ЗАНЯТИЕ 2.6

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Тема: Изучение особенностей различных View и ViewGroup. Создание простого Android-приложения.

На данном практическом занятии предстоит изучить особенности по построению пользовательских интерфейсов в среде разработки Android Studio с использованием XML-макетов с реализацией обработки событий.

Упражнение №1

Создайте новое Android-приложение для телефона с использованием шаблона Empty Activity.

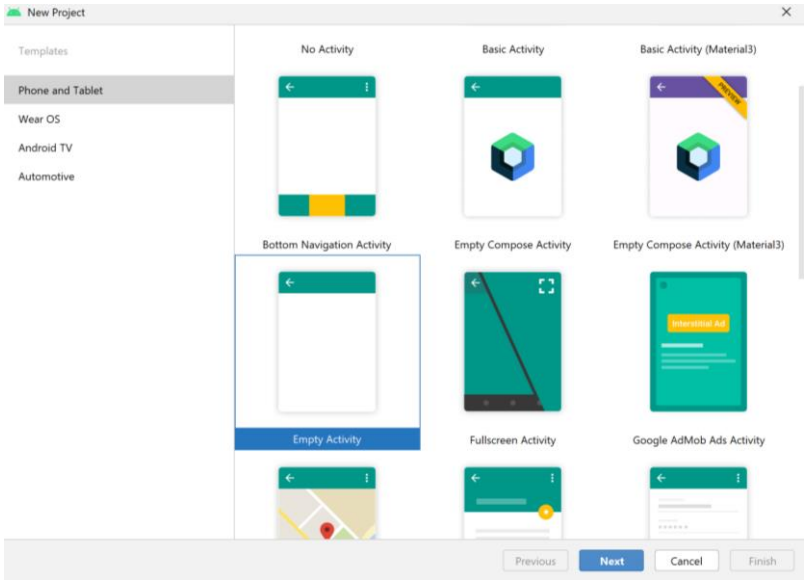


Рисунок 1

Установите минимальную версию API 21:

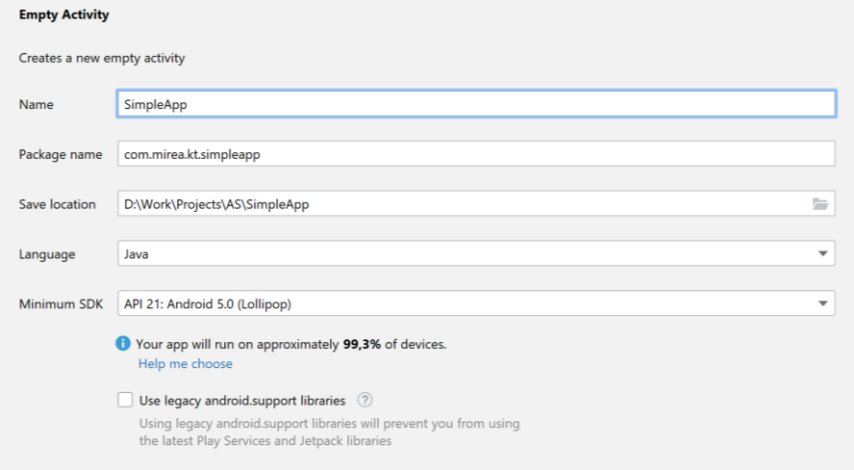


Рисунок 2

Дождитесь создания приложения и корректного обновления плагина Gradle. Перейдите в файл макета **activity_main.xml**. По умолчанию используется контейнер ConstraintLayout.

Замените контейнер на RelativeLayout. Также измените атрибуты виджета TextView, чтобы он отображался точно посредине activity и имел размер шрифта 32sp.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Привет"
        android:textSize="32sp"
        android:layout_centerInParent="true"/>

</RelativeLayout>
```

Рисунок 3

Переключите конструктор в режим «Design» и убедитесь, что компонент TextView отображается корректно.

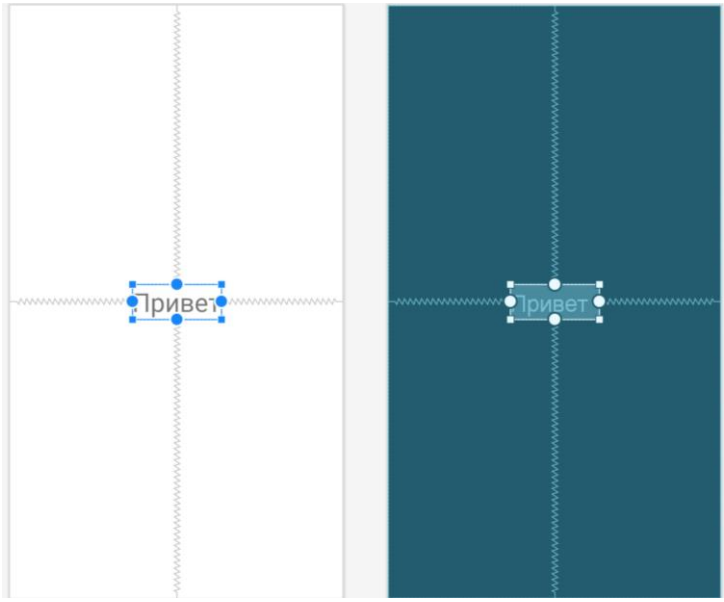


Рисунок 4

Для использования текстовых констант правильно хранить их в соответствующем файле ресурсов strings.xml, а не указывать явно, как на рисунке 3. Перейди в файл strings.xml и добавьте туда слово «Привет», чтобы

в дальнейшем использовать ссылку на этот ресурс. Название строки можно использовать любое (главное, чтобы они были уникальные).

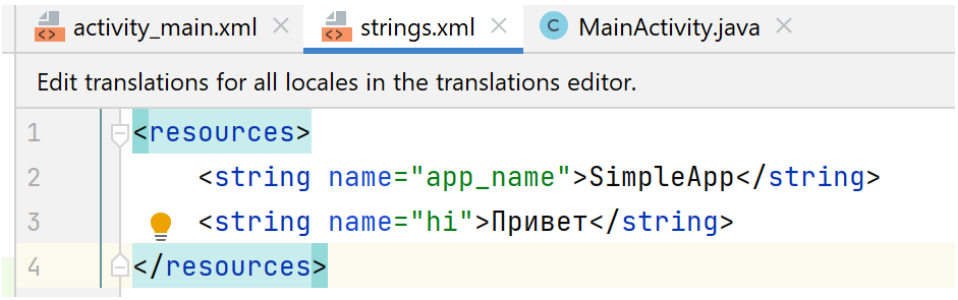


Рисунок 5

Также добавьте в файл colors.xml информацию о цветах, которые будут использоваться в приложении (hex-значения разных цветов можно генерировать онлайн https://www.color-hex.com/). Название цвета можно использовать любое (главное, чтобы они были уникальные).

Рисунок 6

Используйте ссылки на цвета и на значение строки в файле макета activity_main.xml для виджета TextView. Также необходимо придумать уникальный id для этого View-компонента (в данном случае tvWelcome – желательно, чтобы уникальный id отражал суть/назначение компонента).

```
<TextView
    android:id="@+id/tvWelcome"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:text="@string/hi"
    android:textColor="@color/my_custom_blue"
    android:textSize="32sp" />
```

Рисунок 7

Запустите приложение на эмуляторе или телефоне. Убедитесь, что приложение корректно отображает UI.

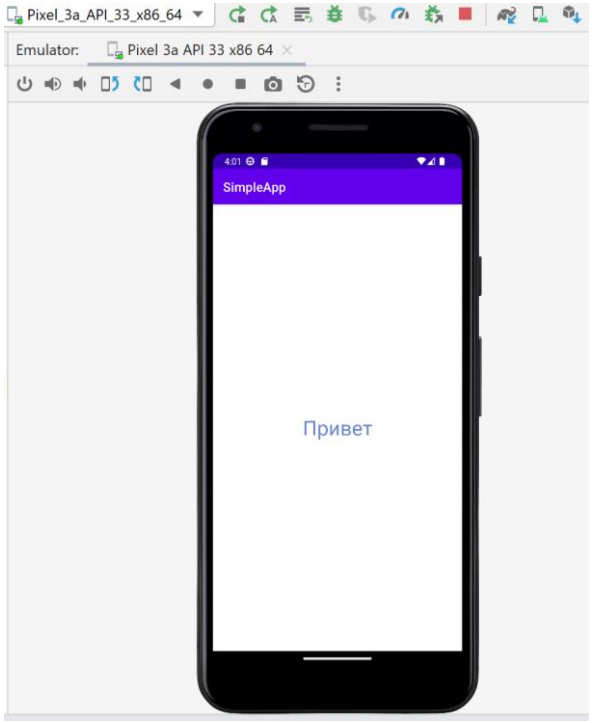


Рисунок 8

Добавьте на макет еще два View-компонента: TextView и Button. Текстовые значения также необходимо добавить в strings.xml. Ширина кнопки имеет значение **match_parent**, то есть занимает всю ширину «родительского» контейнера:

```
<TextView
    android:id="@+id/tvBye"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="@string/bye"
    android:textColor="@color/my_custom_red"
    android:textSize="32sp"
    android:layout_below="@id/tvWelcome"/>

<Button
    android:id="@+id/btnStart"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/start"
    android:layout_centerHorizontal="true"
    android:layout_above="@id/tvWelcome"/>
```

Рисунок 9

Следует обратить внимание на атрибуты **layout_above** и **layout_below**. Эти атрибуты относятся к особенностям используемого контейнера – RelativeLayout. RelativeLayout представляет объект ViewGroup, который располагает дочерние элементы относительно позиции других дочерних элементов разметки или относительно области самой разметки RelativeLayout. Используя относительное позиционирование, можно расположить элемент по правому краю, или в центре, или иным способом, который предоставляет данный контейнер. Для установки элемента в файле xml применяются следующие атрибуты:

android:layout_above	располагает элемент над элементом с указанным Id
android:layout_below	располагает элемент под элементом с указанным Id
android:layout_toLeftOf:	располагается слева от элемента с указанным Id
android:layout_toRightOf	располагается справа от элемента с указанным Id
android:layout_toStartOf	располагает начало текущего элемента, где начинается элемент с указанным Id
android:layout_toEndOf	располагает начало текущего элемента, где завершается элемент с указанным Id
android:layout_alignBottom	выравнивает элемент по нижней границе другого элемента с указанным Id
android:layout_alignLeft	выравнивает элемент по левой границе другого элемента с указанным Id
android:layout_alignRight	выравнивает элемент по правой границе другого элемента с указанным Id
android:layout_alignStart	выравнивает элемент по линии, у которой начинается другой элемент с указанным Id
android:layout_alignEnd	выравнивает элемент по линии, у которой завершается другой элемент с указанным Id
android:layout_alignTop	выравнивает элемент по верхней границе другого элемента с указанным Id
android:layout_alignBaseline	выравнивает базовую линию элемента по базовой линии другого элемента с указанным Id
android:layout_alignParentBottom	если атрибут имеет значение true, то элемент прижимается к нижней границе контейнера
android:layout_alignParentRight	если атрибут имеет значение true, то элемент прижимается к правому краю контейнера
android:layout_alignParentLeft	если атрибут имеет значение true, то элемент прижимается к левому краю контейнера
android:layout_alignParentStart	если атрибут имеет значение true, то элемент прижимается к начальному краю контейнера (при левосторонней ориентации текста - левый край)
android:layout_alignParentEnd	если атрибут имеет значение true, то элемент прижимается к конечному краю контейнера (при левосторонней ориентации текста - правый край)
android:layout_alignParentTop	если атрибут имеет значение true, то элемент прижимается к верхней границе контейнера
android:layout_centerInParent	если атрибут имеет значение true, то элемент располагается по центру родительского контейнера

android:layout_centerHorizontal	при значении true выравнивает элемент по центру по горизонтали
android:layout_centerVertical	при значении true выравнивает элемент по центру по вертикали

Запустите приложение. Убедитесь, что view компоненты располагаются на activity таким образом:

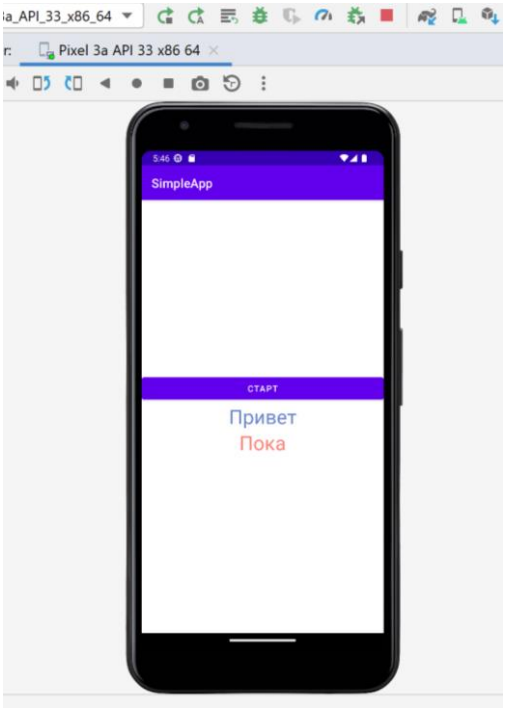


Рисунок 10

Перейдите в класс MainActivity. Реализуйте обработку нажатия на кнопку btnStart (событие OnClick). По нажатию этой кнопки компоненты TextView будут менять содержимое.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button startButton = findViewById(R.id.btnStart);
        TextView textViewFirst = findViewById(R.id.tvWelcome);
        TextView textViewSecond = findViewById(R.id.tvBye);
        startButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Log.i( tag: "simple_app_tag", msg: "Click start button");
                textViewFirst.setText("Начало");
                textViewSecond.setText("Конец");
            }
        });
    }
}
```

Рисунок 11

IDE подсвечивает серым цветом реализацию анонимного класса. Это означает, что код можно оптимизировать. В данном случае заменить на lambda.

```
startButton.setOnClickListener(v -> {
    Log.i( tag: "simple_app_tag", msg: "Click start button");
    textViewFirst.setText("Начало");
    textViewSecond.setText("Конец");
});
```

Рисунок 12

Запустите приложение. Убедитесь, событие нажатия на кнопку корректно работает. Также **обязательно убедитесь**, что в логах отображается информация по тегу «simple_app_tag».

В данном случае в качестве слушателя используется реализация с помощью анонимного класса.

Измените код таким образом, чтобы в качестве слушателя была сама activity (см. лекцию №2.5).

Доп. задание: попробуйте использовать событие OnLongClick. Сравните его работу с предыдущим.

Упражнение №2

В данном приложении необходимо реализовать простой калькулятор, который будет складывать два введенных пользователем числа.

Верните код класса MainActivity к первоначальному виду (введенный код можно закомментировать или удалить).

Перейдите в файл макета **activity_main.xml** удалите виджеты TextView. Замените контейнер RelativeLayout на **LinearLayout** с вертикальной ориентацией (дочерние компоненты будут располагаться друг под другом по вертикали):

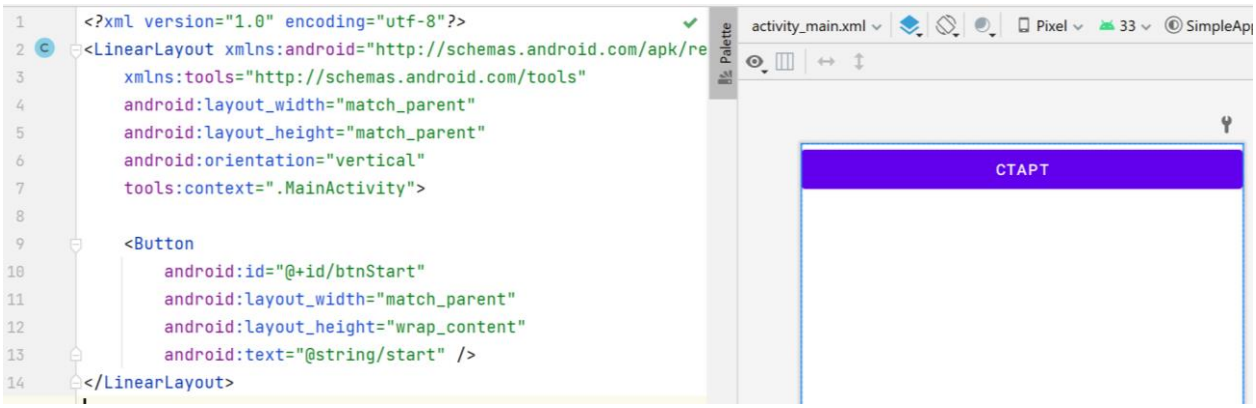
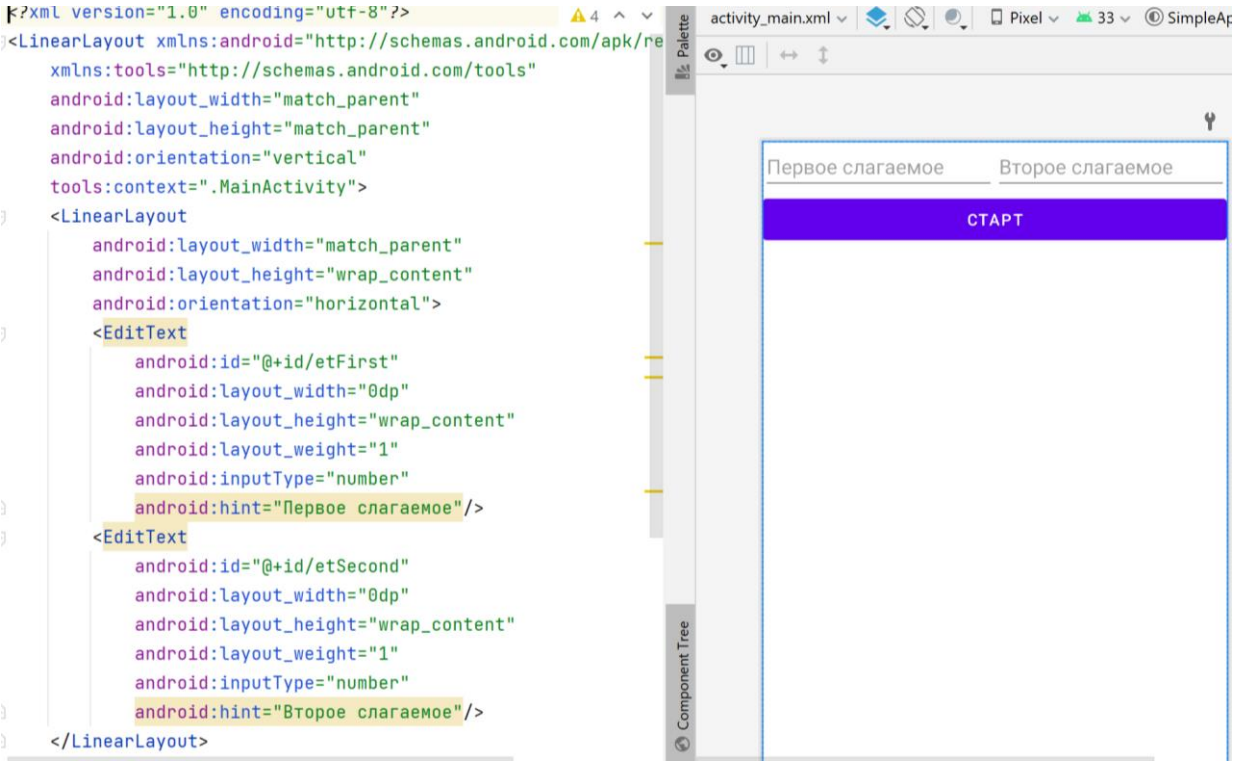


Рисунок 13

Добавьте еще один контейнер **LinearLayout**. Он будет дочерним для главного контейнера. В новом **LinearLayout** будет горизонтальная ориентация и он должен содержать два виджета EditText.

Для того, чтобы два компонента EditText поровну делили экран необходимо использовать атрибут **android:layout_weight**, который при использовании контейнера LinearLayout назначает индивидуальный вес для дочернего элемента. Этот атрибут определяет «важность» представления и позволяет этому элементу расширяться, чтобы заполнить любое оставшееся пространство в родительском представлении. В данном случае у виджетов указан одинаковый вес – 1. Это значит что ширина родительского контейнера будет делится между ними поровну:




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/re
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <EditText
            android:id="@+id/etFirst"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="number"
            android:hint="Первое слагаемое"/>
        <EditText
            android:id="@+id/etSecond"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="number"
            android:hint="Второе слагаемое"/>
    </LinearLayout>
```

Рисунок 14

Следует обратить внимание на атрибуты android:hint и android:inputType компонента EditText. Они являются необязательными к использованию, но представляют дополнительные возможности. Первый является текстом подсказки, которая отображается внутри EditText. Второй – тип вводимых значений. В данном случае предстоит работать с цифрами, поэтому тип number (этот тип не позволит вводить в EditText другие символы). О всех возможных типах можно ознакомиться на сайте для разработчиков: https://developer.android.com/reference/android/text/InputType.

Добавьте в файл макета новый виджет TextView, в котором будет отображаться результат сложения. TextView должен располагаться под кнопкой на расстоянии 32 dp. Реализовать необходимый отступ можно с помощью атрибута **android:layout_margin**.

```
            android:layout_weight="1"
            android:inputType="number"
            android:hint="Второе слагаемое"/>
    </LinearLayout>
    <Button
        android:id="@+id/btnStart"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/start" />
    <TextView
        android:id="@+id/tvResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"/>
</LinearLayout>
```

Рисунок 15

android:layout_marginTop в данном случае указывает, что отступ должен быть сверху (можно реализовать с любой стороны).

В классе MainActivity в методе onCreate необходимо реализовать считывание значений из EditText и обработку события нажатия на кнопку. Обязательно использует логирование для отслеживания работы приложения.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    EditText editTextFirst = findViewById(R.id.etFirst);
    EditText editTextSecond = findViewById(R.id.etSecond);
    TextView textViewResult = findViewById(R.id.tvResult);
    Button btnCalc = findViewById(R.id.btnStart);
    btnCalc.setOnClickListener(v -> {
        Log.i( tag: "simple_app_tag", msg: "Click calc button");
        String firstTermStr = editTextFirst.getText().toString();
        String secondTermStr = editTextSecond.getText().toString();
        if(!firstTermStr.isEmpty() && !secondTermStr.isEmpty()){
            int result = Integer.parseInt(firstTermStr)+ Integer.parseInt(secondTermStr);
            Log.i( tag: "simple_app_tag", msg: "Resul calculation: " + result);
            textViewResult.setText(String.valueOf(result));
        }else{
            Log.w( tag: "simple_app_tag", msg: "Empty term!");
            Toast.makeText(getApplicationContext(), text: "Invalid terms",Toast.LENGTH_LONG).show();
        }
    });
     ...
}
```

Рисунок 16

Считать значение из EditText можно цепочкой методов getText().toString(). Полученное значение типа String необходимо преобразовать к int, чтобы произвести вычисление. Если пользователь не введет одно из слагаемых в EditText, приложение отобразит всплывающее сообщение «Invalid term».

Запустите приложение. Убедитесь, что вычисление корректно работает, а в окне Logcat отображаются логи.

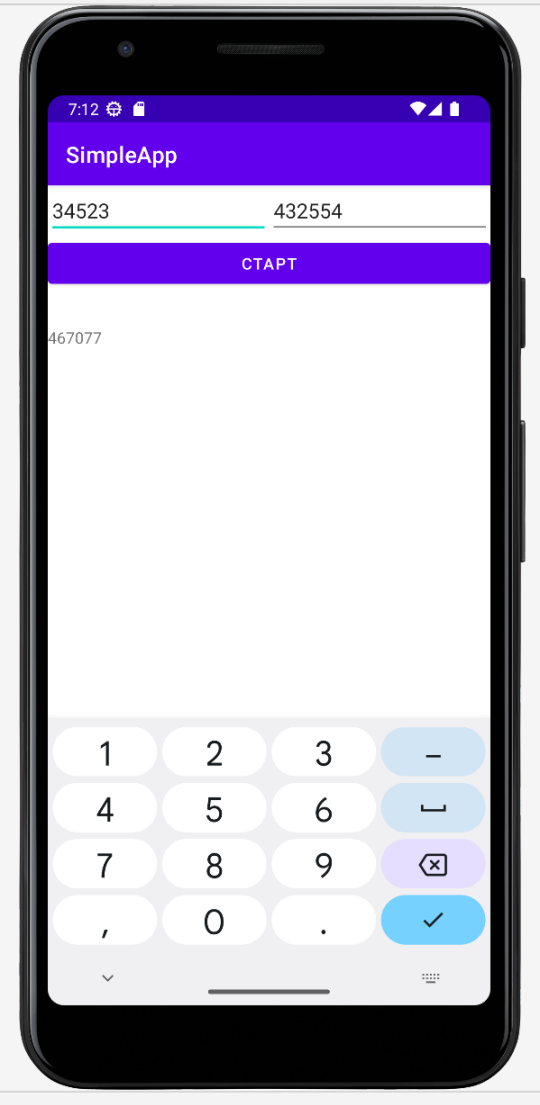


Рисунок 17

Приложение готово! Теперь необходимо подготовить его к релизу. Для этого у разработчика должна быть собственная цифровая подпись. Рассмотрим процесс создания подписи и последующей сборки.

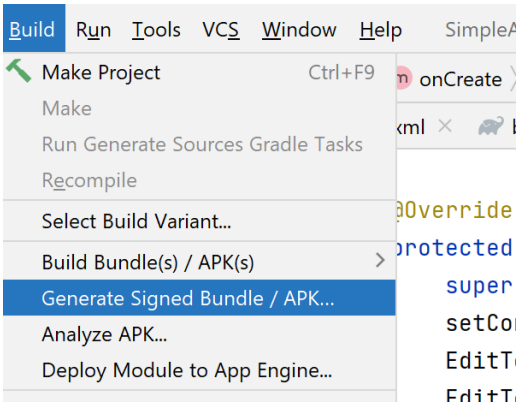


Рисунок 18

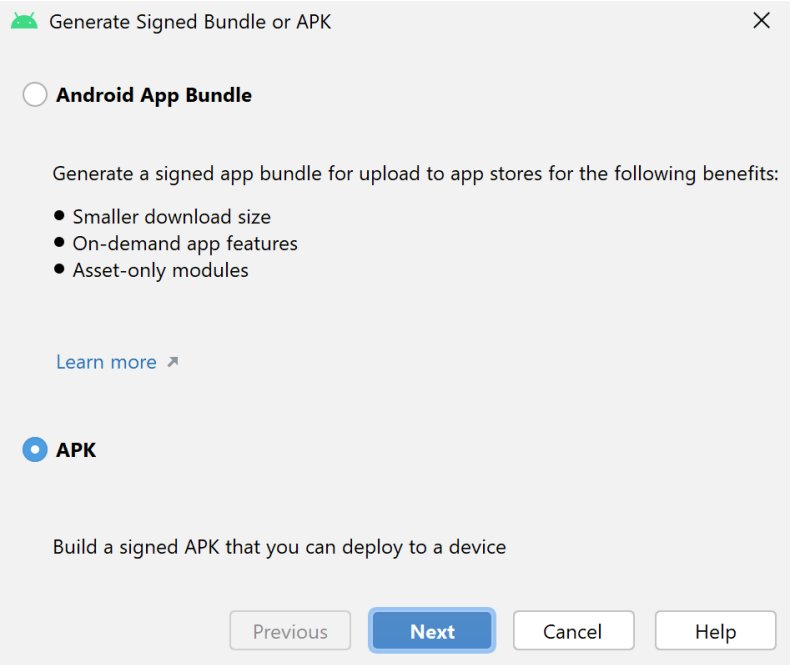


Рисунок 19

Если у вас нет подписи, то нажмите «Create new».

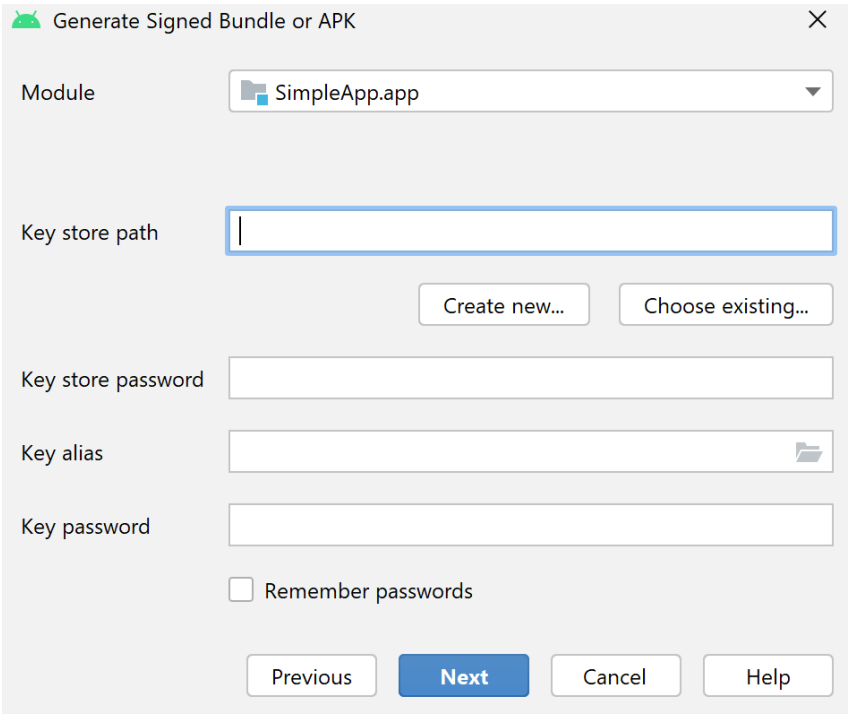


Рисунок 20

В появившемся окне выберете каталог, в котором будет сохранен файл с вашей подписью и придумайте имя файла. Также необходимо придумать пароль от хранилища ключей и от самого ключа (запомните их – это обязательно). **Напишите свои реквизиты и номер группы** в поля First and Last Name, Organizational Unit. Остальные параметры можно заполнить как на картинке или по желанию.

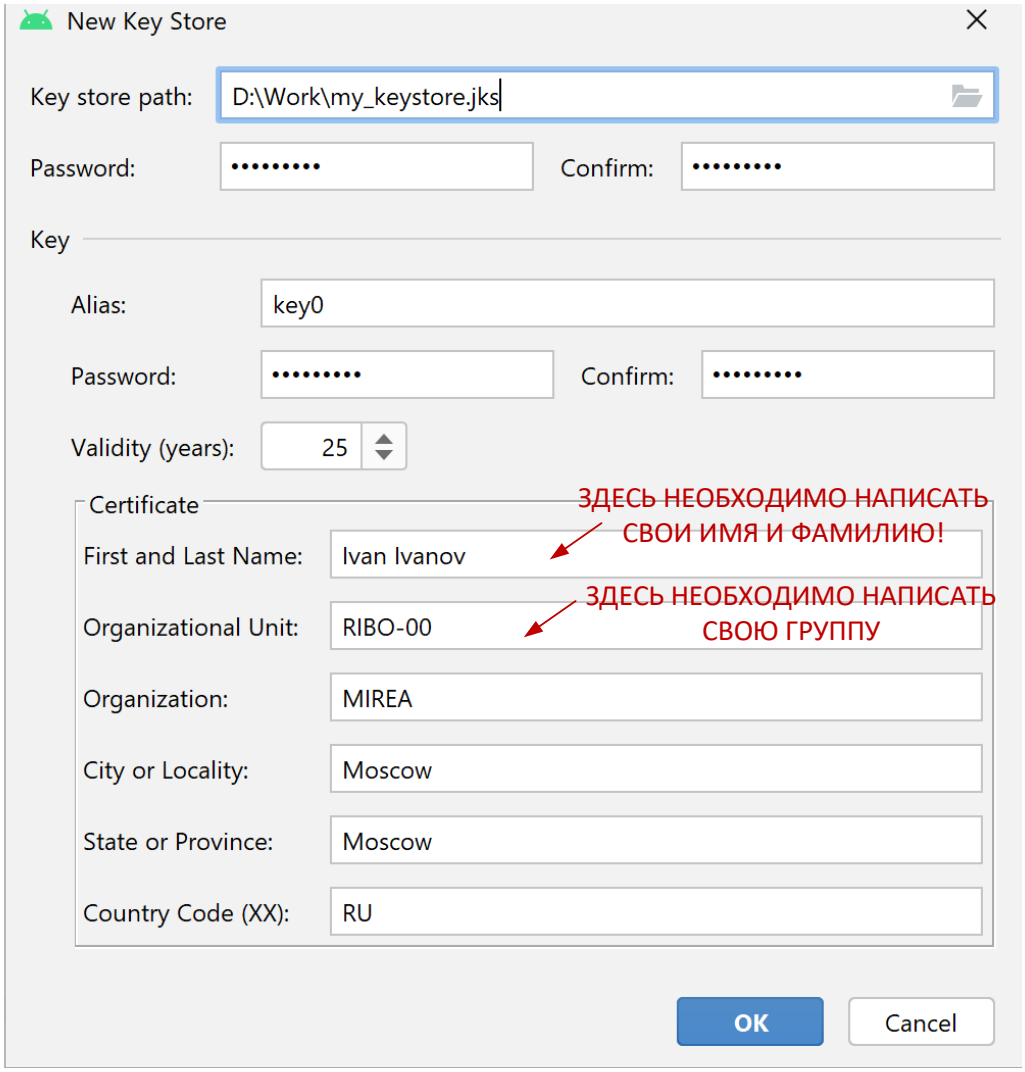


Рисунок 21

Нажмите OK. Цифровая подпись разработчика создана. **Теперь ее необходимо использовать при сборке приложений после выполнения практических заданий и приложения для курсовой работы.**

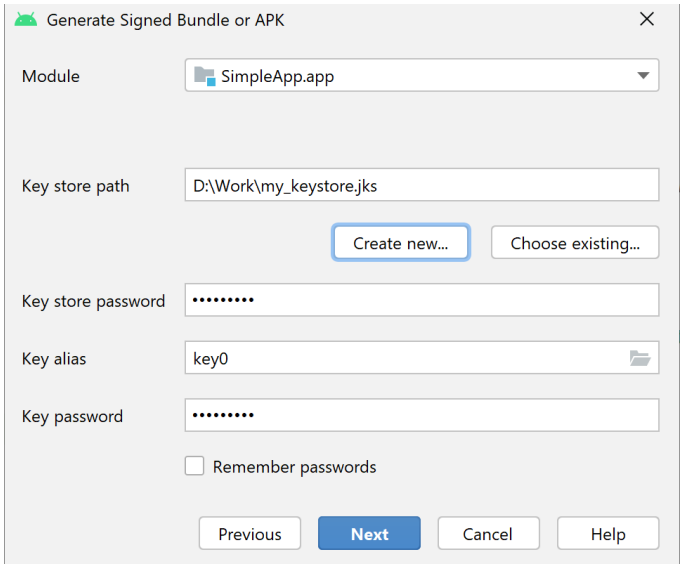


Рисунок 22

Нажмите Next. Далее выберете release и нажмите Finish:

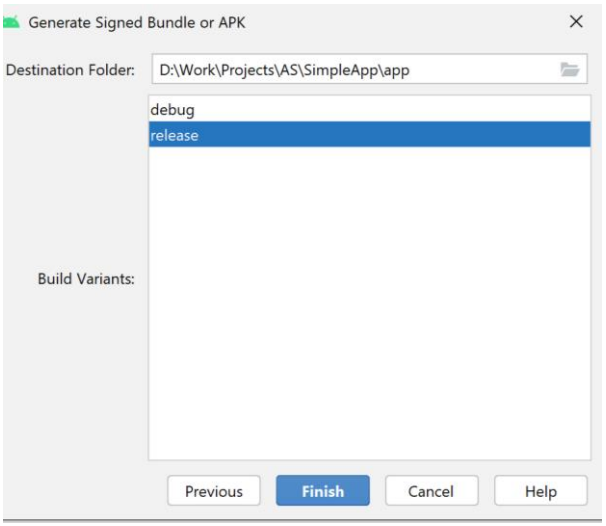


Рисунок 23

Приложение будет собрано в файл apk. Путь к файлу ~PROJECTNAME\app\release\app-release.apk (именно этот файл необходимо прикреплять к исходникам при выполнении индивидуального задания).

Индивидуальное задание

Требования:

1) На сдачу практического задания отводится **14 дней**.
2) Вариант определяется согласно порядковому номеру студента в журнале.
3) В случае дистанционного выполнения практического задания код программы необходимо выложить на Github и предоставить ссылку на него.
4) **Помимо исходного кода необходимо выкладывать файл apk (release-сборка, подписанная личной ЦП студента)**
5) Итоговое приложение должно компилироваться без ошибок, полностью выполнять требуемый функционал и на одном из экранов содержать информацию о номере варианта и ФИО студента.
6) Названия переменных и методов должны отражать суть и нести смысловую нагрузку.
7) **Обязательно использование логирования с приоритетом DEBUG или INFO (как минимум в методах жизненного цикла activity).**
8) Необходимо придерживаться стилистике по написанию Java-кода.
9) Необходимо предусмотреть защиту от ввода «аномальных» (ошибочных) значений.

Задание: разработать Android-приложение в соответствии с вариантом.

Вариант 1. Разработать Android-приложение, вычисляющее площади поверхностей и периметры сторон для прямоугольника и треугольника. Размеры сторон вводит пользователь с виртуальной клавиатуры в виджет EditText. При разработке приложения обязательно использование контейнера **LinearLayout**. При обработке события нажатия на кнопку в качестве слушателя необходимо использовать activity.

Вариант 2. Разработать Android-приложение – банковский калькулятор вклада для вычисления ежемесячного дохода и итогового дохода. Начисление процентов происходит ежемесячно. Длительность вклада (в месяцах), годовой процент по вкладу и начальную сумму вводит пользователь с виртуальной клавиатуры в виджеты EditText. При выполнении задания обязательно использовать цикл for. При разработке приложения обязательно использование контейнера **LinearLayout**. При обработке события нажатия на кнопку в качестве слушателя необходимо использовать activity.

Вариант 3. Разработать Android-приложение по поиску делимых чисел в заданном пользователем диапазоне. То есть пользователь вводит в различные EditText-виджеты значения диапазона и число-делитель. Приложение должно вывести на экран все числа из диапазона, которые делятся на число-делитель без остатка. При выполнении задания обязательно использовать цикл for. При разработке приложения обязательно использование контейнера **ConstraintLayout**.

Вариант 4. Разработать Android-приложение – банковский калькулятор вклада для вычисления ежемесячного дохода и итогового дохода. Начисление процентов происходит ежемесячно. Длительность вклада (в месяцах), месячный процент по вкладу и начальную сумму вводит пользователь с клавиатуры в виджеты EditText. При выполнении задания обязательно использовать цикл while. При разработке приложения обязательно использование контейнера **RelativeLayout**.

Вариант 5. Разработать Android-приложение для изменения входного текста (строки) по трем алгоритмам: разворот строки, удаление гласных букв, увеличение регистра четных букв. Исходный текст вводит пользователь в виджет EditText. Для реализации каждого алгоритма должна быть своя кнопка (Button). Результат (измененный текст) необходимо отобразить в TextView. При разработке приложения обязательно использование контейнера **RelativeLayout**. При обработке события нажатия на кнопку в качестве слушателя необходимо использовать activity.

Дополнительная информация:

1) Перед выполнением задания рекомендуется ознакомиться с Лекциями №2.1, №2.3, №2.5.
2) Рекомендуется использовать обработку исключений.
3) Информация о контейнере ConstraintLayout для выполнения варианта 3:

Constraints – это линии, на основе которых располагается компонент view внутри ConstraintLayout. Constraints могут быть привязаны к сторонам самого ConstraintLayout или к сторонам других view внутри ConstraintLayout. Constraints можно разделить на вертикальные и горизонтальные.

Горизонтальные constraints:

- правой стороны (Right), левой стороны (Left);
- начальной стороны (Start), конечной стороны (End).

Вертикальные constraints:

- верхней стороны (Top), нижней стороны (Bottom);
- базовой линии (Baseline).

Вертикальное и горизонтальное constraints друг с другом не связаны.

Baseline – это линия выравнивания контента элемента. Пример – для TextView это линия строки, на которой пишется текст. Если у view выставлен Baseline constraint, то базовая линия элемента будет находиться на уровне базовой линии view, к которой привязан constraint.

Для начала, проще всего рассматривать constraints, как стороны view. То есть можно, например, привязать левую сторону view B к правой стороне view A – тогда view B будет располагаться справа от view A. Общий формат атрибутов для привязки constraint выглядит следующим образом:

app:layout_constraint{X}_to{Y}Of="{Z}"

Где:

X – constraint привязываемой view;

Y – сторона view, к которой привязываются;

Z – id view, к которой привязываются, или parent, если привязать нужно к стороне ContraintLayout.

Основные правила привязки сторон:

- привязывать между собой можно только Start и End, Left и Right, Top и Bottom. То есть, нельзя, например, привязать Left к Start или Baseline к Top;
- Baseline можно привязать только к Baseline другой view;
- при привязке Start/End игнорируются привязки Left/Right;
- при привязке Baseline игнорируются привязки Top/Bottom;
- не привязывайте view с внешней стороны ConstraintLayout, например, layout_constraintRight_toLeftOf="parent". ConstraintLayout обработает такую привязку, но как себя при этом поведет, сложно предсказать.

Дополнительная информация:
https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout