Обработка нажатия кнопок и взаимодействие с view

Перед добавлением поведения к кнопкам необходимо включить автоматический импорт, чтобы Android Studio автоматически импортировала все классы, необходимые для кода Котлин.

1. В Android Studio откройте Редактор настроек. Перейдите к **File**>**New project Setup>Settigs for new project>Other Settings**
2. Выберите **Auto Imports**. В разделе Java убедитесь, что пункт **Add Unambiguous Imports on the fly** отмечен флажком.
3. Закройте редактор настроек.

Отображение сообщения по нажатию кнопки

1. Откройте класс MainActivity.kt. (раскройте ветвь **app** > **java** > **com.example.android.myfirstapp** чтобы найти MainActivity). Этот класс описывает поведение главного экрана нашего приложения. Пока что класс содержит только одну функцию, onCreate(). Функция onCreate() выполняется, когда активити стартует.
2. Посмотрите внимательно на функцию onCreate(). Обратите внимание на вызов функции setContentView(). Эта строка устанавливает файл ресурсов activity_main.xml в качестве разметки активити главного экрана.

```
setContentView(R.layout.activity_main);
```

3. Добавим новую функцию toastMe() в класс MainActivity. Функция toastMe() принимает один аргумент – View. Это представление, получающее событие нажатия кнопки.Функция создает и отображает всплывающее уведомление. Вот ее код:

```
fun toastMe(view: View) {
  // val myToast = Toast.makeText(this, message, duration);
  val myToast = Toast.makeText(this, "Hello Toast!", Toast.LENGTH_SHORT)
  myToast.show()
}
```

В языке Kotlin, если явно не используется никакого модификатора доступа, то по умолчанию применяется public . Далее идет слово fun, обозначающее функцию, и ее имя. В скобках передаваемый функции аргумент – его имя и тип разделены двоеточием. Далее объявляется переменная val myToast.

Переменной myToast присваивается результат вызова метода makeText java-класса Toast. Метод makeText принимает контекст, сообщение и длительность отображения тоста, и возвращает тост в переменную myToast. Тост затем отображается методом show().

Функция toastMe является примером использования java-кода в kotlin-классе.

4. Откройте файл макета activity_main.xml и добавьте свойство android:onClick кнопкеToast. Значением свойства установите **toastMe**.

```
android:onClick="toastMe"
```

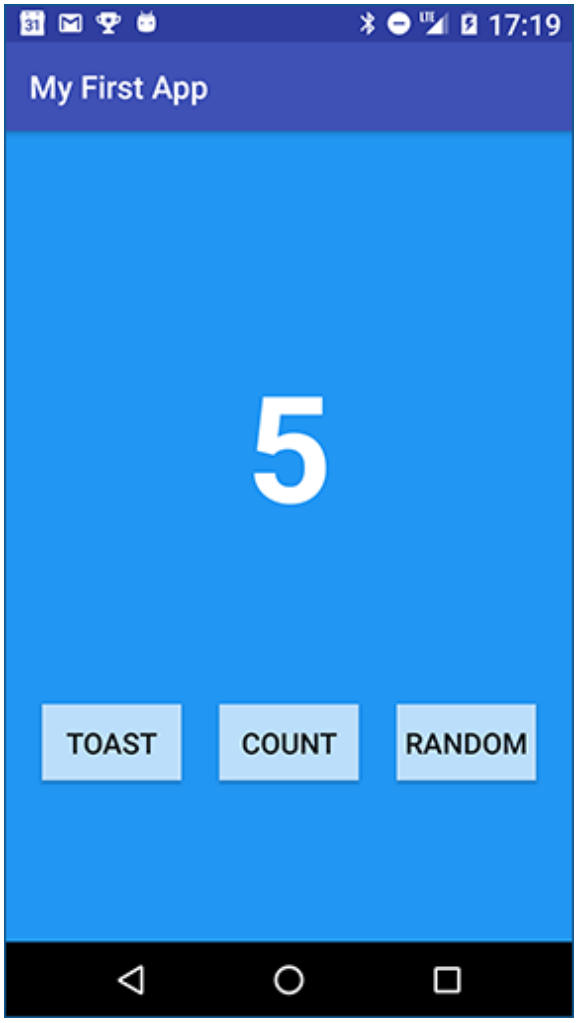
5. Запустите приложение и нажмите кнопку TOAST, вы должны увидеть на экране короткое сообщение с текстом “Hello Toast!”.

Таким образом, что для того, чтобы сделать элемент экрана интерактивным вам нужно:

- Реализовать функцию, определяющую поведение экранного элемента при нажатии на него. Эта функция должна быть public, не возвращать никаких значений, и принимать View в качестве аргумента.
- Установить имя функции в качестве значения свойства **onClick** в экранном элементе.

Реализация счетчика по нажатию кнопки Count

Функция, отображающая тост, очень проста. Она не взаимодействует с другими элементами экрана. На следующем шаге мы будем по нажатию кнопки находить и обновлять другие view-элементы экрана. Напишем функцию для кнопки Count, которая при нажатии будет увеличивать значение текстового поля на 1.



1. В файле макета экрана определим параметр id для TextView:

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

2. Отройте класс ActivityMain.kt и объявите переменную для TextView:

```
private lateinit var textView: TextView
```

3. В методе onCreate инициализируем её:

```
textView = findViewById(R.id.textView)
```

4. В теле класса ActivityMain.kt, добавим функцию countMe(). Эта функция будет вызываться при нажатии кнопки Count, поэтому она должна быть

public, не иметь возвращаемых значений, и получать View в качестве аргумента.

```
fun countMe (view: View) {
}
```

5. Получаем значение текстового поля TextView.

```
fun countMe (view: View) {
   // Get the value of the text view.
   val countString = textView.text.toString()
}
```

6. Конвертируем полученное значение в число, и увеличим его на единицу.

```
fun countMe (view: View) {
   // Get the value of the text view.
   val countString = textView.text.toString()
   // Convert value to a number and increment it
   var count: Int = Integer.parseInt(countString)
   count++
}
```

7. Отображаем новое значение в TextView.

```
fun countMe (view: View) {
   // Get the value of the text view.
   val countString = textView.text.toString()
   // Convert value to a number and increment it
   var count: Int = Integer.parseInt(countString)
   count++
   // Display the new value in the text view.
   textView.text = count.toString();
}
```

8. Функция готова. Теперь нужно вызвать ее при нажатии кнопки COUNT. Для этого нужно установить имя функции в качестве значения свойства**onClick** кнопки в файле макета.

Запустите приложение на устройстве.

При нажатии кнопки COUNT значение текстового поля увеличивается на единицу.