|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-21-23 | Муравьев А. О. |
| **Проверил:**  Старший преподаватель кафедры МОСИТ | Шешуков Л.С. |

Москва 2025 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc192876081)

[1 ТЕОРИТИЧЕСКАЯ ЧАСТЬ 5](#_Toc192876082)

[1.1 Активность 5](#_Toc192876083)

[1.2 Логирование 7](#_Toc192876084)

[1.3 Взаимодействие с элементами пользовательского интерфейса 12](#_Toc192876085)

[1.4 Переход между экранами 16](#_Toc192876086)

[1.5 Передача данных между Activity 21](#_Toc192876087)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 25](#_Toc192876088)

[2.1 Отслеживание работы жизненного цикла активности 25](#_Toc192876090)

[2.1.1 Этап onCreate 25](#_Toc192876091)

[2.1.2 Этап onStart 26](#_Toc192876092)

[2.1.3 Этап onResume 27](#_Toc192876093)

[2.1.4 Этап onPause 27](#_Toc192876094)

[2.1.5 Этап onStop 28](#_Toc192876095)

[2.1.6 Этап onDestroy 29](#_Toc192876096)

[2.2 Переход на активность с помощью кнопки 30](#_Toc192876097)

[2.2.1 Декларативный способ 30](#_Toc192876099)

[2.2.2 Программный способ 32](#_Toc192876100)

[2.3 Переход в активность с передачей данных 35](#_Toc192876101)

[ЗАКЛЮЧЕНИЕ 41](#_Toc192876102)

ВВЕДЕНИЕ

Цель данной работы — ознакомиться с работой жизненного цикла активностей, логирования и переходов между активностями.

В ходе выполнения задания будет изменён проект из предыдущей практической работы. Проект будет включать логирование различных этапов жизненного цикла активности, реализован переход между активностями двумя способами, а также создана активность, при переходе к которой будут передаваться данные из поля ввода.

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.

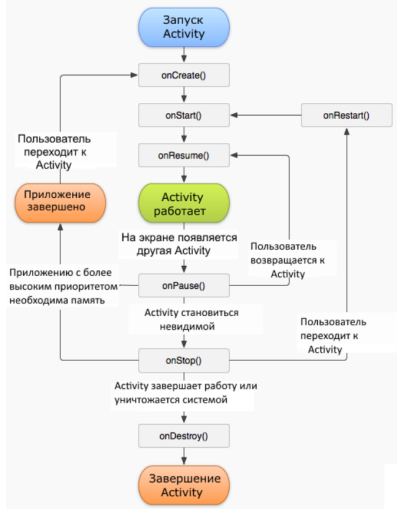
# ТЕОРИТИЧЕСКАЯ ЧАСТЬ

## Активность

Активность в Android представляет собой ключевой компонент приложения, обеспечивающий графический интерфейс для взаимодействия пользователя с программой. Она функционирует как отдельное «окно» пользовательского интерфейса, через которое пользователь может вводить данные, просматривать информацию и выполнять различные действия.

Структура активности включает в себя элементы управления, такие как кнопки, текстовые поля, изображения и другие виджеты, обеспечивающие необходимую функциональность приложения. В операционной системе Android каждая активность проходит определённые стадии жизненного цикла, управление которыми осуществляется системой.

Жизненный цикл активности в Android представлен на рисунке 1.

  
Рисунок 1 – Жизненный цикл активности в Android

Первым этапом в жизненном цикле активности является метод onCreate(), который вызывается при её первоначальном создании. В этом методе обычно выполняется одноразовая инициализация, такая как настройка пользовательского интерфейса, загрузка данных и определение начального состояния активности. После выполнения onCreate() активность переходит в состояние onStart().

Метод onStart() вызывается, когда активность становится видимой для пользователя. На этом этапе можно выполнять подготовительные действия, необходимые для дальнейшего взаимодействия с пользователем. Следующим этапом жизненного цикла является onResume().

Метод onResume() срабатывает, когда активность выходит на передний план и становится доступной для пользовательского ввода. На этом этапе рекомендуется запускать анимации, обновлять интерфейс и выполнять другие задачи, связанные с активной работой приложения.

Когда активность теряет фокус, но остаётся видимой, вызывается onPause(). Этот метод используется для временной приостановки процессов, которые не должны выполняться в фоновом режиме, например, для остановки анимаций или временного освобождения ресурсов.

Если активность полностью исчезает с экрана, система вызывает метод onStop(). В этом состоянии следует завершать ресурсоёмкие операции, не требующие работы в фоновом режиме. Если активность снова становится видимой, выполняется onRestart(), который предшествует повторному вызову onStart(). В методе onRestart() можно восстановить операции, которые были приостановлены в onStop().

Завершающим этапом является метод onDestroy(), который вызывается при окончательном уничтожении активности. Здесь выполняется освобождение всех оставшихся ресурсов и сохранение данных при необходимости. После вызова onDestroy() активность полностью удаляется из памяти системы.

Для анализа работы методов жизненного цикла можно использовать логирование.

## Логирование

Логирование является важным элементом разработки приложений, позволяющим отслеживать их работу и выявлять возможные ошибки. В Android предусмотрено два основных метода логирования: использование класса Log и вывод всплывающих уведомлений с помощью Toast.

Класс Log предназначен для записи отладочной информации и предоставляет несколько уровней логирования: DEBUG (отладочные сообщения), ERROR (ошибки), INFO (информационные сообщения), VERBOSE (подробные данные) и WARN (предупреждения). Эти уровни обозначаются первыми буквами, что позволяет разработчикам классифицировать сообщения по степени важности.

Пример использования логирования представлен на рисунке 2.

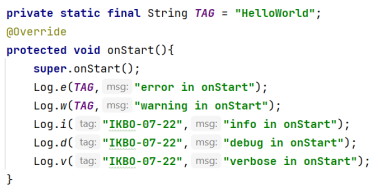
  
Рисунок 2 – Пример использования логирования

Где «MyAppTag» представляет собой пользовательский тег, который используется для идентификации сообщений, поступающих от конкретного приложения, а "Информационное сообщение" — это содержимое самого лог-записи.

Тег можно задать один раз для всего приложения, сохранив его в переменной, что упрощает структурирование логов и их анализ.

Для корректности работы программы в начале перезаписываемого метода нужно вызвать супер-метод родительского класса через «super.onStart()».

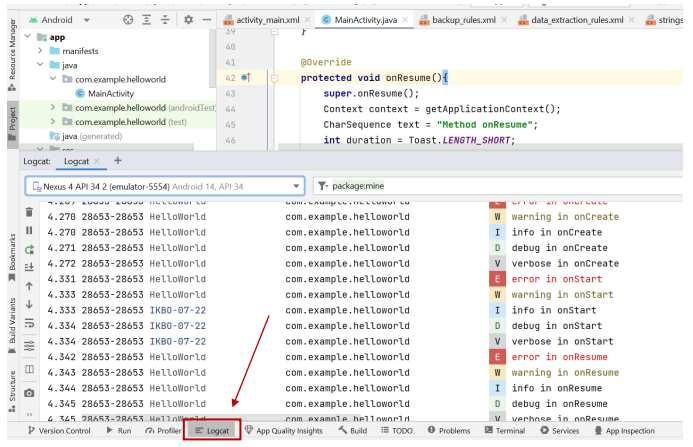
Пример задания пользовательского тега и логирования показан на рисунке 3.

  
Рисунок 3 – Пример использования пользовательского тега в логировании

Сообщения, записанные с использованием класса Log, отображаются в панели Logcat — инструменте для просмотра и анализа логов, встроенном в среду разработки Android Studio.

Logcat позволяет фильтровать сообщения по уровню логирования, тэгу или тексту, что облегчает процесс отладки и диагностики работы приложения.

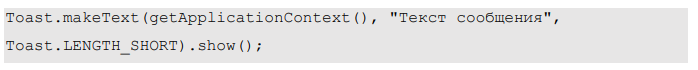
Интерфейс панели Logcat представлен на рисунке 4.

  
Рисунок 4 – Интерфейс панели Logcat, выделена кнопка для открытия панели

С другой стороны, Toast в Android предназначен для кратковременного отображения всплывающих сообщений пользователю. Эти уведомления автоматически исчезают через короткий промежуток времени и не требуют от пользователя какого-либо взаимодействия.

Toast используется для вывода простых уведомлений, таких как подтверждение успешного выполнения действия или предупреждение о некритической ошибке.

Пример использования Toast представлен на рисунке 5.

  
Рисунок 5 – Пример использования Toast

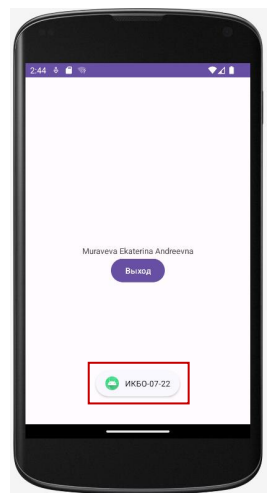
Где "Текст сообщения" представляет собой содержимое уведомления, отображаемого пользователю, а параметр Toast.LENGTH\_SHORT определяет короткую продолжительность его показа.

Данный параметр можно изменить на Toast.LENGTH\_LONG, если требуется более длительное отображение сообщения.

Пример использования Toast внутри события onResume показан на рисунке 6.

  
Рисунок 6 – Пример использования Toast внутри события onResume

При открытии активности появится короткое оповещение, продемонстрированное на рисунке 7.

  
Рисунок 7 – Пример отображения оповещения Toast, выделено оповещение

Основное различие между Log и Toast заключается в их назначении и способе использования. Log предназначен для записи отладочной информации, которая доступна только разработчикам через Logcat, и не отображается в пользовательском интерфейсе приложения. Он используется для диагностики работы программы, выявления ошибок и анализа её поведения.

Toast, в свою очередь, предназначен для кратковременного информирования пользователя. Он отображает всплывающее сообщение на экране, которое исчезает через определённое время без необходимости взаимодействия.

Кроме того, сообщения Log могут сохраняться в Logcat на длительный срок и использоваться для последующего анализа, тогда как уведомления Toast отображаются только в течение короткого времени и не сохраняются в системе.

## Взаимодействие с элементами пользовательского интерфейса

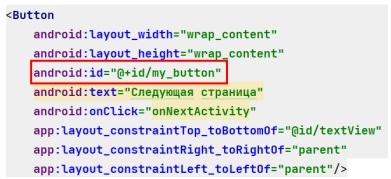
Взаимодействие с элементами пользовательского интерфейса в Android-приложениях осуществляется с помощью специальных методов и механизмов. Задать обработчик события для кнопки можно двумя способами:

* декларативно, с использованием атрибута onClick в XML-разметке;
* программно, в коде приложения, используя метод setOnClickListener.

При программном способе задания обработчика событий ключевую роль играют два метода:

* findViewById — используется для получения ссылки на элемент интерфейса по его уникальному идентификатору, указанному в XML-разметке;
* setOnClickListener — устанавливает обработчик события для кнопки, позволяя выполнить определённое действие при нажатии.

Пусть в XML есть кнопка с идентификатором «@+id/my\_button», как показано на рисунке 8.

  
Рисунок 8 – Кнопка с идентификатором «@+id/my\_button», выделен атрибут идентификатора

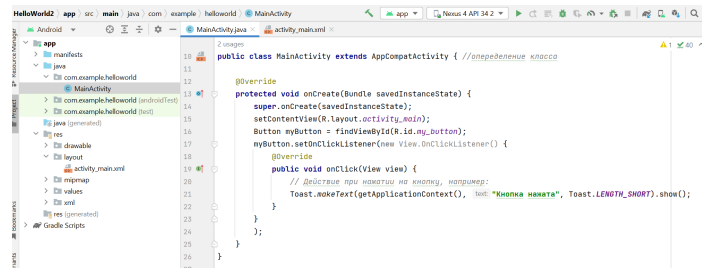
Тогда для получения ссылки на эту кнопку в Java используется код, показанный на рисунке 9.

  
Рисунок 9 – Код для получения кнопку с идентификатором «@+id/my\_button»

После получения ссылки на элемент интерфейса можно взаимодействовать с ним, изменяя его свойства, вызывая методы и выполняя другие операции.

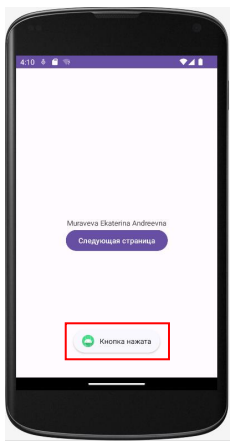
В свою очередь, setOnClickListener — это метод, предназначенный для установки обработчика нажатий на виджет, такой как кнопка. После того как кнопка была найдена с помощью findViewById, можно задать для неё слушатель нажатий.

Слушатель определяет действия, которые будут выполняться при нажатии на кнопку. Пример использования представлен на рисунке 10.

  
Рисунок 10 – Пример использования слушателя

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

Пример сообщения при нажатии на кнопку показан на рисунке 11.

  
Рисунок 11 – Пример всплывающего сообщения при нажатии на кнопку (программно)

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

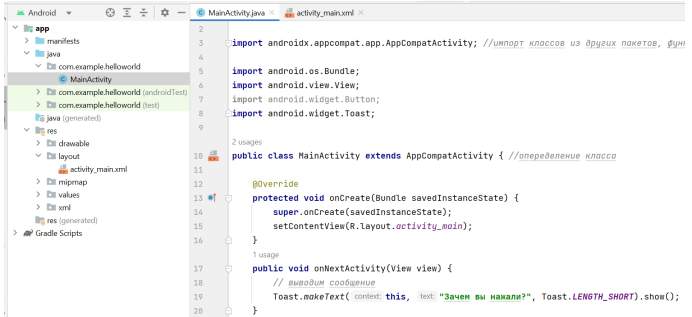
Одним из возможных сценариев использования слушателя нажатий является переход между разными экранами в приложении.

Относительно новым способом взаимодействия с элементами пользовательского интерфейса, специально разработанным для Android, является использование атрибута onClick: «android:onClick=”onNextActivity”».

Имя для обработчика события можно выбрать любое, но рекомендуется придерживаться простых и понятных названий.

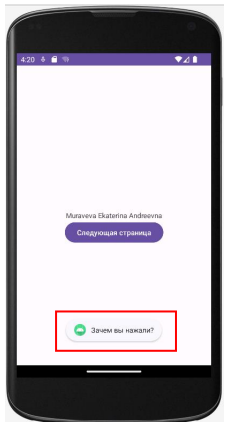
После этого необходимо определить в классе активности метод с указанным именем, который будет обрабатывать нажатие. Этот метод должен быть открытым (public) и принимать в качестве параметра объект View.

Пример объявления метода обработчика показан на рисунке 12.

  
Рисунок 12 – Пример объявления метода обработчика нажатия кнопки

Когда пользователь нажимает на кнопку, вызывается метод onNextActivity, который обрабатывает событие нажатия и выполняет заданное действие. В данном случае он генерирует всплывающее сообщение.

Пример работы метода представлен на рисунке 13.

  
Рисунок 13 – Пример работы обработчика нажатия кнопки (декларативно)

Обратите внимание, что при таком подходе нет необходимости объявлять кнопку с помощью конструкции Button myButton = findViewById(R.id.my\_button), поскольку Android автоматически связывает обработчик с соответствующим элементом интерфейса.

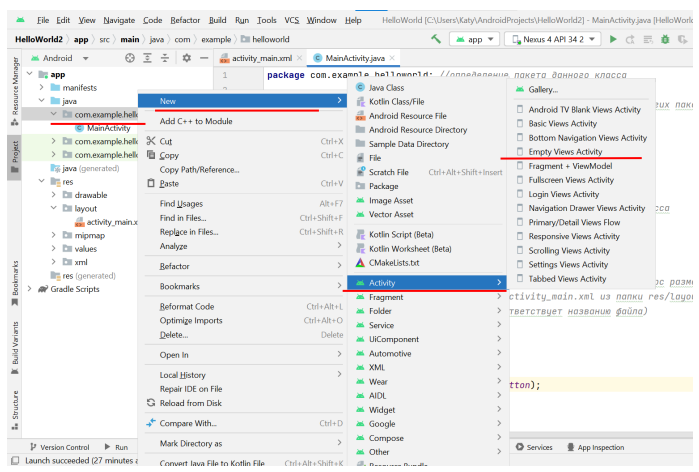
Этот способ можно применять не только к кнопкам, но и к другим элементам интерфейса, что позволяет уменьшить количество строк кода и упростить обработку событий.

## Переход между экранами

Для работы с несколькими экранами в Android-приложении часто требуется создание новых активностей. Каждая активность представляет собой отдельный экран с пользовательским интерфейсом.

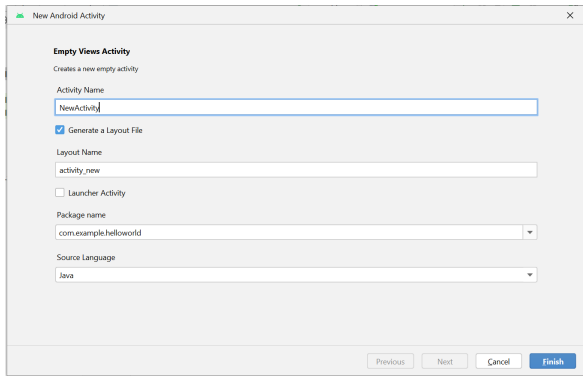
Чтобы добавить новую активность в проект Android Studio, необходимо выполнить несколько шагов. Сначала в панели проекта выбрать модуль app, затем в верхнем меню нажать File, затем New, затем Activity и выбрать нужный тип активности, например Empty Views Activity. Альтернативный способ — нажать правой кнопкой мыши на папку java, затем выбрать New, затем Activity. После этого Android Studio автоматически создаст файлы, необходимые для новой активности, включая XML-файл разметки и класс активности.

На рисунке 14 представлено меню создания новой активности, доступное через верхнее меню или контекстное меню папки Java.

  
Рисунок 14 – Путь к созданию новой активности, выделены нужные элементы

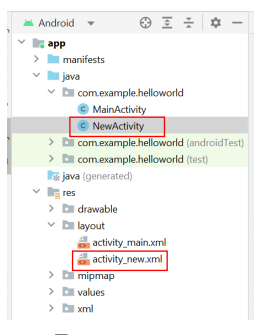
После выбора типа активности откроется окно с параметрами. В этом окне необходимо задать имя активности, её пакет, а также убедиться, что выбран правильный шаблон. Кроме того, можно указать родительскую активность и назначить заголовок окна. После заполнения всех полей следует нажать кнопку Finish для завершения процесса создания.

На рисунке 15 показано окно настройки параметров новой активности.

  
Рисунок 15 – Окно настройки параметров новой активности

После завершения создания активности в проекте появятся новые файлы. Android Studio автоматически добавит новый Java-класс или Kotlin-файл для логики активности, а также XML-файл для разметки пользовательского интерфейса. В файле AndroidManifest.xml будет добавлена новая запись, регистрирующая активность в приложении.

На рисунке 16 представлена структура проекта после создания новой активности.

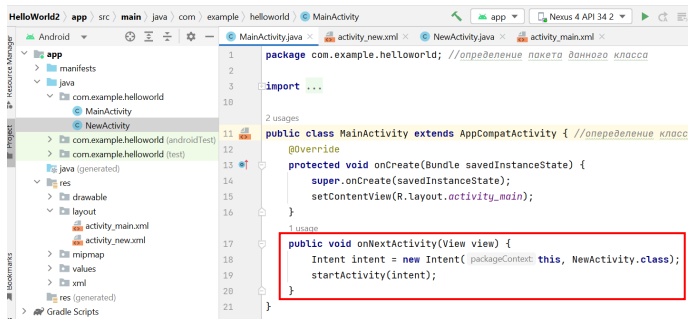
  
Рисунок 16 – Структура проекта после создания новой активности, выделены новые файлы

Переход между различными экранами в Android-приложениях осуществляется с помощью механизма, называемого Intent. Этот механизм играет ключевую роль в обеспечении взаимодействия между компонентами приложения, включая активности, которые представляют собой отдельные экраны.

Intent можно рассматривать как сообщение или запрос, определяющий намерение выполнить определённое действие. В контексте навигации между экранами Intent используется для запуска новой активности. Он не только инициирует переход на другой экран, но и позволяет передавать данные между активностями. Например, если приложение содержит список элементов и экран с подробной информацией о каждом из них, Intent может использоваться для открытия экрана с детальной информацией и передачи данных о выбранном элементе.

Создание Intent требует указания контекста, например текущей активности, и класса активности, которую необходимо запустить.

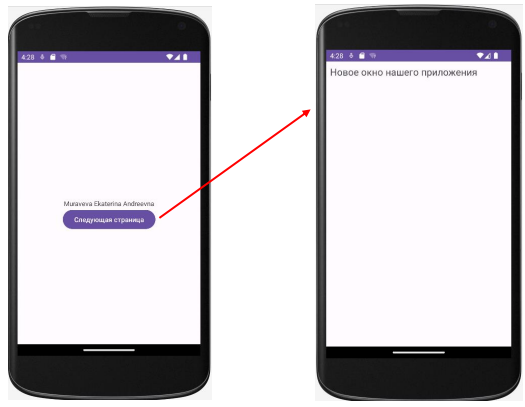
На рисунке 17 представлен пример перехода между активностями с помощью создания Intent.

  
Рисунок 17 – Пример метода перехода между активностями с помощью создания Intent, выделен код метода

Здесь this обозначает текущую активность, а NewActivity — ранее созданную активность, которую необходимо запустить. Далее метод startActivity инициирует выполнение действия, определённого в объекте Intent.

После выполнения этой строки система Android обрабатывает Intent и запускает указанную в нём активность. В результате пользовательский интерфейс переключается с текущей активности на интерфейс новой активности NewActivity.

На рисунке 18 показан процесс переключения между экранами с помощью создания Intent.

  
Рисунок 18 – Процесс переключения между экранами с помощью onClick

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

## Передача данных между Activity

Простой запуск активности может быть недостаточным для полноценного взаимодействия между экранами. В некоторых случаях необходимо передавать данные как в открываемую активность, так и обратно.

Для этого в классе Intent предусмотрена специальная область extraData. Она представляет собой список пар ключ/значение, которые передаются вместе с намерением. В качестве ключей используются строки, а в качестве значений можно передавать примитивные типы данных, массивы примитивов, объекты класса Bundle и другие структуры.

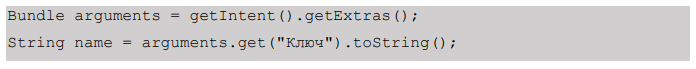
Для передачи данных в другую активность используется метод putExtra.

На рисунке 19 показан пример использования метода putExtra для передачи данных.

  
Рисунок 19 - Использование метода putExtra для передачи данных

Чтобы получить отправленные данные в NewActivity при её загрузке, используется метод get(), в который передаётся ключ объекта.

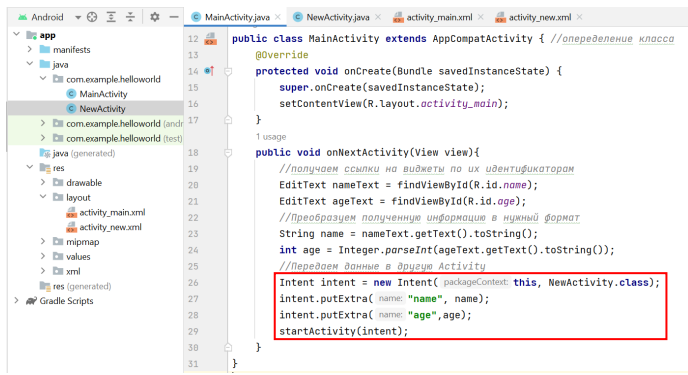
На рисунке 20 представлено получение переданных данных с использованием метода get().

  
Рисунок 20 - Получение переданных данных с использованием метода get()

В зависимости от типа переданных данных можно использовать различные методы объекта Bundle, которые принимают ключ в качестве параметра.

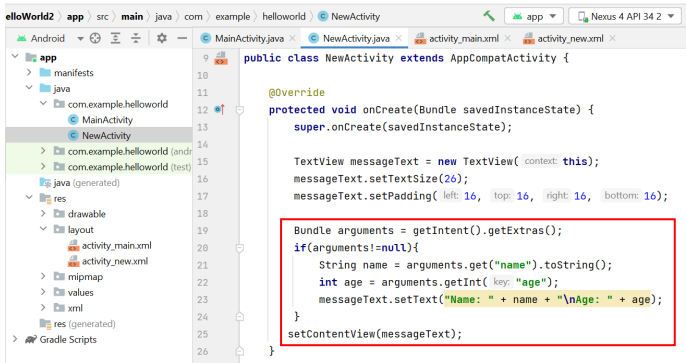
Допустим, у нас есть разметка с полями для ввода имени и возраста. Присваиваем этим полям идентификаторы name и age. После этого в MainActivity передаём введённые данные в другую активность с помощью метода putExtra.

На рисунке 21 показан код передачи данных между активностями

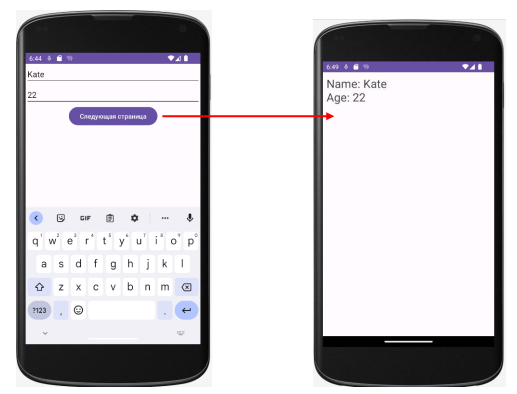
  
Рисунок 21 - Код передачи данных между активностями

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

На рисунке 22 показан процесс получения переданных данных и их валидации.

  
Рисунок 22 - Получение данных в NewActivity и проверка их содержимого

В результате передаем введенные данные из одного окна в другое.

  
Рисунок 23 – Результат создания новой активности с передачей данных

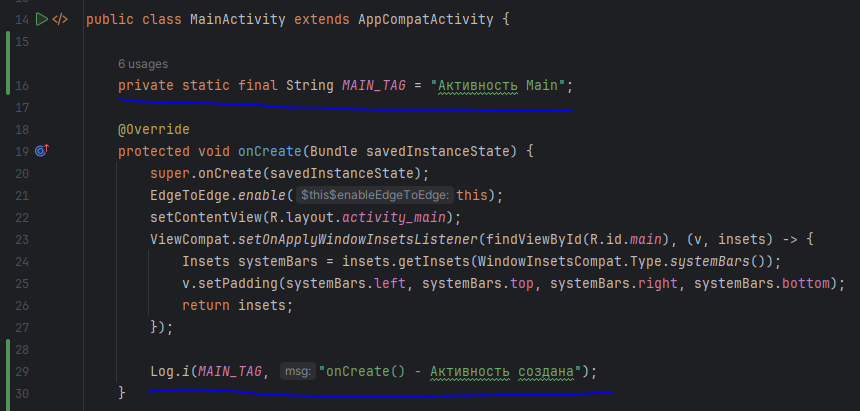
# ПРАКТИЧЕСКАЯ ЧАСТЬ



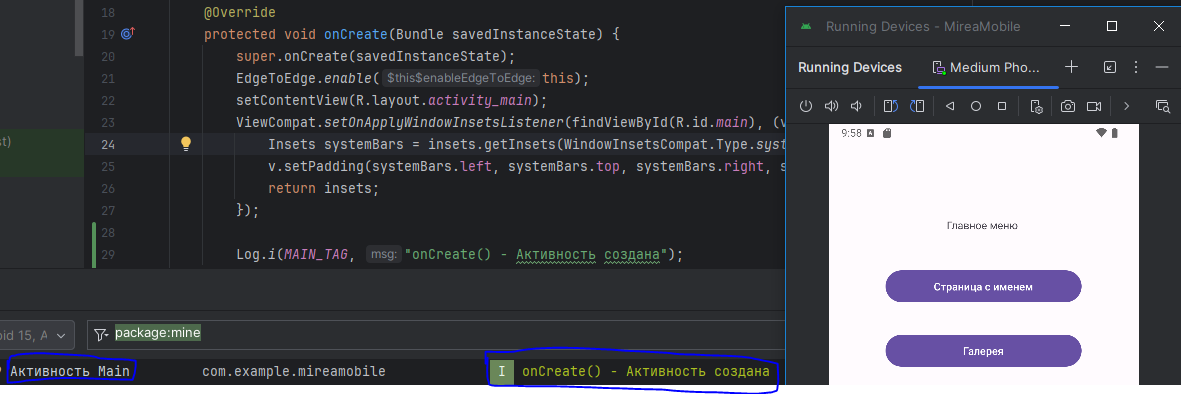
## Отслеживание работы жизненного цикла активности

### Этап onCreate

Для логирования создания активности создадим тег «MAIN\_TAG» с значением «Активность Main», обозначающий работу в MainActivity и добавим логирование в методе onCreate. Полученный код продемонстрирован на рисунке 24.

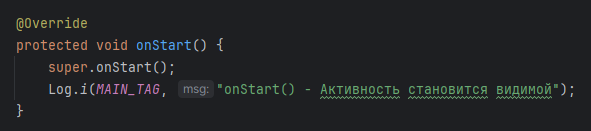
  
Рисунок 24 – Код для логирования метода onCreate, выделены тег и вызов логирования

Теперь при создании активности в Logcat будет отображаться информация, отображённая на рисунке 25.

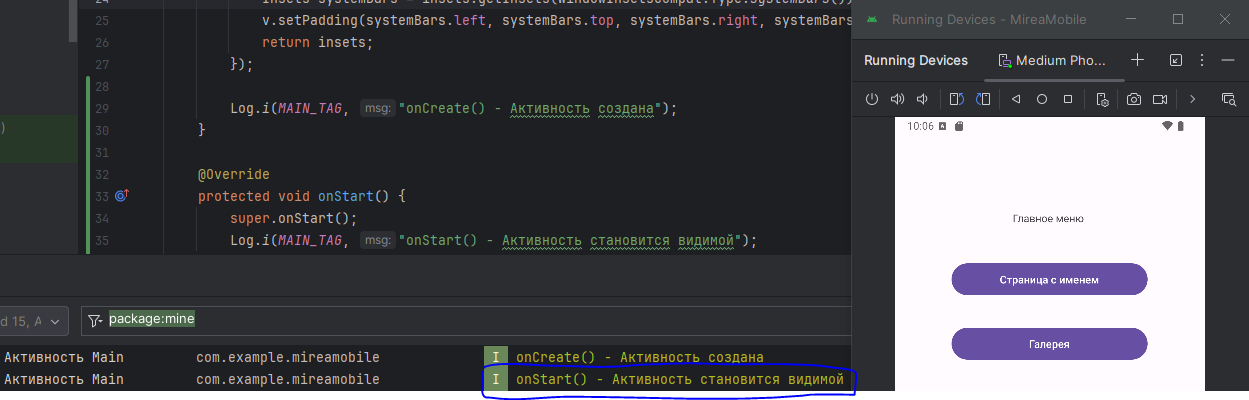
  
Рисунок 25 – Логирование этапа onCreate, выделены тег и текст лога

### Этап onStart

Для логирования этапа onStart нужно перезаписать метод onStart. Новый фрагмент кода показан на рисунке 26.

  
Рисунок 26 – Метод логирования метода onStart

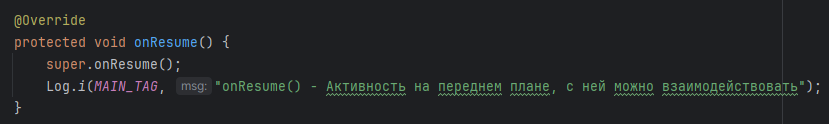
Теперь при открытии активности в Logcat будет отображаться информация, отображённая на рисунке 27.

  
Рисунок 27 – Логирование метода onStart

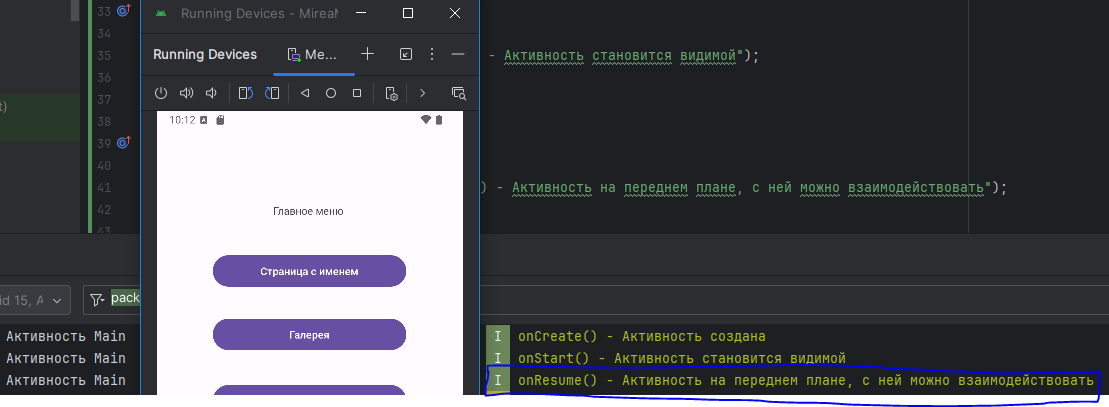
Также этот лог будет появляться каждый раз, когда активность становится видима пользователю.

### Этап onResume

Для логирования этапа onResume нужно перезаписать метод onResume. Новый фрагмент кода показан на рисунке 28.

  
Рисунок 28 – Код для логирования этапа onResume

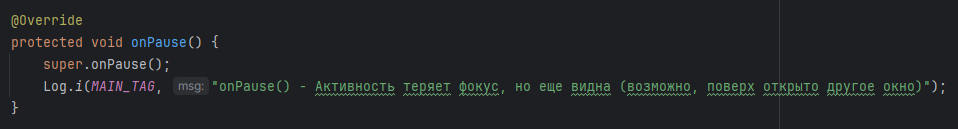
Теперь при возобновлении работы активности в Logcat будет отображаться информация, отображённая на рисунке 29.

  
Рисунок 29 – Логирование этапа onResume

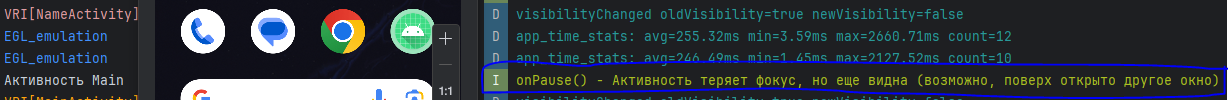
Этот лог будет появляться каждый раз, когда активность оказывается на первом плане и с ней можно взаимодействовать.

### Этап onPause

Для логирования этапа onPause нужно перезаписать метод onPause. Новый фрагмент кода показан на рисунке 30.

  
Рисунок 30 – Метод логирования метода onPause

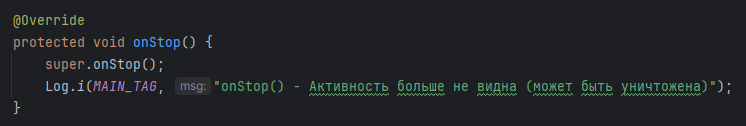
Теперь при смещении фокуса с активности (например, при закрытии приложения) в Logcat будет отображаться информация, отображённая на рисунке 31.

  
Рисунок 31 – Логирование метода onPause

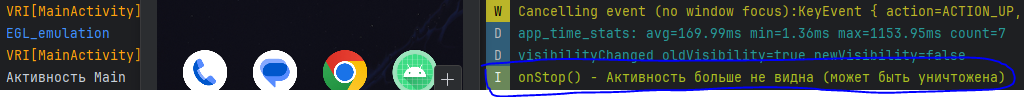
Также этот лог будет появляться каждый раз, когда активность теряет фокус, но всё ещё видима пользователю.

### Этап onStop

Для логирования этапа onStop нужно перезаписать метод onStop. Новый фрагмент кода показан на рисунке 32.

  
Рисунок 32 – Код для логирования этапа onStop

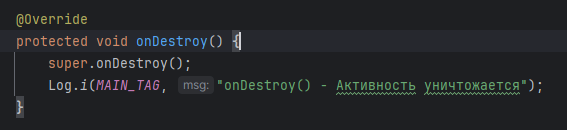
Теперь при скрытии активности (например, при закрытии приложения) в Logcat будет отображаться информация, отображённая на рисунке 33.

  
Рисунок 33 – Логирование этапа onStop

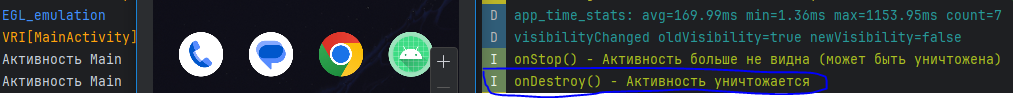
Этот лог будет появляться каждый раз, когда активность перестаёт быть видимой. Этот этап подготавливает активность к удалению её из памяти.

### Этап onDestroy

Для логирования этапа onDestroy нужно перезаписать метод onDestroy. Новый фрагмент кода показан на рисунке 34.

  
Рисунок 34 – Метод логирования метода onDestroy

Теперь при удалении активности из памяти (например, при закрытии приложения) в Logcat будет отображаться информация, отображённая на рисунке 35.

  
Рисунок 35 – Логирование метода onDestroy

Также этот лог будет появляться каждый раз, когда активность удаляется из памяти. Обычно перед этим этапом вызывается onPause или onStop.

Однако система может убить процесс приложения без вызовов onPause, onStop или onDestroy (например, из-за нехватки памяти).

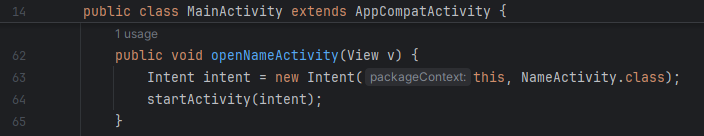
## Переход на активность с помощью кнопки



### Декларативный способ

Для реализации перехода от одной активности к другой, воспользуемся кнопкой и активностями, реализованными в рамках прошлой практической работы. Кнопка будет вести из главного меню (MainActivity) в активность с ФИО студента (NameActivity).

Для работы перехода необходимо написать обработчик события, в котором будет создан Intent. Код обработчика представлен на рисунке 36.

  
Рисунок 36 – Код обработчика события нажатия на кнопку

Далее в атрибут кнопки onClick нужно записать метод, который будет обработчиком события нажатия на кнопку (openNameActivity). Реализация в XML-коде показана на рисунке 37.

  
Рисунок 37 – XML-код кнопки, ведущей на активность с ФИО студента, выделен атрибут onClick

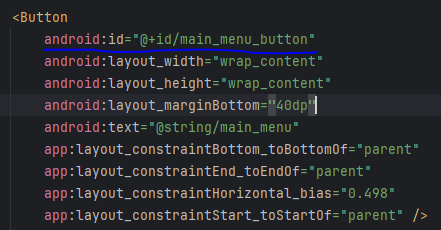
Теперь при нажатии на кнопку «Страница с именем» пользователя перенесёт на активность с ФИО студента. Открывшаяся после нажатия кнопки активность продемонстрирована на рисунке 38.

  
Рисунок 38 – Активность, на которую переносит пользователя при нажатии на кнопку «Страница с именем»

### Программный способ

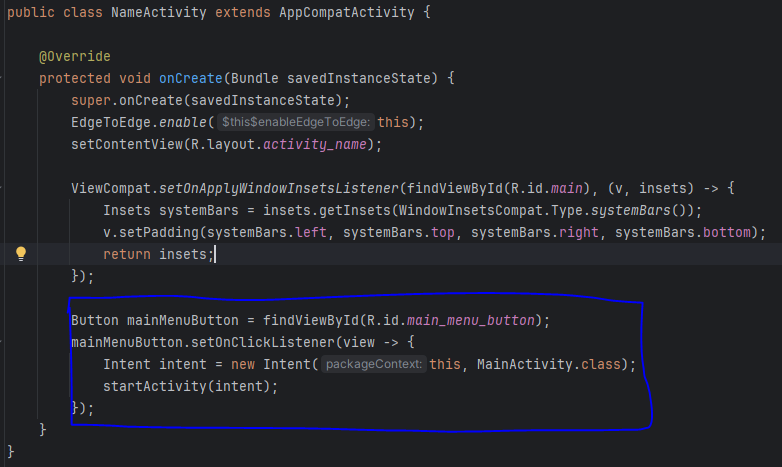
Кнопку «Главное меню», ведущую из активности с ФИО студента обратно в главное меню, реализуется уже программным способом.

Сперва реализуется верстка кнопки и указывается идентификатор кнопки («main\_menu\_button»). Полученный XML-код отображен на рисунке 39.

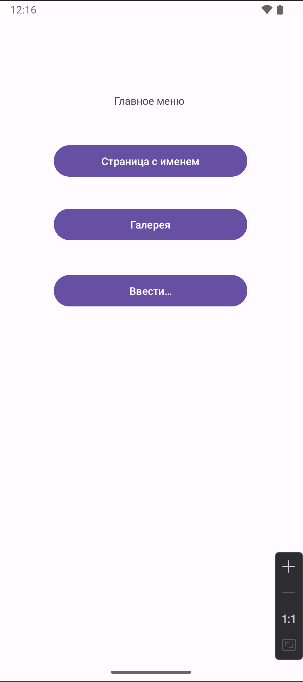
  
Рисунок 39 – XML-код кнопки, ведущей обратно в главное меню, выделен атрибут идентификатора кнопки

Для работы кнопки нужно добавить слушатель к этой кнопке. Его можно добавить в методе onCreate. Для этого необходимо найти нужную кнопку через «R.id.main\_menu\_button» и добавить к нему слушатель, который будет создавать новый Intent, перенося пользователя обратно в активность MainActivity.

Реализованный код активности показан на рисунке 40.

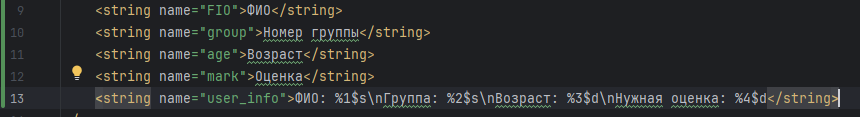
  
Рисунок 40 – Java-код активности NameActivity, выделен фрагмент кода, отвечающий за создание слушателя нажатия на кнопку

В результате при нажатии кнопки «Главное меню» пользователя вернёт в главное меню. Открывшаяся после нажатия кнопки активность продемонстрирована на рисунке 41.

  
Рисунок 41 - Активность, на которую переносит пользователя при нажатии на кнопку «Главное меню»

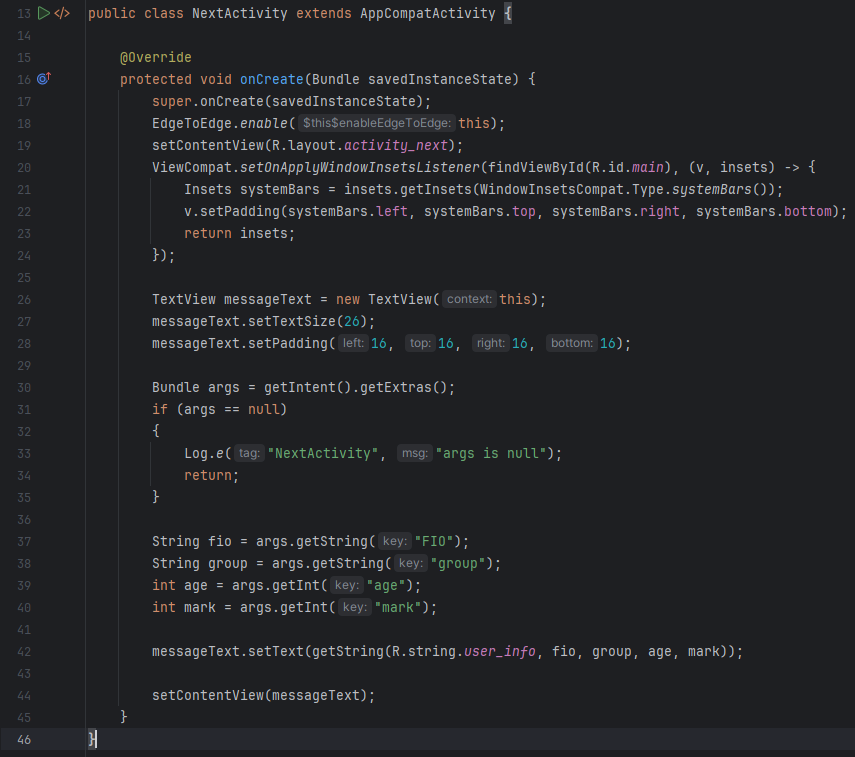
## Переход в активность с передачей данных

Сперва для удобства редактирования нужно добавить заготовленные строки в файл strings.xml, которые понадобятся во время разработки. Новые строки представлены на рисунке 42.

  
Рисунок 42 – Добавленные заготовленные строки

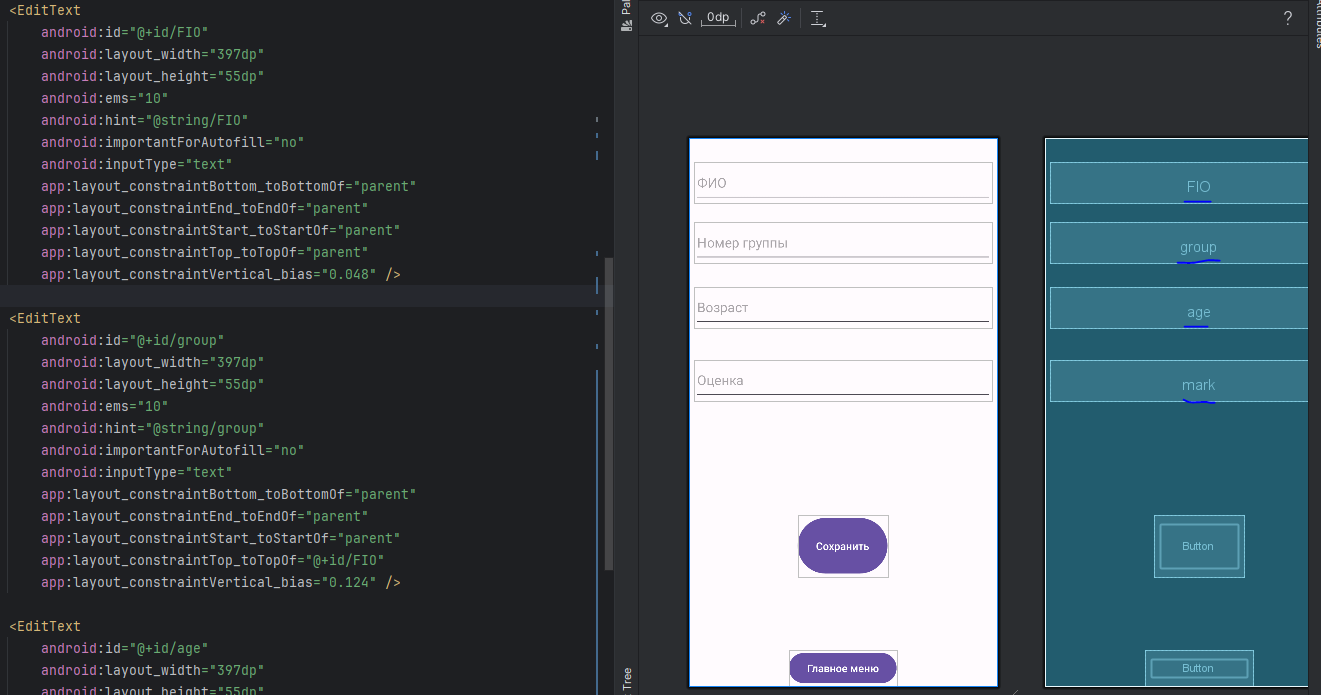
Нужно создать пустую активность NextActivity. Она будет принимать аргументы «FIO», «group», «age», «mark» при запуске и показывать их пользователю.

На рисунке 43 продемонстрирован Java-код активности NextActivity.

  
Рисунок 43 – Java-код активнсти NextActivity

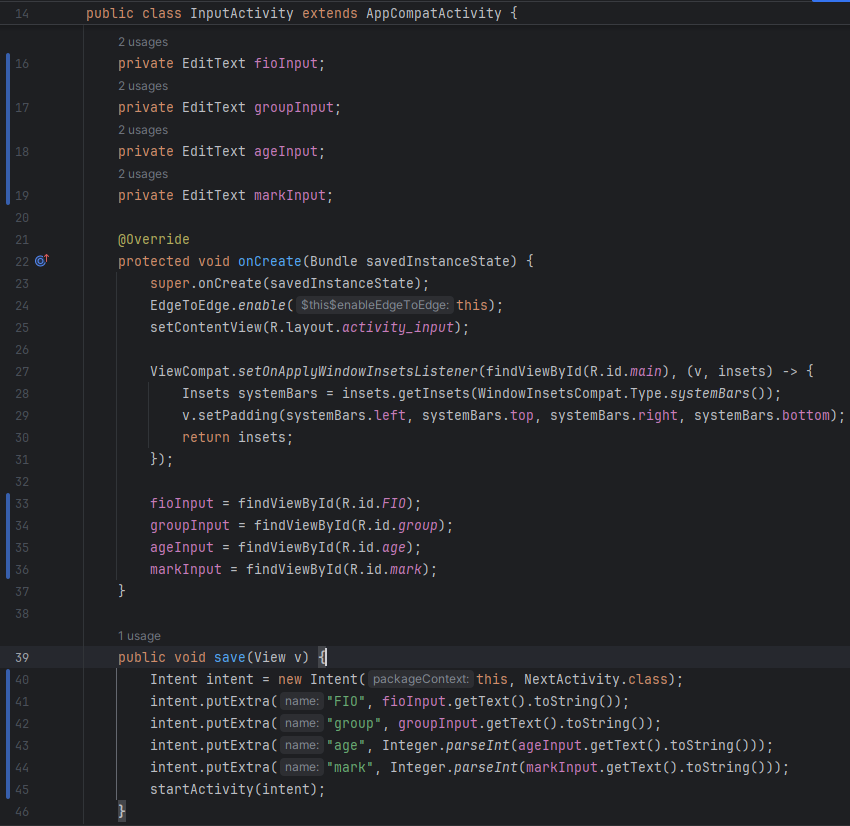
В этом коде при создании активности создаётся элемент TextView, в который вносятся данные, введённые пользователем.

Чтобы протестировать работу новой активности, были изменены поля у активности ввода. Были добавлены новые поля для ввода ФИО, номера группы, возраста и оценки. Модель изменённой активности ввода показана на рисунке 44.

  
Рисунок 44 – Изменённая модель активности ввода, выделены идентификаторы новых полей

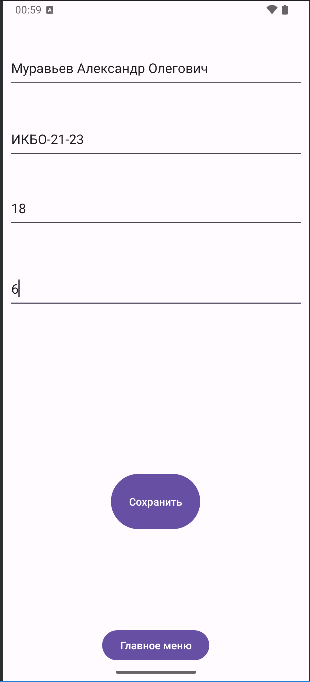
Для создания новой активности с введёнными данными нужно при нажатии кнопки «Сохранить» считать данные полей и создать Intent с добавлением этих полей.

Java-код активности InputActivity представлен на рисунке 45.

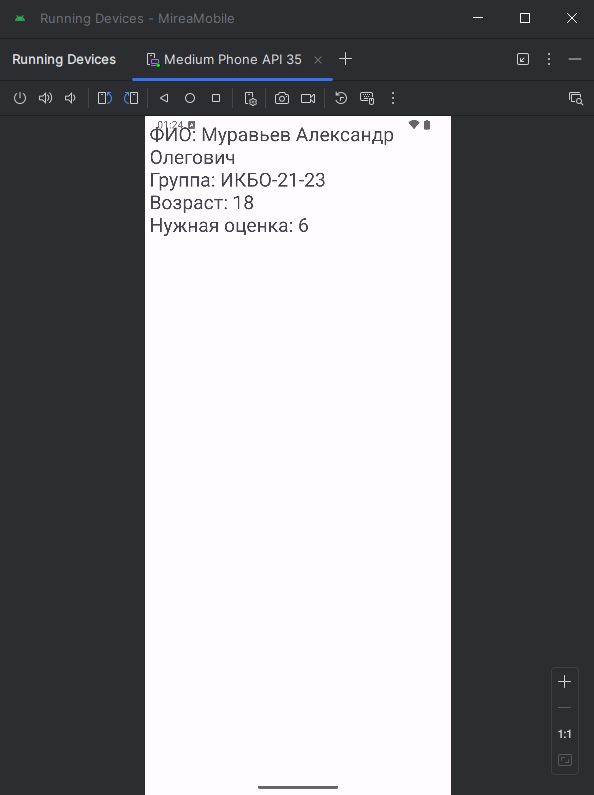
  
Рисунок 45 – Java-код активности InputActivity

Метод save декларативно закреплён за кнопкой «Сохранить».

Для проверки нужно ввести данные студента в поля в приложении. Заполненная форма показана на рисунке 46.

  
Рисунок 46 – Заполненная форма с информацией о студенте

После нажатия кнопки «Сохранить» создаётся активность «NextActivity» с введёнными данными, что продемонстрировано на рисунке 47.

  
Рисунок 47 – Создавшаяся активность с данными студента

ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы были получены навыки работы с логированием и активностями в Android-приложениях, а также их взаимодействием с помощью механизма Intent. Были освоены методы обработки событий пользовательского интерфейса, включая программное назначение обработчиков нажатий через setOnClickListener.

Кроме того, была изучена передача данных между активностями и их последующее получение, что позволяет реализовывать динамическое взаимодействие между экранами приложения.

Таким образом, в рамках данной работы были закреплены базовые знания по навигации в Android-приложениях и взаимодействию между компонентами.

В результате практической работы было модифицировано мобильное приложение, которое получило новый функционал логирования и перехода между активностями в том числе с передачей данных между активностями.