Лабораторная работа №3. Работа с экранами

Задание: создать приложение, которое состоит из нескольких activities. Первая activity содержит элемент TextView с названием или номером activity, текстовое поле EditText для ввода какой-то информации, кнопку Button с названием "Next" или "Перейти на 2 activity/экран/окно" или просто "2". Помимо этого, в 1 activity должен быть TextView с ФИО студента и группой. После нажатия на эту кнопку происходит переход на вторую activity, где содержится TextView с названием или номером activity, TextView с надписью что-то вроде "В первом окне вы напечатали:" и под ним - ещё один TextView с содержимым EditText с первой activity, и, разумеется, кнопка "1" или "Вернуться на 1 экран" или "Вернуться к вводу текста", нажав на которую пользователь может перейти обратно к 1 activity. Запустить на эмуляторе и убедиться, что всё работает.

Создайте новый проект с Empty Views Activity и удалите стандартный TextView с фразой «Hello World!». Теперь в первой activity (файл main_activity.xml) расставьте элементы TextView с текстом «Activity 1», EditText с фоновым текстом (hint) «Введите имя» и кнопку Button с текстом «NEXT». Не забудьте свои ФИО и группу. Пример расстановки элементов показан на рисунке 32.

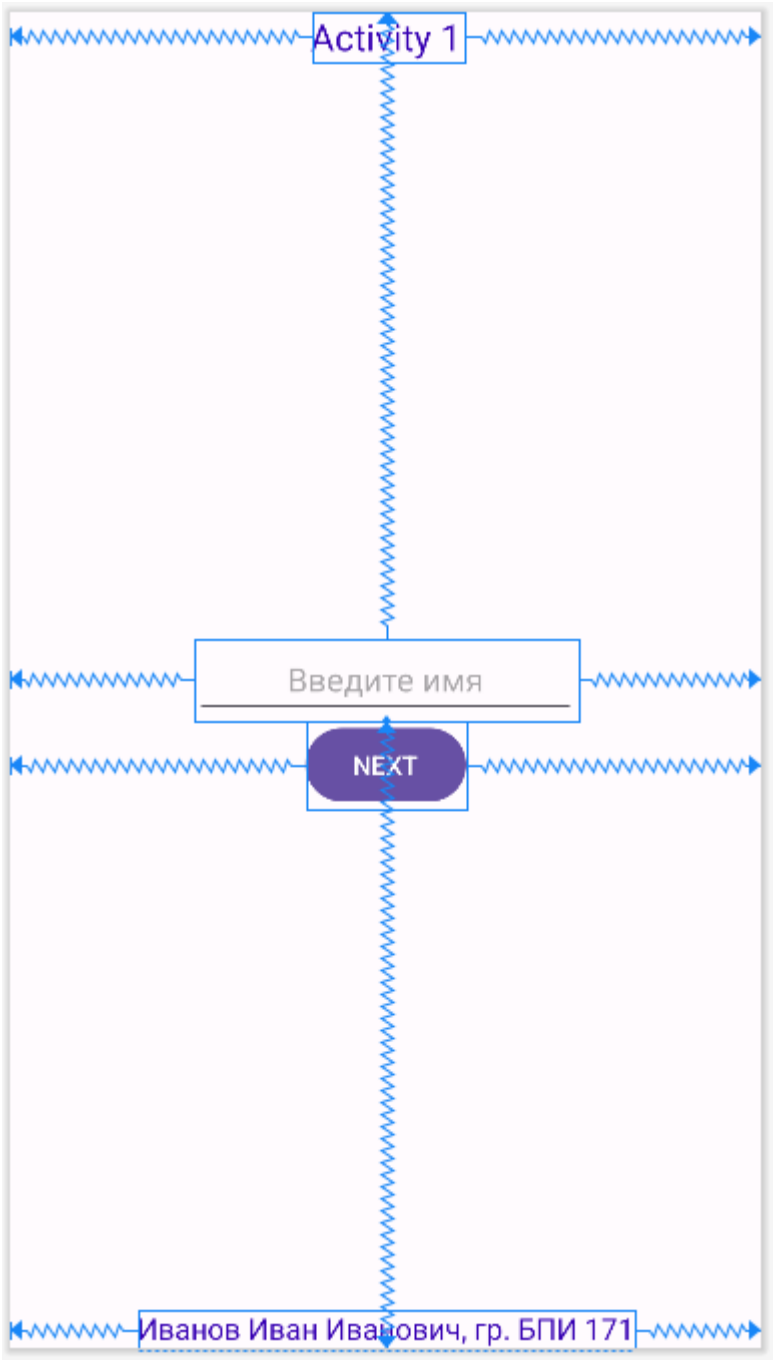

Рисунок 32 – Пример расстановки элементов в файле activity_main.xml

Добавим вторую activity, для этого слева наверху нажмите по папке app правой кнопкой мыши или выберите пункт главного меню FIle и выберите пункт New -> Activity -> Empty Views Activity, как показано на рисунке 33.

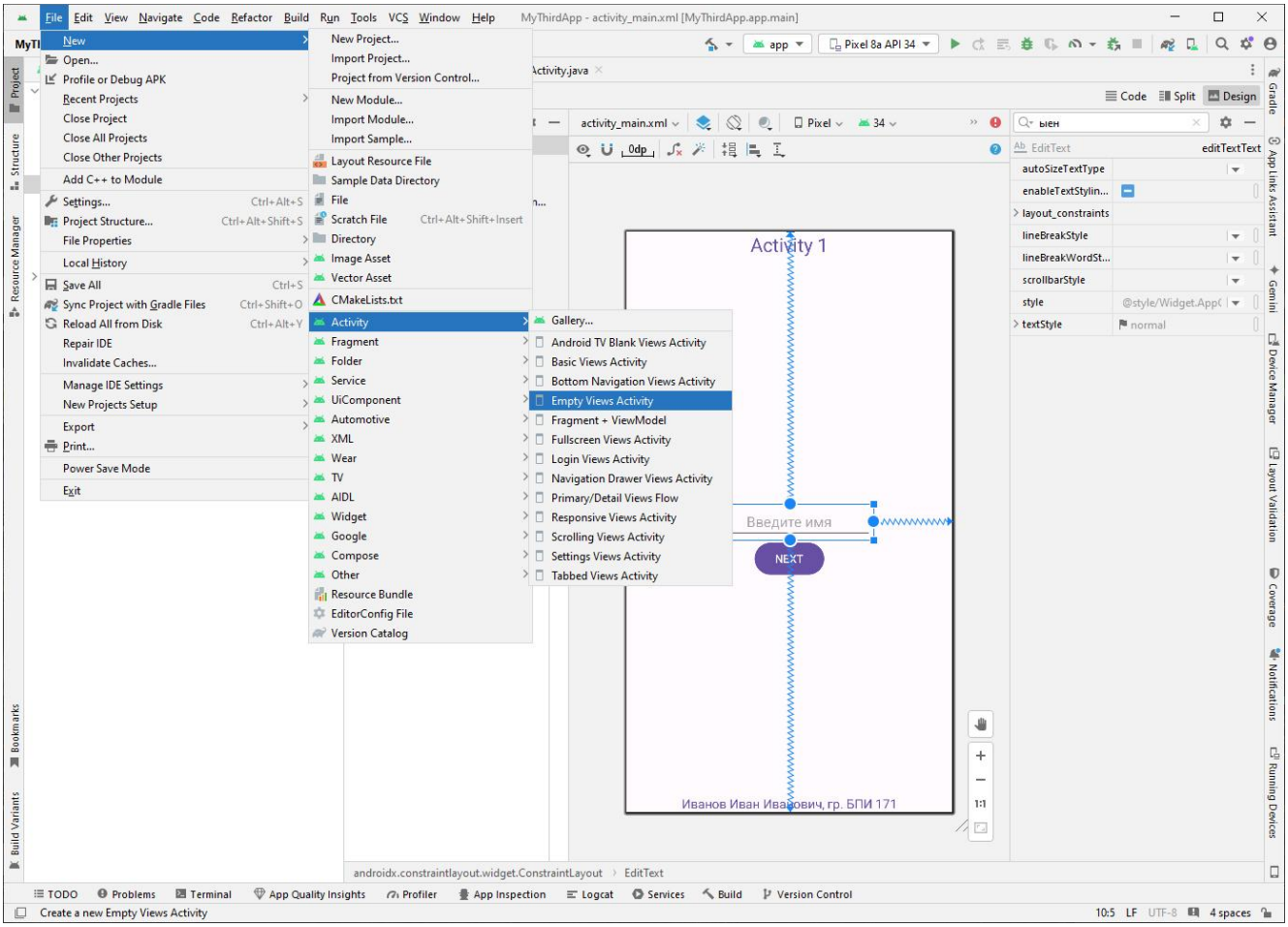

Рисунок 33 – Добавление новой activity

После этого появится стандартное окно создания activity, можете оставить все названия по умолчанию и нажать Finish, либо поменять название в поле Activity Name на «SecondActivity», остальные поля подстроятся под это название автоматически. Таким образом, создалось два новых файла: SecondActivity.java или SecondActivity.kt и activity_second.xml.

Перейдите в файл activity_second.xml и расставьте там следующие элементы: TextView с текстом «Activity 2», TextView с текстом «Вы ввели:», TextView без текста и кнопку Button для перехода обратно в первую activity. Пример расстановки элементов показан на рисунке 34.

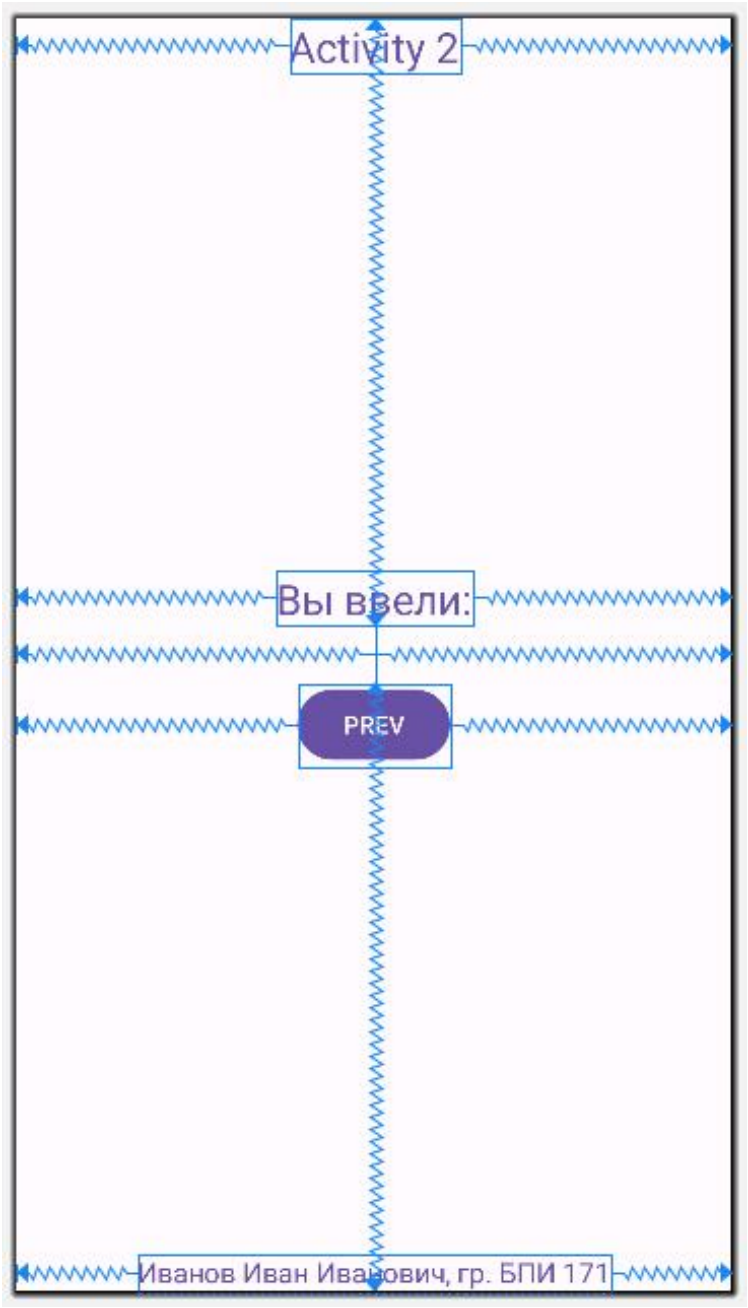

Рисунок 34 – Пример расстановки элементов в файле activity_second.xml

После расстановки интерфейсных элементов в двух activity необходимо написать код для взаимодействия элементов с activity. Перейдите в файл MainActivity и после объявления класса создайте переменную:

Java:

```
public static String text2remember;
```

Kotlin:

Мы используем механизм Extras на Kotlin, пока ничего не надо, никаких переменных

В ней будет храниться текст из поля EditText в первой activity и использоваться из неё в TextView во второй activity. Далее в методе onCreate определите нужные элементы (не забудьте про зависимости, импортировать их можно по подсказке в Android Studio после написания этого кода):

Java:

```
EditText editText = findViewById(R.id.editTextText);
Button button = findViewById(R.id.button);
```

Kotlin:

```
val editText = findViewById<EditText>(R.id.editTextText)
val button = findViewById<Button>(R.id.button)
```

Заметьте, что если раньше каст (cast, или (EditText), (Button) и т.д.) перед методом findViewById нужно было указывать обязательно на языке Java, в новых версиях Android Studio это необязательно. Однако для Kotlin это обязательно. Теперь присвойте текст полю editText текст из переменной text2remember:

Java:

```
editText.setText(text2remember);
```

Kotlin:

```
editText.setText(getIntent().getStringExtra("text2remember"));
```

Делается это для того, чтобы при возвращении из второго activity в первое в поле TextView было написано то, что было написано до перехода во вторую activity. Теперь создайте для кнопки listener и метод onClick (как во второй лабораторной работе) и внутри метода onClick напишите следующее:

Java:

```
text2remember = editText.getText().toString();
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
startActivity(intent);
```

Kotlin:

```
val intent = Intent(this@MainActivity, SecondActivity::class.java)
intent.putExtra("text2remember", editText.text.toString())
startActivity(intent)
```

Здесь переменной text2remember присваивается текст из поля EditText и определяется понятие Intent для запуска второй activity. На Kotlin мы используем механизм Extras передачи данных с одной activity на другую, т.е. intent для второго класса создаётся с прицепленной к нему переменной `text2remember` со значением из поля EditText, т.е. с тем, что ввёл кпользователь в это поле.

Теперь перейдите в файл SecondActivity.java и в методе onCreate определите интерфейсные элементы и присвойте полю TextView, которое располагается ниже поля с текстом "Вы ввели:" или выше кнопки (надо узнать его точный id, это может быть не textView3, как в коде ниже) текст из переменной text2remember:

Java:

```
TextView textView = findViewById(R.id.textView3);
textView.setText(MainActivity.text2remember);
Button button = findViewById(R.id.button2);
```

Kotlin:

```
val textView = findViewById<TextView>(R.id.textView3)
textView.text = getIntent().getStringExtra("text2remember")
val button = findViewById<Button>(R.id.button2)
```

Здесь в коде на Kotlin мы вынимаем переменную из intent, переданную через механизм Extras, и присваиваем её значение элементу TextView, расположенному над кнопкой Prev. Осталось только создать для кнопки listener и метод onClick для возвращения на первую activity. Создайте их и внутри метода onClick напишите:

Java:

```
Intent intent = new Intent(SecondActivity.this, MainActivity.class);
startActivity(intent);
```

Kotlin:

```
val intent = Intent(this@SecondActivity, MainActivity::class.java)
intent.putExtra("text2remember",textView.text.toString())
startActivity(intent)
```

Не забывайте про импорт зависимостей, среда разработки вам об этом напомнит и поможет это сделать. В коде на Kotlin мы посылаем переменную `text2remember` обратно в EditText для первой activity, чтобы, перейдя в неё, EditText содержал ранее введённый пользователем текст.

Заметьте, при создании второй activity в файл манифеста AndroidManifest.xml автоматически добавились строки:

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

Каждая activity приложения должна быть упомянута в файле AndroidManifest.xml. Так как мы использовали шаблон добавления activity из меню Android Studio, она сама добавила эти строки в файл манифеста приложения, иначе нам бы пришло это делать вручную.

Теперь запустите приложение и протестируйте, всё должно работать. Пример работы приложения показан на рисунках 35 и 36.

Тут нужно отметить, что это не очень хороший код, и способ размножения activities в памяти никуда не годится. Во-первых, данные можно передавать не только через переменную, но и через механизмы Extras и Bundles, поэтому переменная, по сути, и не будет нужна. Во-вторых, чтобы вернуться на 1 activity, не надо создавать новую activity и восстанавливать в нужном поле EditText то, что ввёл туда пользователь. В-третьих, с этим связана и главная проблема этого кода: каждый раз нажатие кнопки Next и особенно Prev порождает в памяти всё новые и новые экземпляры классов MainActivity и SecondActivity, в итоге, если очень постараться, Android в конечном итоге убьёт наше приложение, так как оно будет пожирать все ресурсы системы на поддержание жизни сотен, если не тысяч порожденных окон-activities. Чтобы этого избежать, в коде нажатия на кнопку Prev достаточно написать одну строку: finish(). Этот метод завершает работу activity. Соответственно, если мы так сделаем, вторая activity будет

убиваться, и фокус будет передаваться первой activity каждый раз, когда мы будем нажимать на кнопку Prev. Иными словами, в памяти будут в каждый момент времени находиться либо две activities, либо одна – первая. И, как можно увидеть, текст, введённый пользователем в EditText на первой activity после убийства второй будет сохраняться, поэтому и запоминать и восстанавливать его не нужно при возвращении со второй activity на первую.

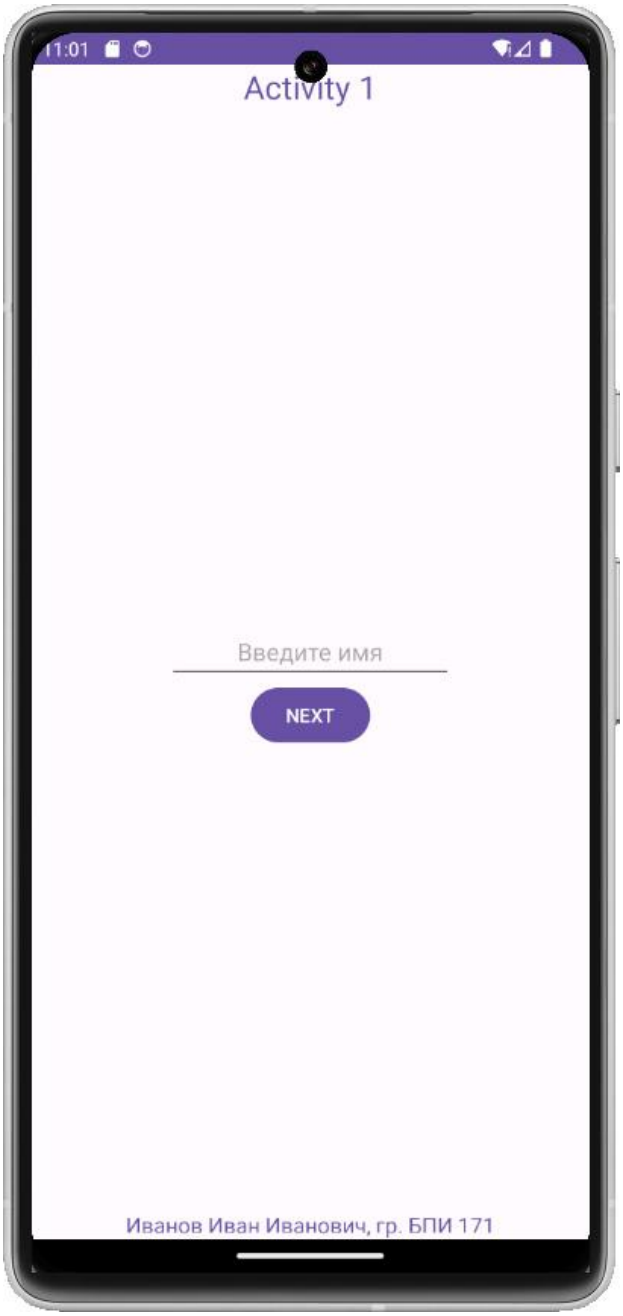

Рисунок 35 – Вид первой activity

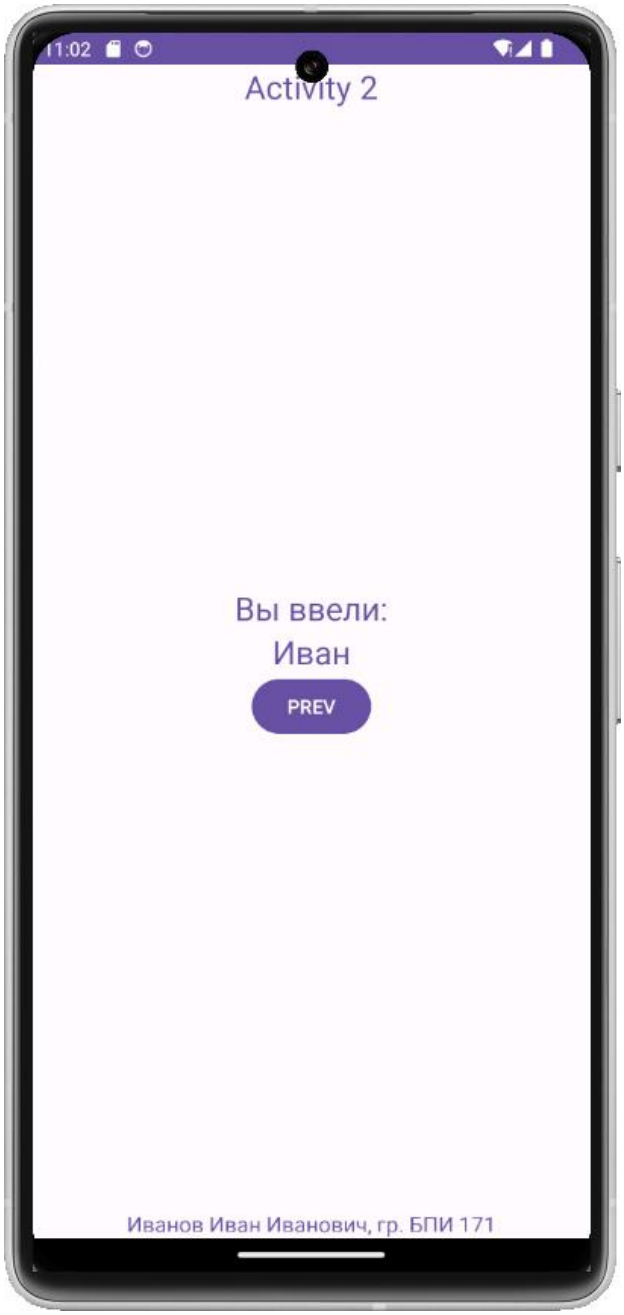

Рисунок 36 – Вид второй activity