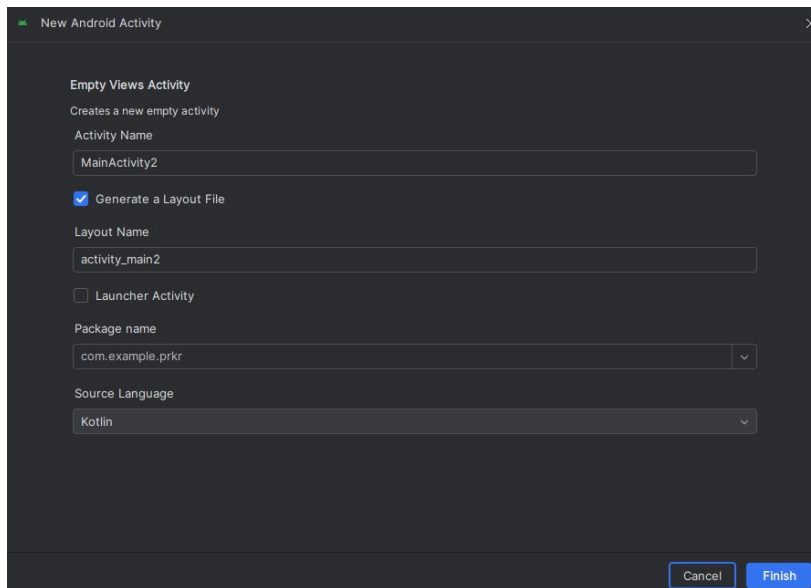


### Практика 3

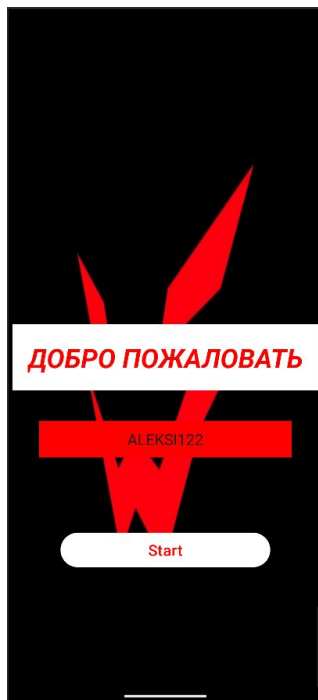
Добавляем новую пустую активность (теперь уже правильно)



И поменяем название активности в коде кнопки (с Activity2 на MainActivity2)

```
val btn : Button = findViewById(R.id.btn_1)
btn.setOnClickListener{
    val intent = Intent( packageContext: this, MainActivity2::class.java)
    startActivity(intent)
}
```

Тестируем



Я изменил background на красный, и если нажать на Start будет белый экран

## Передача значений к другой активности

Для передачи значений используется объект класса Intent. Попробуем передать через него текст из редактируемого поля.

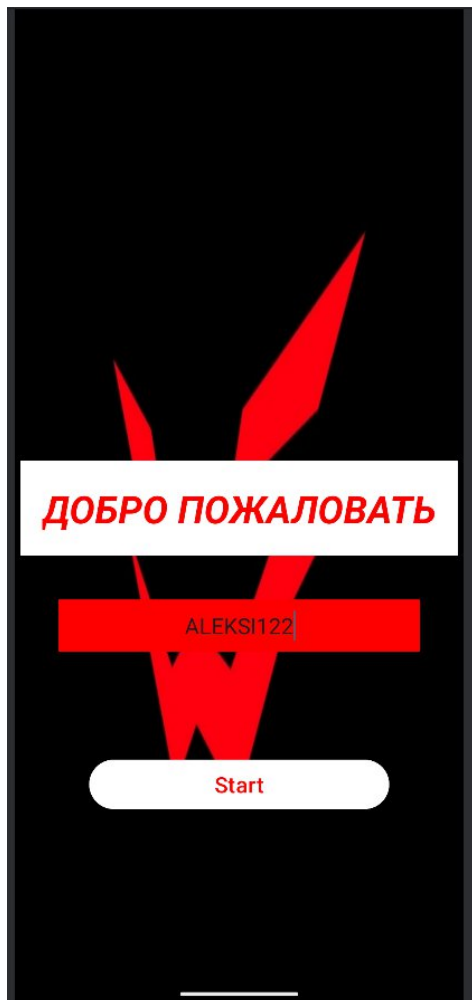
### Отправка данных

```
val btn : Button = findViewById(R.id.btn_1)
btn.setOnClickListener{
    val intent = Intent(packageContext, MainActivity2::class.java)
    intent.putExtra(name: "Name", findViewById<EditText>(R.id.editText).text.toString())
    startActivity(intent)
}
```

### Получение данных

```
val text : TextView = findViewById(R.id.name)
text.text= intent.getStringExtra(name: "Name")
```

### Тестируем



### Задание 1

Для передачи числа используется точно такая же функция, как и для строки. Передадим число 122

```

val btn : Button = findViewById(R.id.btn_1)
btn.setOnClickListener{
    val intent = Intent(packageContext: this, MainActivity2::class.java)
    intent.putExtra(name: "Name", findViewById<EditText>(R.id.editText).text.toString())
    intent.putExtra(name: "num", value: 122)
    startActivity(intent)
}

```

А для получения Int используется функция getIntentExtra, которая работает точно так же

```

val text : TextView = findViewById(R.id.name)
var dann = "Получено:\n"
dann += intent.getStringExtra(name: "Name") + "\n"
dann += intent.getIntExtra(name: "num", defaultValue: 0).toString()
text.text = dann

```

Тестируем



Задание 2

ViewBinding используется для обращения к элементам интерфейса напрямую, а не через поиск по id как это делалось в функции findViewById. Сам процесс подключения немного требует выполнения некоторых шагов, но в дальнейшем обращаться к элементам становится куда проще.

Для его подключения нужно добавить в файл build.gradle.kts viewBinding = true и синхронизировать изменения:

```
android {  
    buildFeatures {  
        viewBinding = true  
    }  
}
```

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

[Sync Now](#) [Ignore these changes](#)

После, внутри основного класса добавить переменную типа ActivityMainBinding, которая автоматически формируется при создании активности. К примеру для MainActivity2 был создан класс ActivityMain2Binding

MainActivity

```
class MainActivity : AppCompatActivity() {  
    lateinit var binding: ActivityMainBinding  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        binding = ActivityMainBinding.inflate(layoutInflater)  
        setContentView(binding.root)  
    }  
}
```

MainActivity2

```
class MainActivity2 : AppCompatActivity() {  
    lateinit var binding: ActivityMain2Binding  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        binding = ActivityMain2Binding.inflate(layoutInflater)  
        setContentView(binding.root)  
    }  
}
```

Функция inflate возвращает макет активности который передается в переменную binding, и уже через неё можно обратиться к любому элементу

MainActivity с findViewById

```
val btn : Button = findViewById(R.id.btn_1)  
btn.setOnClickListener {  
    val intent = Intent(packageContext, MainActivity2::class.java)  
    intent.putExtra("Name", findViewById<EditText>(R.id.editText).text.toString())  
    intent.putExtra("num", 122)  
    startActivity(intent)  
}
```

## MainActivity c BindingView

```
val btn : Button = binding.btn1
btn.setOnClickListener{
    val intent = Intent( packageContext: this, MainActivity2::class.java)
    intent.putExtra( name: "Name", binding.editText.text.toString())
    intent.putExtra( name: "num", value: 122)
    startActivity(intent)
}
```

## MainActivity2 c findViewById

```
val text : TextView = findViewById(R.id.name)
var dann = "Получено:\n"
dann += intent.getStringExtra( name: "Name") + "\n"
dann += intent.getIntExtra( name: "num", defaultValue: 0).toString()
text.text = dann
```

## MainActivity2 c BindingView

```
val text : TextView = binding.name
var dann = "Получено:\n"
dann += intent.getStringExtra( name: "Name") + "\n"
dann += intent.getIntExtra( name: "num", defaultValue: 0).toString()
text.text = dann
```

Результат на лицо. Более того, BindingView хранит все компоненты одной активности отдельно и нельзя по ошибке обратиться к компоненту из другой активности. Например, надпись, приветствующая в начале, остается доступной во второй активности.

Вывод: BindingView гораздо безопаснее, удобней и быстрее, чем findViewById