
               />

        <Button
            android:id="@+id/bt"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="16dp"
            android:layout_marginEnd="16dp"
            android:layout_marginBottom="20dp"
            android:backgroundTint="@color/black"
            android:text="Вход"
            app:layout_constraintBottom_toBottomOf="
               parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="
               parent" />

        <EditText
            android:id="@+id/edLogin"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="16dp"
            android:layout_marginTop="112dp"
            android:layout_marginEnd="16dp"
            android:autoText="false"
            android:contextClickable="false"
            android:ems="10"
```

```
            android:freezesText="true"
            android:hint="Логин"
            android:inputType="textPersonName"
            android:maxLength="12"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="
               parent"
            app:layout_constraintTop_toTopOf="parent"
               />

        <TextView
            android:id="@+id/tverr"
            android:layout_width="0dp"
            android:layout_height="30dp"
            android:layout_marginStart="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="76dp"
            android:background="#E12B2B"
            android:gravity="center"
            android:shadowColor="#FFFFFF"
            android:textColor="#FFFFFF"
            android:textSize="20sp"
            android:visibility="gone"
            app:layout_constraintBottom_toBottomOf="
               parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="
               parent" />

        <EditText
            android:id="@+id/edPass"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:ems="10"
```

```
            android:hint="Пароль"
            android:inputType="textPassword"
            android:onClick='"+7"'
            app:layout_constraintEnd_toEndOf="@+id/
               edLogin"
            app:layout_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="@+id
               /edLogin"
            app:layout_constraintTop_toBottomOf="@+id/
               edLogin" />

        <ImageView
            android:id="@+id/imageView2"
            android:layout_width="233.35dp"
            android:layout_height="221.36dp"
            android:layout_marginEnd="201dp"
            android:layout_marginBottom="50dp"
            android:rotation="0"
            app:layout_constraintBottom_toTopOf="@+id/
               bt"
            app:layout_constraintEnd_toEndOf="parent"
            app:srcCompat="drawable/img_1" />

        <ImageButton
            android:id="@+id/btSeePass"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:backgroundTint="color/
               cardview_shadow_end_color"
            android:scaleType="fitEnd"
            app:layout_constraintBottom_toBottomOf="@+
               id/edPass"
            app:layout_constraintEnd_toEndOf="@+id/
               edPass"
            app:layout_constraintTop_toTopOf="@+id/
```

```
                edPass"
            app:srcCompat="@drawable/img_12" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Третий экран представлен в зависимости от того, какие константные логин и пароль были занесены в предыдущей реализации экрана Авторизации. Экран отображает либо все картинки (1 пользователь), либо часть картинок (2 пользователь), либо картинки полностью отсутствуют (3 пользователь).

```
package ru.summer.test

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import ru.summer.test.databinding.ActivityThirdBinding
import ru.summer.test.databinding.SecondActivityBinding

class third_activity : AppCompatActivity() {

    companion object{
        const val TOTAL_COUNT = "total_count"
    }

    private lateinit var bindingClass :
       ActivityThirdBinding

    override fun onCreate(savedInstanceState: Bundle?)
        {
            super.onCreate(savedInstanceState)
            bindingClass =
            ActivityThirdBinding.inflate(
               layoutInflater)
            setContentView(bindingClass.root)
            show_i()
```

```
        }

        fun show_i() {
                val count = intent.getIntExtra(TOTAL_COUNT
                    , 0)

                if (count == 3) {
                        bindingClass.img3.visibility =
                            View.GONE
                        bindingClass.img1.visibility =
                            View.GONE
                        bindingClass.img2.visibility =
                            View.GONE
                        bindingClass.img4.visibility =
                            View.GONE
                        bindingClass.tv3.visibility = View
                            .GONE
                        bindingClass.tv1.visibility = View
                            .GONE
                        bindingClass.tv2.visibility = View
                            .GONE
                        bindingClass.tv4.visibility = View
                            .GONE
                }

                if (count == 2) {
                        bindingClass.img3.visibility =
                            View.GONE
                        bindingClass.img4.visibility =
                            View.GONE
                        bindingClass.tv3.visibility = View
                            .GONE
                        bindingClass.tv4.visibility = View
                            .GONE
                }
```

```
        }
}
```

Программная реализация дизайна нового экрана для отражения изображений выглядит таким образом:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto
           "
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".third_activity">

        <ImageView
                android:id="@+id/img1"
                android:layout_width="160dp"
                android:layout_height="222dp"
                android:layout_marginStart="16dp"
                android:layout_marginTop="90dp"
                android:visibility="visible"
                app:layout_constraintStart_toStartOf="
                   parent"
                app:layout_constraintTop_toTopOf="parent"
                app:srcCompat="@drawable/img_2" />

        <ImageView
                android:id="@+id/img4"
                android:layout_width="160dp"
                android:layout_height="222dp"
                android:layout_marginTop="346dp"
                android:layout_marginEnd="16dp"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintTop_toTopOf="parent"
```

```
            app:srcCompat="@drawable/img_5" />

        <ImageView
            android:id="@+id/img2"
            android:layout_width="160dp"
            android:layout_height="222dp"
            android:layout_marginTop="90dp"
            android:layout_marginEnd="16dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/img_3" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="17dp"
            android:layout_marginTop="38dp"
            android:text="Галерея"
            android:textColor="color/black"
            android:textSize="24sp"
            app:layout_constraintStart_toStartOf="
               parent"
            app:layout_constraintTop_toTopOf="parent"
               />

        <ImageView
            android:id="@+id/img3"
            android:layout_width="160dp"
            android:layout_height="222dp"
            android:layout_marginStart="16dp"
            android:layout_marginTop="346dp"
            app:layout_constraintStart_toStartOf="
               parent"
            app:layout_constraintTop_toTopOf="parent"
```

```
            app:srcCompat="@drawable/img_4" />

        <TextView
            android:id="@+id/tv1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="Первый день в Surf"
            android:textColor="@color/black"
            android:textSize="14sp"
            app:layout_constraintEnd_toEndOf="@+id/
               img1"
            app:layout_constraintStart_toStartOf="@+id
               /img1"
            app:layout_constraintTop_toBottomOf="@+id/
               img1" />

        <TextView
            android:id="@+id/tv2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="Самый милый корги"
            android:textColor="@color/black"
            app:layout_constraintEnd_toEndOf="@+id/
               img2"
            app:layout_constraintHorizontal_bias="
               0.431"
            app:layout_constraintStart_toStartOf="@+id
               /img2"
            app:layout_constraintTop_toBottomOf="@+id/
               img2" />

        <TextView
            android:id="@+id/tv3"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="Печенье в коробке"
            android:textColor="@color/black"
            app:layout_constraintEnd_toEndOf="@+id/
               img3"
            app:layout_constraintHorizontal_bias="
               0.457"
            app:layout_constraintStart_toStartOf="@+id
               /img3"
            app:layout_constraintTop_toBottomOf="@+id/
               img3" />

        <TextView
            android:id="@+id/tv4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="Чашка свежего кофе"
            android:textColor="@color/black"
            app:layout_constraintEnd_toEndOf="@+id/
               img4"
            app:layout_constraintHorizontal_bias="
               0.509"
            app:layout_constraintStart_toStartOf="@+id
               /img4"
            app:layout_constraintTop_toBottomOf="@+id/
               img4" />

        <ImageButton
            android:id="@+id/imageButton2"
            android:layout_width="120dp"
            android:layout_height="56dp"
            android:layout_marginEnd="148dp"
```

```
            android:backgroundTint="@color/white"
            android:scaleType="fitCenter"
            app:layout_constraintBottom_toBottomOf="
               parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:srcCompat="@drawable/img_7" />

        <ImageButton
            android:id="@+id/imageButton3"
            android:layout_width="120dp"
            android:layout_height="56dp"
            android:backgroundTint="color/white"
            android:scaleType="fitCenter"
            app:layout_constraintBottom_toBottomOf="@+
               id/imageButton2"
            app:layout_constraintStart_toEndOf="@+id/
               imageButton2"
            app:layout_constraintTop_toTopOf="@+id/
               imageButton2"
            app:layout_constraintVertical_bias="0.0"
            app:srcCompat="@drawable/img_8" />

        <ImageButton
            android:id="@+id/imageButton"
            android:layout_width="120dp"
            android:layout_height="56dp"
            android:backgroundTint="@color/white"
            android:scaleType="fitCenter"
            app:layout_constraintBottom_toBottomOf="@+
               id/imageButton2"
            app:layout_constraintEnd_toStartOf="@+id/
               imageButton2"
            app:layout_constraintTop_toTopOf="@+id/
               imageButton2"
            app:srcCompat="@drawable/img_6" />
```

```
        <ImageButton
                android:id="@+id/imageButton4"
                android:layout_width="56dp"
                android:layout_height="63dp"
                android:layout_marginTop="24dp"
                android:layout_marginEnd="24dp"
                android:backgroundTint="@color/white"
                android:scaleType="fitCenter"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintTop_toTopOf="parent"
                app:srcCompat="@drawable/img_9" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Заключение

В данной работе изучались теоретические основы языка программирования Kotlin и их практическая реализация.

В ходе выполнения работы были решены следующие задачи:

— изучены основные компоненты языка и его синтаксис;

— разработано приложение "Галерея"на основе языка Kotlin в среде Android Studio.

Таким образом, все поставленные задачи были решены. Цель обучения можно считать достигнутой.