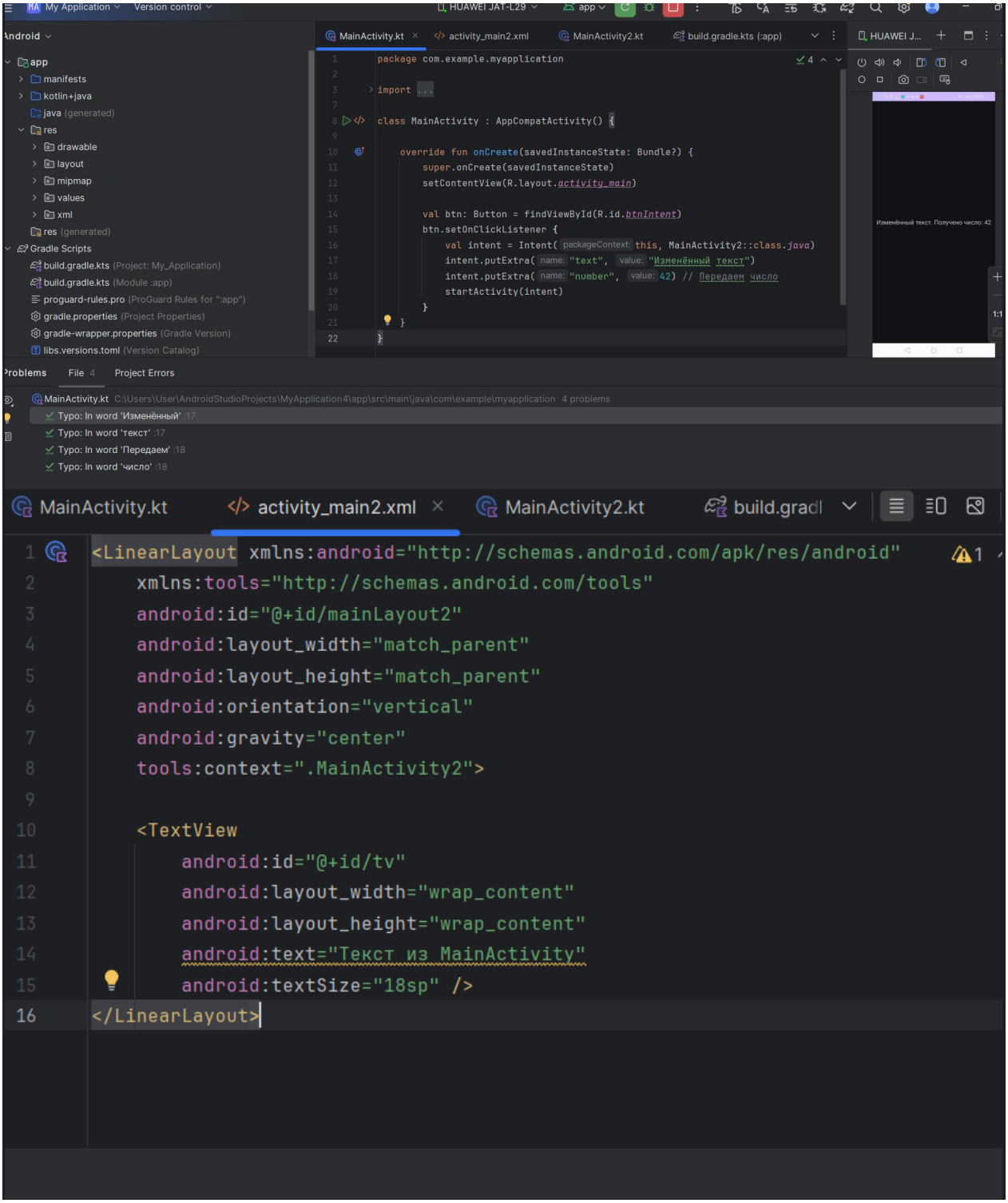
Практическая работа Intent
Переход между Активностями

Выполнила:Окунцова В.ИС233

Задание 1: Протестируйте работу putExtra и getExtra, передайте не только строковые значения, а к примеру числовые.

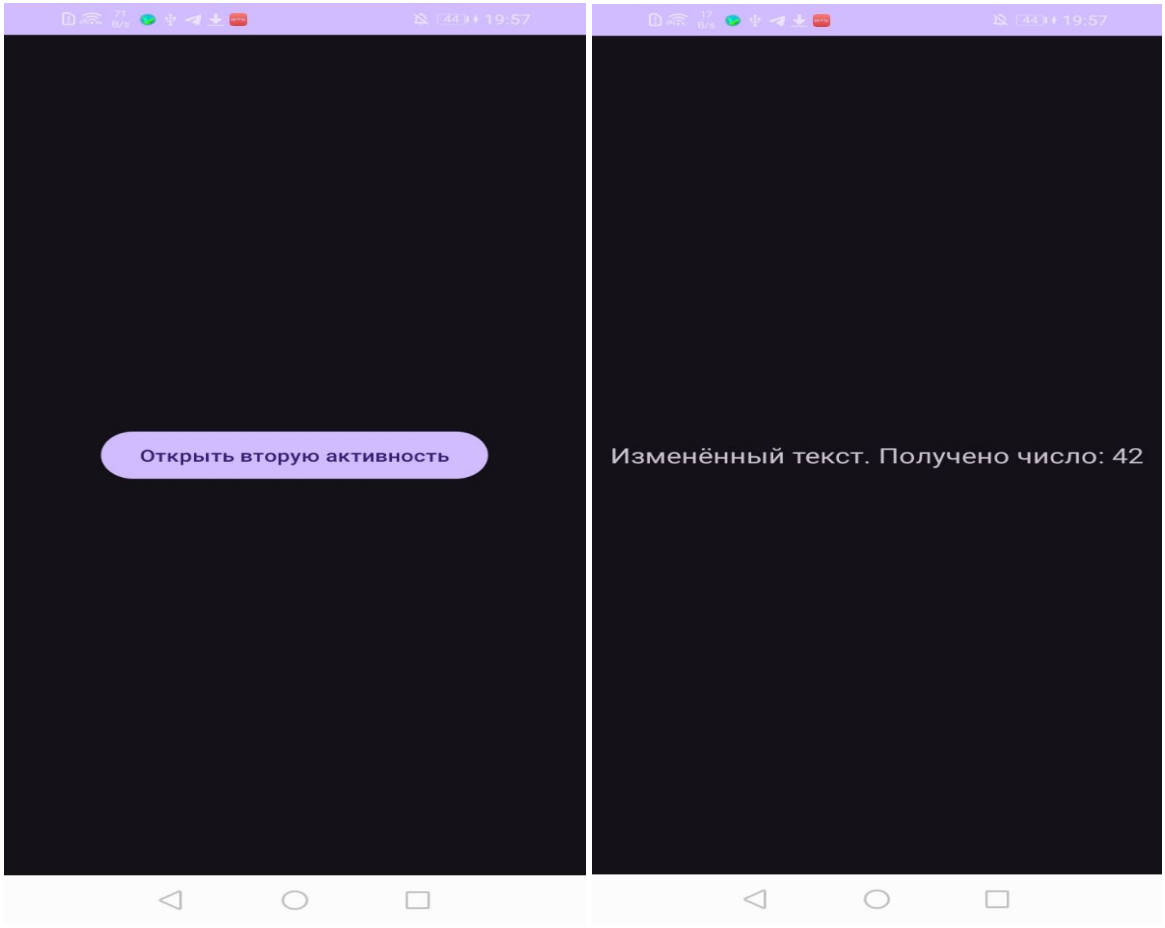


MainActivity.kt | activity_main2.xml | MainActivity2.kt | build.gradle.kts (:app)

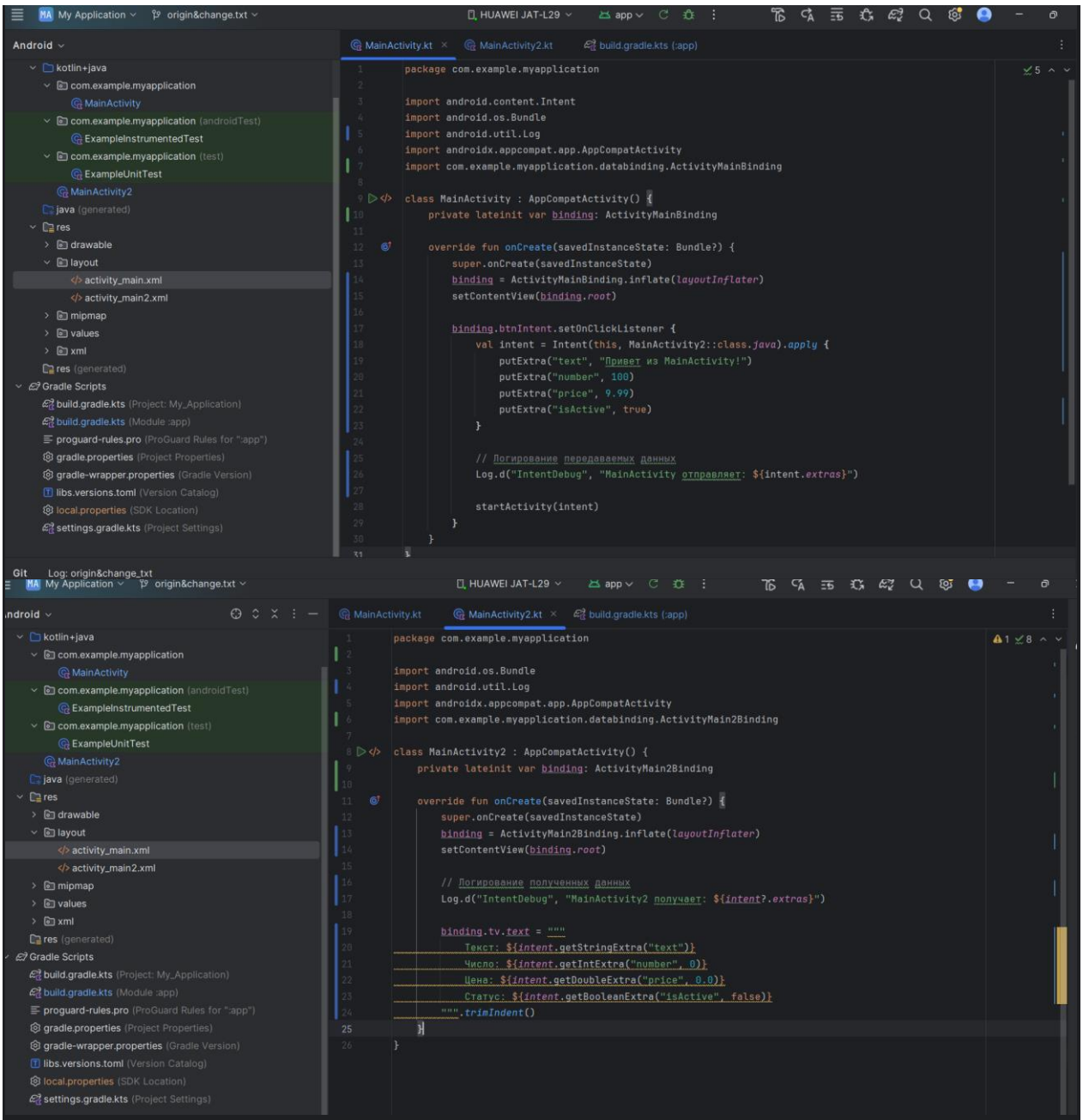
```kotlin
package com.example.myapplication

import ...

class MainActivity2 : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main2)

        val textView: TextView = findViewById(R.id.tv)
        val message = intent.getStringExtra( name: "text")
        val number = intent.getIntExtra( name: "number",  defaultValue: 0)

        textView.text = "$message. Получено число: $number"
    }
}
```

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
}

android {
    namespace = "com.example.myapplication"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.myapplication"
        minSdk = 24
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }

    buildFeatures {
        viewBinding = true
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
```

```
    kotlinOptions {
        jvmTarget = "11"
    }
}
dependencies {

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.activity)
    implementation(libs.androidx.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
}
```



Задание 2:




```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
}

android {
    namespace = "com.example.myapplication"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.myapplication"
        minSdk = 24
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
```

```
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }

    // Здесь уже включен ViewBinding
    buildFeatures {
        viewBinding = true
        // Можно добавить другие фичи, например:
        // compose = true
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }

    kotlinOptions {
        jvmTarget = "11"
    }
}
dependencies {

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.activity)
    implementation(libs.androidx.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
}
```

Поработав с привязкой переменных findViewById, используйте ViewBinding и напишите, как вы лично считаете в чём разница и зачем нужен данный способ привязки.

Раньше для того, чтобы обратиться к кнопке или другому элементу на экране, мы использовали метод findViewById. Нужно было писать длинные строчки кода, искать элемент по его ID и указывать, какого он типа (кнопка, текст и т.д.). Это неудобно и иногда приводило к ошибкам, если ID написан неправильно. ViewBinding — это современный и простой способ работать с элементами интерфейса. Когда включаешь ViewBinding, Android сам создаёт специальный класс, в котором уже есть все элементы из твоего XML-файла с правильными типами и именами. Теперь не нужно писать findViewById — можно сразу обращаться к элементам через объект binding. Это сокращает код, делает его чище и уменьшает вероятность ошибок. Таким образом, ViewBinding облегчает и ускоряет разработку, особенно когда экранов и элементов много. Я считаю, что этот способ гораздо удобнее и надёжнее, поэтому его лучше использовать вместо findViewById.