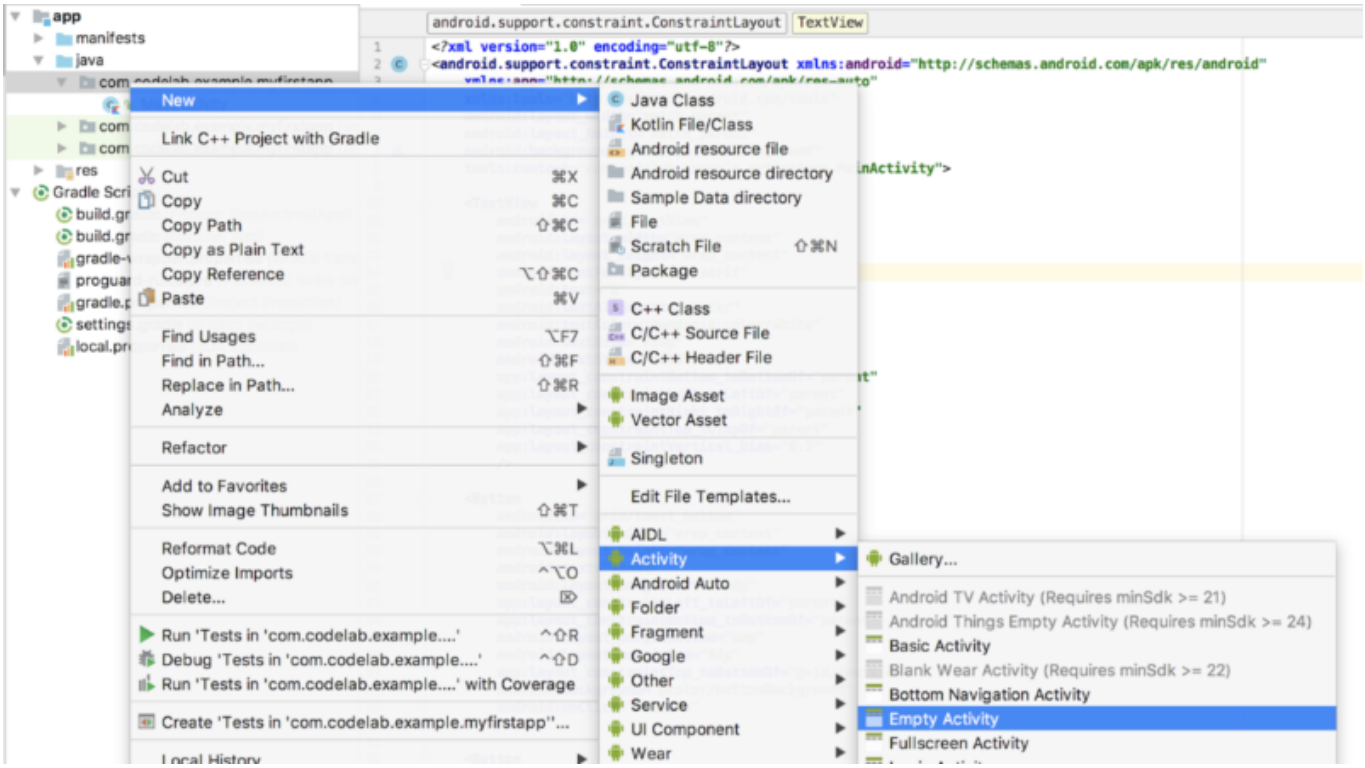
Вы узнаете

Как запустить новое активити с помощью интента

Как передать данные во второе активити через интент

Как получить данные, переданные через интент во втором активити

Создаем новое активити



1. Раскройте пакет с именем com.example.android.myfirstapp по пути **apps** > **java** > **com.example.android.myfirstapp**.
2. Правым кликом выберите команду **File** > **New** > **Activity** > **Empty Activity**.
3. В окне настроек нового активити установите для него имя SecondActivity. Убедитесь, что в поле Source Language установлен Kotlin.

Также проверьте им пакета, оно должно соответствовать вашему приложению. Если вы не выделили имя пакета в меню при вызове команды добавления нового активити, установите в это поле android.example.com.myfirstapp.

4. Нажмите **Finish**. Android Studio создаст файл на языке Kotlin и файл макета на языке XML для нового активити.

5. Дождитесь синхронизации gradle в Android Studio.Gradle это система сборки, которая используется в Android Studio для компиляции и постройки приложения. Вы будете видеть сообщения от gradle о прогрессе сборки приложения внизу экрана.

Изучите файл манифеста

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="android.example.com.myfirstapp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My First App"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondActivity"></activity>
    </application>

</manifest>
```

1. Откройте файл манифеста приложения по пути **app** > **manifests** > **AndroidManifest.xml**.Файл манифеста содержит информацию о приложении, включая список всех активити.
2. Обратите внимание на текст между тегами <activity> для MainActivity:Как видите, тег <activity> содержит параметр имени активити. MainActivity запускается после при старте приложения, поэтому включает дополнительные параметры, которые определяют его как стартовое активити.
3. Изучите тег <activity> для второго активити. Он содержит только параметр имени.

Каждое активити должно быть определено в файле манифеста. Вы можете добавить

новое активити в приложение не через меню добавления активити, но тогда вам нужно будет вручную прописать его в манифесте.

Изучите файл Kotlin **нового активити**

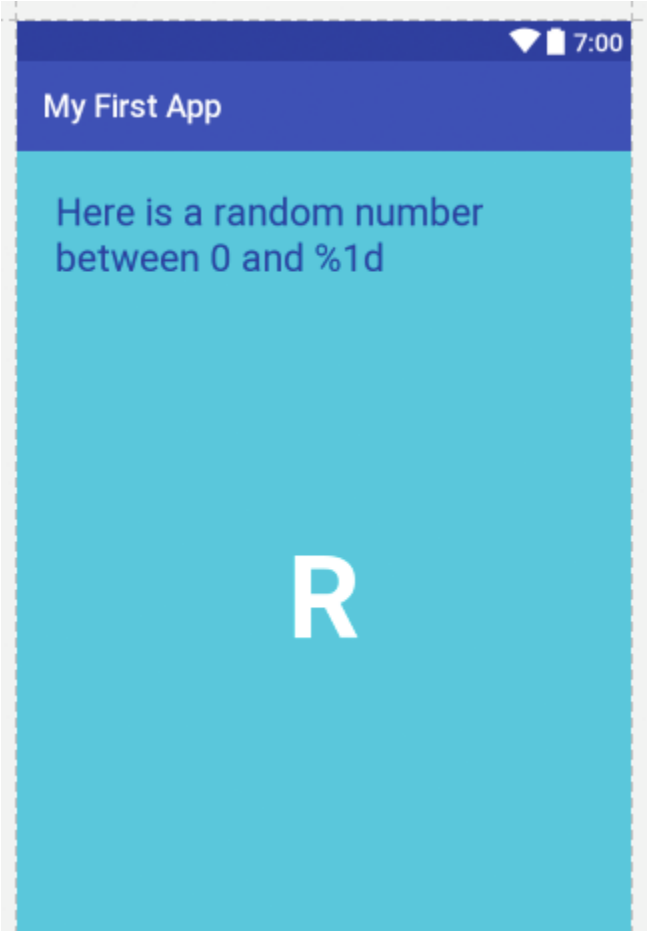
1. Откройте файл SecondActivity.kt.
2. Обратите внимание на метод onCreate(). Постмотрите на вызов setContentView(). Этот метод указывает файл макета activity_second.xml как разметку нового активити.

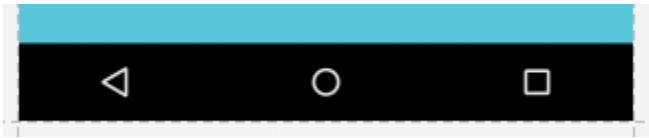
```
setContentView(R.layout.activity_second);
```

В этом активити мы добавим метод отображения случайного числа. Но сначала нам нужно определить поле в макете для его отображения.

Создаем макет для нового активити

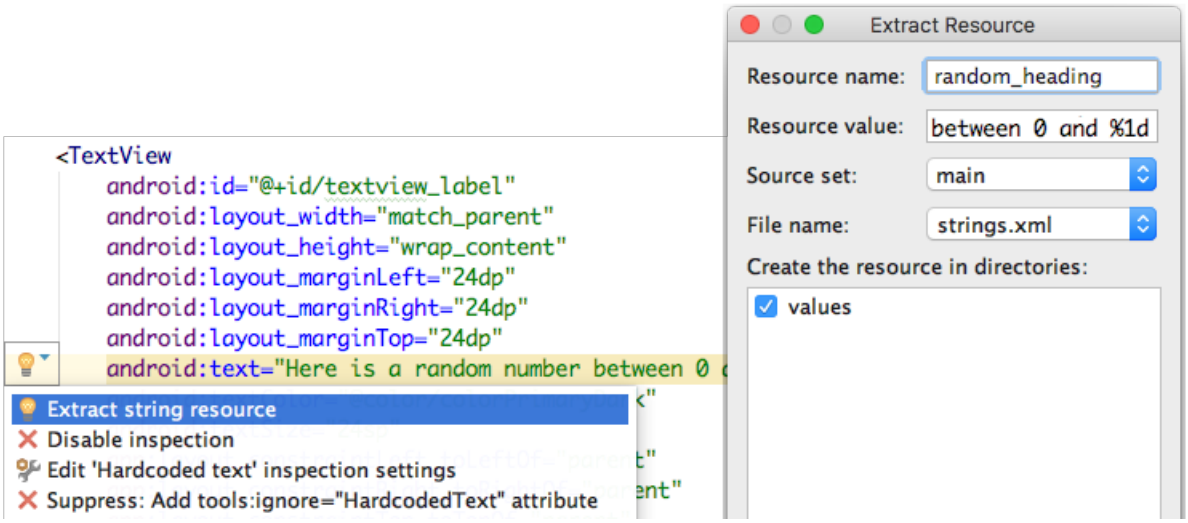
Экран нового активити будет отображать заголовок и случайный номер. На вкладке дизайн редактора макетов можно увидеть, как будет выглядеть экран:

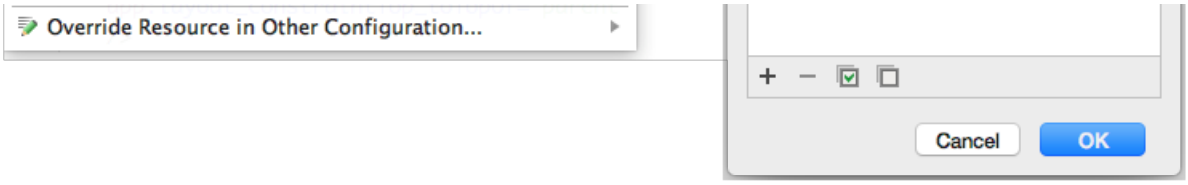




Добавьте TextView для заголовка

1. Откройте файл activity_second.xml. Вы увидите пустой ConstraintLayout. Это макет по умолчанию для шаблона Empty Activity.
2. Добавьте **TextView** из палитры компонентов. Этот TextView будет использовано для отображения заголовка вверху экрана.
3. Ограничение верхнего края TextView установим по верхней границе окна, левый край по левой стороне, а правый край по правой стороне экрана. Нижний край не ограничиваем.
4. Установите значение ширину **match_parent**, а высоту **wrap_content**, поэтому высота изменится в соответствии с высотой содержимого.
5. Установите идентификатор **@+id/textview_label**.
6. Установите верхний, левый и правый отступ **24dp**. Левый и правый отступы могут также иметь значение «start» и «end» для поддержки локализации языков с написанием справа налево.
7. Установите значение параметра цвета текста **@color/colorPrimaryDark** и размер текста **24sp**.
8. Поместите текст в TextView «**Это случайное число между 0 и %1d**.» Спецификатор **%1d будет заменен числом в процессе работы приложения.**
9. Поместите текст в ресурсы с именем **random_heading**.





Это код XML для TextView, которое отображает заголовок:

```
<TextView

android:id="@+id/textview_label"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_marginLeft="24dp"

android:layout_marginRight="24dp"

android:layout_marginTop="24dp"

android:text="@string/random_heading"

android:textColor="@color/colorPrimaryDark"

android:textSize="24sp"

app:layout_constraintLeft_toLeftOf="parent"

app:layout_constraintRight_toRightOf="parent"

app:layout_constraintTop_toTopOf="parent"

/>
```

Добавьте TextView для отображения случайного числа

1. Добавьте TextView для отображения случайного числа.

2. Установите значения параметров высоты и ширины **wrap_content**.
3. Этот TextView будет находиться ниже TextView заголовка. Ограничьте верхний край по нижнему краю TextView заголовка. Остальные ограничения установите по остальным сторонам экрана.
4. Установите идентификатор **@+id/textview_random**.
5. Установите значение верхнего, леаого и правого отступов **24dp**.
6. Установите значение параметров **textColor** в **@android:color/white**, **textSize** в **72sp**, и **textStyle** в **bold**.
7. Напишите букву **R** в текстовом поле. Этот текст является просто заменителем, пока не будет сгенерировано случайное число.
8. Установите значение параметра **layout_constraintVertical_bias** в **0.45**.Этот TextView имеет ограничения по всем краям, так что лучше использовать вертикальные смещения, чем отступы для регулировки вертикального положения, чтобы макет одинаково хорошо выглядел в разных ориентациях и размерах экрана.

Это код XML для TextView которое отображает случайный номер:

```
<TextView

android:id="@+id/textview_random"

android:text="R"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:textColor="@android:color/white"

android:textSize="72sp"

android:textStyle="bold"

app:layout_constraintTop_toBottomOf="@+id/textview_label"
```

```
app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintLeft_toLeftOf="parent"

app:layout_constraintRight_toRightOf="parent"

app:layout_constraintVertical_bias="0.45"

/>
```

Изменение цвета фона макета

Установите новому активити новый цвет фона, отличающийся от первого:

1. В файле colors.xml, добавьте новый цвет:

```
<color name="screenBackground2">#26C6DA</color>
```

2. В макете для второго активити, установите элементу ConstraintLayout новый цвет. Это можно сделать в панели свойств или в коде XML:

```
android:background="@color/screenBackground2"
```

Макет для второго активити готов. Вы можете посмотреть его на вкладке Дизайн.

Теперь пришло время реализовать переход на второй экран.

Запуск второго активити

Интент – это объект для обмена между активити, который абстрактно представляет собой намерение выполнить какое-либо действие. В основном интенты используются для запуска активити. Как только интент отправляется, его получает система Android, и считывает информацию в нем. Для открытия второго экрана, нам нужно создать и отправить объект Intent с указанием активити, которое нужно открыть. Затем вызвать метод startActivity() с передачей объекта Intent, который отправит это сообщение фреймворку Android, который откроет это активити.

Выполните такие шаги:

1. Откройте класс MainActivity.kt.
2. Добавьте метод randomMe(), который будет вызываться, когда нажата кнопка Random.Если вы забыли, как реализовать такой метод, посмотрите на код методов toastMe() и countMe().
3. Добавьте код для запуска второго активити в тело метода randomMe():

```
// Create an Intent to start the second activity

val randomIntent = Intent(this, SecondActivity::class.java)



// Start the new activity.

startActivity(randomIntent)
```

4. Что еще нужно сделать, чтобы связать метод randomMe() с кнопкой Random? Сделайте это самостоятельно. Ищите подсказки в прошлом уроке.

5. Запустите приложение. Нажмите кнопку Random. Второе активити откроется, но оно пока не показывает случайный номер.

Передача информации между активити

На втором экране при открытии должно отображаться случайное число из диапазона от нуля до текущего значения счетчика на первом экране. Для этого второму активити требуется текущее значение счетчика. Мы можем передать эту информацию в интенте, используя метод putExtra(). Второе активити может получить интент, который запустил его, и извлечь данные Extra Data из этого интента.

Обновите MainActivity для передачи дополнительных данных второму активити

1. В классе SecondActivity.kt, объявите статическую переменную TOTAL_COUNT, для использования ее в качестве ключа для Extra Data.

```
companion object {

const val TOTAL_COUNT = "total_count"
```

```
}
```

2. Изменим метод randomMe(). Сначала получим текущее значение счетчика из TextView. Конвертируем значение в тип int, затем с помощью метода putExtra() добавим значение в Intent. Метод putExtra() принимает ключ и значение; в качестве ключа укажем константу TOTAL_COUNT из SecondActivity.kt.

```
randomIntent.putExtra(SecondActivity.TOTAL_COUNT, count)
```

Вот полный код метода randomMe():

```
fun randomMe (view: View) {

// Create an Intent to start the second activity

val randomIntent = Intent(this, SecondActivity::class.java)



// Get the current value of the text view.

val countString = textView.text.toString()



// Convert the count to an int

val count = Integer.parseInt(countString)



// Add the count to the extras for the Intent.

randomIntent.putExtra(SecondActivity.TOTAL_COUNT, count)



// Start the new activity.
```

```
startActivity(randomIntent)

}
```

Обновите SecondActivity для вычисления и отображения случайного числа

Мы написали код для отправки текущего значения счетчика во второе активити. На следующем шаге настроим SecondActivity.kt для получения и использования значения счетчика.

Напишем метод для извлечения значения счетчика и вычисления на его основе случайного числа

1. В классе SecondActivity.kt, добавим метод showRandomNumber() без аргументов.
2. Пропишем в этом методе генерацию и отображение случайного числа, максимальное значение которого соответствует значению счетчика.
3. Используем метод getIntent() для получения интента, запустившего это активити. Используем метод getIntExtra() для получения extra data из интента; аргументами являются ключ или значение по умолчанию при его отсутствии.

Внимание: при получении данных из интента, вы должны знать их тип. Используйте соответствующий getter метод, например getStringExtra() или getIntExtra().

Полный код метода showRandomNumber():

```
//Other class code...



fun showRandomNumber() {

   // Get the count from the intent extras

   val count = intent.getIntExtra(TOTAL_COUNT, 0)
```

```
    // Generate the random number

    val random = Random()

    var randomInt = 0

    // Add one because the bound is exclusive

    if (count > 0) {

        // Add one because the bound is exclusive

        randomInt = random.nextInt(count + 1)

    }



    // Display the random number.

    textview_random.text = Integer.toString(randomInt)



    // Substitute the max value into the string resource

    // for the heading, and update the heading

    textview_label.text = getString(R.string.random_heading, count)

}
```

Обновите метод onCreate()

Метод onCreate() вызывается при старте активити. Текущий метод onCreate() для SecondActivity устанавливает макет разметки для экрана. Добавим вызов метода showRandomNumber() в метод onCreate(), он будет обновлять текстовое поле для отображения случайного числа.

1. Изменим метод onCreate() в SecondActivity. Добавим в конце вызов метода showRandomNumber().

```
override fun onCreate(savedInstanceState: Bundle?) {

    super.onCreate(savedInstanceState)

    setContentView(R.layout.activity_second)

    showRandomNumber()

}
```

2. Запустим приложение. Нажмем кнопку счетчика на некоторое время, затем нажмем кнопку Random button. В новом окне должно отобразиться случайное число.