Лекция #19. Передача данных между Activity. Сериализация

Для передачи данных между двумя **Activity** используется объект **Intent**. Через его метод **putExtra()** можно добавить ключ и связанное с ним значение.

Например, передача из текущей **activity** в **SecondActivity** строки *"Hello World"* с ключом *"hello":*

```
// создание объекта Intent для запуска SecondActivity
val intent: Intent = Intent(this@MainActivity,
SecondActivity::class.java)
// передача объекта с ключом "hello" и значением "Hello World"
intent.putExtra("hello", "Hello World")
// запуск SecondActivity
startActivity(intent)
```

Для передачи данных применяется метод **putExtra()**, который в качестве значения позволяет передать данные простейших типов - **String**, **Int**, **Float**, **Double**, **Long**, **Short**, **Byte**, **Char**, **массивы** этих типов, либо объект интерфейса **Serializable**.

Чтобы получить отправленные данные при загрузке **SecondActivity**, можно воспользоваться методом **get()**, в который передается ключ объекта:

```
// создание Bundle с содержимым переданным при запуске Activity
val arguments: Bundle? = intent.extras
// проверка arguments на null
if (arguments != null){
    // получение значение с ключом "hello" из arguments
    val name: String = arguments.get("hello").toString()
}
```

В зависимости от типа отправляемых данных при их получении мы можем использовать ряд методов объекта Bundle. Все они в качестве параметра принимают ключ объекта. Основные из них:

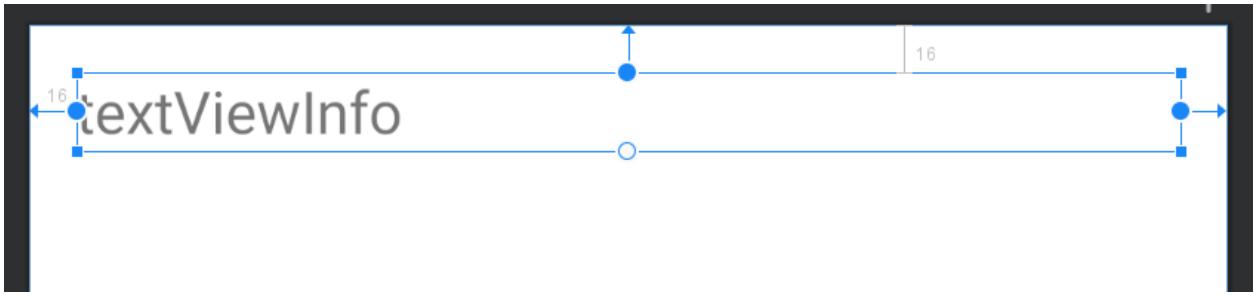
- **get()**: универсальный метод, который возвращает значение типа Object. Соответственно поле получения данное значение необходимо преобразовать к нужному типу
- **getString()**: возвращает объект типа **String**
- **getInt()**: возвращает значение типа **Int**
- **getByte()**: возвращает значение типа **Byte**
- **getChar()**: возвращает значение типа **Char**
- **getShort()**: возвращает значение типа **Short**
- **getLong()**: возвращает значение типа **Long**
- **getFloat()**: возвращает значение типа **Float**
- **getDouble()**: возвращает значение типа **Double**
- **getBoolean()**: возвращает значение типа **Boolean**
- **getCharArray()**: возвращает массив объектов **Char**
- **getIntArray()**: возвращает массив объектов **Int**
- **getFloatArray()**: возвращает массив объектов **Float**
- **getSerializable()**: возвращает объект интерфейса **Serializable**

Пусть у нас в проекте будет определено две activity: **MainActivity** и **SecondActivity**.

В коде **SecondActivity** определим получение данных:

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

    val textView: TextView = findViewById(R.id.textViewInfo)

    // создание Bundle с содержимым переданным при запуске Activity
    val arguments: Bundle? = intent.extras
    // проверка arguments на null
    if (arguments != null){
        val name: String = arguments.get("name").toString()
        val company: String? = arguments.getString("company")
        val age: Int = arguments.getInt("age")

        textView.text = "Name: $name \nCompany: $company \nAge: $age"
    }
}
```

Так же добавим **textViewInfo** в интерфейс **SecondActivity**:



В данном случае в **SecondActivity** получаем все данных из объекта **Bundle** и выводим их в текстовое поле **TextView**. Предполагается, что данной **activity** будут передаваться три элемента - две строки с ключами ***name*** и ***company*** и число с ключом ***age***.

Теперь определим передачу в SecondActivity данных. Например, определим для MainActivity следующий интерфейс в файле **activity_main.xml**:

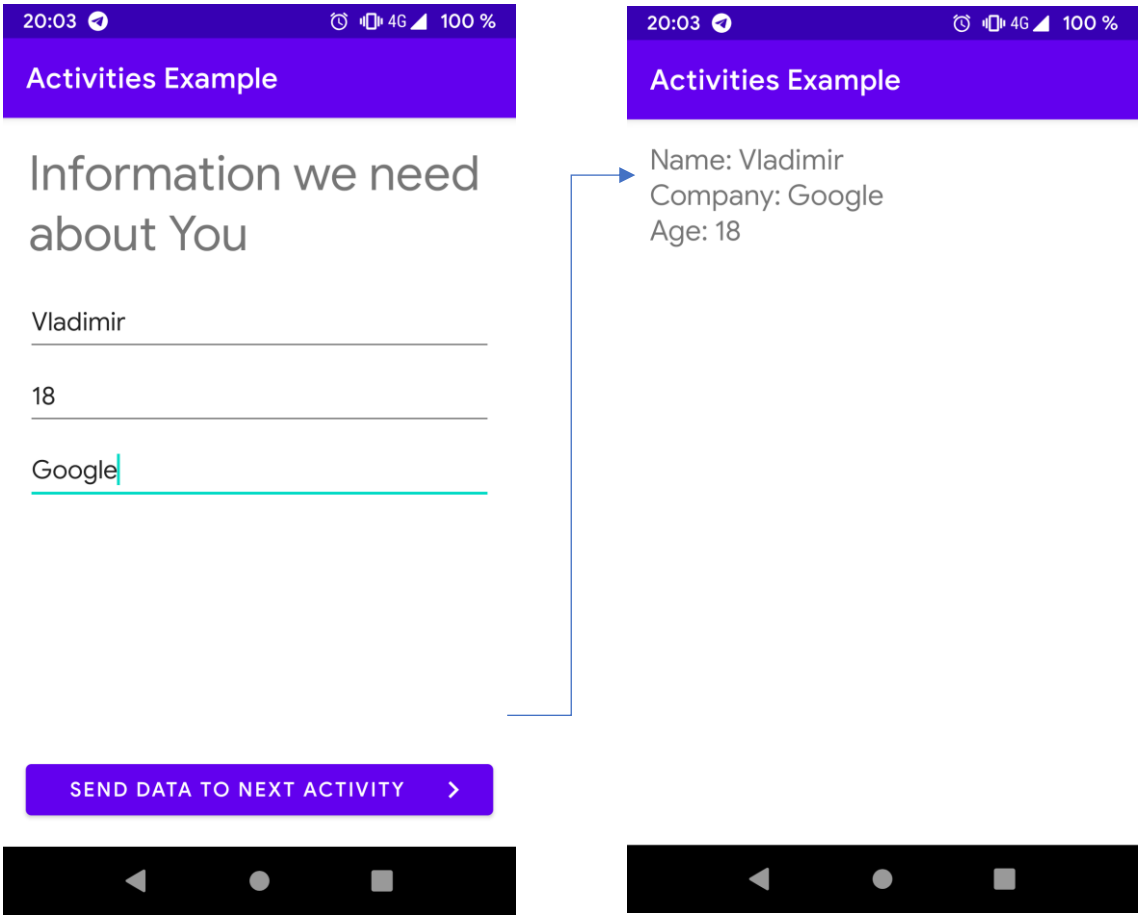
Здесь определены три текстовых поля для ввода данных и кнопка.

В классе **MainActivity** определим обработчик нажатия на кнопку:

```
fun buttonSendClick(view: View){
    val nameText: TextView = findViewById(R.id.editTextName)
    val ageText: TextView = findViewById(R.id.editTextAge)
    val companyText: TextView = findViewById(R.id.editTextCompany)

    val name = nameText.text.toString()
    val age = ageText.text.toString().toInt()
    val company = companyText.text.toString()

    val intent: Intent =
Intent(this@MainActivity,SecondActivity::class.java)

    intent.putExtra("name", name)
    intent.putExtra("age", age)
    intent.putExtra("company", company)

    startActivity(intent)
}
```

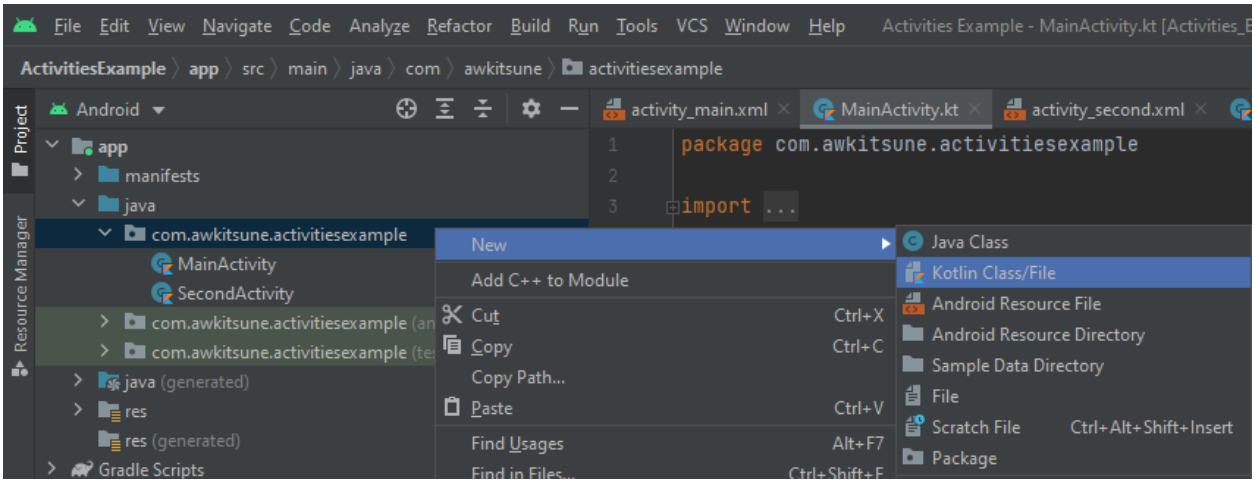
В обработчике нажатия кнопки получаем введенные в текстовые поля EditText данные и передаем их в объект **Intent** с помощью метода **putExtra()**. Затем запускаем **SecondActivity**.

В итоге при нажатии на кнопку запустится SecondActivity, которая получит некоторые введенные в текстовые поля данные.

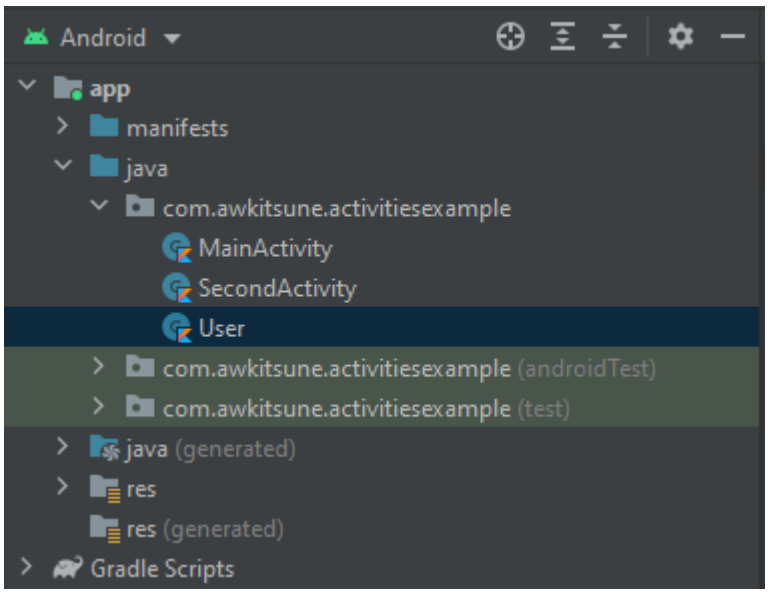


Передача сложных объектов

В примере выше передавались простые данные - числа, строки. Но также мы можем передавать более сложные данные. В этом случае используется механизм сериализации. Для этого нажмем на папку пакета, где находятся классы **MainActivity** и **SecondActivity**, правой кнопкой мыши и в контекстном меню выберем **New -> Kotlin Class**:



Назовем новый класс **User** - пусть он будет представлять пользователя.


Android
app
manifests
java
com.awkitsune.activitiesexample
MainActivity
SecondActivity
User
com.awkitsune.activitiesexample (androidTest)
com.awkitsune.activitiesexample (test)
java (generated)
res
res (generated)
Gradle Scripts

Пусть класс **User** имеет следующий код:

```
package com.awkitsune.activitiesexample

import java.io.Serializable

class User() : Serializable {
    private var name: String = ""
    private var company: String = ""
    private var age: Int = 0

    constructor(_name: String, _company: String, _age: Int) : this() {
        name = _name
        company = _company
        age = _age
    }

    fun setName(_name: String){
        name = _name
    }

    fun getName(): String {
        return name
    }

    fun setCompany(_company: String){
        company = _company
    }

    fun getCompany(): String {
        return company
    }

    fun setAge(_age: Int){
        _age = age
    }

    fun getAge(): Int {
        return age
    }

}
```

Стоит отметить, что данный класс реализует интерфейс **Serializable**. Теперь изменим код обработчика нажатия на кнопку:

```
fun buttonSendClick(view: View){
    val nameText: TextView = findViewById(R.id.editTextName)
    val ageText: TextView = findViewById(R.id.editTextAge)
    val companyText: TextView = findViewById(R.id.editTextCompany)

    val name = nameText.text.toString()
    val age = ageText.text.toString().toInt()
    val company = companyText.text.toString()

    val user: User = User(name, company, age)

    val intent: Intent = Intent(this@MainActivity,
SecondActivity::class.java)
    intent.putExtra(user.javaClass.simpleName, user)
    startActivity(intent)
}
```

Теперь вместо трех разрозненных данных передается один объект **User**. В качестве ключа используется результат метода **user.class.getSimpleName**, который по сути возвращает название класса.

И изменим метод **onCreate** класса **SecondActivity**:

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

    val textView: TextView = findViewById(R.id.textViewInfo)

    var user: User = User()

    val arguments: Bundle? = intent.extras
    if (arguments != null){
        user = arguments.getSerializable(user.javaClass.simpleName) as
User

        textView.text = "Name: ${user.getName()} \nCompany:
${user.getCompany()} \nAge: ${user.getAge()}"
    }
}
```

Для получения данных применяется метод **getSerializable()**, поскольку класс **User** реализует интерфейс **Serializable**. Таким образом, мы можем передать один единственый объект вместо набора разрозненных данных.