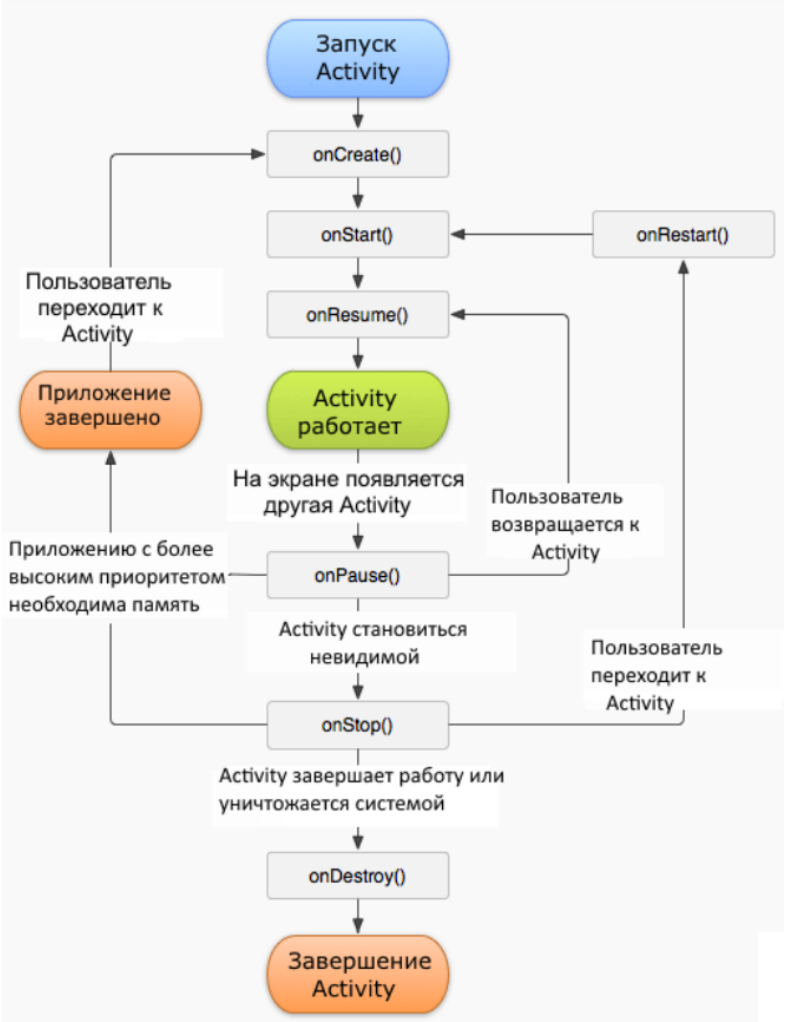
**Часть 1. Активность**

Активность в Android — это основной компонент приложения, который предоставляет интерфейс для взаимодействия пользователя с приложением. Она действует как одно «окно» в пользовательском интерфейсе, через которое пользователь может взаимодействовать с приложением, например, вводить данные, просматривать информацию или выполнять другие действия.

Каждая активность обычно заполняется различными элементами управления, такими как кнопки, текстовые поля, изображения и другие виджеты, которые обеспечивают функциональность приложения. В Android каждая активность имеет свой жизненный цикл, который управляется операционной системой.




Первым методом в жизненном цикле активности является "**onCreate()**". Этот метод вызывается при первоначальном создании активности. Здесь разработчики обычно размещают код для инициализации, который нужно выполнить один раз: настройка пользовательского интерфейса, инициализация данных и состояния активности. После "**onCreate()**", активность переходит в состояние "**onStart()**".

Метод "**onStart()**" вызывается, когда активность становится видимой для пользователя. В этом методе можно производить подготовительные действия, чтобы активность была готова к взаимодействию с пользователем. После "**onStart()**", следующим шагом в жизненном цикле является "**onResume()**".

"**onResume()**" активируется, когда активность готова начать взаимодействие с пользователем. На этом этапе активность находится на переднем плане и может принимать пользовательский ввод. Это идеальное место для запуска анимаций или выполнения вычислений, необходимых для интерфейса.

Когда активность переходит в фоновый режим, вызывается метод "**onPause()**". Этот метод используется для приостановки операций, которые не должны продолжаться, пока активность не находится на переднем плане. Это включает в себя остановку анимаций, приостановку выполнения сложных вычислений или освобождение системных ресурсов.

Если активность становится полностью невидимой для пользователя, система вызывает "**onStop()**". В этом методе следует останавливать более сложные операции, которые ненужны или неэффективны, когда активность не видна. Если активность снова станет видимой, вызывается "**onRestart()**", что означает возврат к "**onStart()**".

Метод "**onRestart()**" используется для перезапуска операций, которые были остановлены или приостановлены в "**onStop()**". Он подготавливает активность к повторному запуску и восстановлению её работы.

Наконец, "**onDestroy()**" вызывается при уничтожении активности. Этот метод используется для окончательной очистки ресурсов и сохранения данных, если это необходимо. Это последний вызов в жизненном цикле активности, после которого активность полностью уничтожается системой.

Для того чтобы лучше разобраться в работе методов жизненного цикла применим логирование.

**Часть 2. Логирование**

Логирование — это ключевой аспект разработки приложений, который позволяет разработчикам отслеживать работу приложения и диагностировать возможные проблемы. В Android предусмотрены два основных способа логирования: использование класса Log и отображение всплывающих сообщений с помощью Toast.

Класс Log в Android используется для записи отладочной информации.

Log предоставляет различные уровни логирования, такие как DEBUG (отладка), ERROR (ошибка), INFO (информация), VERBOSE (подробности) и WARN (предупреждение) и запоминается по первым буквам, позволяя разработчикам классифицировать важность сообщений.

Например, запись лога может выглядеть так:

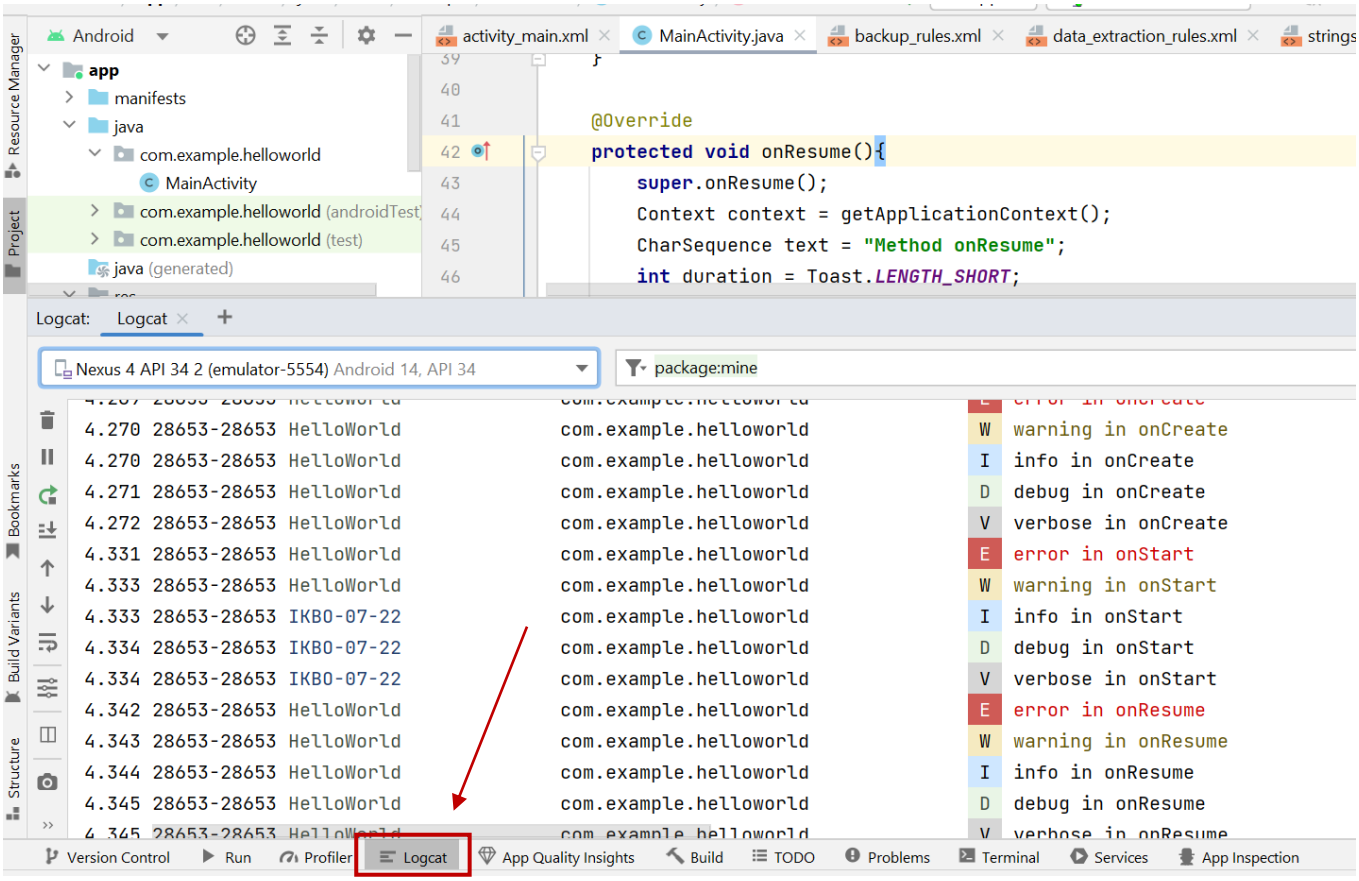
```
Log.i("MyAppTag", "Информационное сообщение");
```

где "MyAppTag" — это пользовательский тег для идентификации сообщений из приложения, а "Информационное сообщение" — это текст самого сообщения.

Пользовательский тег также можно определить один для всего приложения через перемененную.

```
private static final String TAG = "HelloWorld";
@Override
protected void onStart(){
    super.onStart();
    Log.e(TAG, msg: "error in onStart");
    Log.w(TAG, msg: "warning in onStart");
    Log.i( tag: "IKBO-07-22", msg: "info in onStart");
    Log.d( tag: "IKBO-07-22", msg: "debug in onStart");
    Log.v( tag: "IKBO-07-22", msg: "verbose in onStart");
}
```

Сообщения, залогированные с помощью Log, выводятся в панели Logcat, инструмент для просмотра логов, доступный в среде разработки Android Studio.

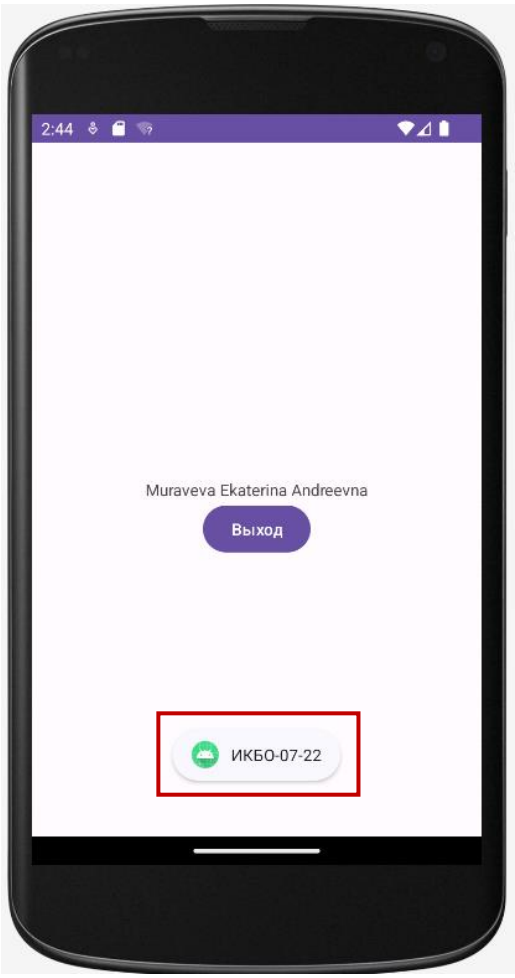



С другой стороны, Toast в Android предназначен для отображения коротких всплывающих сообщений пользователю. Эти сообщения автоматически исчезают после небольшого промежутка времени и не требуют активного взаимодействия пользователя. Toast является эффективным способом для отображения простых уведомлений, таких как подтверждение выполнения какого-либо действия. Пример использования Toast может выглядеть так:

```
Toast.makeText(getApplicationContext(), "Текст сообщения",
Toast.LENGTH_SHORT).show();
```

где "Текст сообщения" — это содержание уведомления, а Toast.LENGTH_SHORT указывает на короткую продолжительность его отображения.

```
@Override
protected void onResume(){
    super.onResume();
        Toast.makeText(getApplicationContext(), text: "ИКБО-07-22", Toast.LENGTH_SHORT).show();
}
```




Главное отличие между Log и Toast заключается в их целях и способах использования. Log ориентирован на логирование информации для разработчиков и отладку приложения, при этом сообщения Log видны только в Logcat и не отображаются в пользовательском интерфейсе приложения. В отличие от этого, Toast предназначен для взаимодействия с пользователем, показывая короткие информационные сообщения непосредственно в интерфейсе приложения. Также важно отметить, что сообщения Log могут сохраняться в Logcat в течение длительного времени, в то время как Toast отображается только на ограниченный период и затем автоматически исчезает.

**Часть 3. Взаимодействие с элементами пользовательского интерфейса**

Взаимодействие с элементами пользовательского интерфейса в Android-приложениях, осуществляется посредством специализированных методов и механизмов. Задать метод обработчик события можно двумя способами:

- декларативно, при помощи атрибута кнопки onClick;
- программно, в коде вашего приложения используя метод, устанавливающий обработчик событий для кнопки setOnClickListener().

При программном варианте задания обработчика события два основных метода, которые играют ключевую роль в этом процессе, это "**findViewById**" и "**setOnClickListener**".

**findViewById** — это метод, используемый для получения ссылки на виджет по его идентификатору. Каждый элемент интерфейса в XML-разметке имеет уникальный ID, который используется в коде для обращения к этому элементу.

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/my_button"
    android:text="Следующая страница"
    android:onClick="onNextActivity"
    app:layout_constraintTop_toBottomOf="@id/textView"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"/>
```

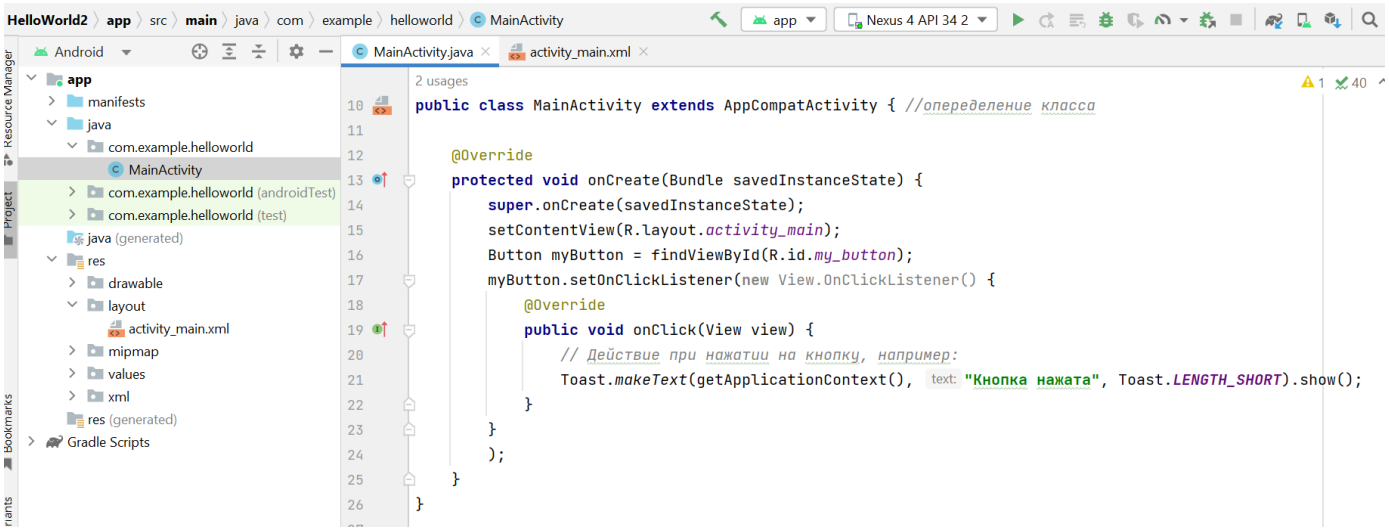
Например, если в XML есть кнопка с идентификатором @+id/my_button, то для получения ссылки на эту кнопку в Java используется следующий код:

```
    Button myButton = findViewById(R.id.my_button);
```

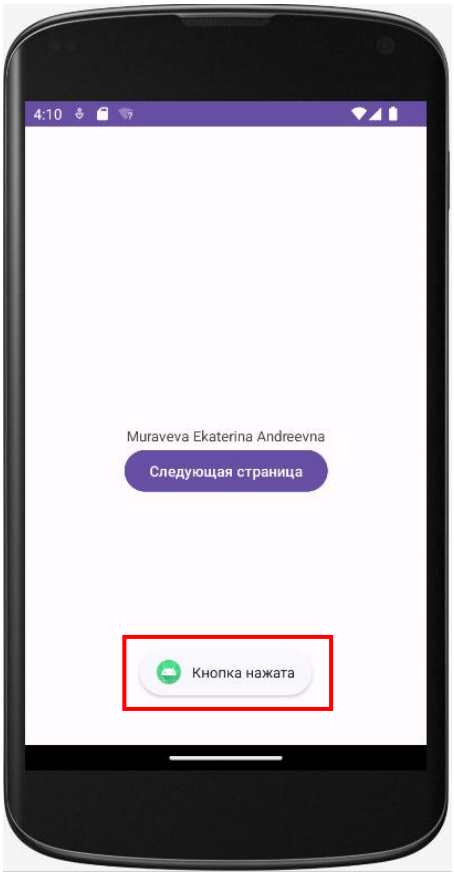
После получения ссылки на элемент интерфейса можно взаимодействовать с ним, изменяя его свойства, вызывая методы и так далее.

В свою очередь, "**setOnClickListener**" — это метод, который используется для установки обработчика нажатий на виджет, например, на кнопку. После того как кнопка найдена с помощью "findViewById", можно установить для неё слушатель нажатий. Этот слушатель определяет действия, которые будут выполняться при нажатии на кнопку, например:

```
    myButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Действие при нажатии на кнопку, например:
            Toast.makeText(getApplicationContext(), "Кнопка нажата",
Toast.LENGTH_SHORT).show();
        }
    });
```




В этом примере при нажатии на кнопку на экране появится короткое сообщение Toast с текстом «Кнопка нажата».




Таким образом, "setOnClickListener" позволяет определить интерактивное поведение элементов пользовательского интерфейса, реагирующее на действия пользователя.

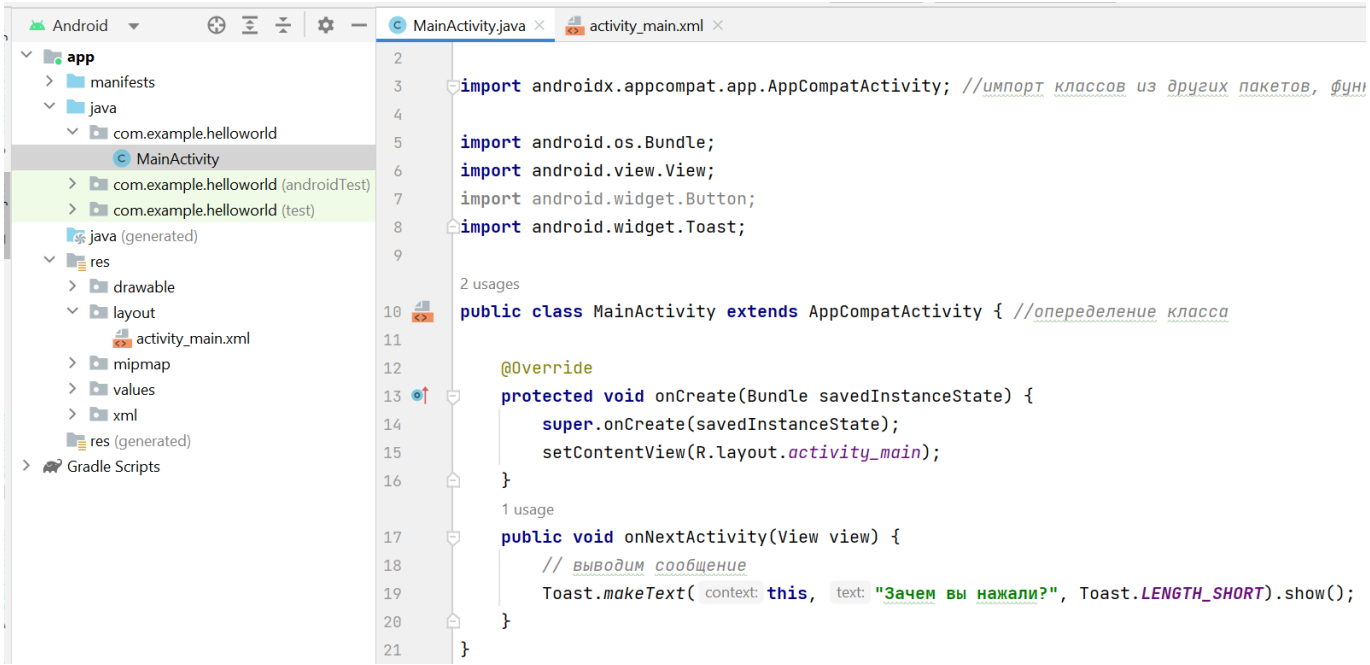
Так же одним из вариантов того, что можно разместить в слушателе нажатий, является переход между разными экранами в приложении.

Относительно новый способ для взаимодействия с элементами пользовательского интерфейса, специально разработанный для Android - использовать атрибут **onClick**:

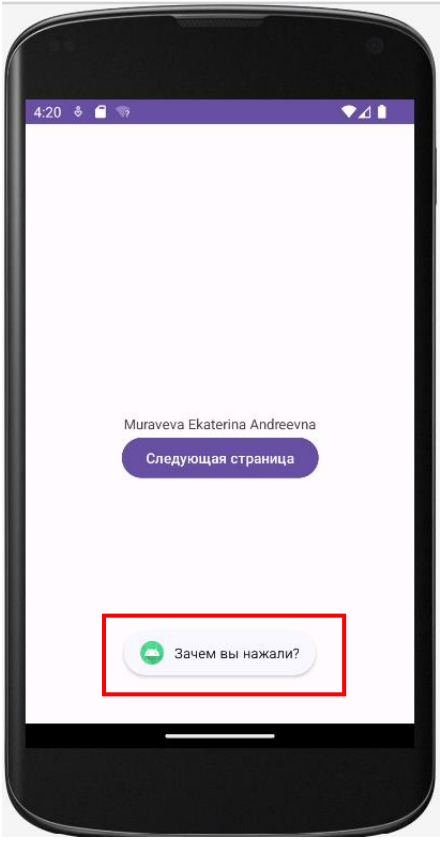
```
android:onClick="onNextActivity"
```

Имя для события можно выбрать произвольное, но лучше не выпендриваться. Далее нужно прописать в классе активности придуманное вами имя метода, который будет обрабатывать нажатие. Метод должен быть открытым (public) и с одним параметром, использующим объект **View**.

```
public void onNextActivity(View view) {
    // выводим сообщение
Toast.makeText(this, "Зачем вы нажали?", Toast.LENGTH_SHORT).show();
}
```



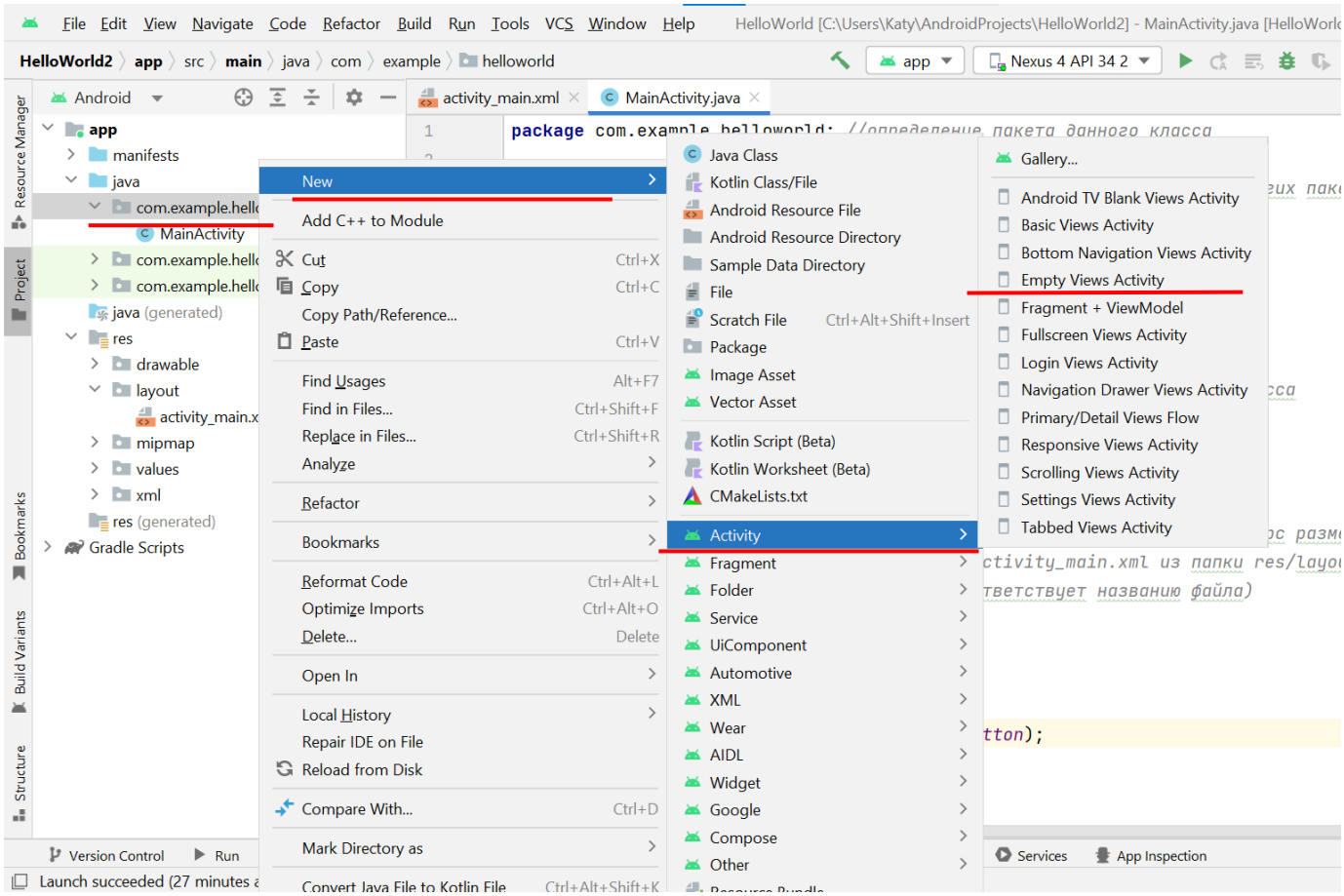

Когда пользователь нажимает на кнопку, то вызывается метод **onNextActivity()**, который в свою очередь генерирует всплывающее сообщение.

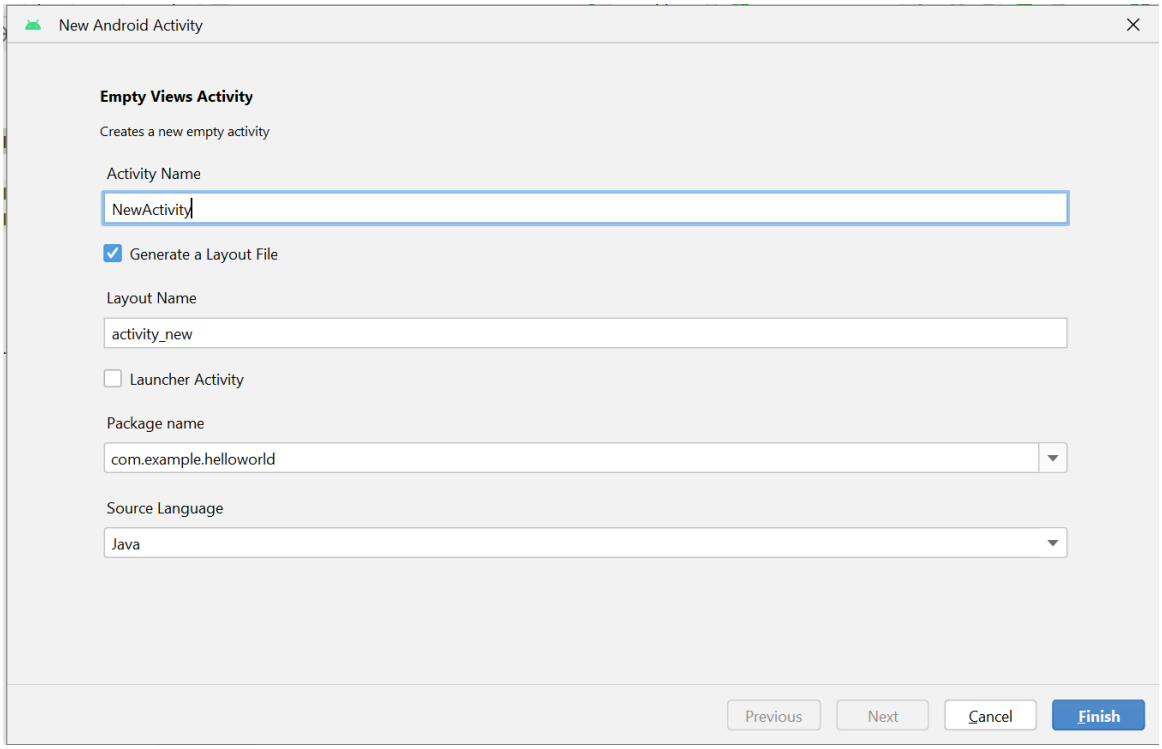


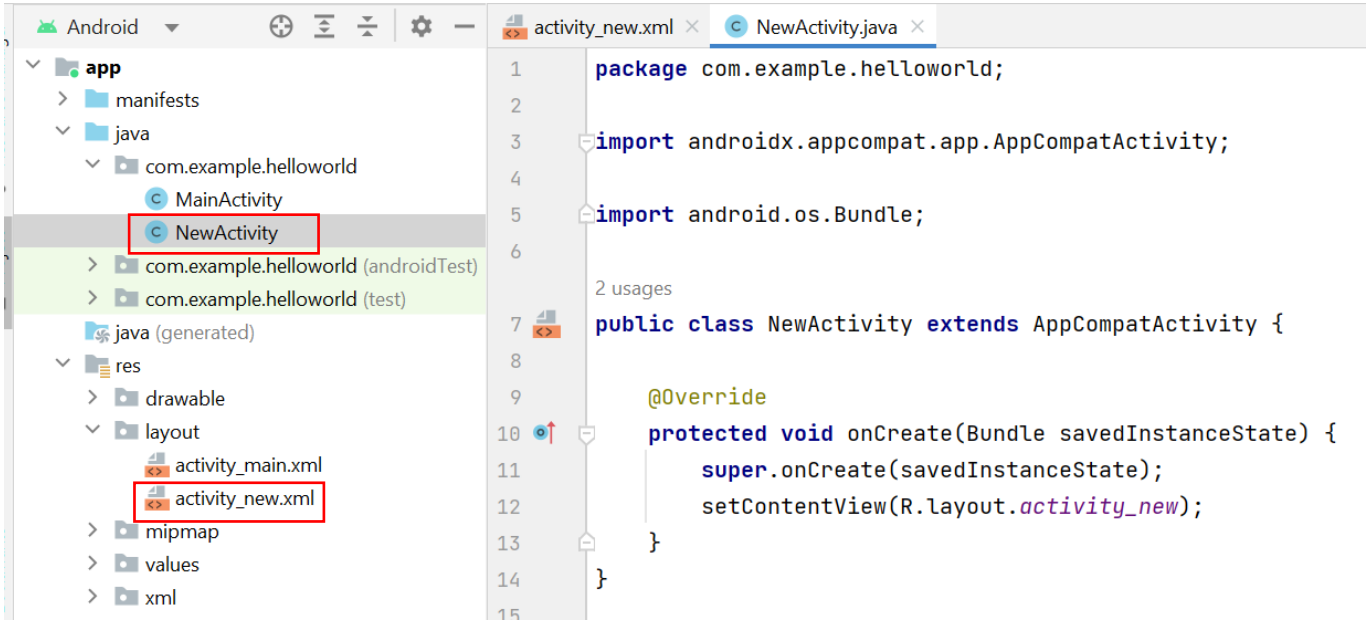

Обратите внимание, что при подобном подходе вам не придётся даже объявлять кнопку через конструкцию **Button myButton = findViewById(R.id.my_button)**, так как Android сама поймёт, что к чему. Данный способ применим не только к кнопке, но и к другим элементам и позволяет сократить количество строк кода.

**Часть 4. Переход между экранами**

В Android-приложениях часто используется несколько активностей, каждая из которых представляет собой отдельный экран или пользовательский интерфейс. Для создания новой активности в Android Studio необходимо выполнить несколько шагов. Во-первых, в проекте нужно создать новый класс активности. Это можно сделать, нажав на модуль **"app"**, затем в верней панели выбрав **"File"** → **"New"** → **"Activity"** и затем выбрав тип активности, например **"Empty Views Activity"** либо нажать правой кнопкой мыши на **"java"**→ **"New"** → **"Activity"**. Android Studio сгенерирует необходимые файлы и ресурсы, такие как файл разметки XML для интерфейса и Java-класс для логики активности.


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
HelloWorld [C:\Users\Katy\AndroidProjects\HelloWorld2] - MainActivity.java [HelloWorld
HelloWorld2 app src main java com example helloworld
app
Nexus 4 API 34 2
Android
activity_main.xml
MainActivity.java
app
manifests
java
com.example.hello
MainActivity
java (generated)
res
drawable
layout
mipmap
values
xml
res (generated)
Gradle Scripts
New
Add C++ to Module
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Analyze
Refactor
Bookmarks
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Open In
Local History
Repair IDE on File
Reload from Disk
Compare With... Ctrl+D
Mark Directory as
Java Class
Kotlin Class/File
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Package
Image Asset
Vector Asset
Kotlin Script (Beta)
Kotlin Worksheet (Beta)
CMakeLists.txt
Activity
Fragment
Folder
Service
UiComponent
Automotive
XML
Wear
AIDL
Widget
Google
Compose
Other
Gallery...
Android TV Blank Views Activity
Basic Views Activity
Bottom Navigation Views Activity
Empty Views Activity
Fragment + ViewModel
Fullscreen Views Activity
Login Views Activity
Navigation Drawer Views Activity
Primary/Detail Views Flow
Responsive Views Activity
Scrolling Views Activity
Settings Views Activity
Tabbed Views Activity
Version Control
Run
Services
App Inspection
Launch succeeded (27 minutes a


New Android Activity
Empty Views Activity
Creates a new empty activity
Activity Name
NewActivity
Generate a Layout File
Layout Name
activity_new
Launcher Activity
Package name
com.example.helloworld
Source Language
Java
Previous
Next
Cancel
Finish

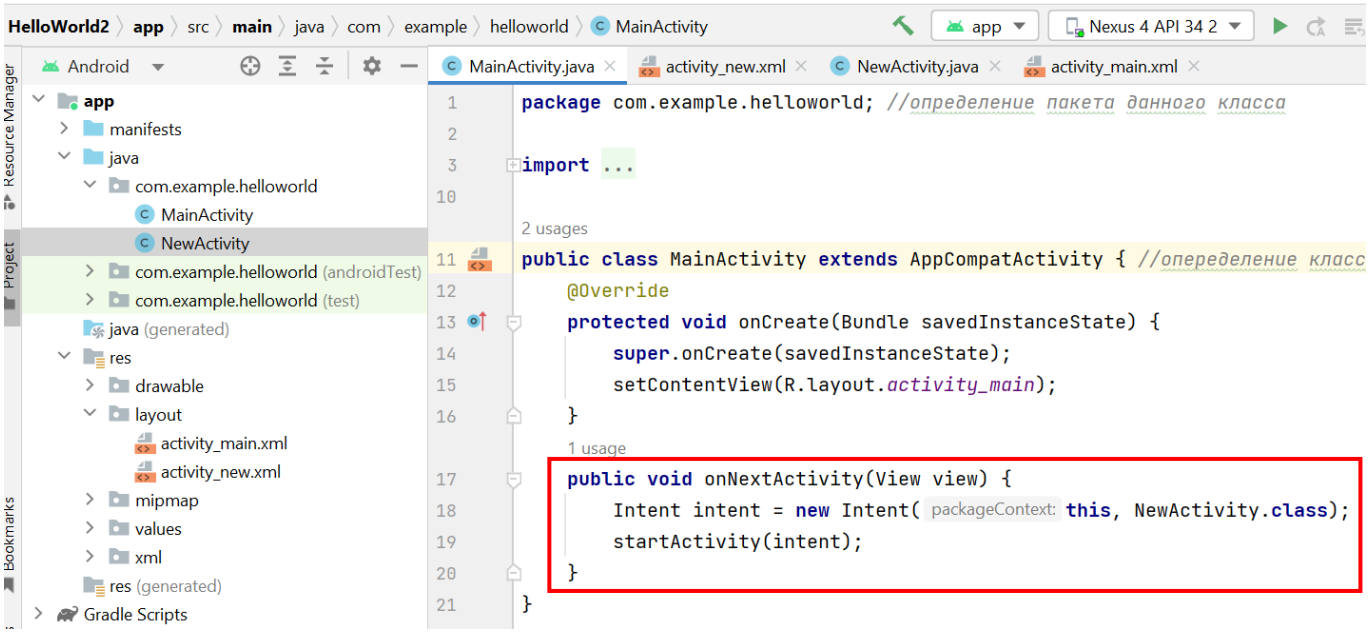



Переход между разными экранами в приложениях Android осуществляется при помощи механизма, называемого **"Intent"**(или намерение). Это основная концепция в Android, которая играет важную роль в обеспечении взаимодействия между различными компонентами приложения, в том числе активностями, которые представляют собой отдельные экраны в приложении.

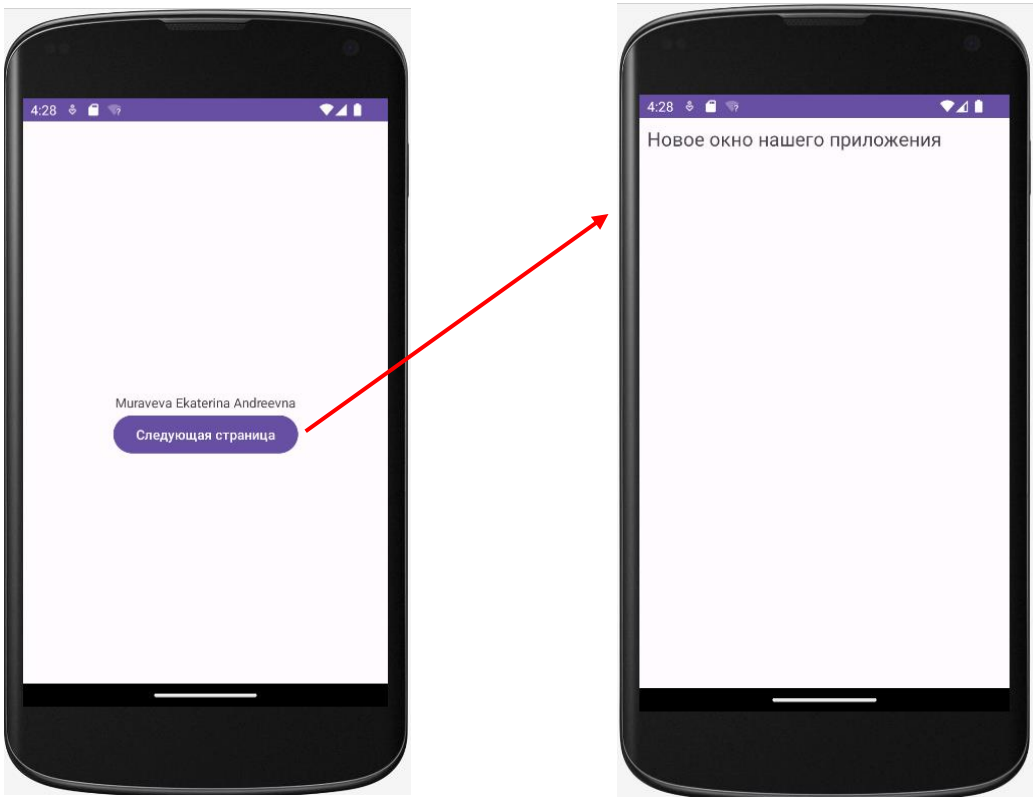
Intent можно описать как сообщение или запрос, который указывает на намерение выполнить определенное действие. В контексте перехода между экранами, Intent используется для запуска новой активности. Он не только указывает системе, что нужно перейти на другой экран, но и может переносить информацию от одного экрана к другому. Например, если приложение содержит список элементов и детальный просмотр каждого элемента на отдельном экране, Intent может использоваться для запуска активности детального просмотра и передачи данных о выбранном элементе.

Создание Intent включает в себя указание контекста (например, текущей активности) и класса активности, которую необходимо запустить. Пример создания Intent для запуска новой активности выглядит следующим образом:

```
Intent intent = new Intent(this, NewActivity.class);
startActivity(intent);
```




Здесь "this" обозначает текущую активность, а "NewActivity" — созданную ранее активность, которую нужно запустить, далее метод "startActivity", начинает выполнение действия, определенного в объекте Intent. По выполнении этой строки, система Android обрабатывает Intent и запускает активность, указанную в нем. Это приводит к переключению пользовательского интерфейса с текущей активности на интерфейс новой активности "NewActivity".




Таким образом, использование нескольких активностей и механизма Intent позволяет создавать многоуровневые и взаимосвязанные пользовательские интерфейсы в Android-приложениях.

**Часть 5. Передача данных между Activity**

Простой запуск Activity может быть недостаточен для разработки целых приложений и в определенный момент может понадобиться передать некоторые данные как в открываемое Activity, так и обратно.

В этом случае нужно задействовать специальную область **extraData**, который имеется у класса Intent.

Область **extraData** – это список пар ключ/значение, который передаётся вместе с намерением. В качестве ключей используются строки, а для значений можно использовать любые примитивные типы данных: String, int, float, double, long, short, byte, char; массивы примитивов, объекты класса Bundle и др.

Для передачи данных в другую активность используется метод putExtra():

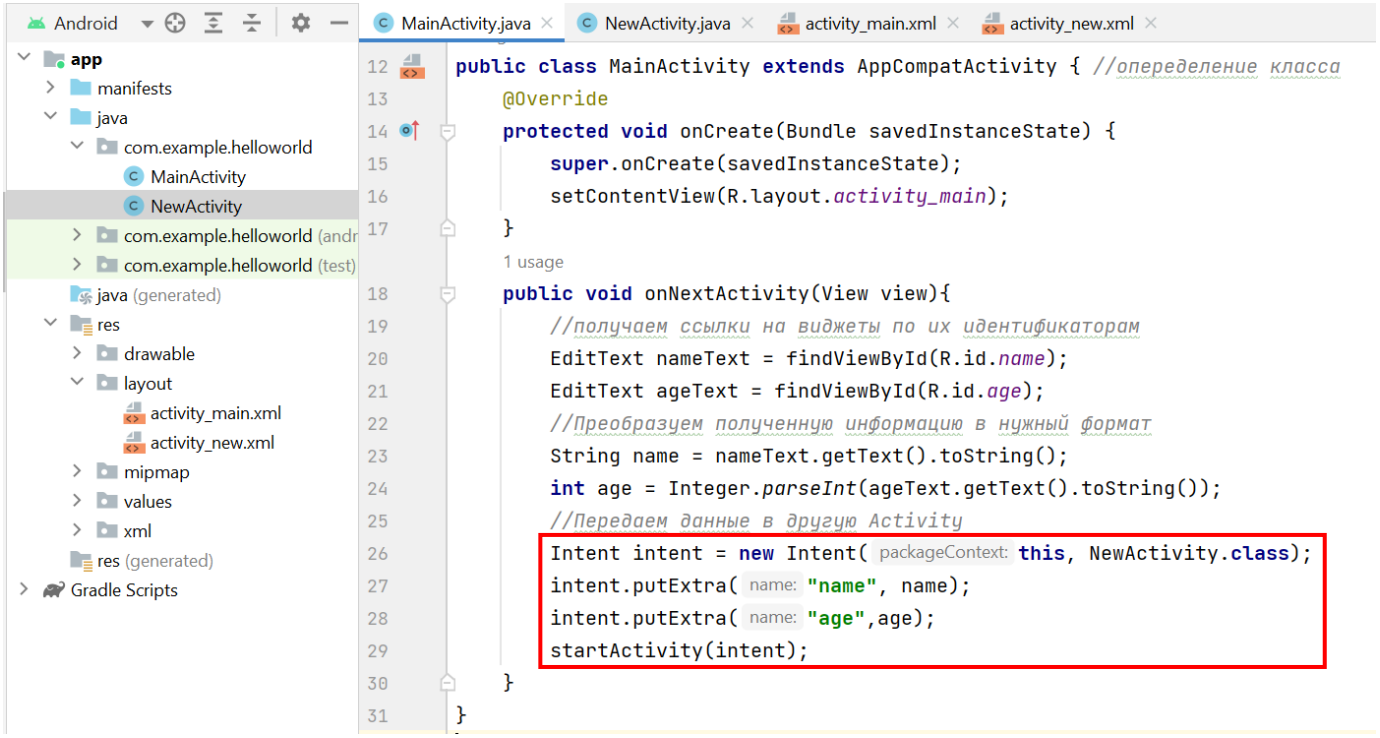
```
intent.putExtra("Ключ", "Значение");
```

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

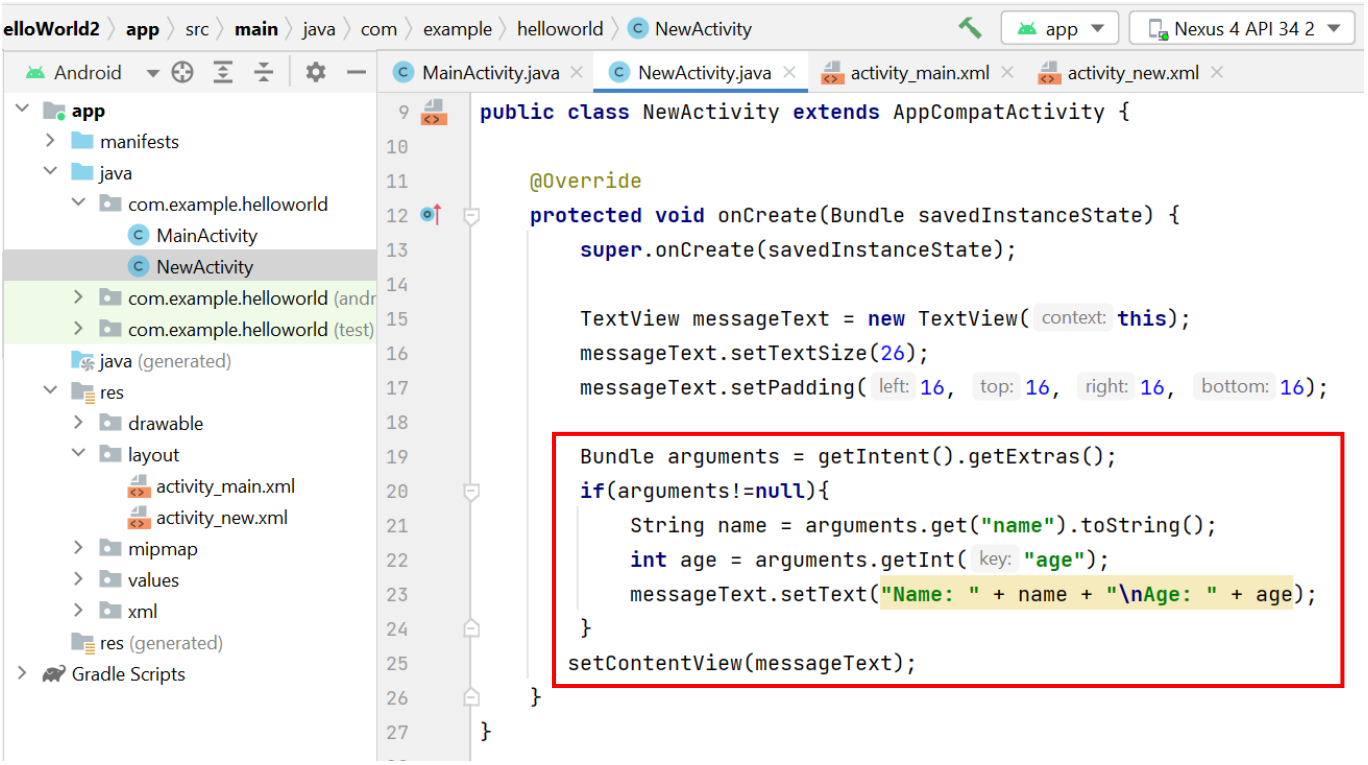
```
Bundle arguments = getIntent().getExtras();
String name = arguments.get("Ключ").toString();
```

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

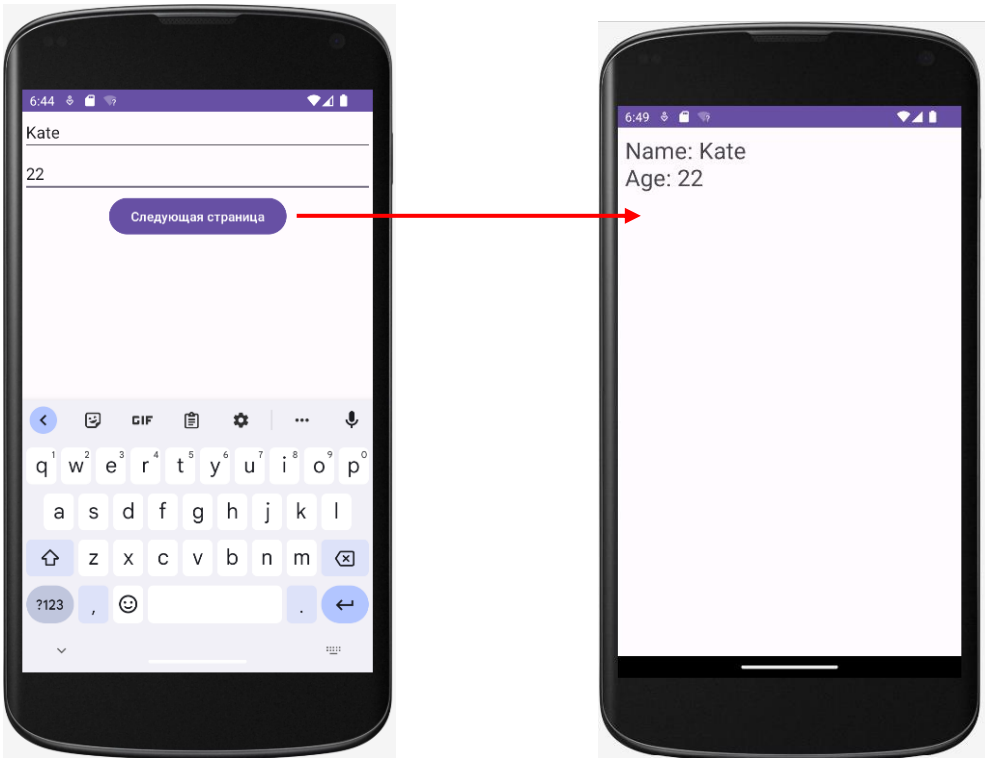
Пусть у нас имеется разметка для ввода двух полей: имени и возраста. Создадим поля и присвоим им идентификаторы name и age соответственно. После этого в файле MainActivity с помощью метода putExtra передаем данные в другую Activity.




А в другой NewActivity наоборот получаем отправленные данные с помощью метода get(). Также проверяем, что аргументы имеют хотя бы один символ и не являются пустыми.




В результате передаем введенные данные из одного окна в другое.




**Задание**

1. Отследить работу жизненного цикла активности при помощи логирования на всех этапах жизненного цикла: **onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy()** (сделать в каждом методе свою запись в логе, проверить работу логов в LogCat).
2. Реализовать переход на другую активность по нажатии на кнопку двумя способами: декларативно и программно.
3. Создать второе Activity. Произвести открытие второй Activity с передачей в нее данных. В качестве передаваемых данных использовать поля для ввода ФИО, номера группы, возраста, оценки, которую хотите получить за практику.

**Источники**

1) https://developer.alexanderklimov.ru/android/layout/constraintlayout.php?ysclid=lskrkw0gjj526736941
2) https://metanit.com/java/android/3.2.php?ysclid=lskrmxb4r48479533
3) https://metanit.com/java/android/3.3.php
4) https://metanit.com/java/android/3.5.php
5) https://developer.android.com/develop/ui/views/layout/declaring-layout

6) https://green-willow.ru/item/669-android-select-activity?ysclid=lskrpd7fhs638441914