МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Отчет по практической работе №2

по дисциплине «Разработка мобильных приложений»

Выполнили:
Студенты группы ИКБО-20-23 Комисарик М.А.

Проверил:
Старший преподаватель кафедры МОСИТ Шешуков Л.С.

Москва 2025 г.

СОДЕРЖАНИЕ

1 ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

1.1 Activity

Activity 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 activity обычно заполняется различными элементами управления, такими как кнопки, текстовые поля, изображения и другие виджеты, которые обеспечивают функциональность приложения. В Android каждая activity имеет свой жизненный цикл, который управляется операционной системой (Рисунок 1).

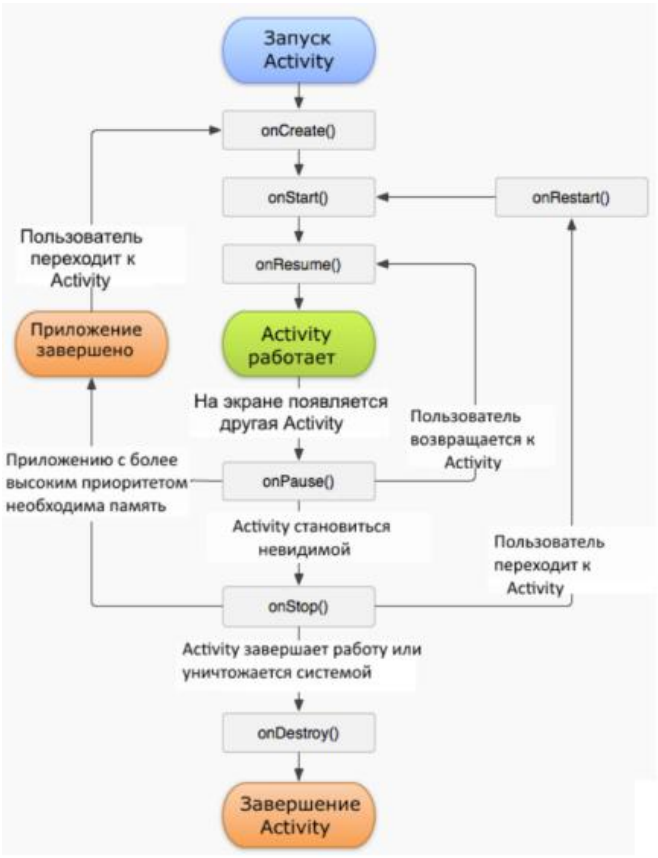

Рисунок 1 – Жизненный цикл activity

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

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

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

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

Если activity становится полностью невидимой для пользователя, система вызывает "onStop()". В этом методе следует останавливать более сложные операции, которые ненужны или неэффективны, когда activity не видна. Если activity 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 activity к повторному запуску и восстановлению её работы.

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

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

1.2 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. Приведём пример записи лога информационного сообщения с пользовательским тегом "MyAppTag" для идентификации (Рисунок 2).

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

Рисунок 2 – Пример записи лога

Пользовательский тег также можно определить один для всего приложения через перемененную (Рисунок 3).

```
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");
}
```

Рисунок 3 – Определение пользовательского тега для всего приложения

Сообщения, залогированные с помощью Log, выводятся в панели Logcat, инструменте для просмотра логов, доступном в среде разработки Android Studio (Рисунок 4).

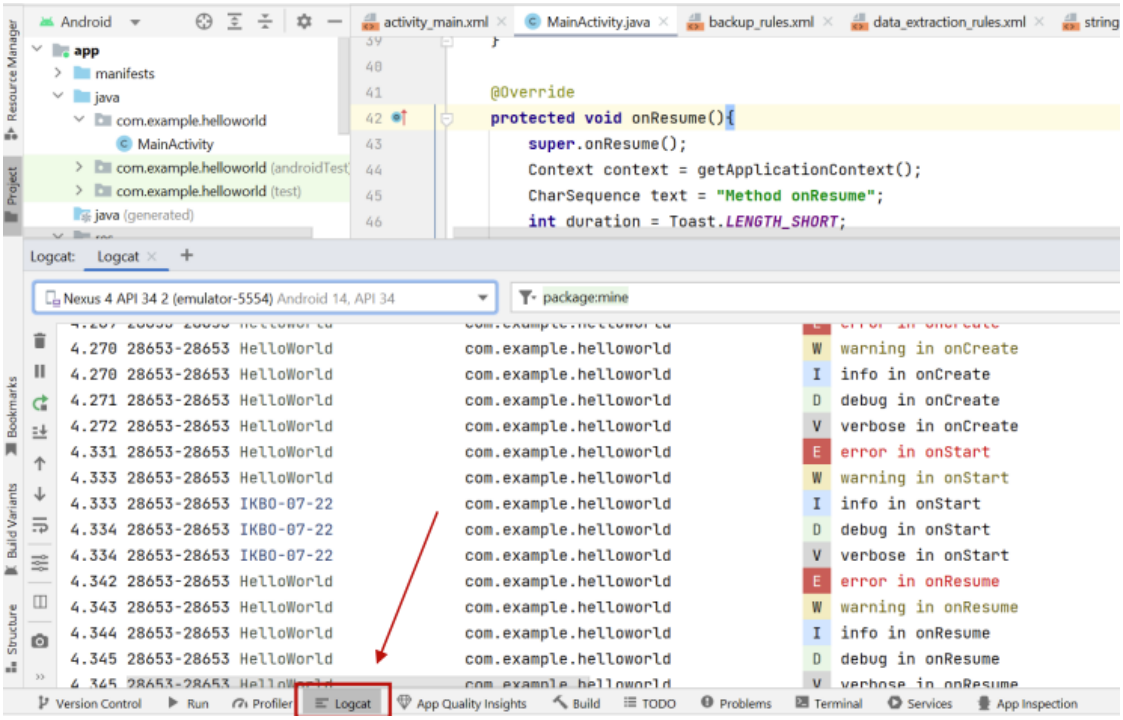

Рисунок 4 – Панель Logcat

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. Приведём пример использования Toast, где где "Текст сообщения" — это содержание уведомления, а Toast.LENGTH_SHORT указывает на короткую продолжительность его отображения (Рисунок 5).

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

Рисунок 5 – Пример создания сообщения с помощью Toast

Приведём еще один пример создания и отображения Toast (Рисунки 6-7).

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

Рисунок 6 – Создание сообщения с помощью Toast

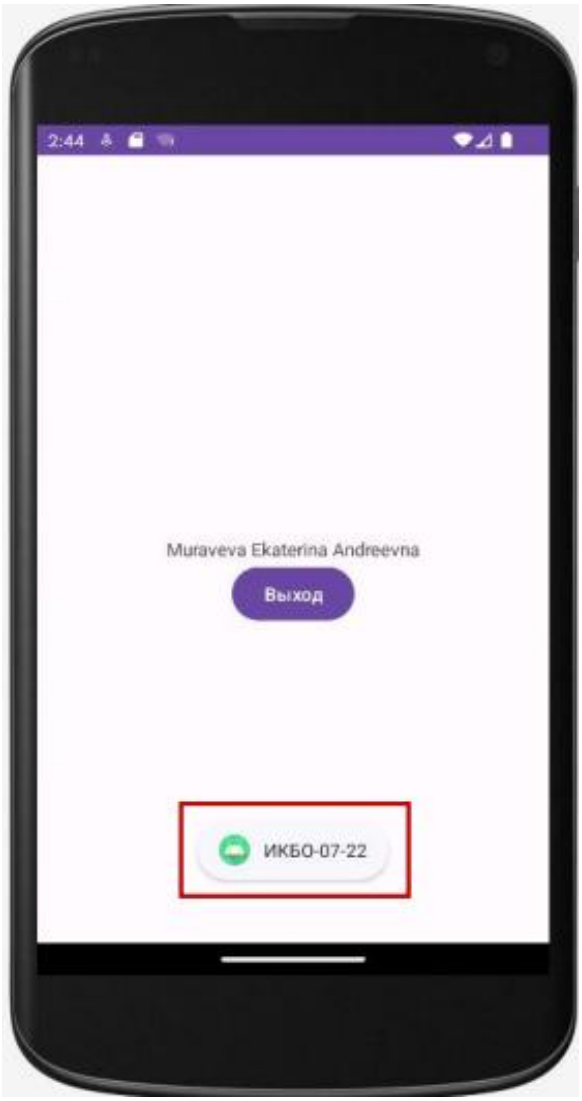

Рисунок 7 – Отображение созданного Toast-сообщения

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.

1.3 Взаимодействие с элементами пользовательского интерфейса

Взаимодействие с элементами пользовательского интерфейса в приложениях для Android, 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 (Рисунок 8).

```
<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"/>
```

Рисунок 8 – Пример id элемента

Например, если в XML есть кнопка с идентификатором @+id/my_button, то для получения ссылки на эту кнопку в Java используется следующий код, представленный на рисунке 9.

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

Рисунок 9 – Получение ссылки на кнопку по её id

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 (Рисунок 10).

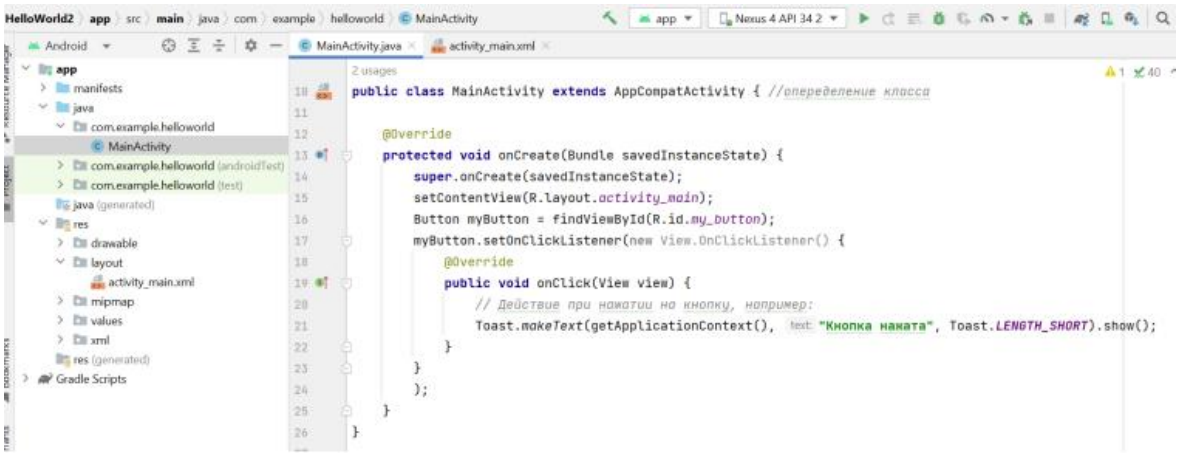

Рисунок 10 – Пример создания обработчика событий

В этом примере при нажатии на кнопку на экране появится короткое сообщение Toast с текстом «Кнопка нажата» (Рисунок 11).

Рисунок 11 – Демонстрация работы обработчика событий

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.

Использование атрибута onClick – это относительно новый способ для взаимодействия с элементами пользовательского интерфейса, специально разработанный для Android (Рисунок 12).

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

Рисунок 12 – Назначение обработчика клика в XML

Далее нужно прописать в классе activity придуманное вами имя метода, который будет обрабатывать нажатие. Метод должен быть открытым (public) и с одним параметром, использующим объект View (Рисунок 13).

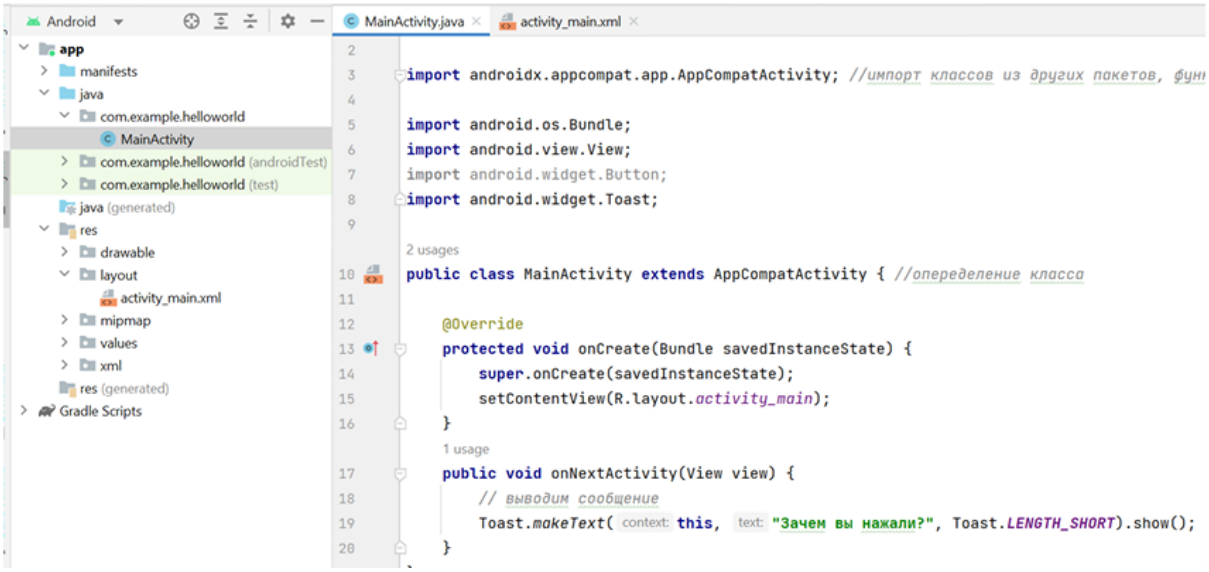

Рисунок 13 – Описание обработчика события onClick в Java

Когда пользователь нажимает на кнопку, то вызывается метод onNextActivity(), который в свою очередь генерирует всплывающее сообщение (Рисунок 14).

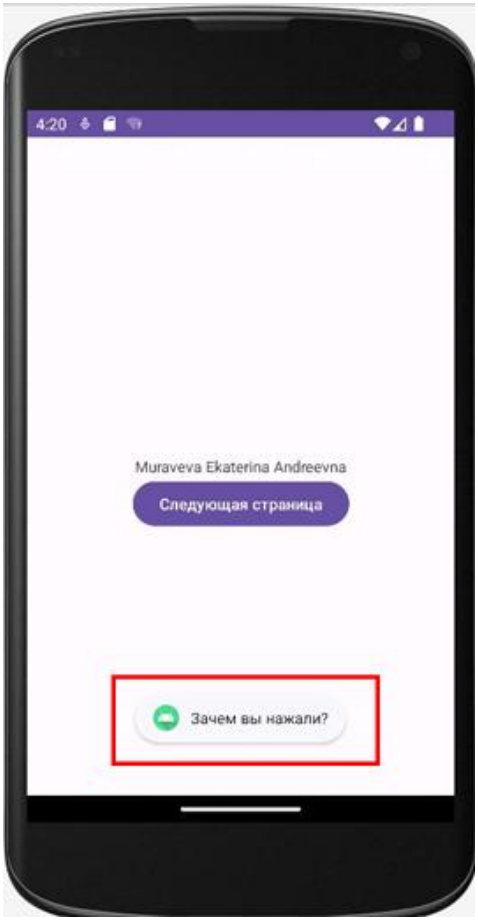

Рисунок 14 – Результат работы обработчика событий

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.

1.4 Переход между экранами

В Android-приложениях часто используется несколько activity, каждая из которых представляет собой отдельный экран или пользовательский интерфейс. Для создания новой activity в Android Studio необходимо выполнить несколько шагов.

Во-первых, в проекте нужно создать новый класс activity. Это можно сделать, нажав на модуль "app", затем в верней панели выбрав "File" → "New" →

"Activity" и затем выбрав тип activity, например "Empty Views Activity" либо нажать правой кнопкой мыши на "java"→ "New" → "Activity" (Рисунки 15-16).

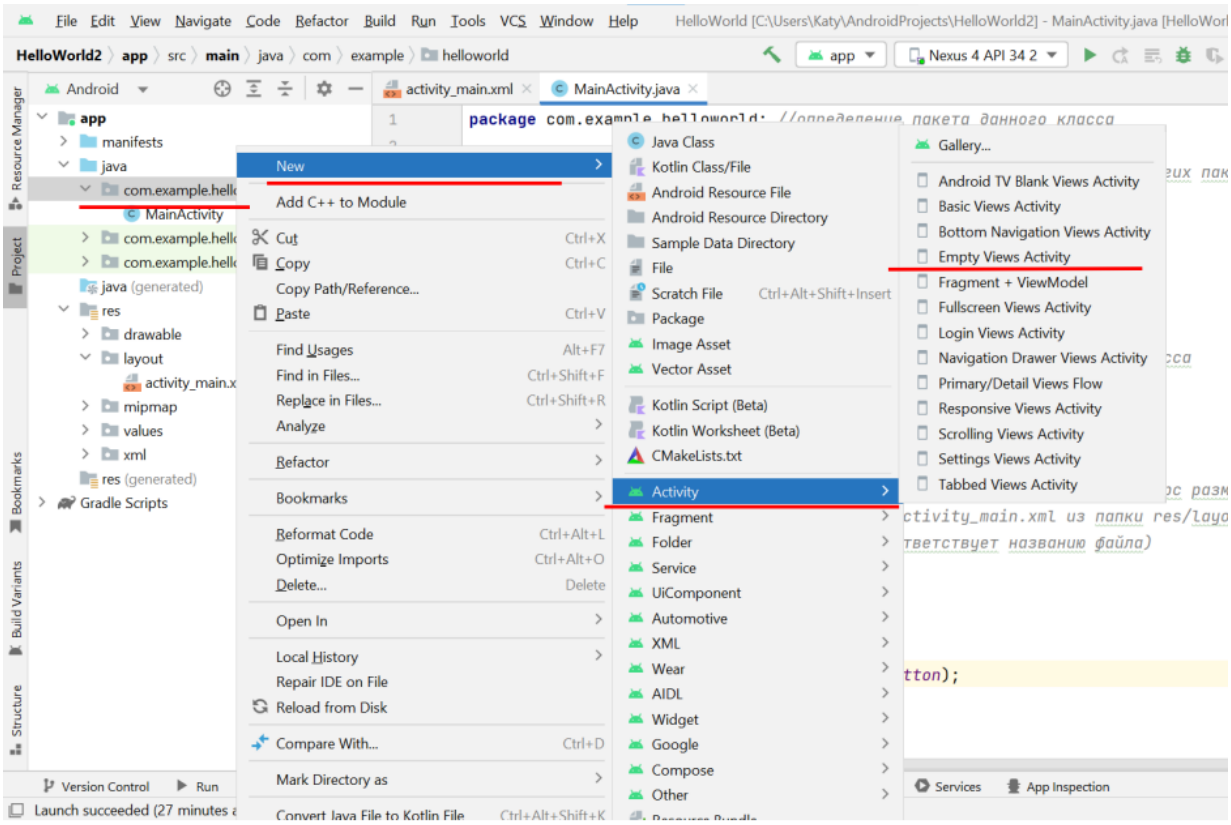

Рисунок 15 – Создание новой activity

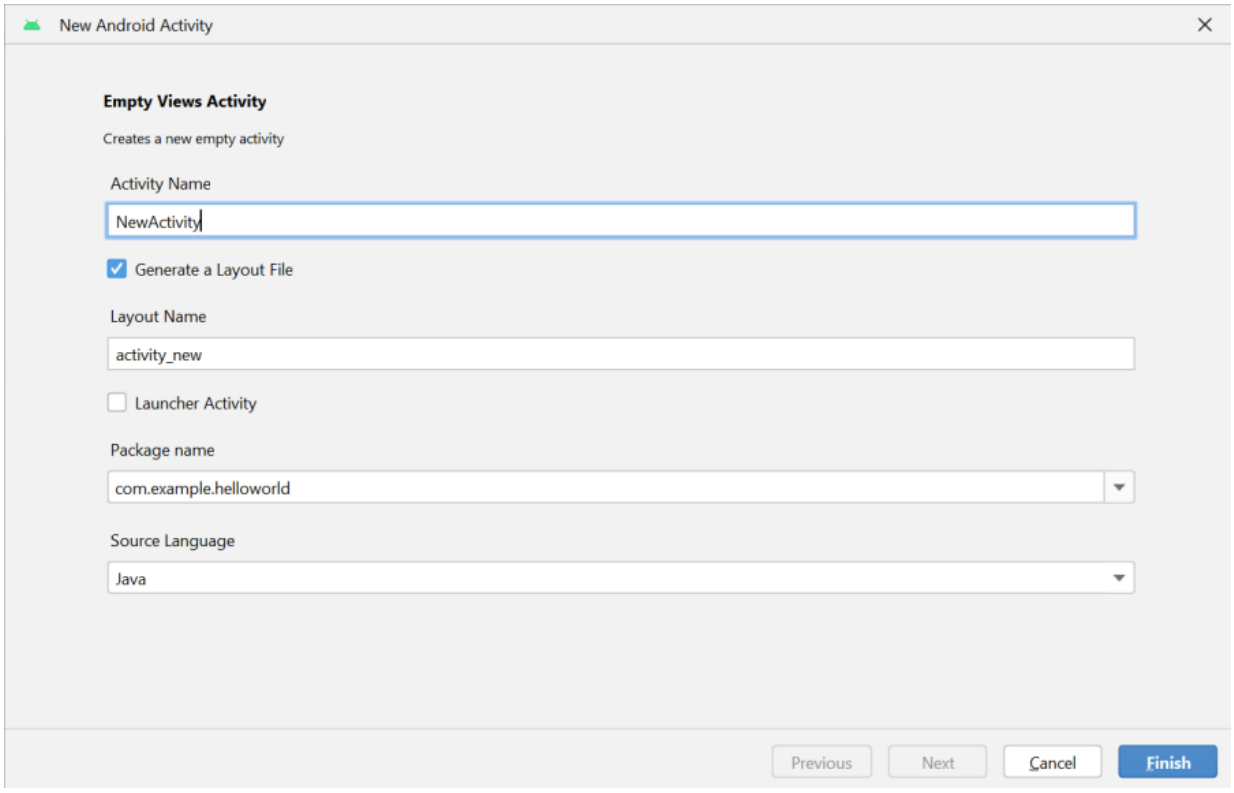

Рисунок 16 – Первичная настройка при создании новой activity

Android Studio сгенерирует необходимые файлы и ресурсы, такие как файл разметки XML для интерфейса и Java класс для логики activity (Рисунок 17).

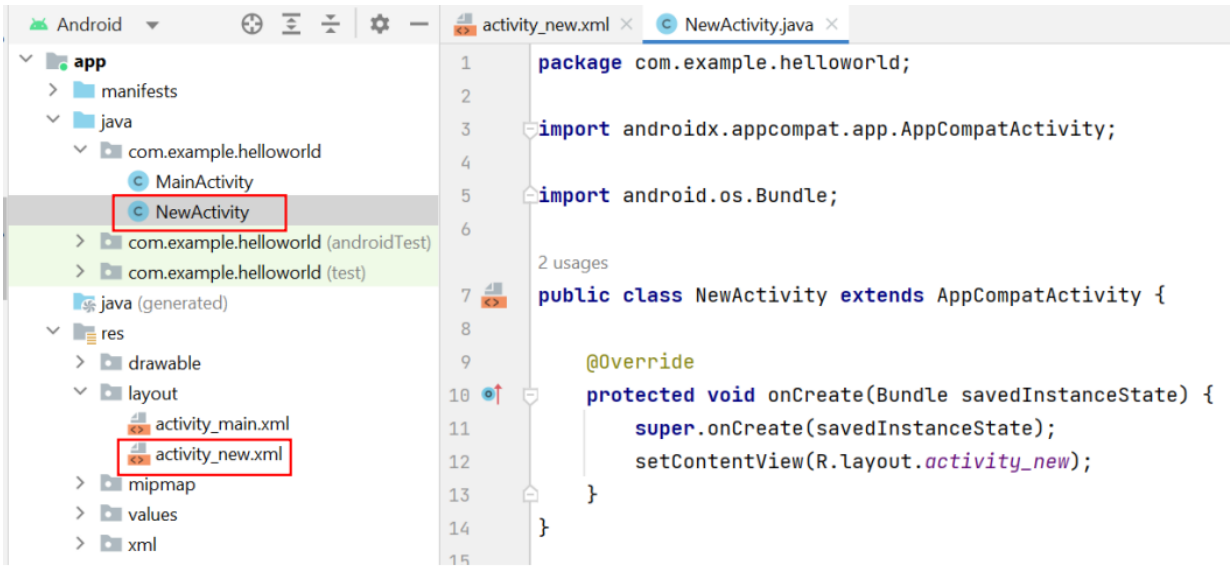

Рисунок 17 – Сгенерированные файлы, связанные с новой activity

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 activity, 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 activity. 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 activity детального просмотра и передачи данных о выбранном элементе.

Создание Intent включает в себя указание контекста (например, текущей activity) и класса activity, которую необходимо запустить. Пример создания Intent для запуска новой activity представлен на рисунке 18.

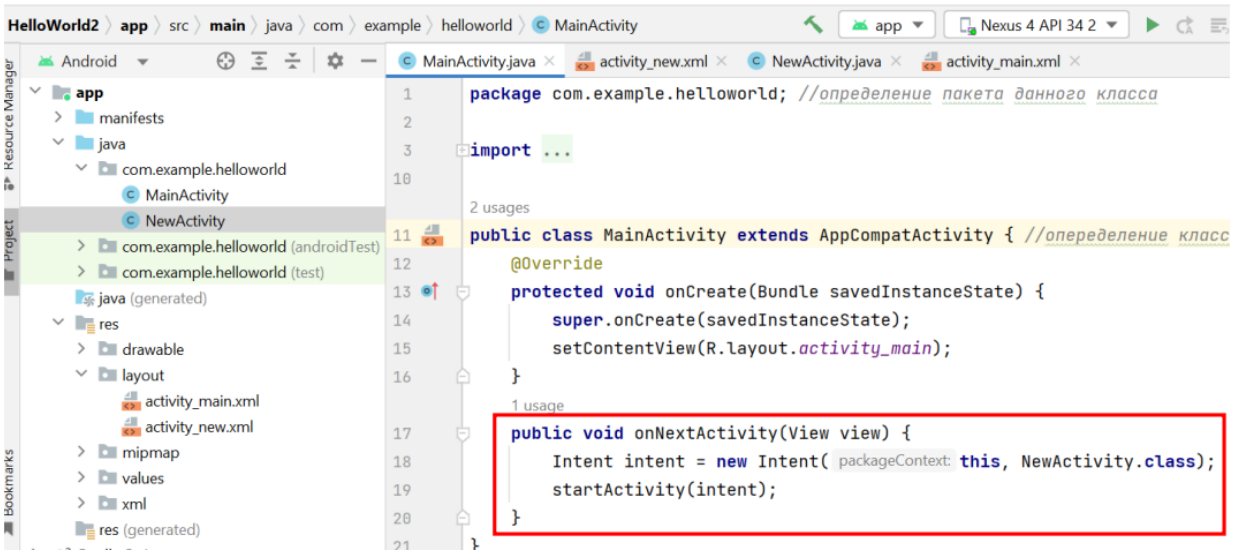

Рисунок 18 – Пример создания Intent для запуска новой activity

Здесь "this" обозначает текущую activity, а "NewActivity" — созданную ранее activity, которую нужно запустить, далее метод "startActivity", начинает выполнение действия, определенного в объекте Intent. По выполнении этой строки, система Android обрабатывает Intent и запускает activity, указанную в нем. Это приводит к переключению пользовательского интерфейса с текущей activity на интерфейс новой activity "NewActivity" (Рисунок 19).

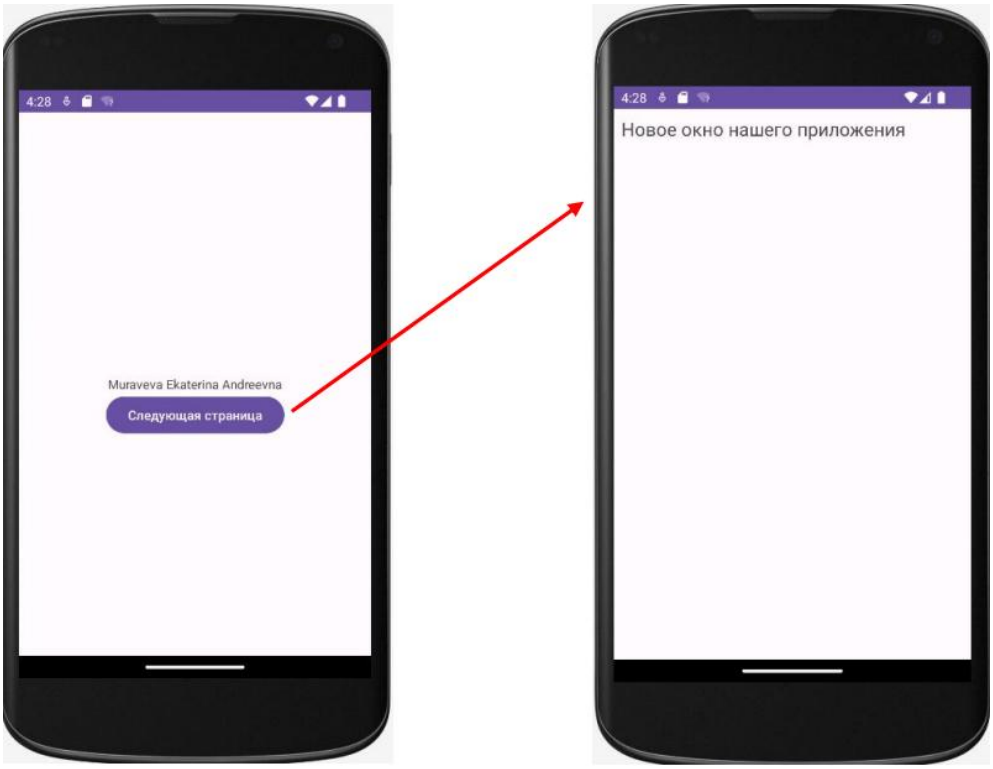

Рисунок 19 – Переход от одной activity к другой

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

1.5 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 activity используется метод putExtra() (Рисунок 20).

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

Рисунок 20 – Передача данных в другую activity

Чтобы получить отправленные данные при загрузке NewActivity, можно воспользоваться методом get(), в который передается ключ объекта (Рисунок 21).

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

Рисунок 21 – Получение данных в новой activity

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 (Рисунок 22).

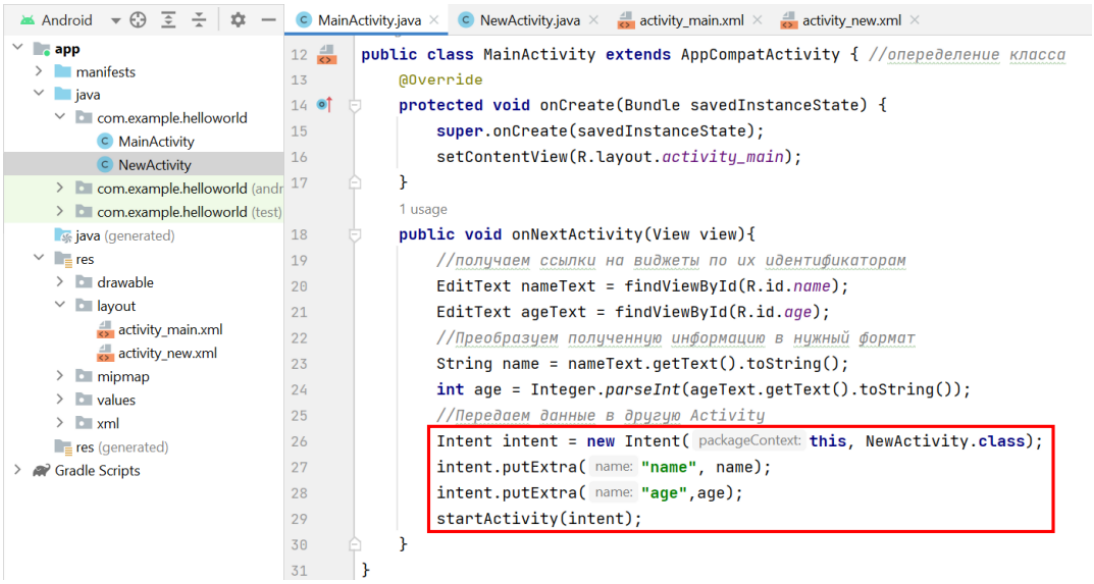

Рисунок 22 – Передача данных из MainActivity

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 (Рисунок 23).

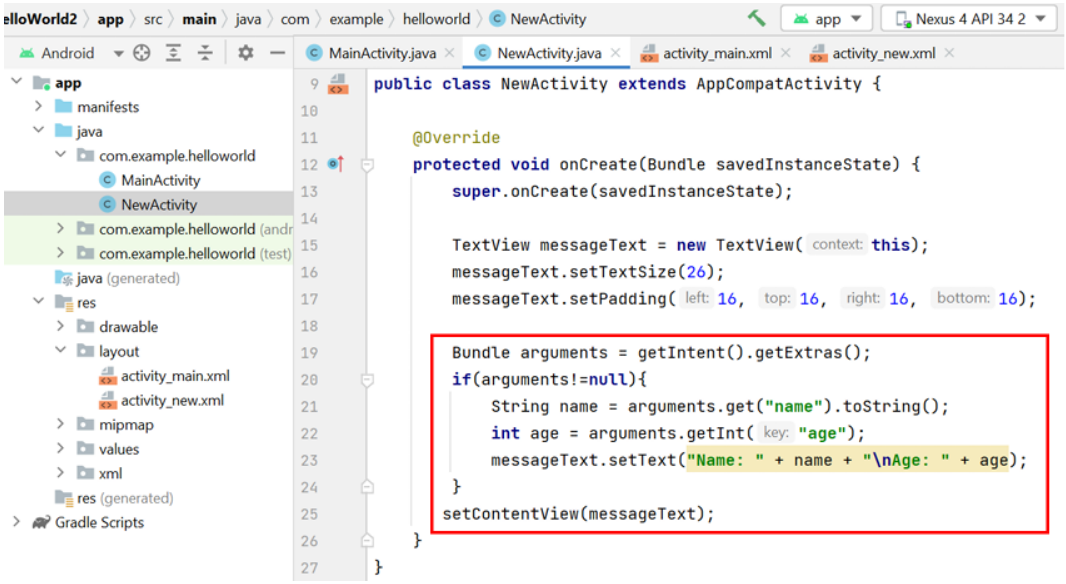

Рисунок 23 – Получение данных в NewActivity

В результате передаем введенные данные из одного окна в другое (Рисунок 24).

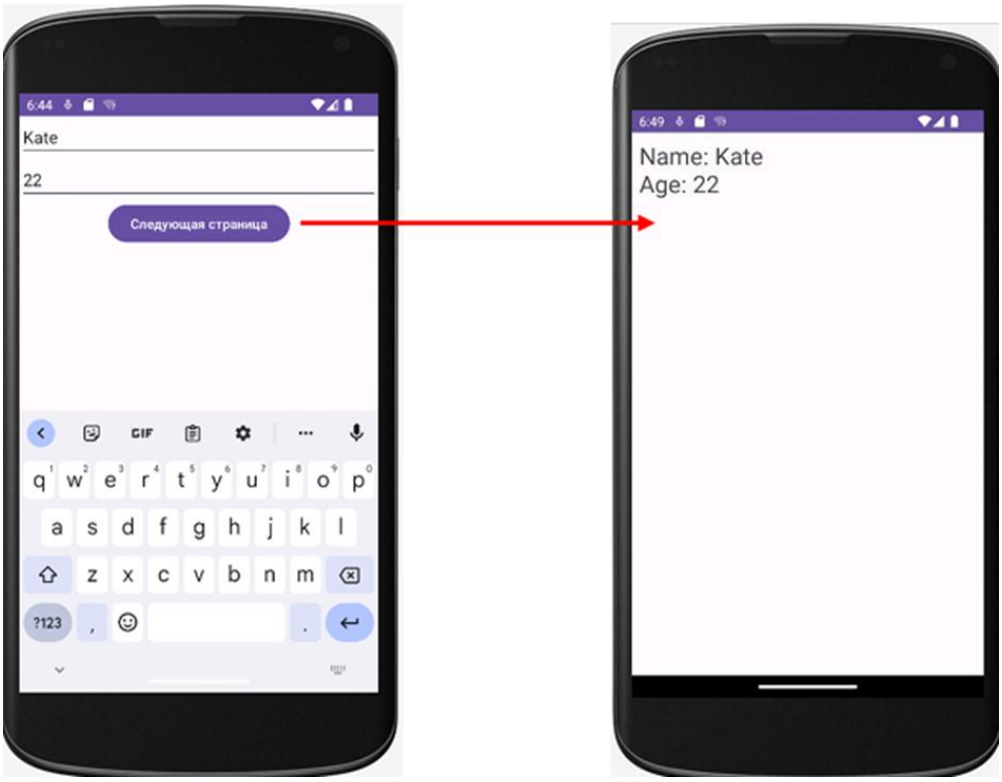

Рисунок 24 – Результат передачи данных из одного окна в другое

2 ПРАКТИЧЕСКОЕ ЗАДАНИЕ

2.1 Отслеживание жизненного цикла activity при помощи логирования

2.1.1 Реализация логирования в методах жизненного цикла

Для создания проекта был выбран шаблон "Empty Views Activity". (Рисунок 25).

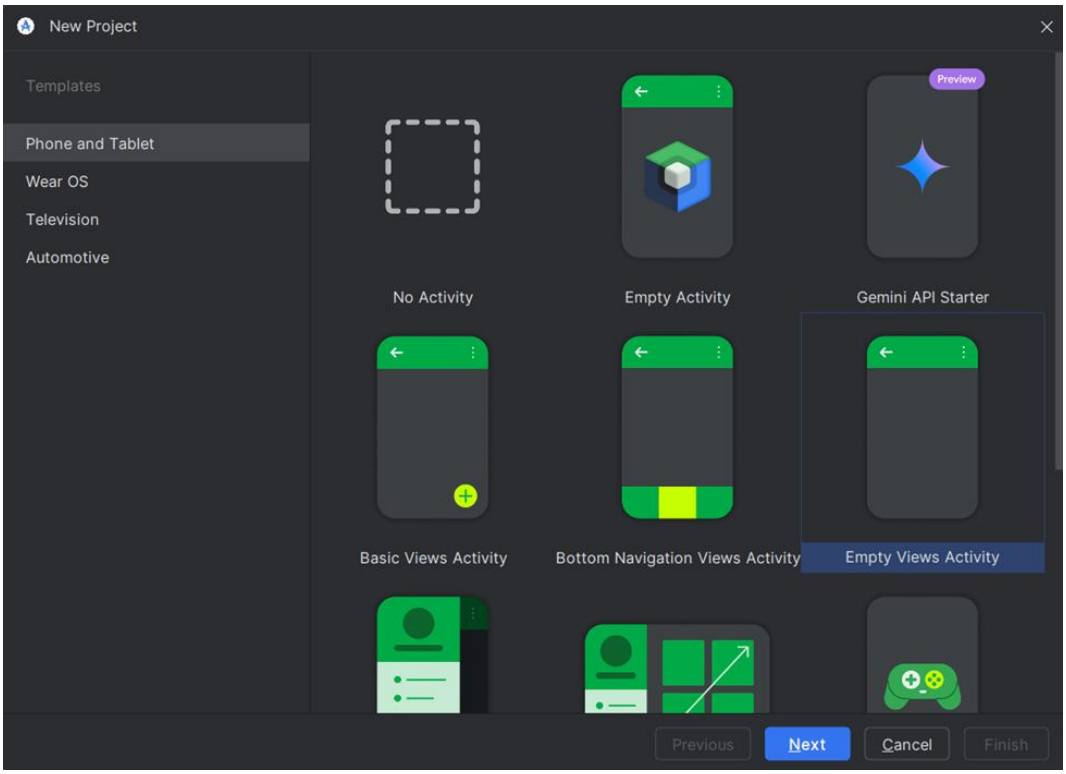

Рисунок 25 – Выбор шаблона проекта в Android Studio

MainActivity изначально состоит из метода onCreate(), который устанавливает разметку activity_main, включает поддержку EdgeToEdge и настраивает обработку системных отступов через ViewCompat.setOnApplyWindowInsetsListener() (Рисунок 26).

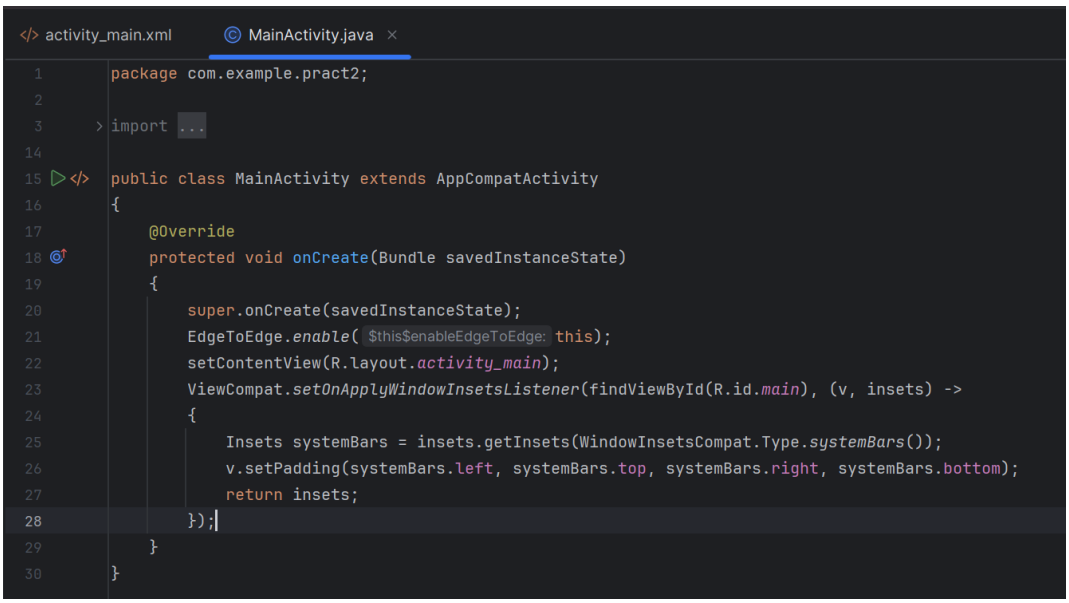

Рисунок 26 – Класс MainActivity при создании проекта

Определим в поле класса MainActivity пользовательский тег как строковую константу для идентификации сообщений логирования (Рисунок 27).

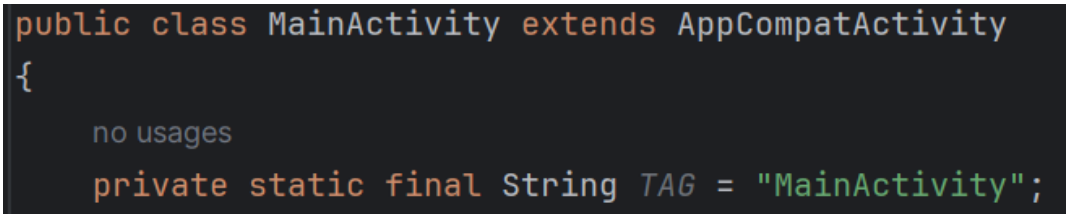

Рисунок 27 – Определение тега для логирования

Добавим в конец метода onCreate() запись лога с информационным сообщением с помощью вызова метода Log.i(), где в качестве тега используется ранее объявленное поле TAG класса MainActivity, а в качестве сообщения используется название метода, то есть "onCreate" (Рисунок 28).

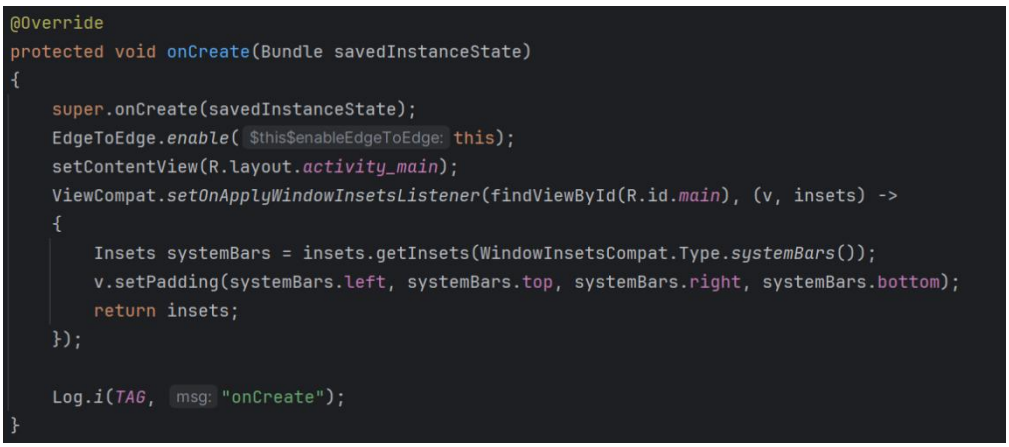

Рисунок 28 – Сообщение логирования в методе OnCreate()

Далее добавим сообщения логирования в методы onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy(). Воспользуемся возможностями Android Studio и откроем окно выбора методов для определения. Для этого нужно нажать правую кнопку мыши по любому месту в файле, потом **Generate** (Alt+Insert), потом **Override Methods**. Далее с помощью поиска, найдём нужные нам методы (Рисунок 29).

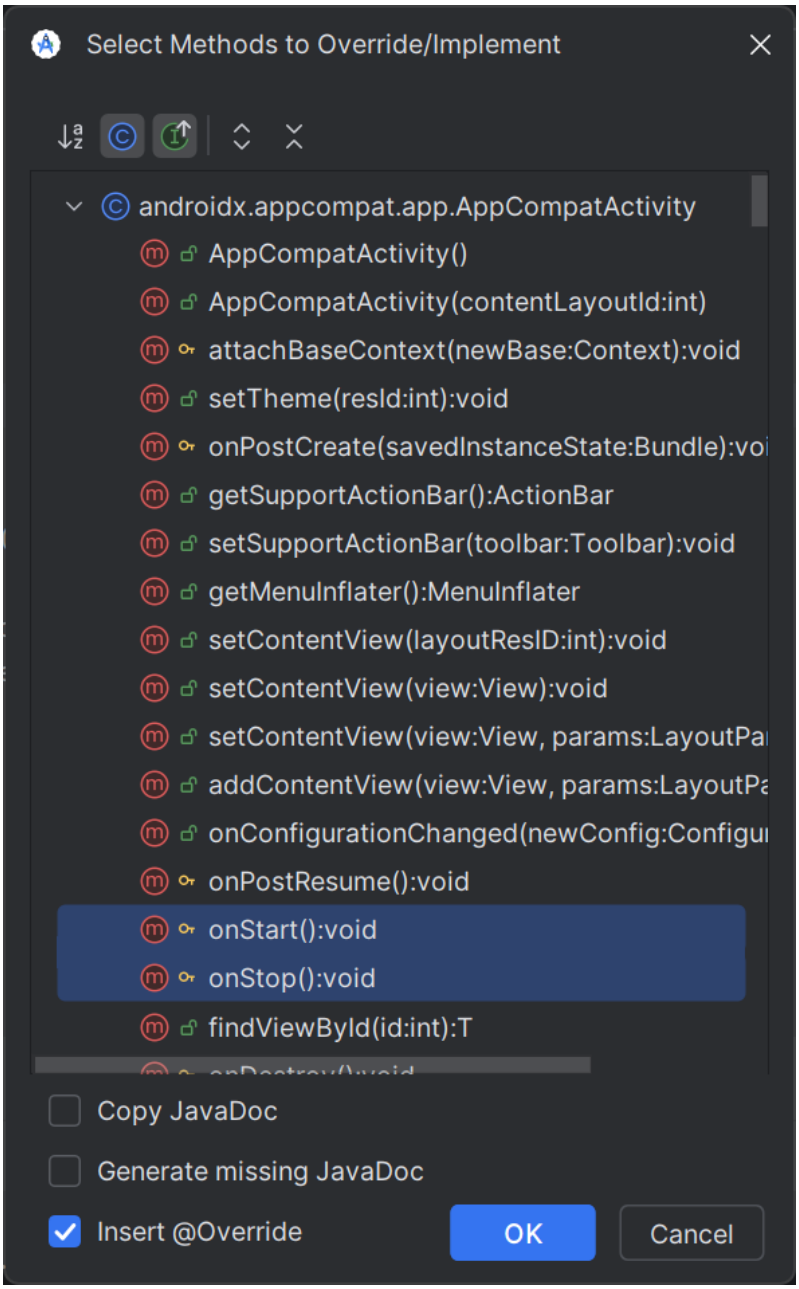

Рисунок 29 – Окно переопределения методов

Android Studio автоматически создаст переопределённый метод и пометит его аннотацией **@Override**, а в теле поместит вызов переопределяемого метода с помощью ключевого слова super (Рисунок 30).

```
@Override
protected void onDestroy()
{
    super.onDestroy();
}
```

Рисунок 30 – Метод OnDestroy(), переопределенный автоматически

Добавим в конец тела каждого из таких методов вывод записей лога по аналогии с добавлением записи лога в метод onCreate() (Рисунки 31-32).

```
@Override
protected void onDestroy()
{
    super.onDestroy();
    Log.i(TAG, msg: "onDestroy");
}

@Override
protected void onStart()
{
    super.onStart();
    Log.i(TAG, msg: "onStart");
}

@Override
protected void onRestart()
{
    super.onRestart();
    Log.i(TAG, msg: "onRestart");
}
```

Рисунок 31 – Добавление логирования в переопределенные методы, часть 1

```
@Override
protected void onStop()
{
    super.onStop();
    Log.i(TAG,  msg: "onStop");
}

@Override
protected void onResume()
{
    super.onResume();
    Log.i(TAG,  msg: "onResume");
}

@Override
protected void onPause()
{
    super.onPause();
    Log.i(TAG,  msg: "onPause");
}
```

Рисунок 32 – Добавление логирования в переопределенные методы, часть 2

После добавления методов для записи в лог в каждый из нужных методов. Откроем панель Logcat, расположенную в левом нижнем углу (Рисунок 33).

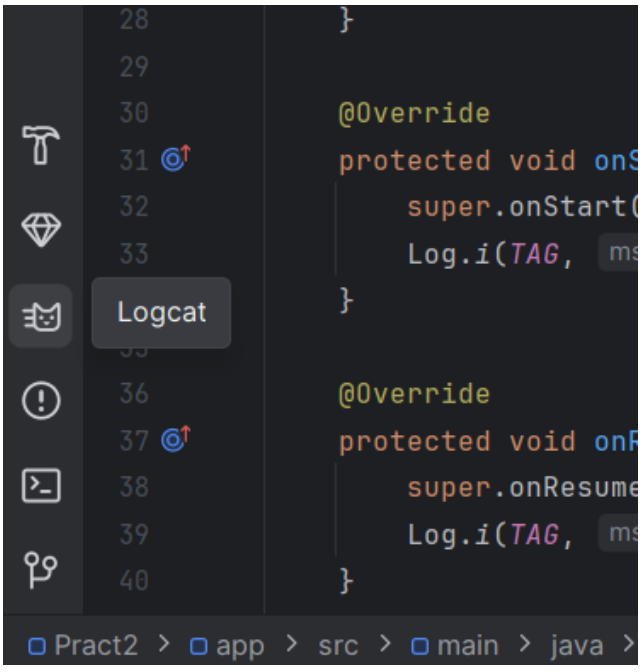

Рисунок 33 – Панель Logcat

В ней выберем созданный ранее пользовательский тег в качестве фильтра записей (Рисунок 34).

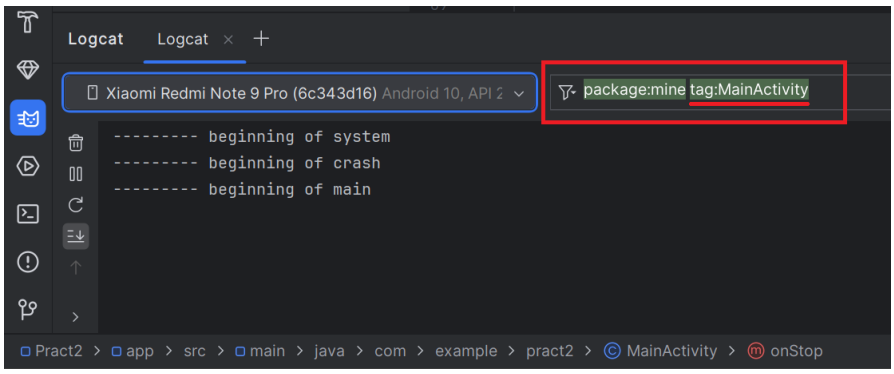

Рисунок 34 – Выбор созданного тега в качестве фильтра записей

2.1.2 Отслеживание жизненного цикла activity

Протестируем отслеживание жизненного цикла activity. Запустим наше приложение и увидим, что в панели Logcat отобразились сообщения из методов onCreate(), onStart() и onResume(), поскольку MainActivity была создана, стала видимой и приняла готовность к взаимодействию с пользователем (Рисунок 35).

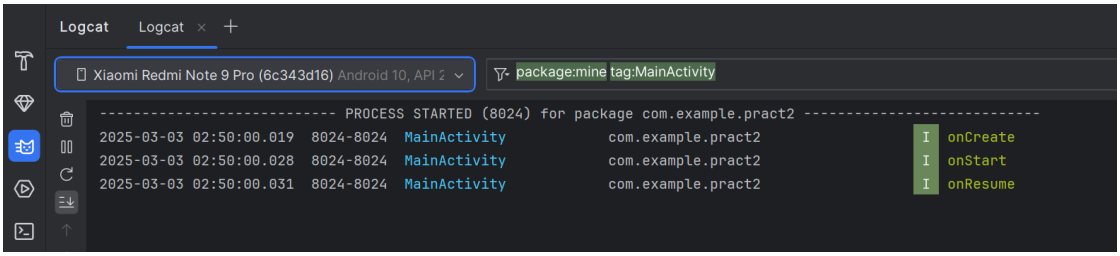

Рисунок 35 – Тестирование логирования жизненного цикла activity при запуске приложения

Перейдём на домашний экран Android и увидим, что в панели Logcat отобразились сообщения из методов onPause() и onStop(), так как MainActivity сначала перестала быть доступной для взаимодействия с пользователем, а затем перестала быть видимой (Рисунок 36).

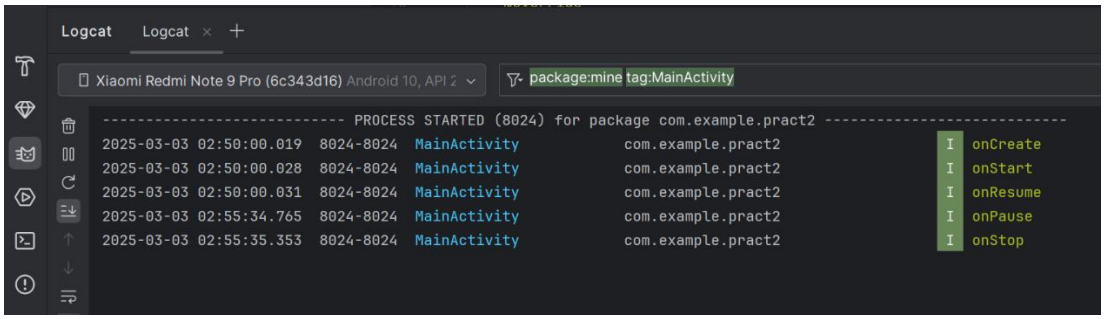

Рисунок 36 – Тестирование логирования жизненного цикла activity при свертывании приложения

Вернёмся к нашему приложению и увидим, что вновь были вызваны методы onStart() и onResume(), поскольку MainActivity вновь стала видимой и готовой к взаимодействию с пользователем (Рисунок 37).

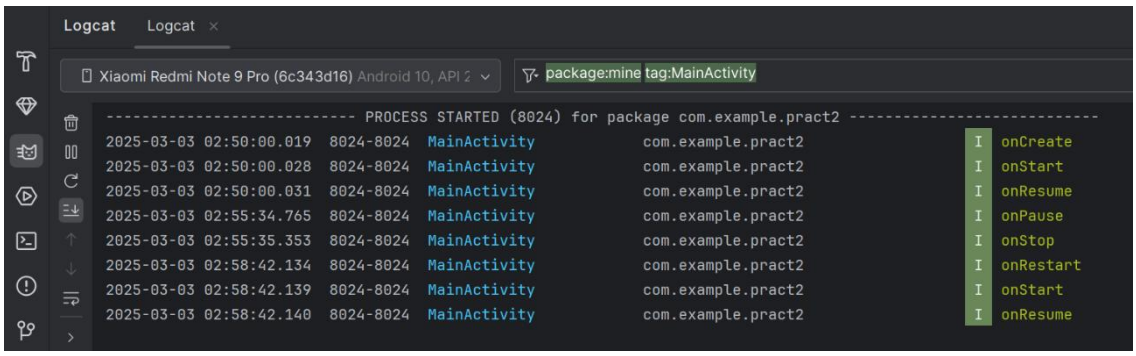

Рисунок 37 – Тестирование логирования жизненного цикла activity при возвращении к приложению

Завершим работу с приложением, закрыв его полностью, и увидим записи логов из методов onPause(), onStop() и onDestroy(), так как MainActivity перестала быть доступной для взаимодействия с пользователем, перестала быть видимой и завершила свою работу (Рисунок 38). Можно заключить, что логирование работы жизненного цикла activity работает корректно.

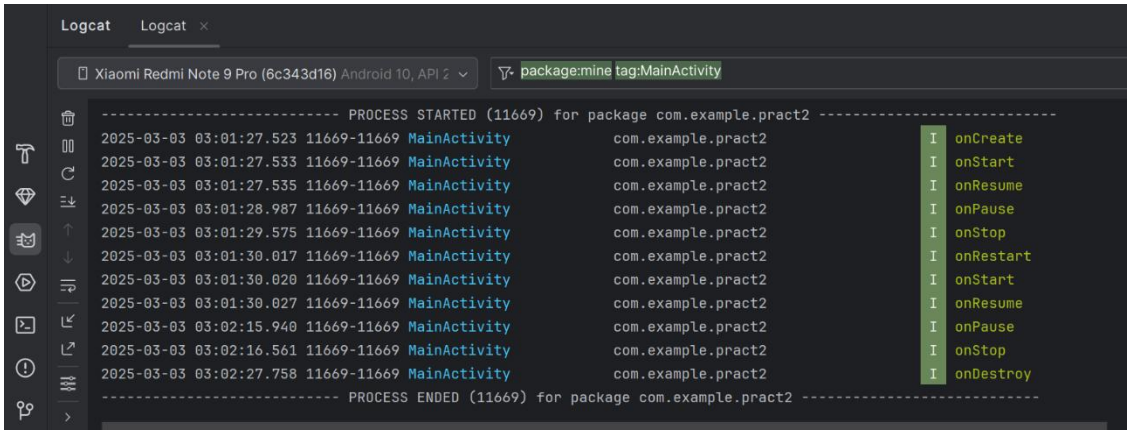

Рисунок 38 – Тестирование логирования жизненного цикла activity при полном закрытии приложения

2.2 Переход на другую activity

2.2.1 Создание разметки первой activity

В файле activity_main.xml корневым элементом установим LinearLayout, с вертикальной ориентацией, выравниванием элементов по центру (Рисунок 39).

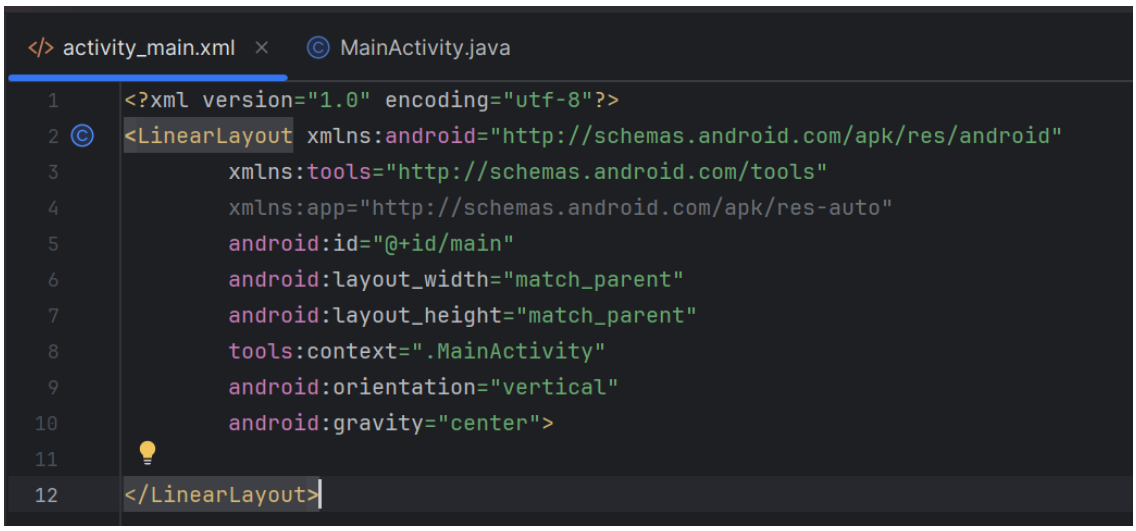

Рисунок 39 – Создание коневого элемента разметки первой activity

Далее добавим вложенный LinearLayout, в котором будут располагаться поля для ввода, с вертикальной ориентацией, и горизонтальными внутренними отступами, шириной равной ширине родительского компонента и высотой в зависимости от содержимого (Рисунок 40).

Рисунок 40 – Вложенный LinearLayout

В качестве дочерних элементов рассмотренного выше LinearLayout добавим четыре поля для ввода типа EditText (Рисунок 41). Каждое из них будет занимать всю ширину родителя, иметь высоту в зависимости от содержания. Значения текстов-подсказок: «Введите ФИО», «Введите номер группы», «Введите возраст», «Введите оценку» соответственно. Каждому компоненту присвоим id.

```
<EditText
        android:id="@+id/nameField"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:hint="Введите ФИО"
        android:inputType="text" />

<EditText
        android:id="@+id/groupField"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:hint="Введите номер группы"
        android:inputType="text" />

<EditText
        android:id="@+id/ageField"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:hint="Введите возраст"
        android:inputType="number" />

<EditText
        android:id="@+id/gradeField"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:hint="Введите оценку"
        android:inputType="text" />
```

Рисунок 41 – Разметка полей для ввода

Затем под группой полей для ввода добавим два компонента-кнопки (Рисунок 42).

```
<Button
        android:id="@+id/declarativeButton"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:text="Подтвердить 1" />

<Button
        android:id="@+id/programmedButton"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:text="Подтвердить 2" />
```

Рисунок 42 – Разметка кнопок

На рисунке 43 представлена разметка MainActivity в графическом режиме.

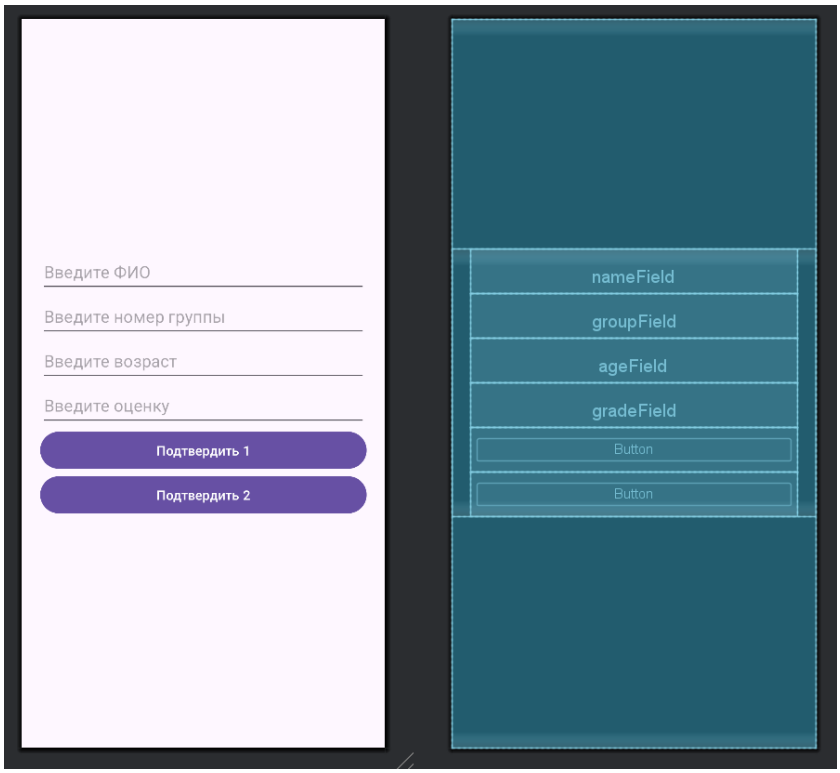

Рисунок 43 – Разметка MainActivity

2.2.2 Создание второй activity и её разметки

Создадим вторую activity и присвоим ей название SecondActivity (Рисунки 44-45).

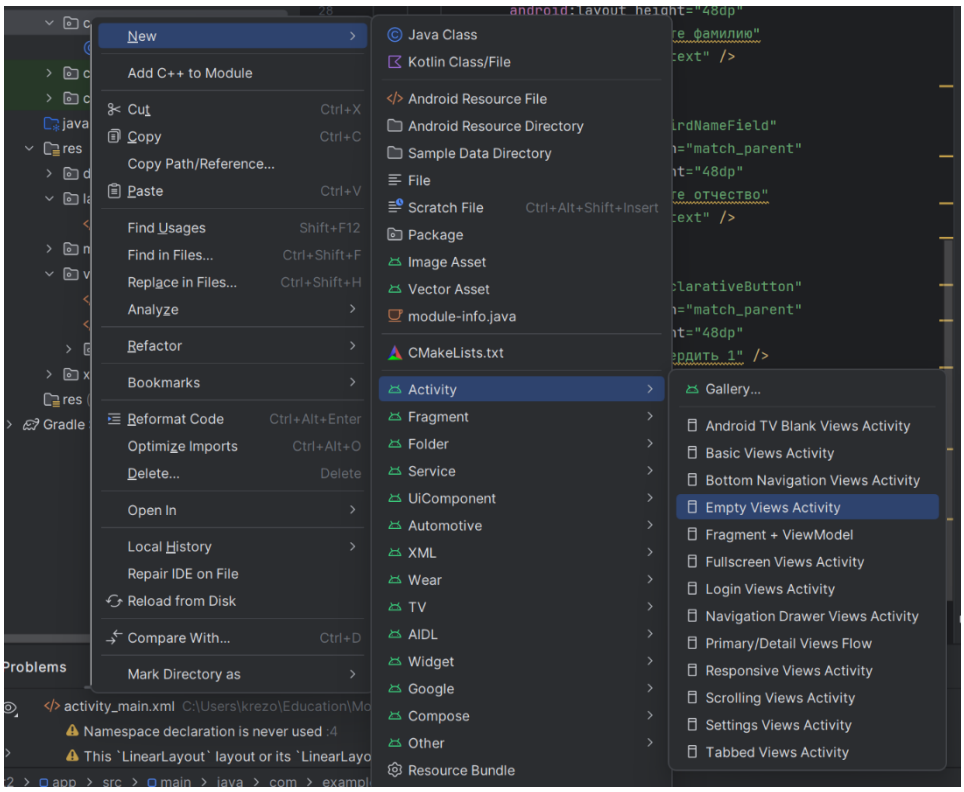

Рисунок 44 – Создание второй activity с помощью выпадающего меню

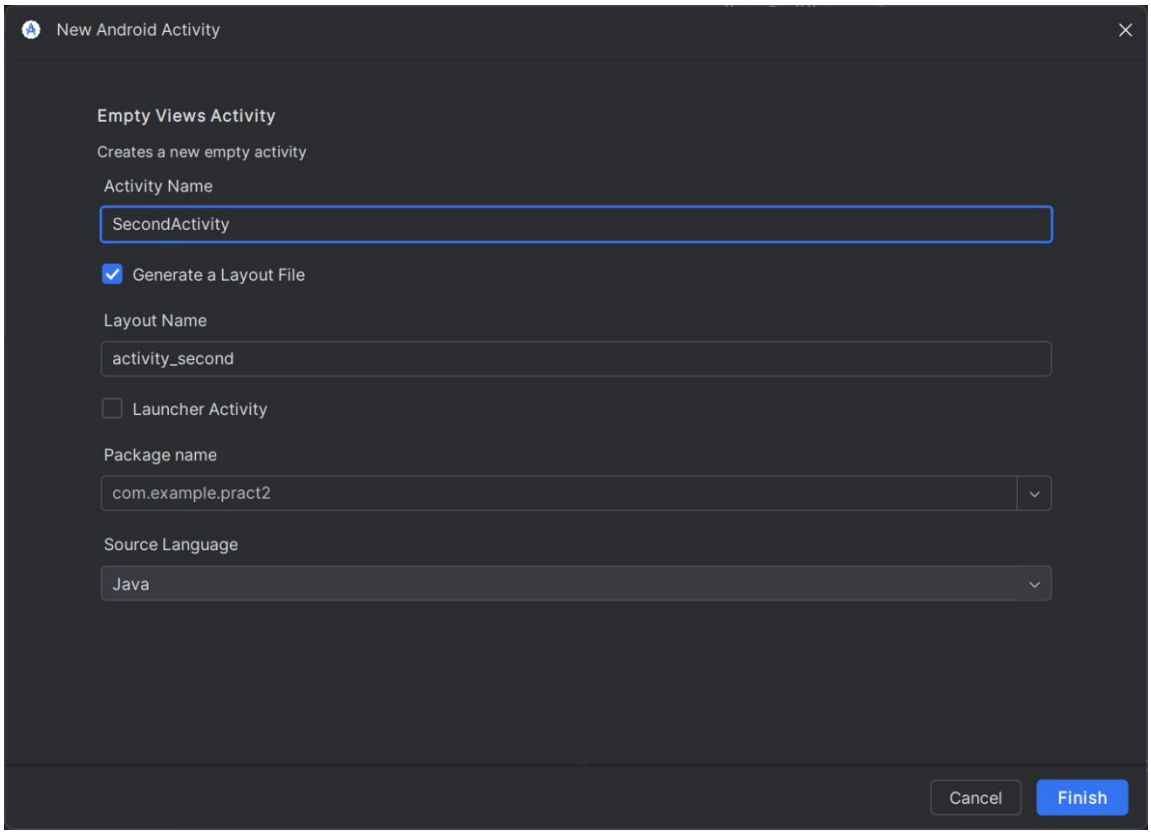

Рисунок 45 – Настройка создания второй activity

В файле activity_second.xml корневым элементом установим LinearLayout, с вертикальной ориентацией, выравниванием элементов по центру (Рисунок 46).

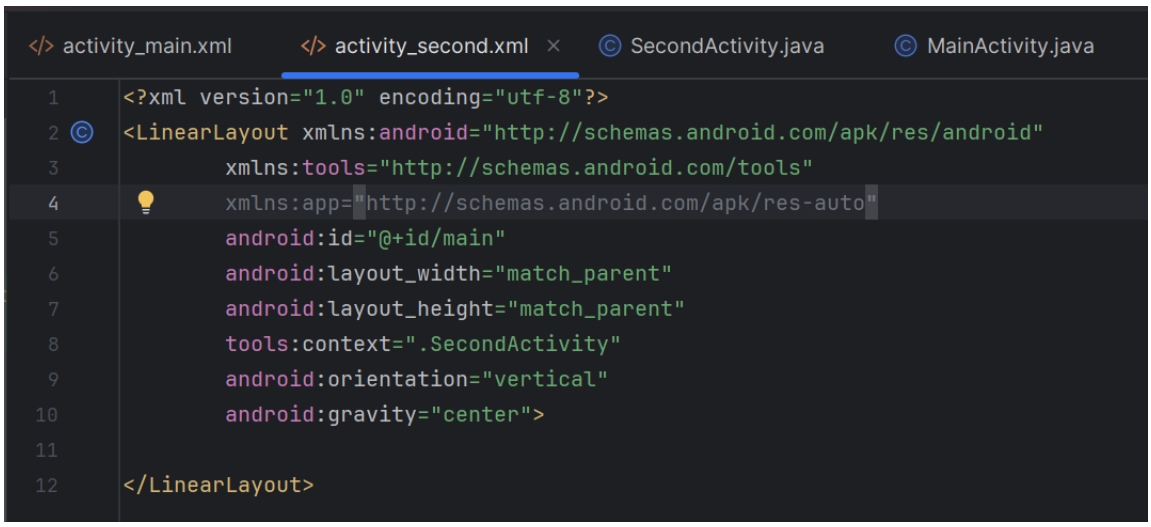

Рисунок 46 – Создание корневого элемента разметки первой activity

Далее добавим четыре компонента TextView для отображения текста (Рисунок 47). Присвоим им id и текстовые значения: «ФИО», «Номер группы», «Возраст», «Оценка» соответственно.

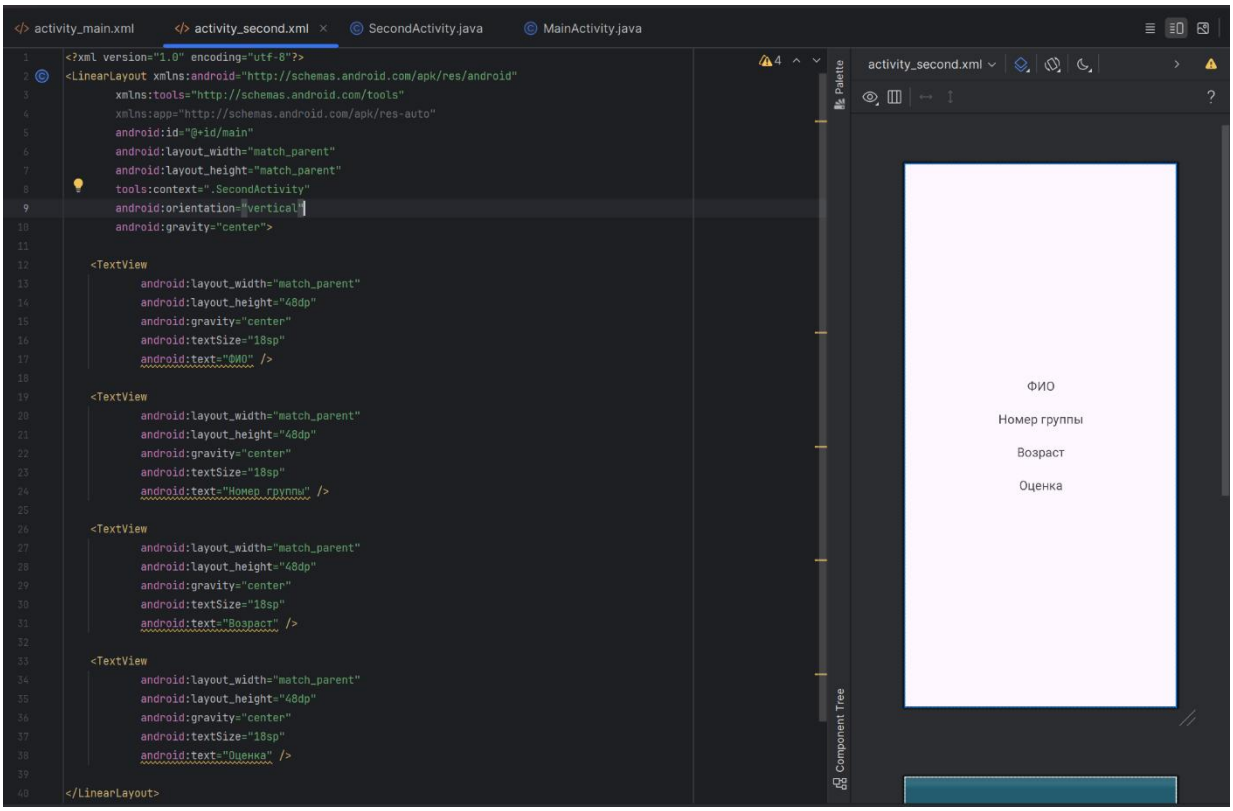

Рисунок 47 – Разметка второй activity

2.2.3 Реализация перехода на другую activity

В классе MainActivity напишем метод обработчик событий moveToSecondPage(). В нём создаётся, а затем передаётся методу startActivity(),

объект класса Intent, который используется для указания системе, что нужно запустить новую activity. В нашем случае из MainActivity запускается SecondActivity (Рисунок 48).

```
public void moveToSecondPage(View v)
{
    Intent i = new Intent( packageContext: this, SecondActivity.class);
    startActivity(i);
}
```

Рисунок 48 – Обработчик событий для перехода на другую activity

Затем в методе onCreate() класса MainActivity получаем ссылку на компонент кнопки с id programmedButton и программно устанавливаем метод moveToSecondPage() в качестве его обработчика событий с помощью метода setOnClickListener() (Рисунок 49).

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_main);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) ->
    {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });

    Button button = findViewById(R.id.programmedButton);
    button.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            moveToSecondPage(v);
        }
    });
}
```

Рисунок 49 – Установка обработчика событий для кнопки программно

Для кнопки с id declarativeButton устанавливаем тот же метод moveToSecondPage() в качестве обработчика событий декларативным способом в файле разметки activity_main.xml с помощью атрибута onClick (Рисунок 50).

```
<Button
        android:id="@+id/declarativeButton"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:onClick="moveToSecondPage"
        android:text="Подтвердить 1" />
```

Рисунок 50 – Установка обработчика событий для кнопки декларативно

2.2.4 Реализация передачи данных из первой activity во вторую

В начале тела обработчика событий moveToSecondPage() получим ссылки на каждое из полей ввода MainActivity (Рисунок 51).

```
public void moveToSecondPage(View v)
{
    EditText nameField = findViewById(R.id.nameField);
    EditText groupField = findViewById(R.id.groupField);
    EditText ageField = findViewById(R.id.ageField);
    EditText gradeField = findViewById(R.id.gradeField);
```

Рисунок 51 – Получение ссылок на поля ввода

Далее извлечём из них текстовые значения с помощью методов getText() и toString() и с помощью метода putExtra() добавим в объект класса Intent (Рисунок 52).

```
2 usages
public void moveToSecondPage(View v)
{
    EditText nameField = findViewById(R.id.nameField);
    EditText groupField = findViewById(R.id.groupField);
    EditText ageField = findViewById(R.id.ageField);
    EditText gradeField = findViewById(R.id.gradeField);

    Intent i = new Intent( packageContext: this, SecondActivity.class);
    i.putExtra( name: "name", nameField.getText().toString());
    i.putExtra( name: "group", groupField.getText().toString());
    i.putExtra( name: "age", ageField.getText().toString());
    i.putExtra( name: "grade", gradeField.getText().toString());

    startActivity(i);
}
```

Рисунок 52 – Добавление данных в объект класса Intent

В методе onCreate() класса SecondActivity выполняется получение переданных данных из Intent, который вызвал SecondActivity. Метод getIntent() возвращает Intent, который запустил эту activity, а getExtras() извлекает Bundle с дополнительными данными, если они были переданы.

Затем, если объект класса Intent содержал в себе какие-то дополнительные данные, из объекта класса Bundle по ключу извлекаются переданные значения

ФИО, группы, возраста и оценки и присваиваются строковым переменным (Рисунок 53).

```
Bundle args = getIntent().getExtras();
if (args == null) return;

String name = args.getString( key: "name");
String group = args.getString( key: "group");
String age = args.getString( key: "age");
String grade = args.getString( key: "grade");
```

Рисунок 53 – Извлечение переданных в activity данных

Далее по id получаем ссылки на текстовые компоненты SecondActivity и с помощью метода setText() устанавливаем значение их текста в соответствии с полученными из MainActivity данными (Рисунок 54).

```
TextView nameOut = findViewById(R.id.nameOut);
TextView groupOut = findViewById(R.id.groupOut);
TextView ageOut = findViewById(R.id.ageOut);
TextView gradeOut = findViewById(R.id.gradeOut);

nameOut.setText(name);
groupOut.setText(group);
ageOut.setText(age);
gradeOut.setText(grade);
```

Рисунок 54 – Заполнение текстовых полей переданными значениями

Проведём тестирование созданного приложения. Введём данные в поля ввода первой activity (Рисунки 55-56).

Рисунок 55 – Начальная страница приложения до ввода данных

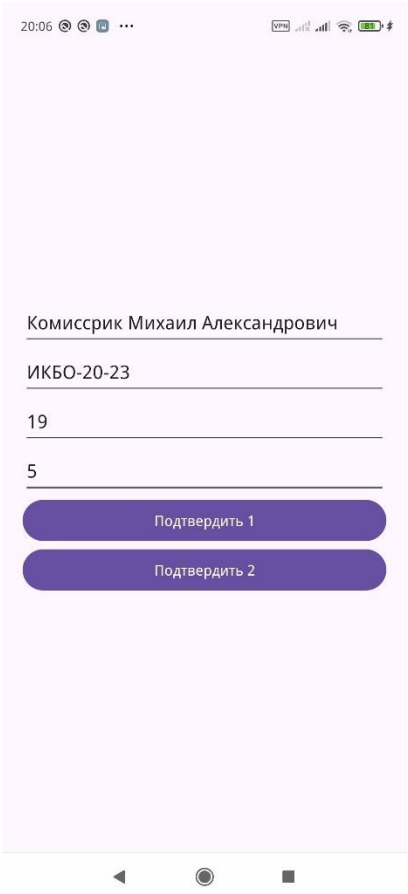

Рисунок 56 – Начальная страница приложения после ввода данных

Затем нажмём на первую кнопку, и убедимся, что в окне новой активности отображаются введённые ранее данные (Рисунок 57). Нажатие на вторую кнопку также приводит к желаемому результату.

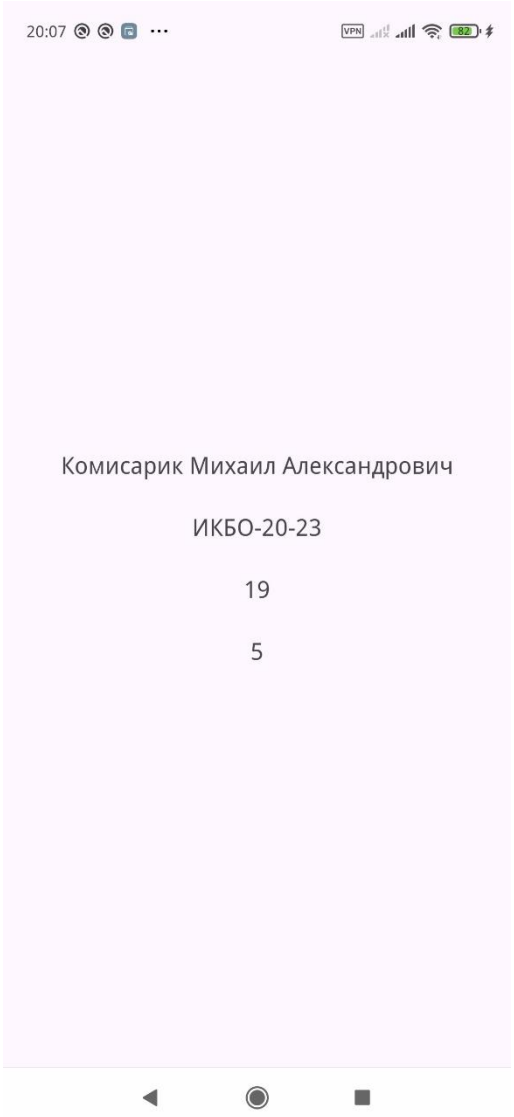

Рисунок 57 – Вторая страница приложения

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были изучены основы работы с инструментами логирования в Android Studio, а также классами пользовательского интерфейса activity. Была отслежена работа жизненного цикла acitvity при помощи логирования на всех этапах жизненного цикла, был реализован переход на другую activity при нажатии на кнопку несколькими способами, а также с передачей в нее данных.