|  |
| --- |
|  |

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

|  |
| --- |
|  |

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2**

**Работа с Activity и верстка**

по дисциплине

**«Разработка мобильных приложений»**

|  |  |
| --- | --- |
| Выполнил студент группы ИНБО-03-22 | Алтухов А.В. |
|  |  |
| Принял ассистент кафедры МОСИТ | Шешуков Л.С. |

|  |  |  |
| --- | --- | --- |
| Практическая  работа выполнена | «\_\_» \_\_\_\_\_\_\_\_\_2024 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |
| «Зачтено» | «\_\_» \_\_\_\_\_\_\_\_\_2024 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |

Москва 2024

СОДЕРЖАНИЕ

[1. ПОСТАНОВКА ЗАДАЧИ 3](#_Toc160734727)

[2. ВЕРСТКА ПРИЛОЖЕНИЯ 4](#_Toc160734728)

[3. РАБОТА С РЕСУРСАМИ 7](#_Toc160734729)

[4. ВЗАИМОДЕЙСТВИЕ ПРОГРАММНОГО КОДА С ВЕРСТКОЙ 11](#_Toc160734730)

[5. СОЗДАНИЕ НОВОГО ACTIVITY И ПЕРЕДАЧА ДАННЫХ 12](#_Toc160734731)

[6. ВЫВОДЫ 17](#_Toc160734732)

[7. ПРИЛОЖЕНИЕ 1 18](#_Toc160734733)

# ПОСТАНОВКА ЗАДАЧИ

1. **Создать верстку приложения на основе предметной области.**

* Реализовать несколько файлов разметки с применением следующих контейнеров: LinerLayout, RelativeLayout, Constraint Layout;
* Добавить в созданные файлы разметки базовые компоненты: Текст, Кнопка, Картинка.

**2. Работа с ресурсами.**

* Добавить в проект несколько строковых, размерных, цветовых и drawable ресурсов.
* Добавить в проект несколько изображений png и svg.
* Подключить созданные ресурсы в подготовленную на прошлом шаге верстку.

**3.** Произвести инициализацию нескольких компонентов в верстке данными из программного кода. Произвести инициализацию текстового компонента строковым ресурсом и компонента картинки – ресурсом картинки.

**4.** Произвести вывод сообщение в Log при нажатии на клавишу. Задать метод обработчик события двумя способами: декларативно и программно.

**5.** Создать второе Activity на основе предметной области. Произвести открытие второй Activity с передачей в нее данных, обоснованных предметной областью. При закрытии второй Activity, вернуть данные в изначальную, обоснованные предметной областью.

# ВЕРСТКА ПРИЛОЖЕНИЯ

Для размещения view-компонентов используются специальные контейнеры (ViewGroup), называемые Layout. **Layout** бывают различных типов (LinearLayout, RelativeLayout, FrameLayout, TableLayout, ConstraintLayout и т.д.) и отвечают за то, как будут расположены их содержимое (дочерние view-компоненты) на экране – таблицей, строкой, столбцом.

Android Studio по умолчанию использует **ConstraintLayout** в качестве корневого контейнера для создания разметки экрана и размещения компонентов. Данный контейнер обладает широким спектром возможностей, что позволяет реализовывать сложные взаиморасположения элементов на экране. Для добавления на экран требуемого компонента необходимо найти его в списке Palette и переместить мышкой на экран в режиме Design или режиме Blueprint. Добавим в Constraint Layout компонент текста. После этого компонент появится на обоих экранах, а также в окне Component Tree (Рисунок 2.1).

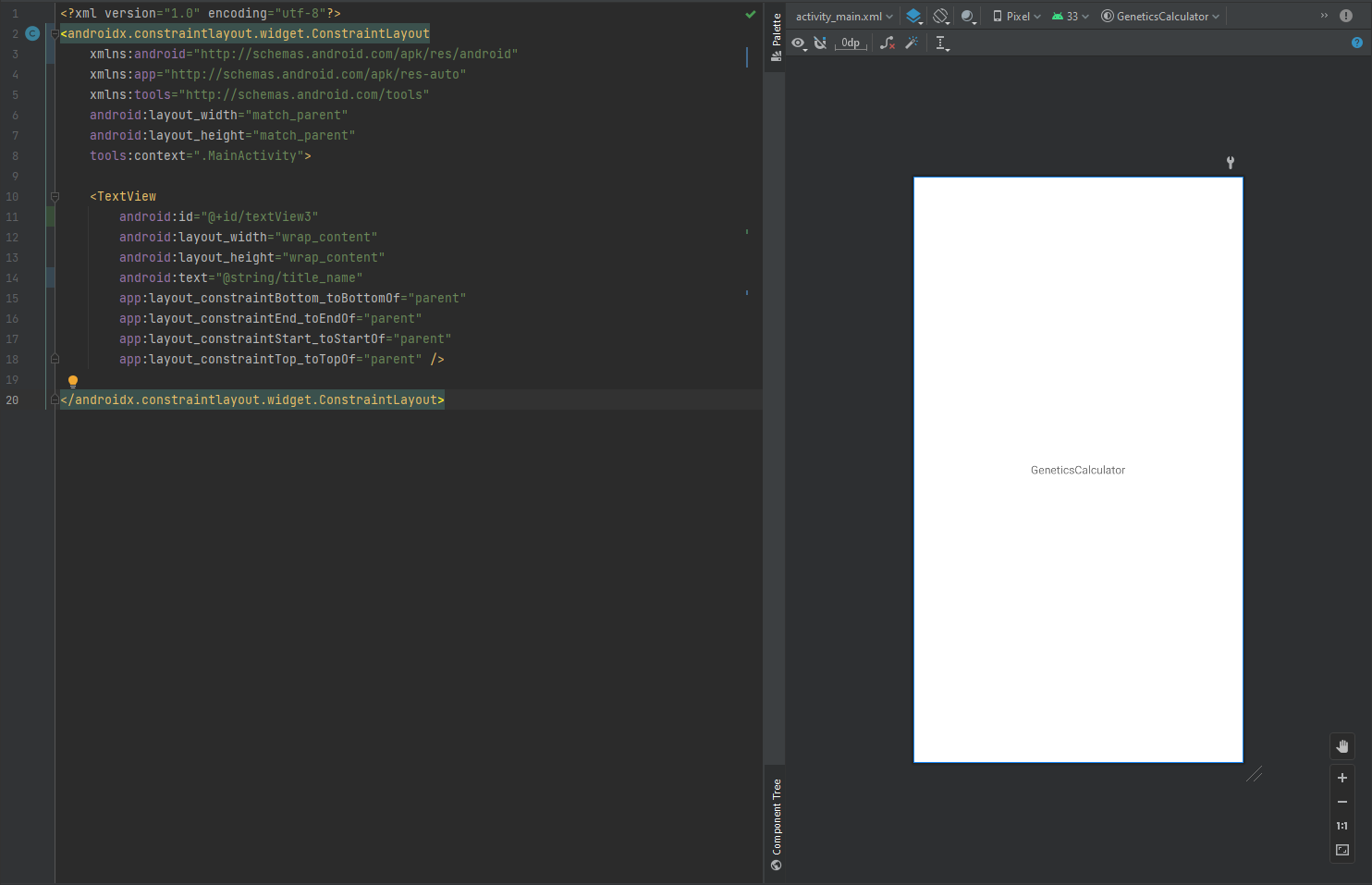


Рисунок 2.1 — Добавление в Constraint Layout компоненты TextView

Квадратные опорные точки в углах компонента позволяют изменять его размеры. Круглые опорные точки, расположенные по сторонам виджета, позволяют создавать привязки (constraints) к сторонам контейнера или другим компонентам и управлять отступами от краев экрана и других компонентов.

Для создания своей собственной привязки необходимо нажать круглую опорную точку на одной из сторон виджета, которую требуется привязать. Далее, не отпуская кнопку мыши, прочертить линию к виджету или краю контейнера. В результате будет создана связь constraint. Создадим LinerLayout контейнер и добавим в него кнопку (Рисунок 2.2).

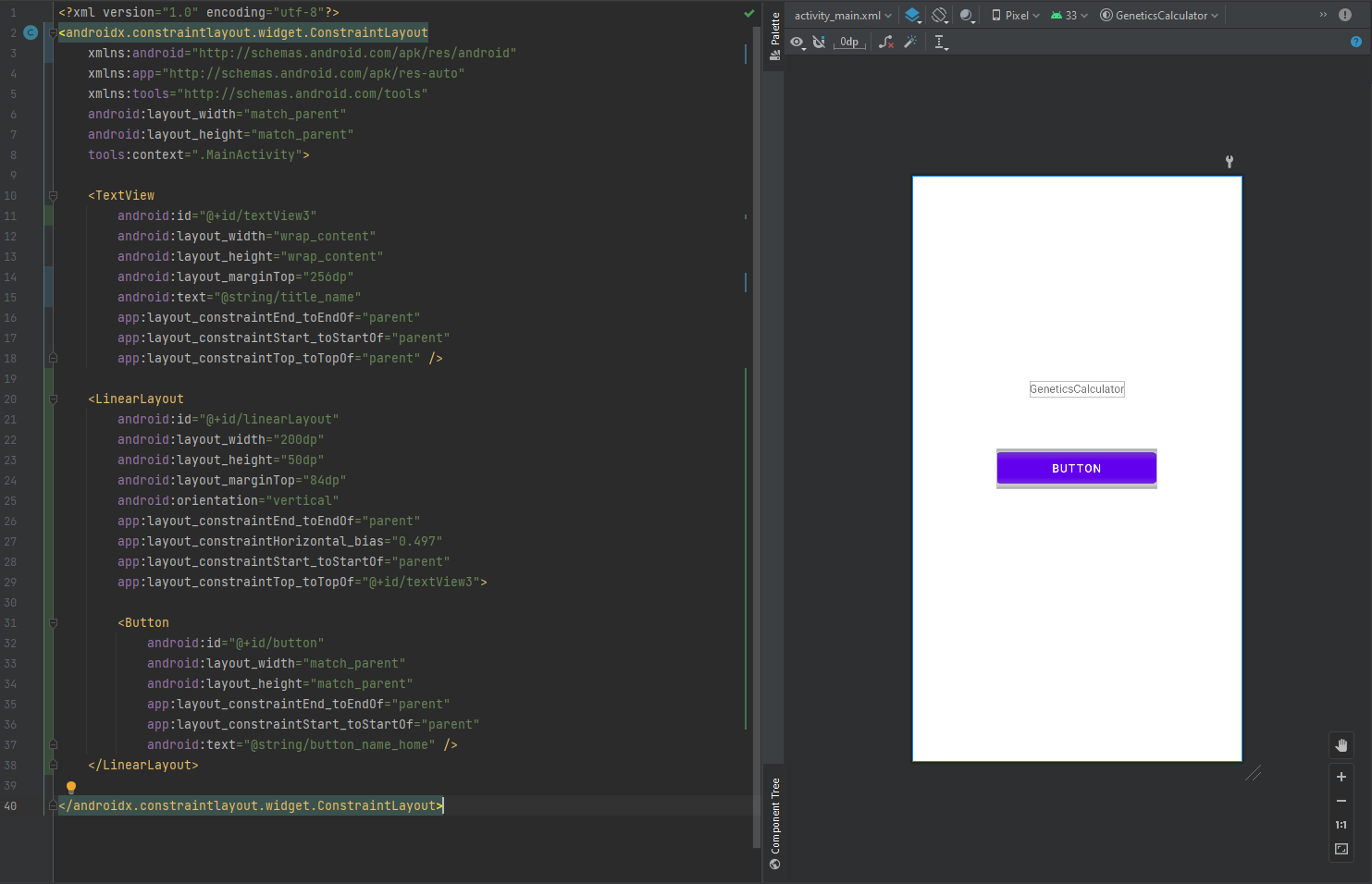


Рисунок 2.2 — Добавление в LinerLayout компоненты Button

Теперь доведем наш макет “до идеала”, разнообразив его различными компонентами, а также изменив текста у кнопок (Рисунки 2.3-2.4).

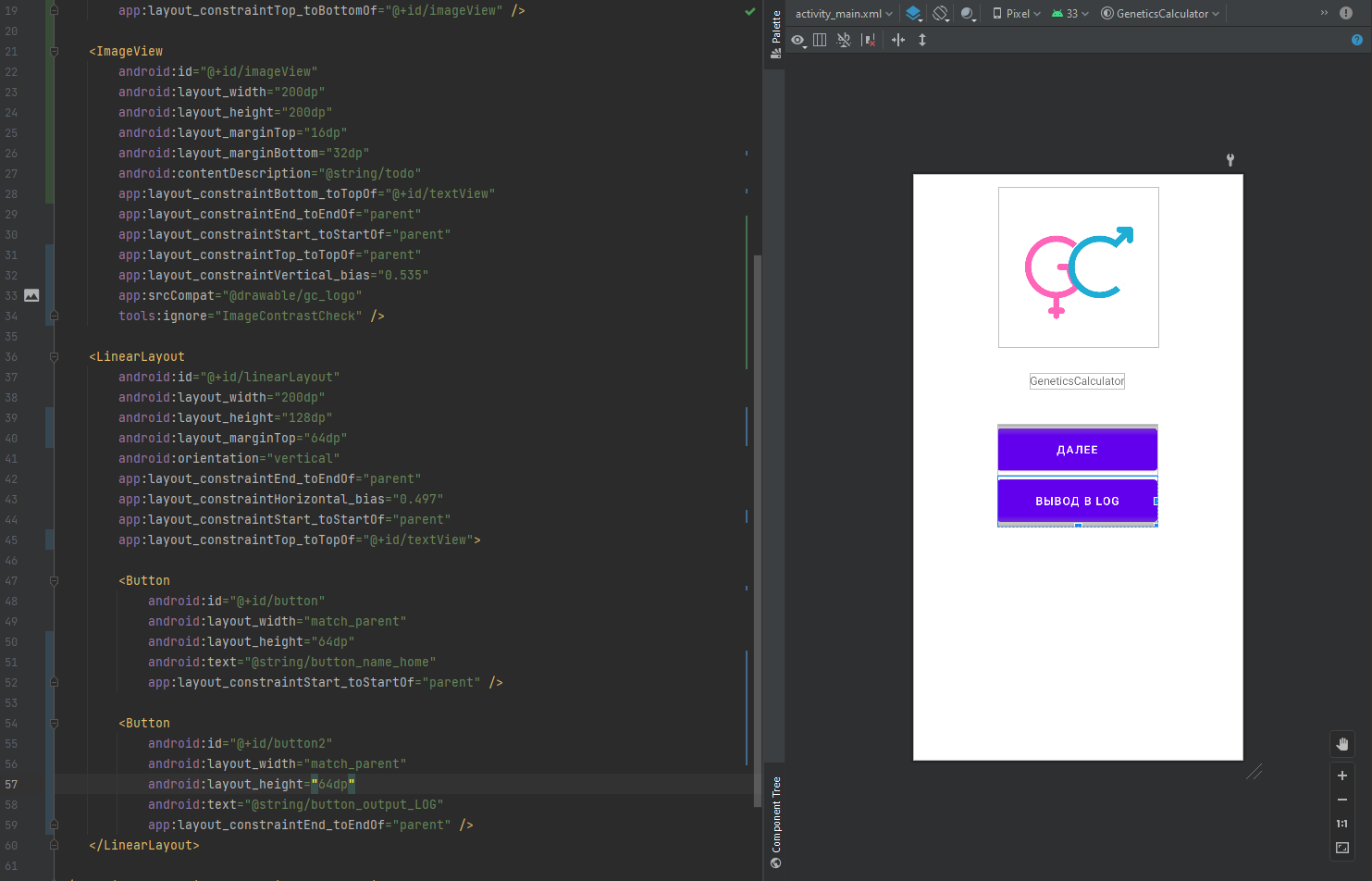


Рисунок 2.2 — Добавление компонентов Image и Button

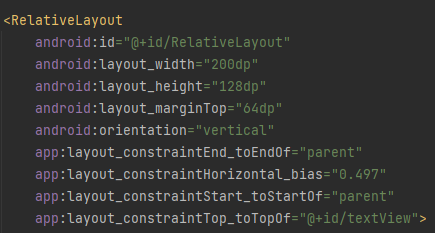


Рисунок 2.3 — Реализация RelativeLayout

# 3. РАБОТА С РЕСУРСАМИ

В структуре проекта особую роль играет папка res (app\src\main\res\). Она используется для файлов-ресурсов различного типа например:

* res/**drawable/** – для изображений (PNG, JPEG и т.д.), адаптированных для различных разрешений экрана;
* res/**layout/** – для xml-файлов графической разметки;
* res/**mipmap/** – для служебных иконок (значков);
* res/**values/** – для хранения различных переменных – строки, массивы и пр.

Для хранения строк, цветов, стилей и звуков и многократного обращения к ним в проекте используются **строковые ресурсы**. Создадим новый строковый ресурс, в который напишем свое ФИО (Рисунок 3.1), также в ресурсы цвета добавим значения двух новых оттенков (Рисунок 3.2). Осталось добавить только размерный ресурс, для этого добавим в папку res/values новый элемент Values Resources File, который назовем “dimens.xml”, теперь определим в нем два ресурса для отступов **horizontal\_margin** и **vertical\_margin** которые хранят соответственно значения 64dp и 32dp, и ресурс **text\_size**, который хранит высоту шрифта - 32sp (Рисунок 3.3).

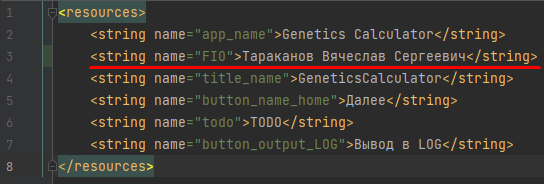


Рисунок 3.1 — Добавление строкового ресурса

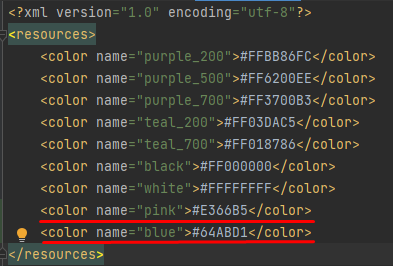


Рисунок 3.2 — Добавление ресурсов цвета

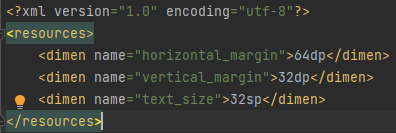


Рисунок 3.3 — Добавление ресурсов разметки

Теперь добавим в приложение наши ресурсы (Рисунки 3.4-3.5).

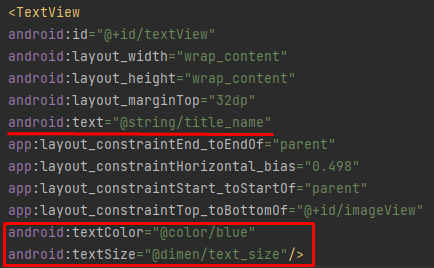


Рисунок 3.4 — Добавление ресурсов в код

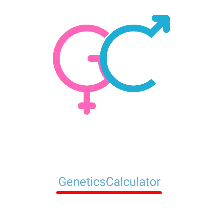


Рисунок 3.5 — Отображение в приложении

**Ресурсами приложения** могут быть и изображения – как растровые, так и векторные. С учетом разнообразия размеров и разрешений экранов мобильных устройств стоит отдавать предпочтение именно векторным изображениям, т.к. в них отсутствует эффект пикселизации.

В открытом проекте нажмем правой кнопкой по папке drawable. Здесь выбираем **New - Vector Asset**. Откроется окно мастера **Asset Studio**, в котором можно выбрать локальный файл или изображение из клипарта (Рисунок 3.6).

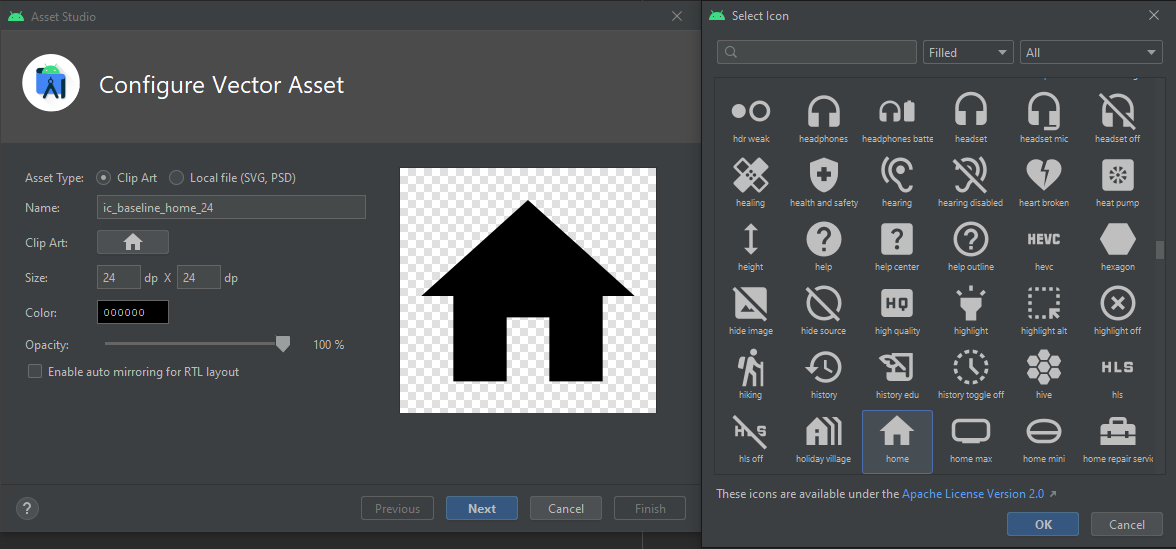


Рисунок 3.6 — Asset Studio

Выбираем векторный значок дома, так как он пригодится нам позже. Мастер создаст xml-файл этого ресурса, откроем его и изучим содержимое. В нём можно изменить вручную цвет и размер (Рисунок 3.7).

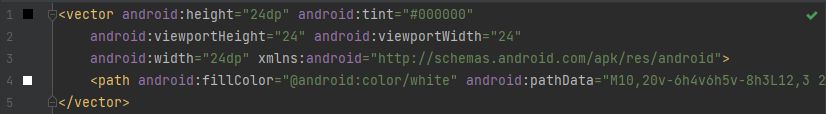


Рисунок 3.7 — xml-файл домика

Теперь добавим изображение, которое используем как логотип, по прошлому примеру, создадим Image Asset (Рисунок 3.8), далее в параметре Res Directory, сначала выбираем вариант – main, и нажимаем кнопку Finish. Но чтобы ваш новый значок, отображался во всех случаях, и не было показа старого стандартного значка, надо повторить все ранее указанные шаги, и в появившемся всплывающем меню выбираем New-Image Asset, но выбрать вариант - debug, затем снова повторяя ранние шаги, выбираем вариант – release (Рисунок 3.9).

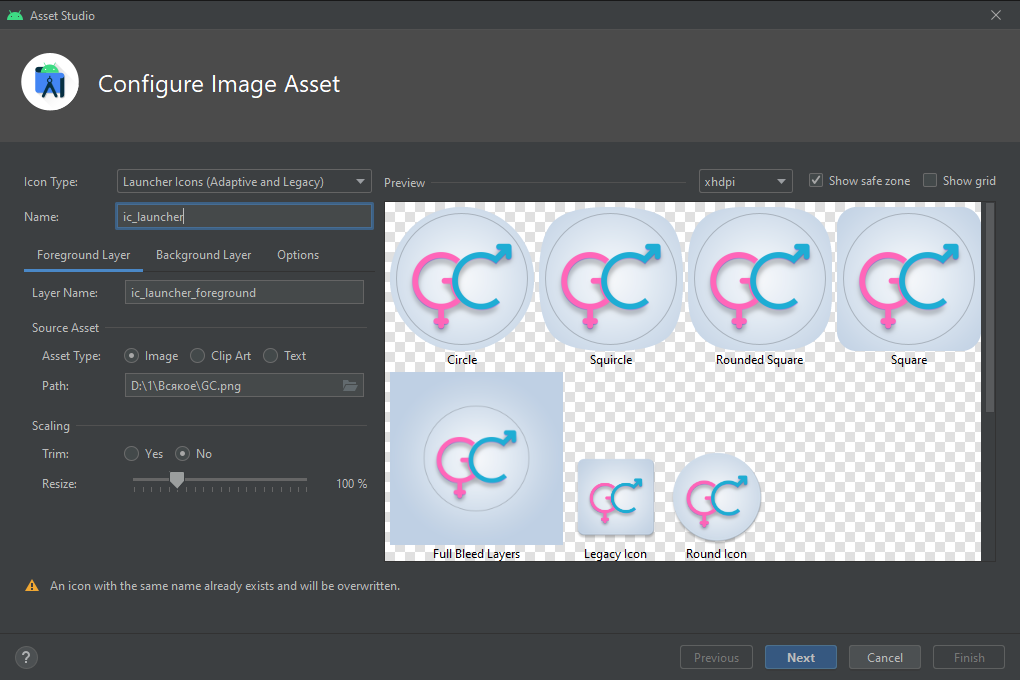


Рисунок 3.8 — Создание Image Asset

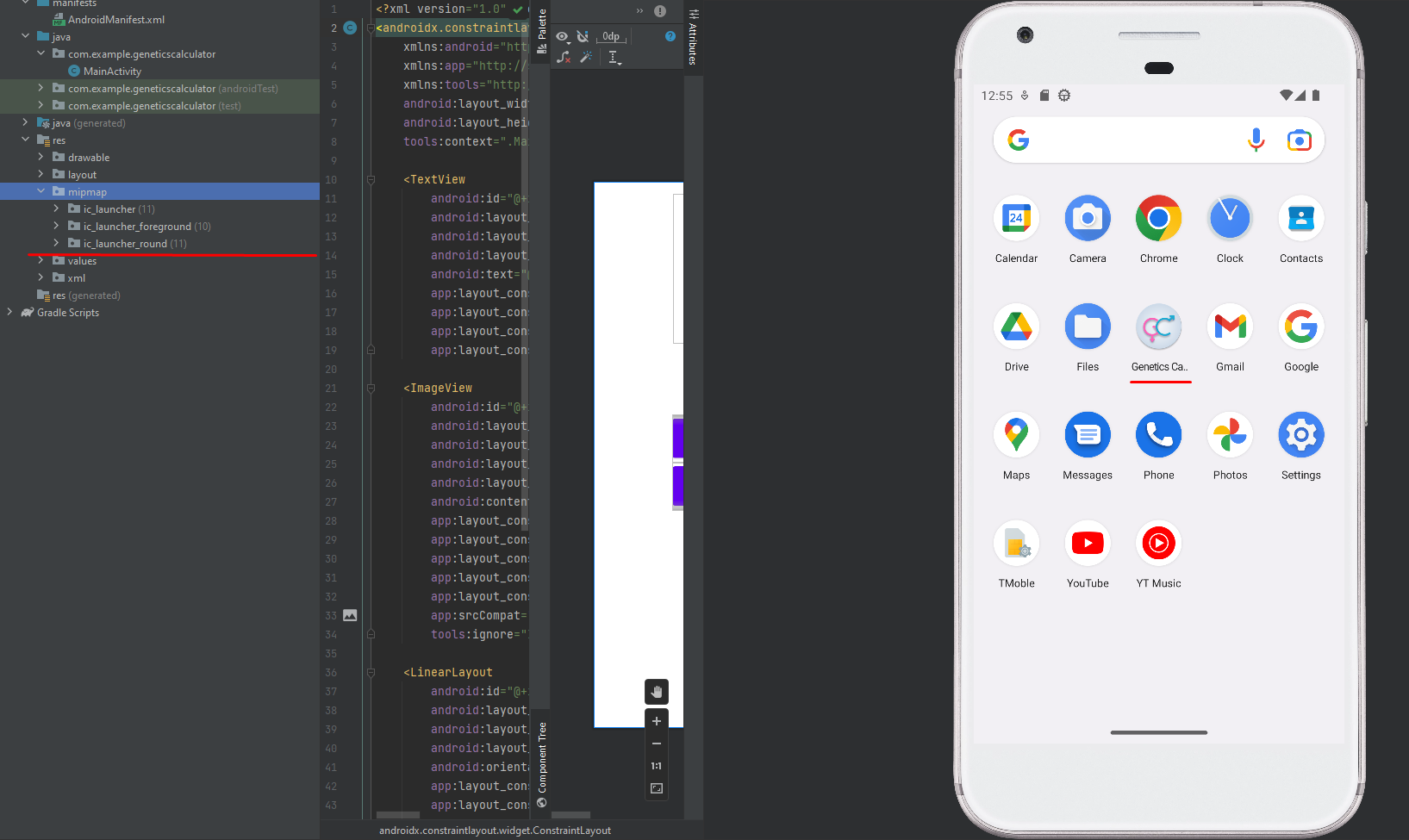


Рисунок 3.9 — Изменение логотипа приложения

1. **ВЗАИМОДЕЙСТВИЕ ПРОГРАММНОГО КОДА С ВЕРСТКОЙ**

По условию задания, необходимо добавить в наше приложение Log по нажатию на кнопку. А также задать метод обработчика события двумя способами: декларативно и программно.

Программный метод является более традиционным для разработки. Для того, чтобы создать прослушивателя нажатия на кнопку, в первую очередь необходимо получить доступ к компоненту кнопки на пользовательском интерфейсе. После чего, необходимо создать нового слушателя действий компонентов и подключить к полученной кнопке. Для этого необходимо создать объект класса OnClickListener и переназначить в нем абстрактный метод onClick. После того, как объект класса получен, его необходимо передать в качестве параметра ранее изученному методу setOnClickListener (Рисунок 4.1).

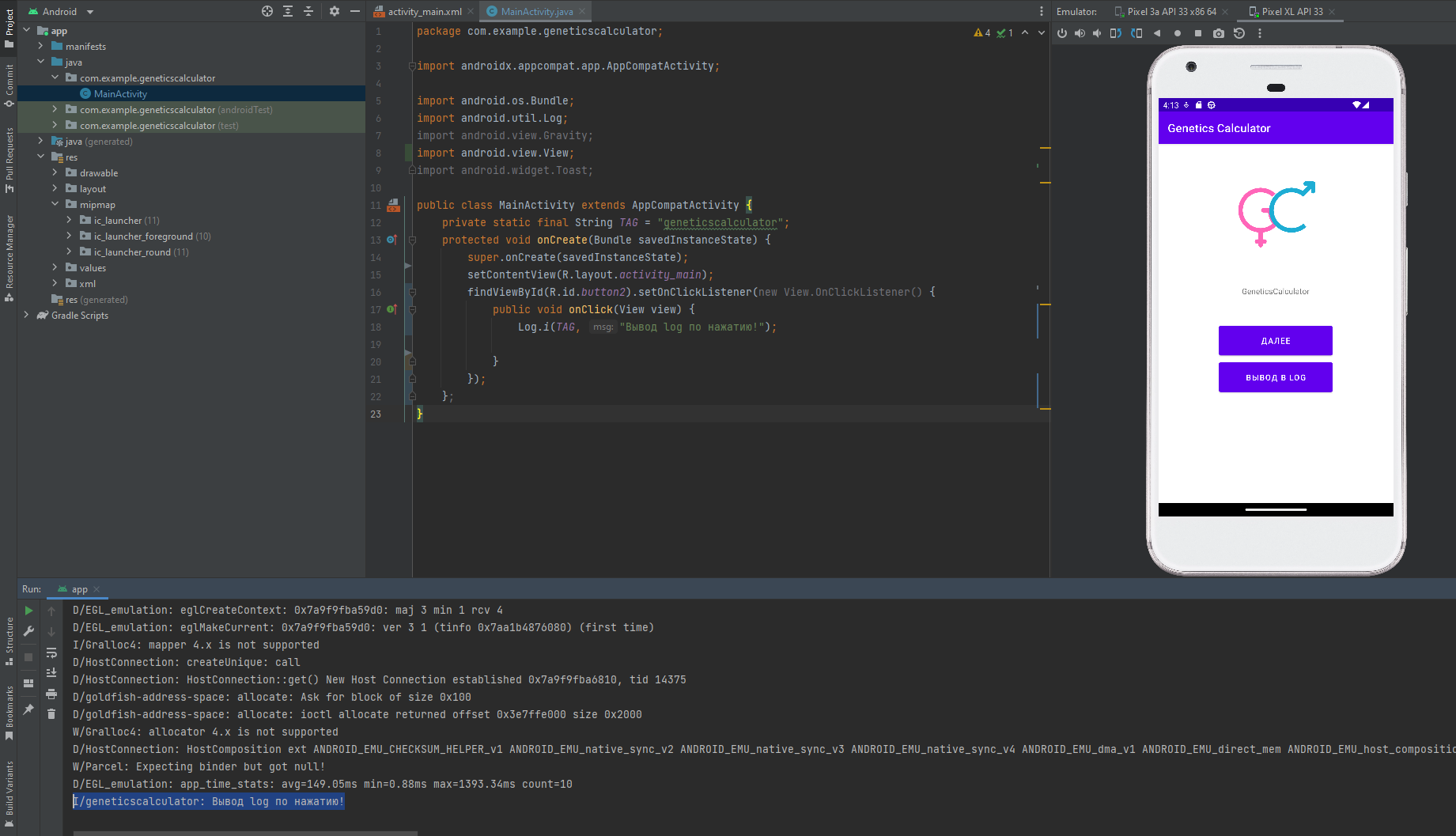


Рисунок 4.1 — Добавление логики кнопке “Вывод в LOG”

# СОЗДАНИЕ НОВОГО ACTIVITY И ПЕРЕДАЧА ДАННЫХ

Для создания второй Activity необходимо нажать правой кнопкой мыши в папку, в которой находится класс MainActivity, и затем в контекстном меню выбрать New->Activity- >Empty Activity (Рисунок 5.1).

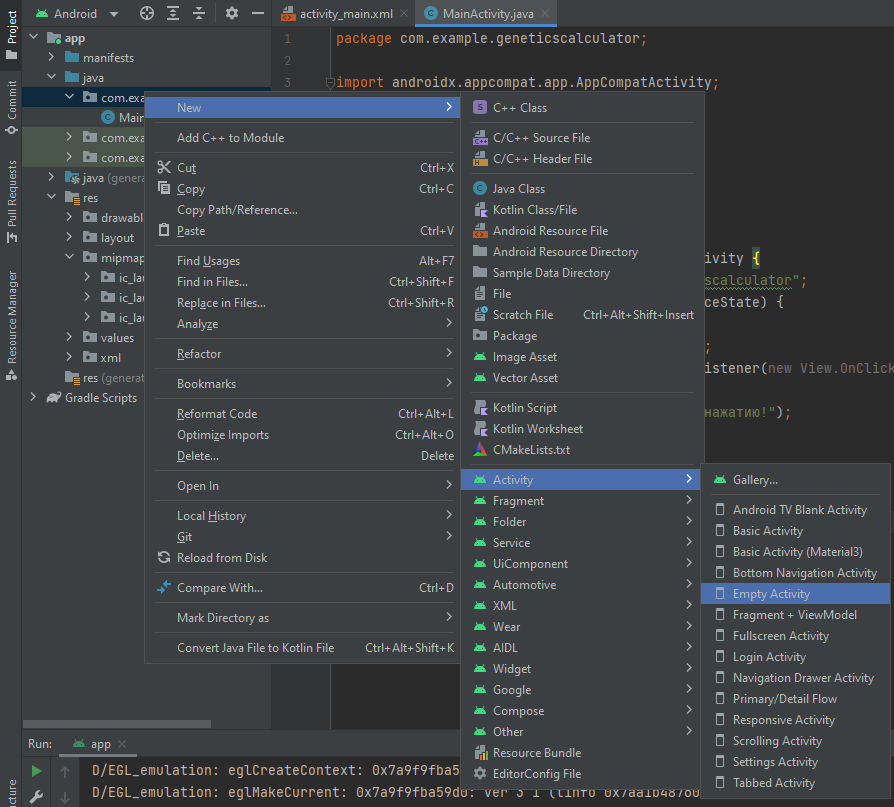


Рисунок 5.1 — Создание дополнительной Activity

Новый класс Activity назовем SecondActivity, а все остальные настройки оставим по умолчанию (Рисунок 5.2).

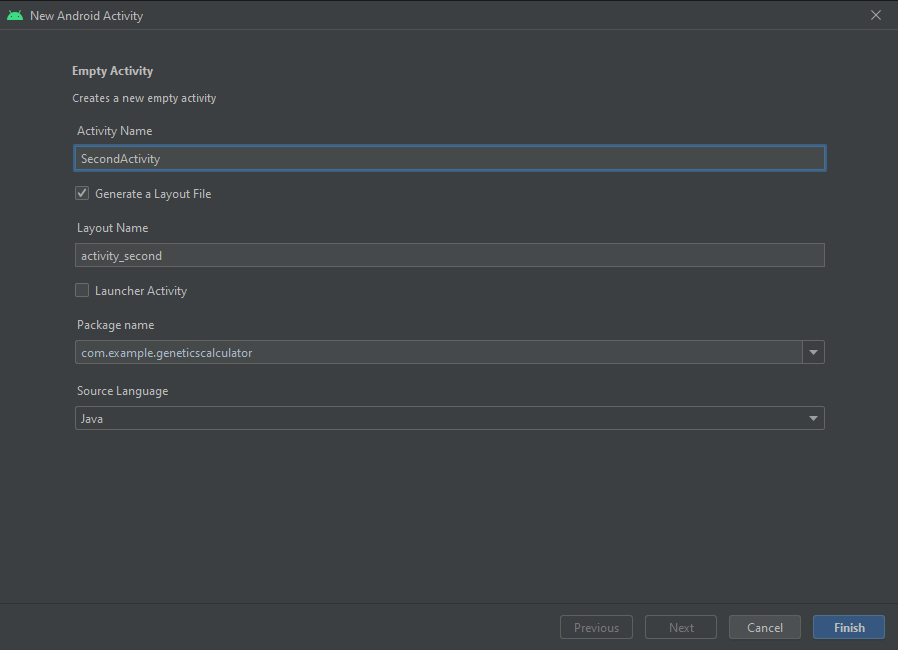


Рисунок 5.2 — Название дополнительной Activity

И после этого в проект будет добавлена новая Activity – SecondActivity. Все используемые классы activity должны быть описаны в файле AndroidManifest.xml с помощью элемента. Каждый подобный элемент содержит как минимум один атрибут android:name, который устанавливает имя класса activity.

Однако, по сути, activity – это стандартный класс java, который наследуется от класса Activity или его наследников. Поэтому вместо встроенных шаблонов в Android Studio можно добавлять обычные классы, и затем их наследовать от класса Activity. Однако в этом случае нужно будет вручную добавлять в файл манифеста данные об activity.

Чтобы из MainActivity запустить SecondActivity, надо вызвать метод startActivity(). В качестве параметра в метод startActivity передается объект Intent. Для своего создания Intent в конструкторе принимает два параметра: контекст выполнения (в данном случае это текущий объект MainActivity) и класс, который используется объектом Intent и представляет передаваемые в задачу данные (фактически класс activity, которую мы будем запускать). Изменим логику наших кнопок и протестируем работоспособность нашего приложения (Рисунок 5.3).

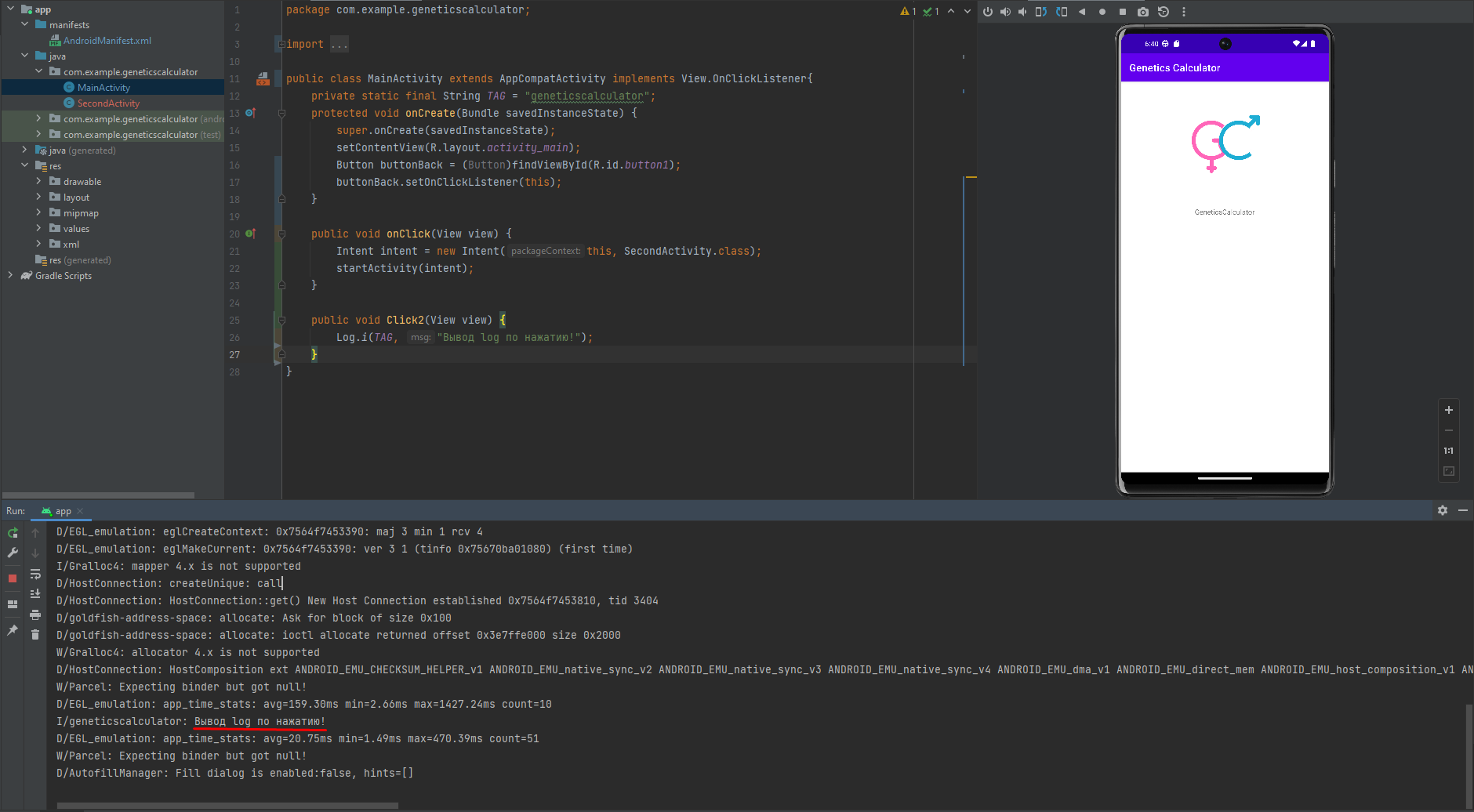


Рисунок 5.3 — Переход в новую Activity по нажатию кнопки

Для передачи данных между двумя Activity используется объект Intent. Через его метод putExtra() можно добавить ключ и связанное с ним значение. Для передачи данных применяется метод putExtra(), который в качестве значения позволяет передать данные простейших типов - String, int, float, double, long, short, byte, char, массивы этих типов, либо объект интерфейса Serializable.

Чтобы получить отправленные данные при загрузке SecondActivity, можно воспользоваться методом get(), в который передается ключ объекта (Рисунки 5.4-5.5).

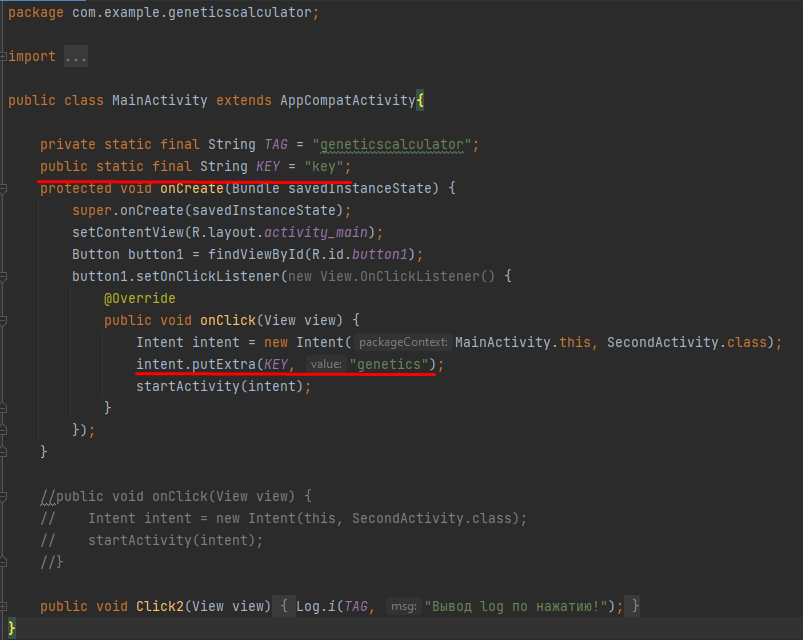


Рисунок 5.4 — Передача данных в активити

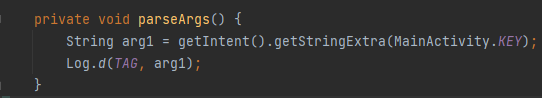


Рисунок 5.5 — Функция получения данных и вывод в log

Когда активность завершается, мы можем вызвать метод **setResult()**, чтобы возвратить данные назад в родительскую активность до метода finish(). Этот метод возвращает код результата закрытия активности (Рисунок 5.6).

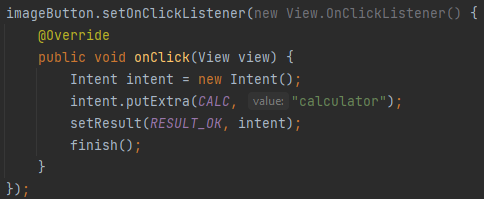


Рисунок 5.6 — Возвращение данных в родительскую активность

**ActivityResultContract** определяет контракт: данные какого типа будут подаваться на вход и какой тип будет представлять результат.

**ActivityResultCallback** представляет интерфейс с единственным методом onActivityResult(), который определяет обработку полученного результата. Когда вторая activity закончит работу и возвратит результат, то будет как раз вызываться этот метод.

Результат передается в метод в качестве параметра. При этом тип параметра должен соответствовать типу результата, определенного в ActivityResultContract (Рисунок 5.7).

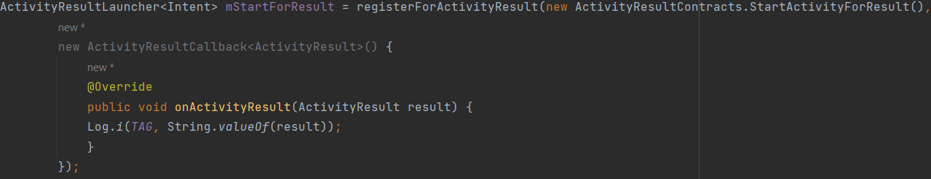


Рисунок 5.7 — Метод приема данных в родительской активити

Также добавим метод lauch() в котором передается объект того типа, который определен объектом ActivityResultContracts в качестве входного (Рисунок 5.8).



Рисунок 5.8 — Вызов метода launch

# 6. ВЫВОДЫ

В данной работе была создана верстка приложения на основе предметной области, освоена работа с ресурсами, а также закреплены и углублены знания об использовании Android Studio.

# 7. ПРИЛОЖЕНИЕ 1

Ссылка на репозиторий с проектом на GitHub:

<https://github.com/MLPeekk/RMP_2_3>