ЗАНЯТИЕ 2.8

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Тема: Создание приложений с несколькими activity. Явные и неявные намерения (Intent). Работа с фрагментами.

Упражнение №1

Создайте новое Android-приложение для телефона с использованием шаблона Empty Activity (или Empty View Activity).

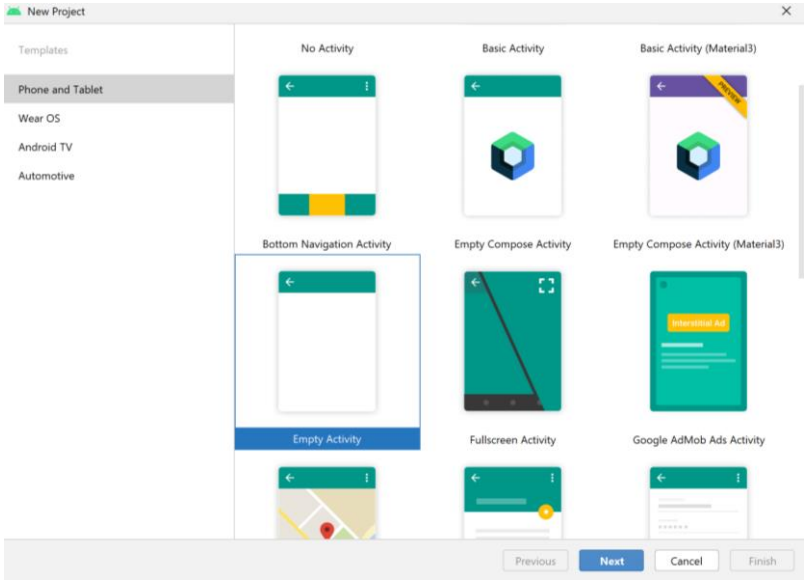

Рисунок 1

Установите минимальную версию API 21. **Язык разработки – Java (не Kotlin)**:

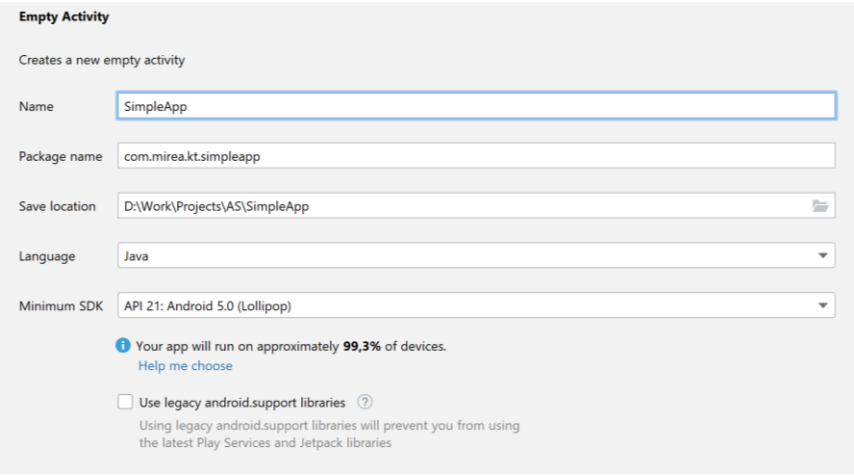

Рисунок 2

Дождитесь создания приложения и корректного обновления плагина Gradle. Перейдите в файл макета **activity_main.xml**. К существующему виджету TextView добавьте кнопку Button, которая будет располагаться ниже:

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tvText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hi"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnStartActivity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/start"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tvText" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 1

Обратите внимание, что на строки используются из файла strings.xml:

```
<string name="app_name">SimpleApp</string>
<string name="hi">Привет</string>
<string name="bye">Пока</string>
<string name="start">Старт</string>
```

Рисунок 2

Измените фон контейнера ConstraintLayout, который занимает весь экран. Это сделает ваше приложение более веселым и привлекательным.

В качестве фона можно использовать сплошной цвет или картинку. Впишите в атрибут background значение цвета, которое вам нравится (убедитесь, что ранее вы добавили этот цвет в файл colors.xml):

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/teal_700"
    tools:context=".MainActivity">
```

Рисунок 3

Дизайн макета вашей activity обновился:

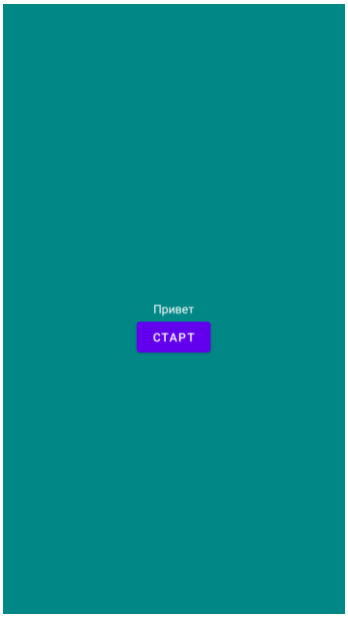

Рисунок 4

Создайте еще одну activity. Быстро это можно сделать через контекстное меню проекта:

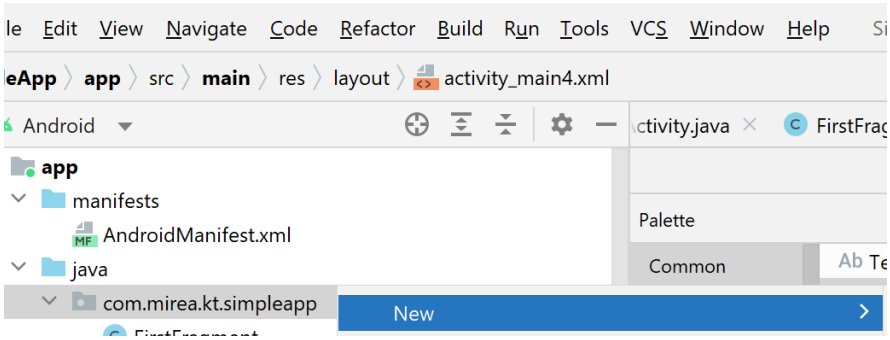

Рисунок 5

Выбирайте шаблон Empty Activity (или Empty Views Activity в зависимости от версии IDE).

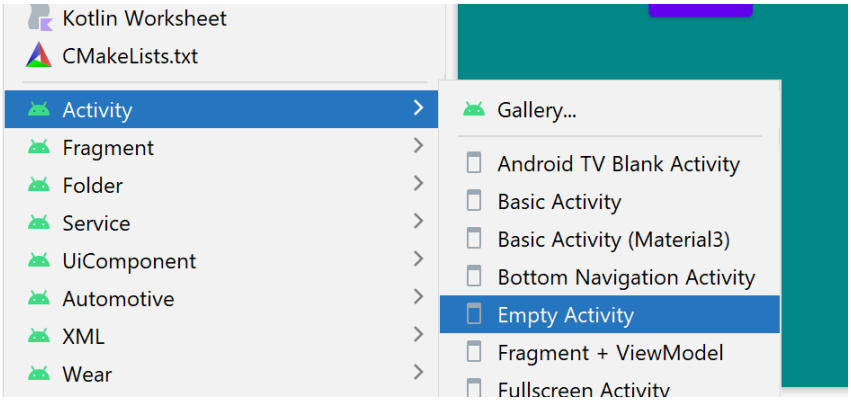

Рисунок 6

Назовите activity понятным для вас названием и завершите создание:

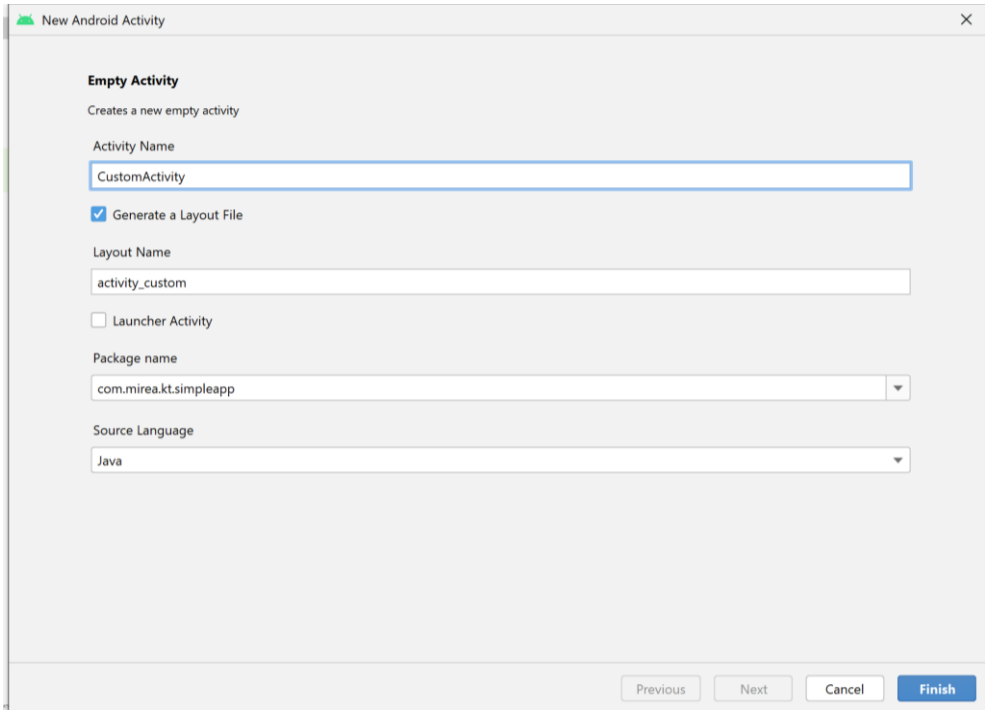

Рисунок 7

Убедитесь, что информация о созданной activity отображается в файле AndroidManifest.xml:

```
<activity
    android:name=".CustomActivity"
    android:exported="false">
</activity>
```

Рисунок 8

Для новой activity автоматически создался layout-файл макета. Откройте его. Он пустой (только контейнер ConstraintLayout). Добавьте внутрь контейнера ConstraintLayout другой контейнер – RelativeLayout с параметрами ширины и высоты – match_parent. Внутрь RelativeLayout поместите два виджета - EditText и Button.

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".CustomActivity">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <EditText
            android:id="@+id/etValue"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
            android:hint="@string/enter_value_edit" />
        <Button
            android:id="@+id/btnExit"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/exit"
            android:layout_below="@id/etValue"
            android:layout_centerHorizontal="true"/>
    </RelativeLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 9

Обратите внимание на атрибут **margin** у виджета EditText. Он позволяет установить отступ с каждой из сторон (сверху, снизу, слева, справа). В данном случае отступ реализован сразу для всех 4 сторон. Значение отступа 16 dp.

Button располагается под EditText – за это отвечает android:layout_below. В этом же время кнопка располагается по центру по горизонтальной линии родителя (RelativeLayout).

Для строковых значений используются ссылки на ресурсы:

```
<string name="enter_value_edit">Enter value</string>
<string name="exit">Exit</string>
```

Рисунок 10

Убедитесь, что ваш макет выглядит подобным образом в режиме Design:

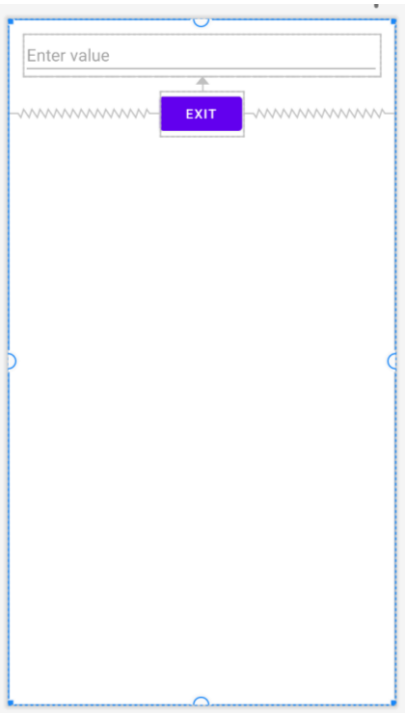

Рисунок 11

Перейдите в класс главной activity и реализуйте событие OnClick для кнопки (также получите ссылку на TextView – она нам понадобится позже):

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private Button btnStart;
    private TextView tvTextValue;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnStart = findViewById(R.id.btnStartActivity);
        tvTextValue = findViewById(R.id.tvText);
        btnStart.setOnClickListener(this);
//...
    }

    @Override
    public void onClick(View v) {
        if(v.getId() == R.id.btnStartActivity){

        }
    }
```

Рисунок 12

Реализуйте в методе OnClick код перехода на созданную activity с помощью явного намерения (Intent). При использовании явного намерения имя компонента указывается «напрямую».

```
@Override
public void onClick(View v) {
    if(v.getId() == R.id.btnStartActivity){
        Intent actIntent = new Intent(getApplicationContext(),CustomActivity.class);
        startActivity(actIntent);
    }
}
```

Рисунок 13

Запустите приложение, убедитесь, что после нажатия на кнопку запускается еще одна activity, на которой располагаются EditText и Button.

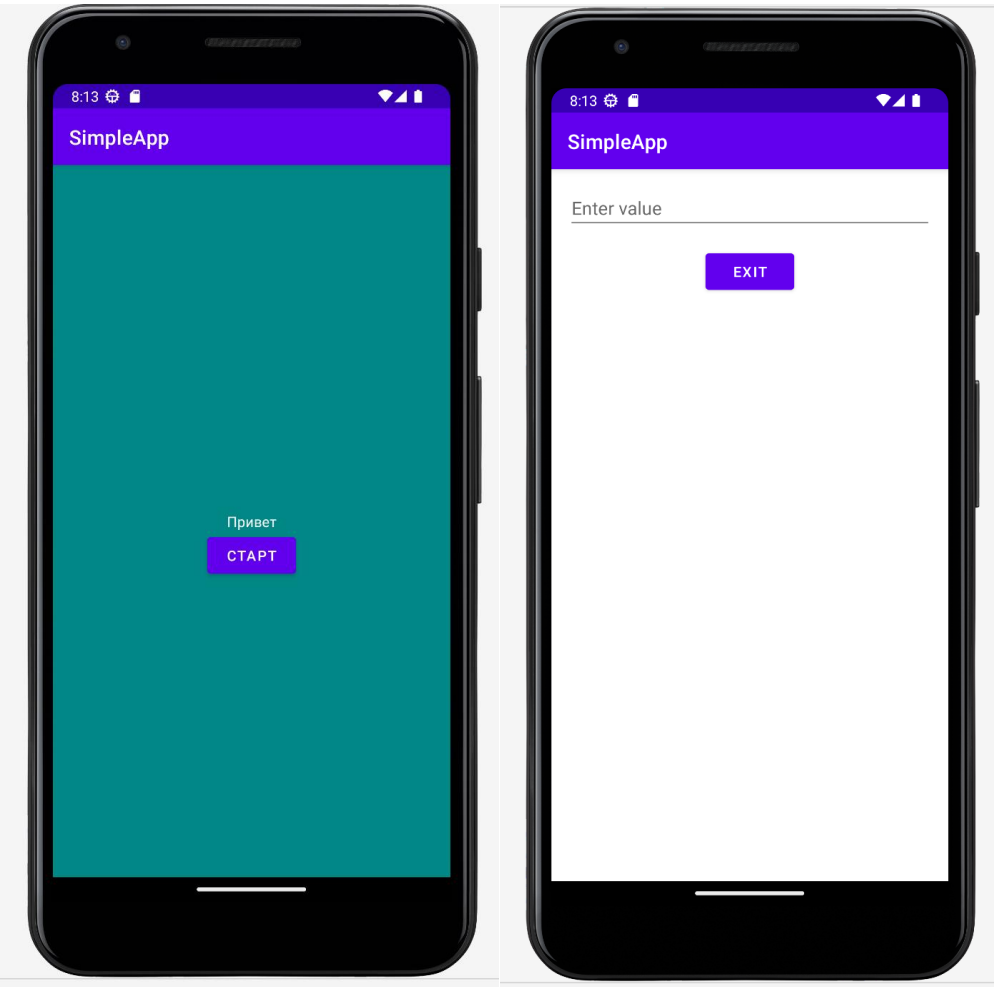

Рисунок 14

Если после попытки «собрать» приложение возникла ошибка подобного рода:

```
7.  Dependency 'androidx.annotation:annotation-experimental:1.3.0' requires libraries and applications that
    depend on it to compile against version 33 or later of the
    Android APIs.

    :app is currently compiled against android-32.

    Recommended action: Update this project to use a newer compileSdkVersion
    of at least 33, for example 33.

    Note that updating a library or application's compileSdkVersion (which
    allows newer APIs to be used) can be done separately from updating
    targetSdkVersion (which opts the app in to new runtime behavior) and
    minSdkVersion (which determines which devices the app can be installed
    on).
```

Рисунок 15

Необходимо изменить версию SDK, которая используется для сборки. Сделать это можно в файле build.gradle:

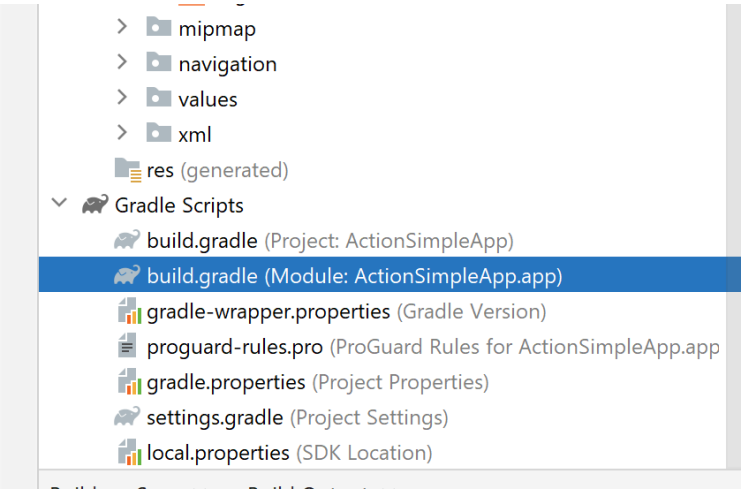

Рисунок 16

В данном случае требуется поменять версию SDK с 32 на 33 и нажать «Sunc Now».

Рисунок 17

Теперь необходимо реализовать возврат каких-либо данных из второй activity в первую. Для этого измените код вызова метода запуска activity:

```
@Override
public void onClick(View v) {
    if(v.getId() == R.id.btnStartActivity){
        Intent actIntent = new Intent(getApplicationContext(),CustomActivity.class);
        startActivityForResult(actIntent,  requestCode: 12345); // метод deprecated, но его можно использовать
    }
}
```

Рисунок 18

В классе второй activity необходимо реализовать сбор данных из поля ввода и отправку этих данных «обратно» в первую activity:

```
public class CustomActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_custom);
        EditText etVal = findViewById(R.id.etValue);
        Button btnExit = findViewById(R.id.btnExit);
        btnExit.setOnClickListener(v -> {
            String strData = etVal.getText().toString(); //забираем данные из EditText
            Intent intent = new Intent();
            intent.putExtra( name: "data",strData); // помещаем их в intent
            setResult(RESULT_OK, intent); // возвращаем intent
            finish(); // завершаем работу activity
        });
    }
}
```

Рисунок 19

Теперь эти данные необходимо «принять». Переопределите метод **onActivityResult** в классе главной activity:

```
        btnStart.setOnClickListener(this);
//...
    }

    onAct
    protected void onActivityResult(int …   FragmentActivity
    public void onActionModeFinished(ActionMode …   Activity
    public void onActionModeStarted(ActionMode m…   Activity
    public void onActivityReenter(int resultCode…   Activity
    public void onSupportActionModeFinished   AppCompatActi…
    public void onSupportActionModeStarted   AppCompatActiv…
    public ActionMode onWindowStartingActionMode…   Activity
    public ActionMode onWindowStartingActionMode…   Activity
    public void onGetDirectActions(CancellationS…   Activity
```

Рисунок 20

И реализуйте обработку данных, которые вернулись после закрытия второй activity (если они, конечно, есть).

```
@Override
public void onClick(View v) {
    if(v.getId() == R.id.btnStartActivity){
        Intent actIntent = new Intent(getApplicationContext(),CustomActivity.class);
        startActivityForResult(actIntent,  requestCode: 12345);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode == 12345){
        if(data != null){
            // обработка полученных данных и запись их в лог
            String str = data.getStringExtra( name: "data");
            Log.d( tag: "simple_app_tag",str);
        }
    }
}
```

Рисунок 21

Запустите приложение. Перейдите на вторую activity и введите в EditText любой текст. Нажмите Exit.

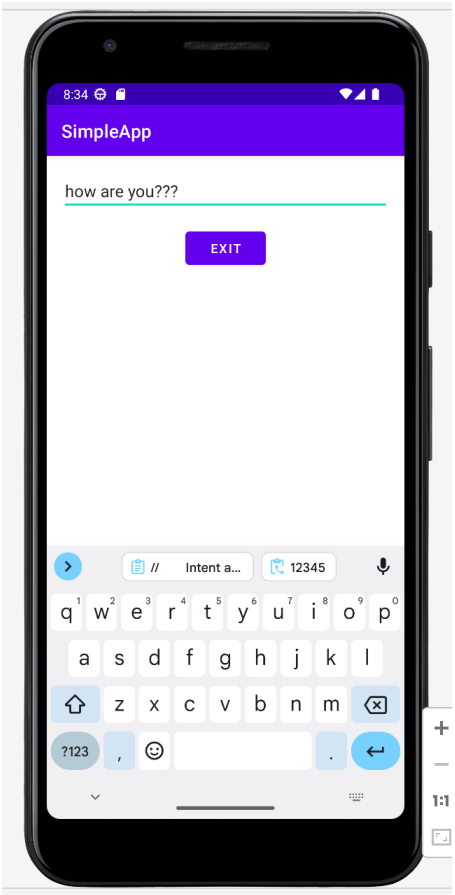

Рисунок 22

Приложение вернется в главную activity. Убедитесь, что данные успешно передались и отобразились в логах (окно Logcat):

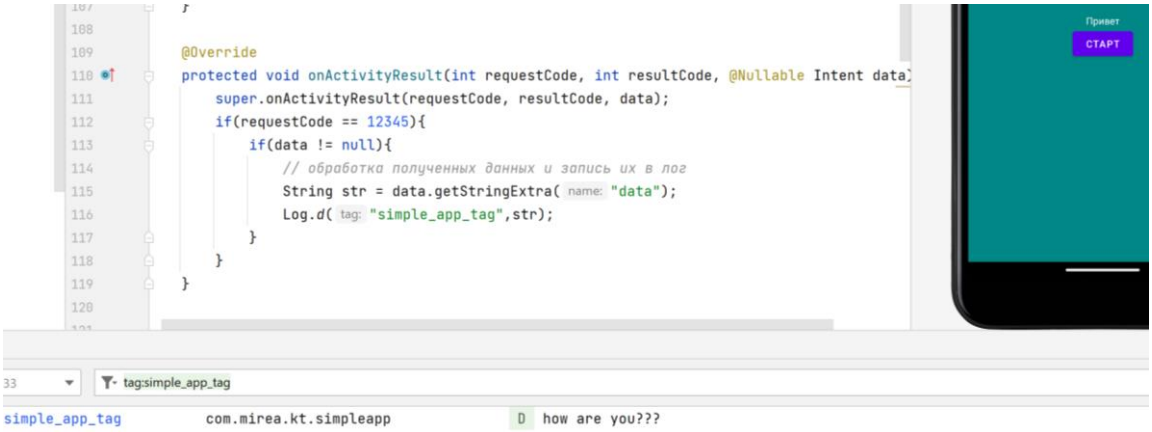

Рисунок 23

Закомментируйте метод onActivityResult и измените код запуска activity в соответствии с новыми рекомендациями от Google:

```
ActivityResultLauncher<Intent> activityResultLaunch = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        new ActivityResultCallback<ActivityResult>() {
            @Override
            public void onActivityResult(ActivityResult result) {
                if (result.getData() != null) {
                    // обработка полученных данных и запись их в лог
                    String str = result.getData().getStringExtra( name: "data");
                    Log.d( tag: "simple_app_tag",str);
                }
            }
});

@Override
public void onClick(View v) {
    if(v.getId() == R.id.btnStartActivity){
        Intent actIntent = new Intent(getApplicationContext(),CustomActivity.class);
        activityResultLaunch.launch(actIntent); // запуск новой activity
    }
}
```

Рисунок 24

Если у вас класс ActivityResultLauncher и прочие выделены красным, то проверьте, что они импортированы.

Запустите приложение, убедитесь, что оно работает корректно, и в логах отображаются данные, как и в первом случае.

Упражнение №2

Продолжайте работу в текущем приложении. Замените вызов activity так, чтобы использовался неявный Intent. Например, чтобы открылась activity для отображения координат:

```
@Override
public void onClick(View v) {
    if(v.getId() == R.id.btnStartActivity){
        Intent actIntent = new Intent(getApplicationContext(),CustomActivity.class);
        activityResultLaunch.launch(actIntent); // запуск новой activity
        Uri uriLocation = Uri.parse("geo:55.754283,37.62002");
        Intent newIntent = new Intent(Intent.ACTION_VIEW);
        newIntent.setData(uriLocation);
        startActivity(newIntent);
    }
}
```

Рисунок 25

Если на вашем устройстве есть приложение, в котором реализован ACTION_VIEW для данных со схемой «geo», то произойдет запуск этой activity или предложен выбор из нескольких. Если такого приложения не найдено, но произойдет завершение работы. Запустите приложение и проверьте:

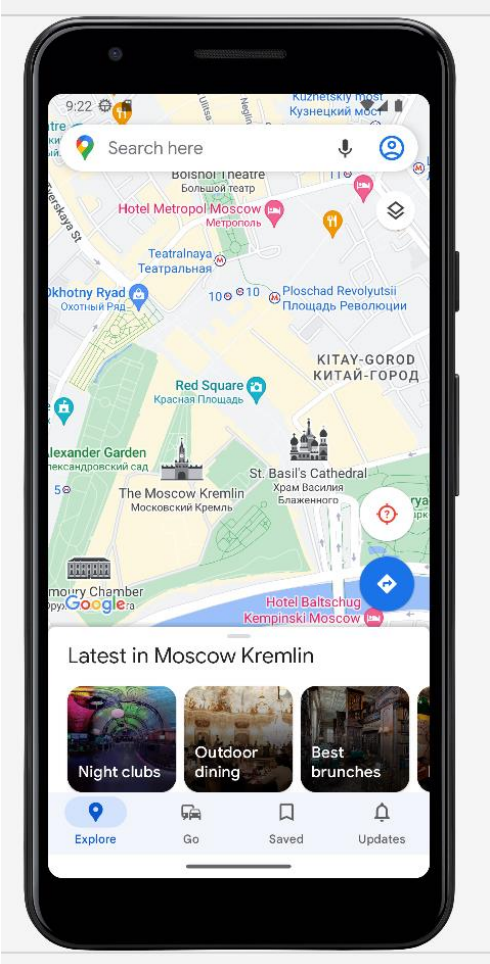

Рисунок 26

Самостоятельно реализуйте запуск activity любого приложения с ACTION_SEND для передачи текста через (см. прошлую лекцию).

В этом случае данные помещаются не в Uri, а в сам объект намерения методом **putExtra**.

Упражнение №3

Продолжайте работу в текущем приложении. Создайте еще одну activity – TestFragmentActivity.

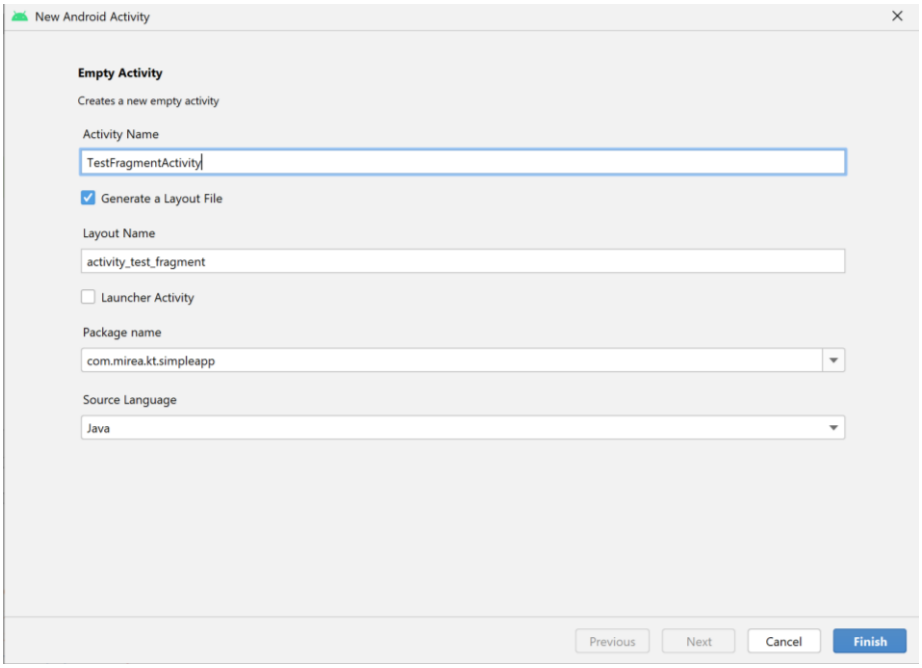

Рисунок 27

Измените файл макета данной activity так, чтобы в нем контейнер RelativeLayout располагался под кнопкой с текстом Change:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".TestFragmentActivity">
    <Button
        android:id="@+id/btnChangeFragment"
        android:text="Change"
        android:layout_width="match_parent"
        android:layout_height="56dp"
        app:layout_constraintTop_toTopOf="parent"/>
    <RelativeLayout
        android:id="@+id/relativeContainer"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="56dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnChangeFragment"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 28

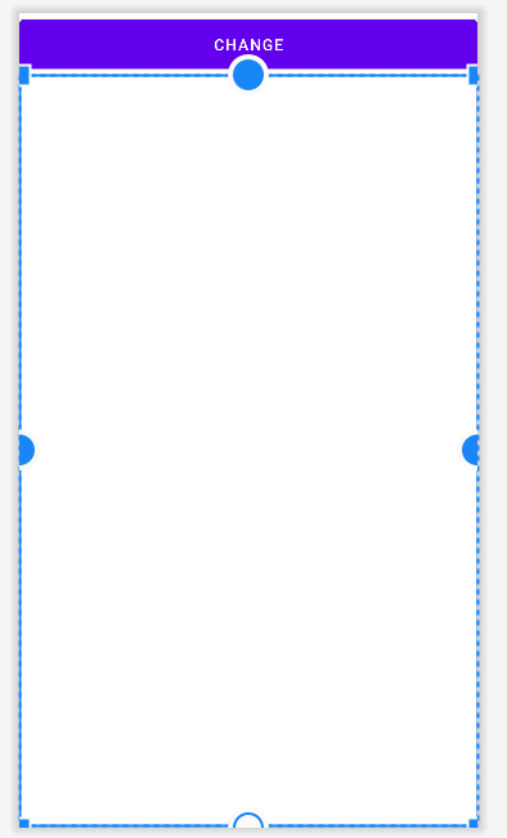

Рисунок 29

Через контекстное меню проекта (правой кнопкой мыши по имени пакета) создайте два класса фрагмента по шаблону Blank:

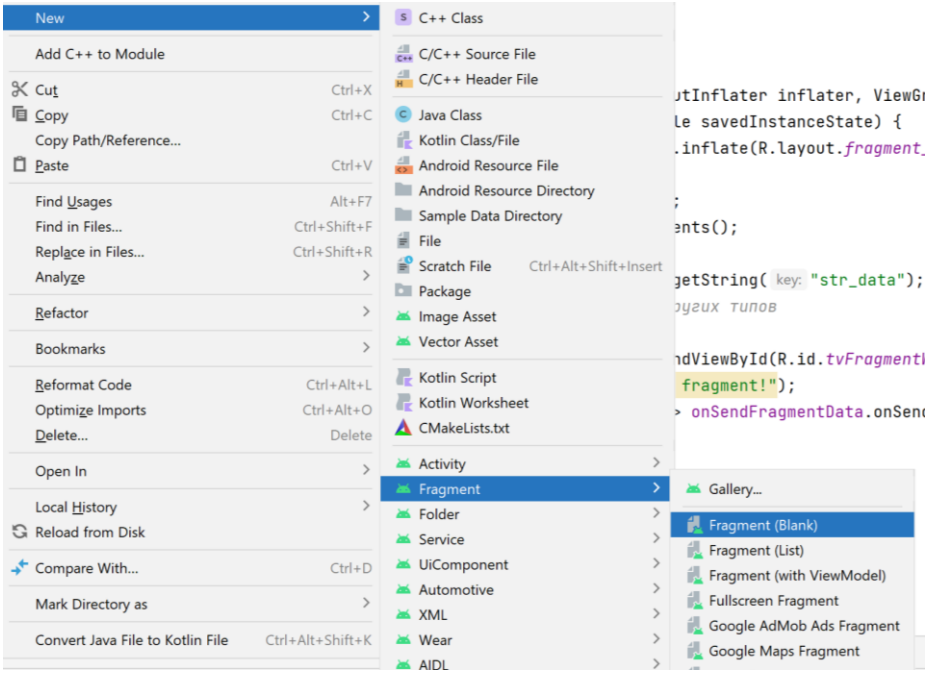

Рисунок 30

Первый класс назовите FirstFragment, второй – SecondFragment.

AndroidStudio сгенерирует код для этих классов. Удалите в созданных классах методы **newInstance** и **OnCreate** – в данном задании они не понадобятся.

Также были сгенерированы файлы макетов для каждого фрагмента.

Перейдите в файл макета первого фрагмента (R.layout.fragment_first). Разместите текстовое поле посередине макета:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".FirstFragment">
    <TextView
        android:id="@+id/tvFragmentWelcome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textSize="32sp"/>
</RelativeLayout>
```

Рисунок 31

В файле макета второго фрагмента (R.layout.fragment_second) разместите по центру виджет **ImageView**. Он предназначен непосредственно для вывода изображений. Но где взять изображение? Добавим картинку в ресурсы приложения. Загрузите png файл: https://goo.su/RB05N. Загруженный файл поместите в каталог ~\PROJECT_NAME\app\src\main\res\drawable\.

Он отобразится в ресурсах drawable:

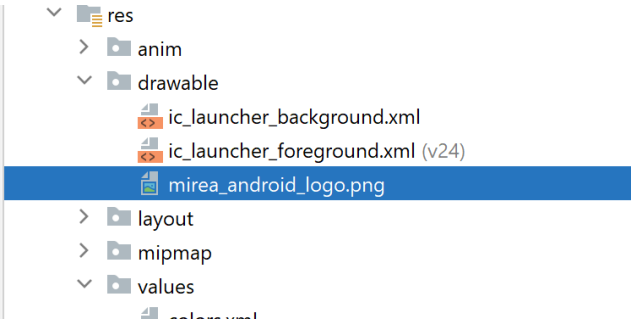

Рисунок 32

Теперь к этому рисунку можно обращаться и xml-разметки и Java-кода!

В итоге файл макета второго фрагмента будет выглядеть следующим образом:

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

    <ImageView
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_centerInParent="true"
        android:src="@drawable/mirea_android_logo"/>

</RelativeLayout>
```

Рисунок 33

В режиме дизайна:

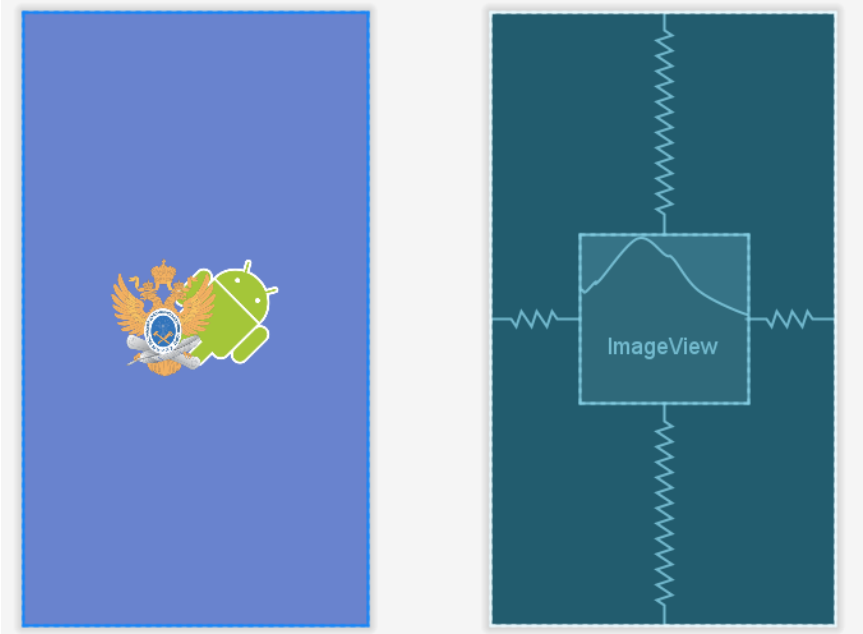

Рисунок 34

Если у вас нет цвета с именем my_custom_blue, добавьте его в файл colors.xml.

Перейдите в класс первого фрагмента и реализуйте в методе **onCreateView** изменение значения текстового поля:

```java
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View rootView =  inflater.inflate(R.layout.fragment_first,
            container,
            attachToRoot: false);
    TextView tv = rootView.findViewById(R.id.tvFragmentWelcome);
    tv.setText("It is a first fragment!");

    return rootView;
}
```

Рисунок 35

LayoutInflater – это класс, который умеет из содержимого layout-файла создать View-элемент. Метод, который это делает называется inflate. Есть несколько реализаций этого метода с различными параметрами. Но все они используют друг друга и результат их выполнения один – View.

Мы рассмотрим эту реализацию – public View inflate (int resource, ViewGroup root, boolean attachToRoot)

Как видим, на вход метод принимает три параметра:

resource - ID layout-файла, который будет использован для создания View.

root – родительский ViewGroup-элемент для создаваемого View. LayoutParams от этого ViewGroup присваиваются создаваемому View.

attachToRoot – присоединять ли создаваемый View к root. Если true, то root становится родителем создаваемого View. Т.е. это равносильно команде root.addView(View). Если false – то создаваемый View просто получает LayoutParams от root, но его дочерним элементом не становится.

Перейдите в метод **onCreate** класса **TestFragmentActivity** и реализуйте в нем механизм смены создания и смены фрагментов в контейнере RelativeLayout:

```
public class TestFragmentActivity extends AppCompatActivity {
    private int counterFragments = 1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_test_fragment);
        Fragment exampleFrg1 = new FirstFragment();
        Fragment exampleFrg2 = new SecondFragment();
//...
        Button btnChange = findViewById(R.id.btnChangeFragment);
        btnChange.setOnClickListener(v -> {
            FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
            //ft.setCustomAnimations(R.anim.right_in,R.anim.rigth_out);
            if(counterFragments % 2 == 0){
                ft.replace(R.id.relativeContainer,exampleFrg2);
            }else{
                ft.replace(R.id.relativeContainer,exampleFrg1);
            }
            ft.commit();
            counterFragments++;
        });
    }
}
```

Рисунок 36

И последний шаг – необходимо реализовать запуск самой **TestFragmentActivity**. Вернитесь в класс главной activity и напишите код аналогичный первому упражнению (только теперь запуск **TestFragmentActivity**) в методе onClick.

Рисунок 37

Запустите приложение. **Убедитесь**, что фрагменты корректно меняются при нажатии на кнопку Change.