Лабораторная работа №2. Работа с элементами

Задание: создать приложение, которое выводит в элемент TextView надпись, введённую пользователем в текстовом поле EditText после нажатия на кнопку Button. Помимо этого, в Activity должен быть TextView с ФИО студента и группой. Запустить на эмуляторе и убедиться, что всё работает.

Создайте новый проект (File -> New -> New Project) с Empty Views Activity, удалите из интерфейса стандартный TextView с фразой «Hello World!» и поместите поле EditText. Для этого из панели слева (Palette) перетащите на экран смартфона объект под названием Plain Text из категории Text. Plain Text - самый простой вариант текстового поля для пользовательского ввода. Затем расположите поле TextView и элемент Button. Пример расположения элементов показан на рисунке 29. Не забудьте про элемент с ФИО и группой.

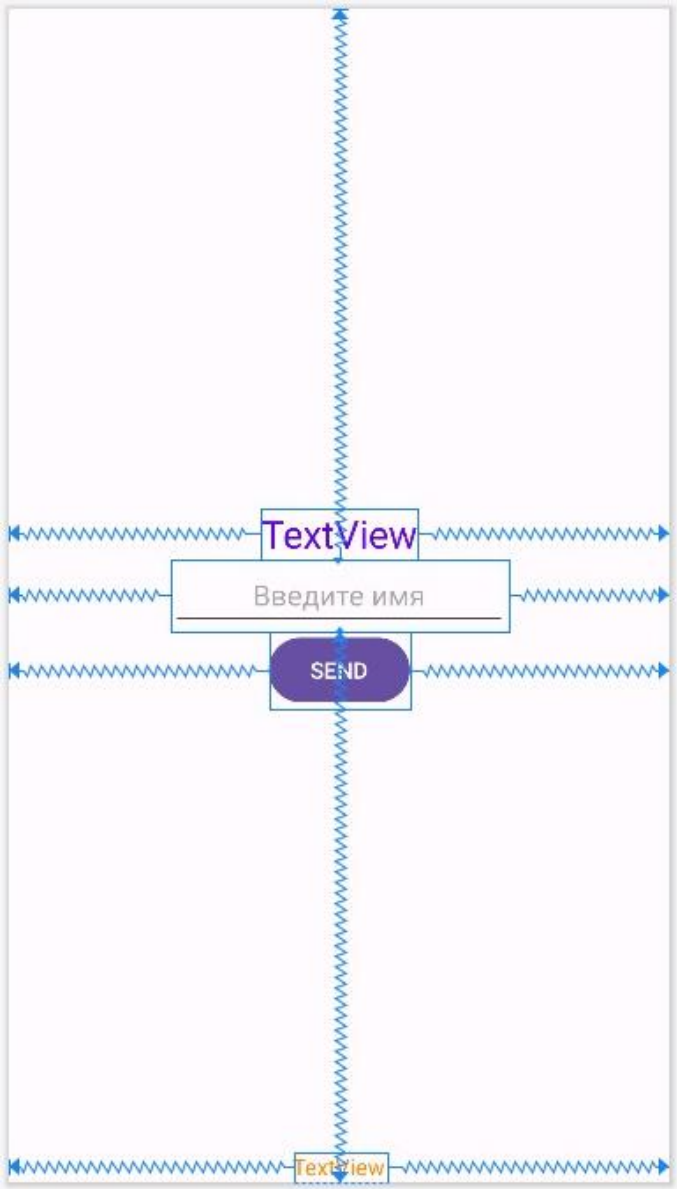

Рисунок 29 – Расположение элементов на экране приложения

Что касается интерфейса, нужно помнить, что, если вы работаете с ConstraintLayout, вам нужно в окне дизайна привязать используемые 4 элемента интерфейса к границам экрана и связать их между собой, иначе весь интерфейс будет в беспорядке. Как правило, обычно можно выбрать один интерфейсный элемент, который у нас будет главным по логике приложения, расположить его в центре экрана, и затем остальные элементы привязать к нему. В данном случае на рисунке 29 видно, что таким главным элементом сделан EditText, а остальне элементы расположены по центру по горизонтали и зависят от EditText-элемента по вертикали (все, кроме элемента с ФИО). Старайтесь всегда делать интерфейс аккуратно и красиво, так, чтобы при повороте экрана, например, с интерфейсом ничего плохого не произошло бы. В данном случае, для EditText установлено свойство gravity в значение center, для TextView изменён цвет и шрифт.

В поле EditText мы будем вводить текст, и после нажатия на кнопку этот текст будет отображаться в фиолетовом элементе TextView. Дополнительно в поле EditText можно указать какой-нибудь фоновый текст, чтобы пользователь понимал, что делать с этим полем. Чтобы добавить фоновый текст в EditText (Введите имя, см. рисунок 29), необходимо выделить этот элемент и в панели его

свойств справа найти пункт hint, туда можно вписать любую фразу. Можно написать текст просто словами, либо же через файл strings.xml. В нём достаточно добавить строку:

```
<string name="hint_editText">Введите имя</string>
```

А в поле hint нужно сослаться на данную строку следующим образом:

```
@string/hint_editText
```

Можете поступать так, как вам удобнее, но всё же правильнее писать все текстовые переменные в файле strings.xml. Затем также переименуйте кнопку (свойство text в значение SEND), позаботьтесь о поле ФИО внизу экрана, как это было сделано в первой лабораторной работе, и - переходим к написанию кода в файл MainActivity.

В объявлении класса MainActivity необходимо убедиться, что класс наследуется от AppCompatActivity. Так нужно будет делать всегда, если в лабораторной работе не указано иное.

Далее в методе onCreate необходимо написать код для взаимодействия элементов. Метод onCreate выполняется всегда при загрузке Activity. Для начала определим в коде нужные нам элементы, с которыми мы будем работать, для этого в методе onCreate напишем следующее:

Java:

```
final TextView textView = findViewById(R.id.textView);
final EditText editText = findViewById(R.id.editTextText);
Button button = (Button) findViewById(R.id.button);
textView.setText(""); // чтобы при старте не был виден
```

Kotlin:

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

Ошибки в коде исправляются путём импорта соответствующих классов. Так как перенос текста из EditText в TextView должен происходить после нажатия на кнопку, то для кнопки необходимо создать событие onClick. Для этого напишите следующее:

Java:

```
button.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
    textView.setText("Привет, " + editText.getText().toString() + "!");
```

```
  }
});
```

Kotlin:

```
button.setOnClickListener { textView.text = "Привет, " + editText.text + "!" }
```

На самом деле среда разработки может написать большую часть кода за вас, просто начните писать button.se, например, на Java, а дальше можно нажать Enter, так как у вас в подсказке (выпадающем списке) будет нужная вам конструкция, получится следующее:

```
button.setOnClickListener();
```

Теперь в скобках начните писать «new » и на первой строчке в подсказке будет View.OnClickListener{…} (android.view.View), нажимайте Enter и основная часть конструкции написана автоматически:

Java:

```
button.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {


      }
});
```

Kotlin:

```
button.setOnClickListener {}
```

Осталось в метод onClick написать нужный нам код:

Java:

```
textView.setText("Привет, " + editText.getText().toString() + "!");
```

Kotlin:

```
textView.text = "Привет, " + editText.text + "!"
```

Всё довольно легко, и к данным подсказкам легко привыкнуть. В данной строчке кода элементу textView присваивается текст «Привет, [текст из editText]!». Запустите приложение и проверьте его работу, введя имя и нажав на кнопку. Результат должен быть примерно таким, как изображено на рисунке 30. На рисунке видно, что имя введено кириллицей. Чтобы иметь возможность вводить данные на русском языке, нужно перейти в меню эмулятора (смартфона, показана инструкция для Pixel 8a) Settings -> System -> Languages -> System Languages -> Add a language, рисунок 31, слева, и затем выбрать Русский, Россия из списка. В результате получите установленный русский язык на эмулятор, как

показано на рисунке 31, справа. И теперь в приложении на экранной клавиатуре можно переключить язык, если нажать на изображение глобуса (которого раньше не было) слева от пробела.

У приложения есть некоторый недостаток: если ничего не ввести в текстовое поле, оно все равно напишет «Привет, !». Этого можно избежать, если добавить в метод onClick условие проверки содержимого поля EditText:

Java:

```
if (!editText.getText().toString().isEmpty()) {
   textView.setText("Привет, " + editText.getText().toString() + "!");
}
```

Kotlin:

```
if (!editText.text.isEmpty()) {
    textView.text = "Привет, " + editText.text + "!"
}
```

Казалось бы, этого достаточно, но – нет! В случае, если мы сначала ввели какое-то имя, а потом очистили текстовое поле, нам всё равно будет отображаться предыдущий привет. Поэтому без else тут не обойтись:

Java:

```
if (!editText.getText().toString().isEmpty()) {
   textView.setText("Привет, " + editText.getText().toString() + "!");
} else {
   textView.setText("");
}
```

Kotlin:

```
if (!editText.text.isEmpty()) {
    textView.text = "Привет, " + editText.text + "!"
} else {
    textView.text = ""
}
```

Теперь все должно работать корректно.

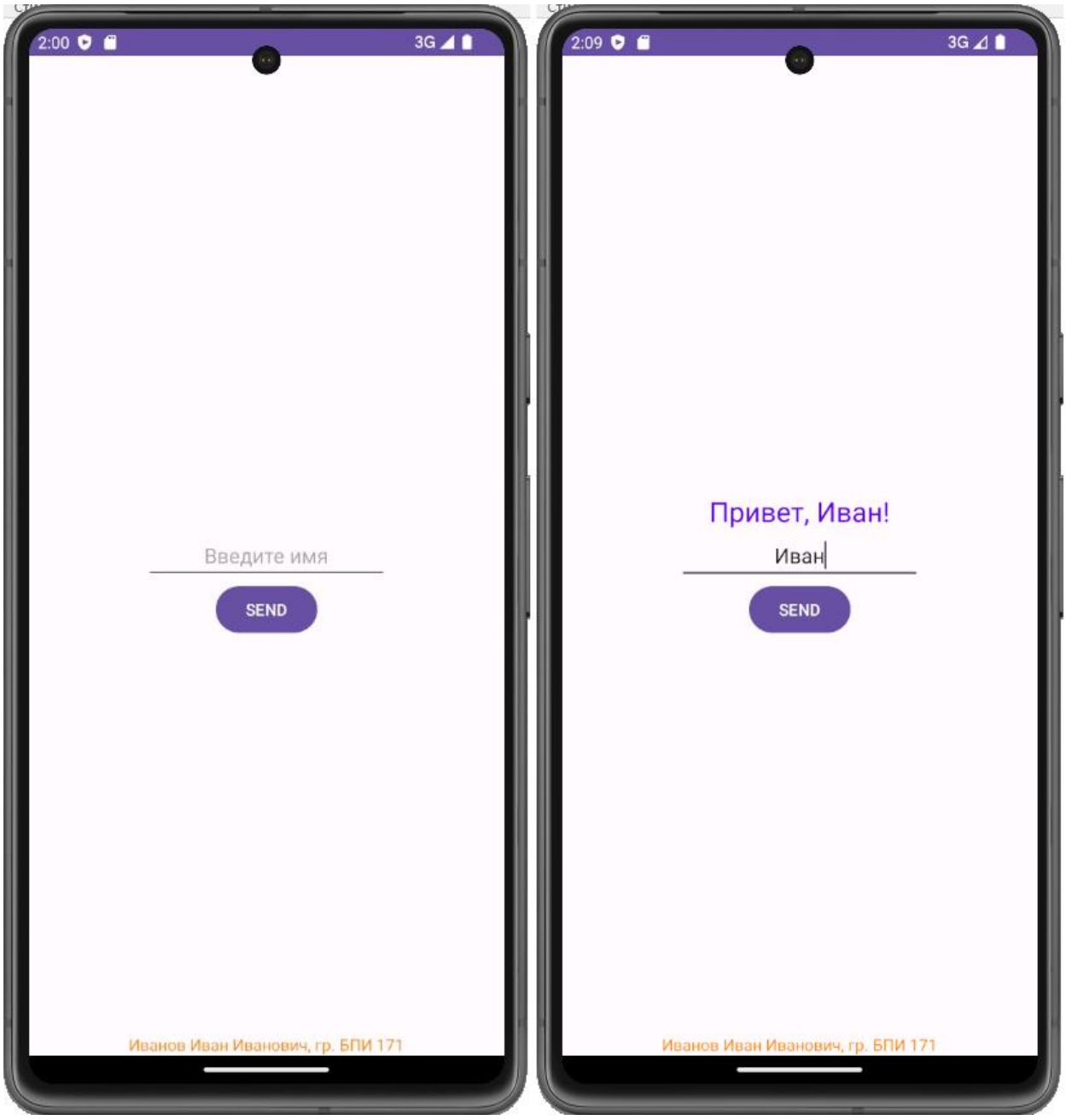

Рисунок 30 – Вид приложения до ввода текста и после нажатия кнопки

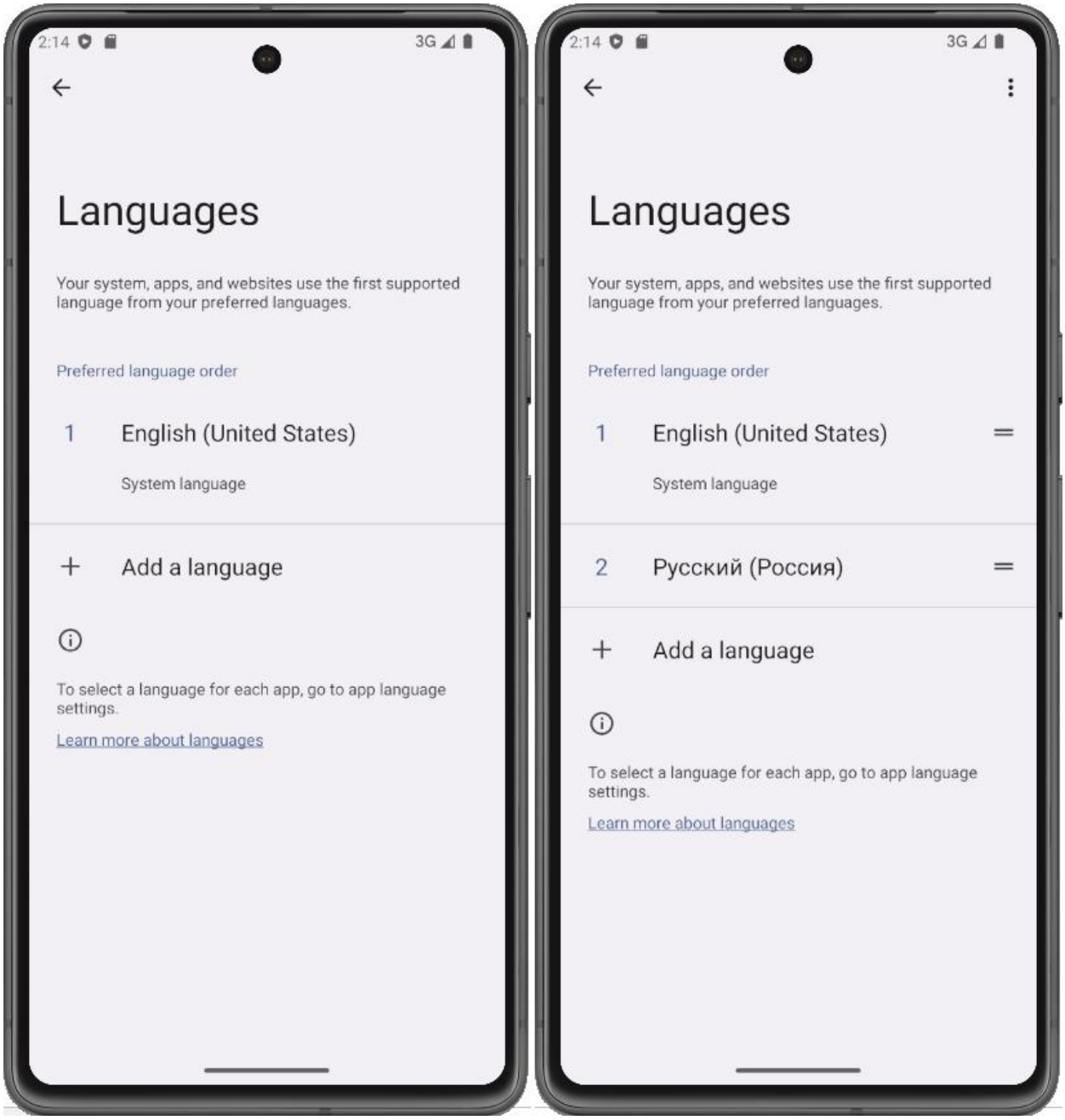

Рисунок 31 – Добавление нового языка на эмулятор или смартфон