**Учебное пособие по выполнению практических заданий по курсу "Разработка облачных и мобильных приложений на платформе Google Android" в среде Android Studio**



Авторы: Майоров В. С., Макаров С.Л.

Москва 2018

Содержание

Лабораторная работа №1. Создание первого приложения 3

Лабораторная работа №2. Работа с элементами 20

Лабораторная работа №3. Работа с экранами 25

Лабораторная работа №4. Стили и темы 32

Лабораторная работа №5. Работа со списками 36

Лабораторная работа №6. Работа с анимацией 42

Лабораторная работа №7. Работа с картами 47

Лабораторная работа №8. Создание виджета 53

Лабораторная работа №9. Работа с меню 60

Лабораторная работа №10. Работа с диалоговыми окнами 66

Лабораторная работа №11. Работа с уведомлениями 70

Лабораторная работа №12. Работа с аудио 72

Лабораторная работа №13. Работа с видео 73

Лабораторная работа №14. Работа с камерой 75

Лабораторная работа №15. Работа с настройками и БД 78

### **Лабораторная работа №1. Создание первого приложения**

Задание: скачать и установить последние версии Java Development Kit и Android Studio, установить необходимые компоненты SDK, сконфигурировать эмулятор, создать новый проект, изменить TextView с надписью "Hello world" на свои ФИО и запустить проект на эмуляторе.

Для того, чтобы начать программировать под Android, необходимо скачать и установить две вещи: Java Development Kit (JDK) и Android Studio. Первым делом необходимо установить JDK, для этого нужно перейти по адресу <http://www.oracle.com/technetwork/java/javase/downloads/index.html>, выбрать актуальную версию (Java SE Development Kit 10.0.2), принять лицензию, скачать и установить с параметрами по умолчанию. Вторым шагом идёт установка Android Studio. Перейдите по адресу <http://developer.android.com/sdk/index.html>, нажмите на кнопку «Download Android Studio», примите лицензию, скачайте и установите приложение с параметрами по умолчанию (версия 3.1.4).

После установки запустите Android Studio - появится окно приветствия, которое показано на рисунке 1.

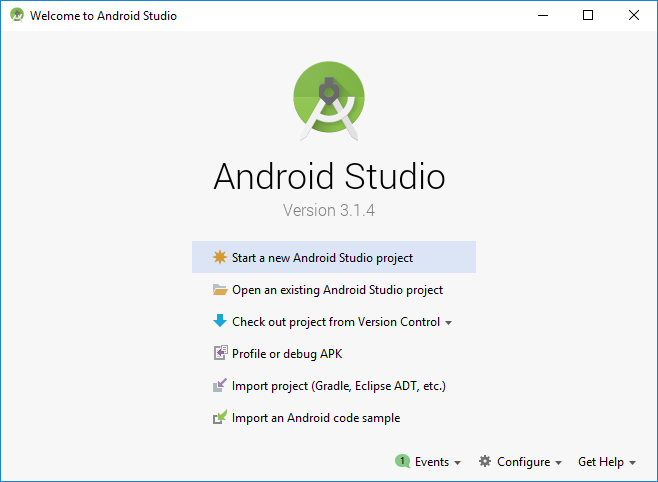


Рисунок 1 – Окно приветствия в Android Studio

Нажмите «Start a new Android Studio project», укажите название приложения в поле «Application name», корневой пакет приложения в поле «Company Domain», который является идентификатором приложения (по умолчанию формируется автоматически в виде имя\_учётной\_записи\_компьютера.example.com), и путь для сохранения проекта «Project location» (лучше оставить по умолчанию). Пример заполнения показан на рисунке 2.

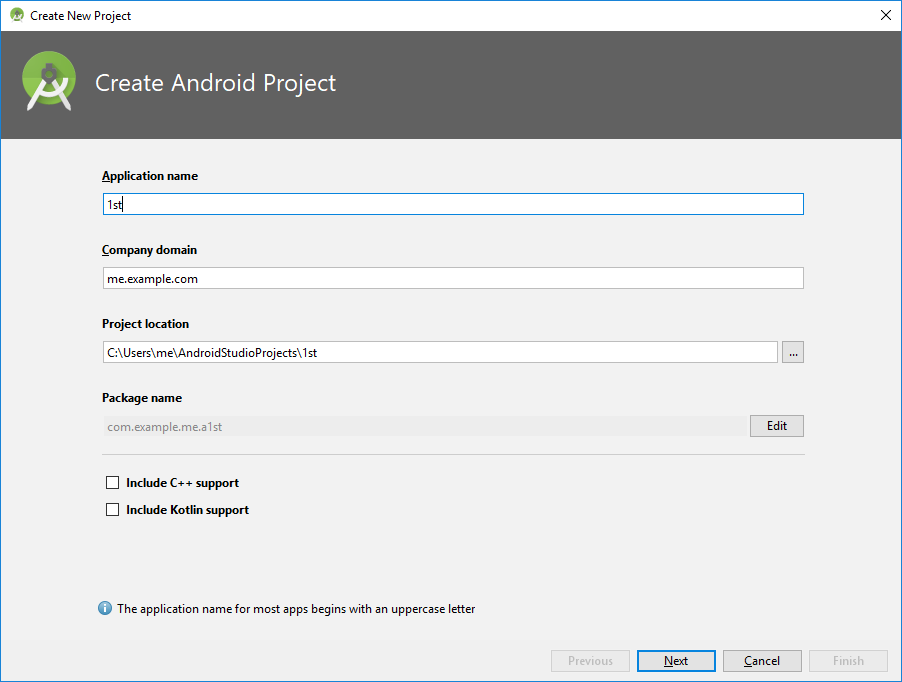


Рисунок 2 – Создание проекта в Android Studio

Нажмите «Next» и увидите страницу с выбором поддерживаемых устройств (телефон и планшет, телевизор, часы, очки) и версий операционной системы (рисунок 3). В рамках данного курса будут разрабатываться приложения для смартфонов. Google рекомендует использовать как можно меньшую версию API, чтобы приложение можно было установить на большее количество устройств, однако, чем меньше версия API, тем меньше функций будет поддерживать приложение. Пока что Google по умолчанию советует использовать API 15, что позволяет установить разрабатываемое приложение более чем на 90% устройств. Для корректного выполнения лабораторных работ не рекомендуется использовать API меньше 15, далее выбор за вами. Если у вас есть устройство под управлением Android, было бы логично выбрать соответствующую версию API.

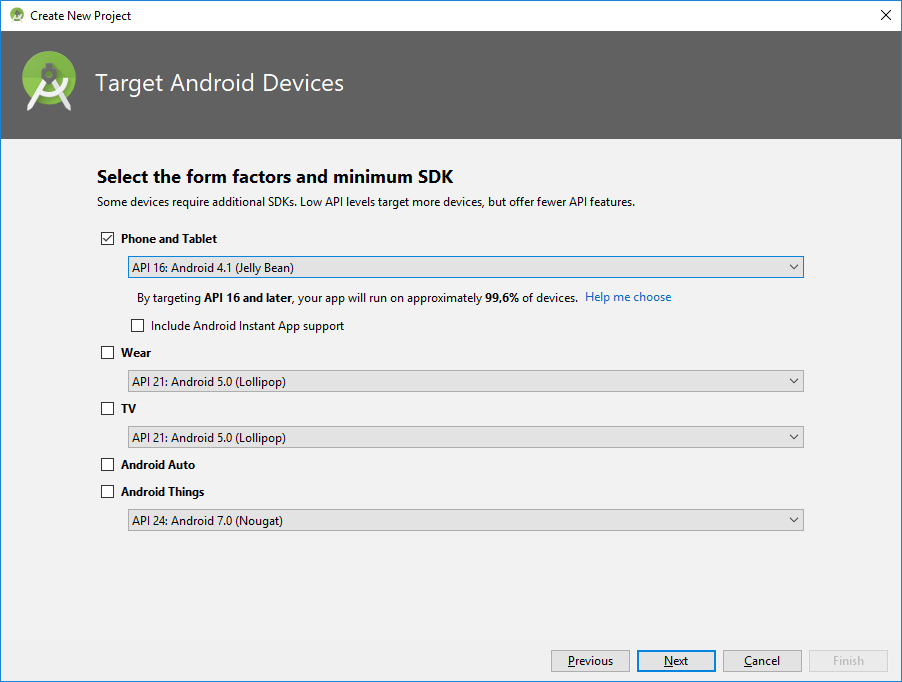


Рисунок 3 – Выбор минимальной версии

Нажмите «Next» и перейдёте на страницу с шаблонами различных Activity (рисунок 4). В большинстве лабораторных работ необходимо выбирать «Empty Activity», в иных случаях об этом будет сказано дополнительно.

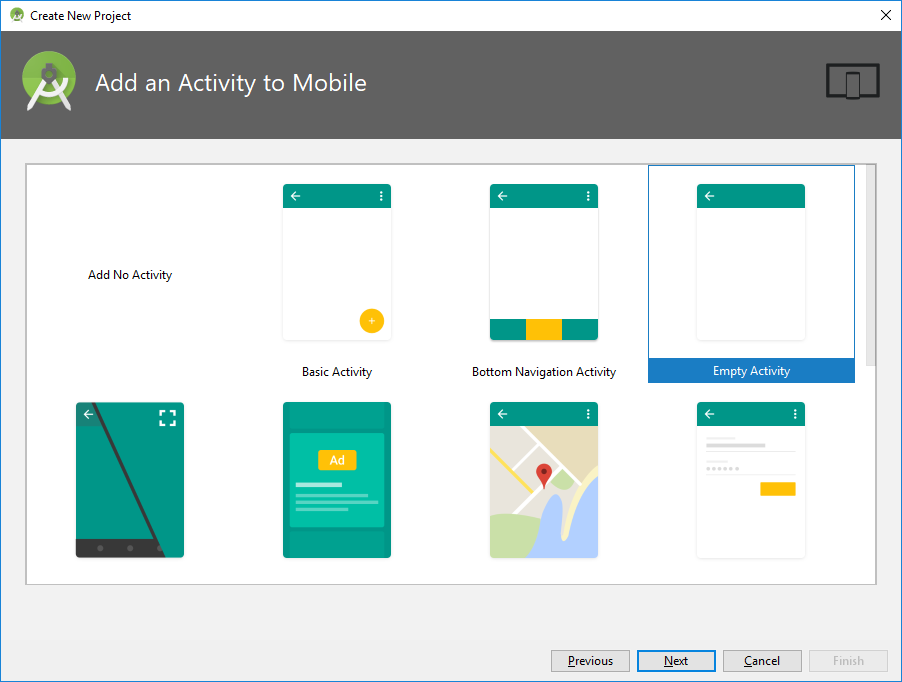


Рисунок 4 – Выбор шаблона Activity

Нажмите «Next» и перейдёте на заключительную страницу, где можно дать название создаваемым файлам activity (имя для файла .java, в котором содержится исходный код приложения) и layout (имя файла .xml, где содержится интерфейс приложения, точнее – того activity, которое было упомянуто выше; не может быть с большой буквы). Для первого Activity обычно оставляют названия по умолчанию, как показано на рисунке 5.

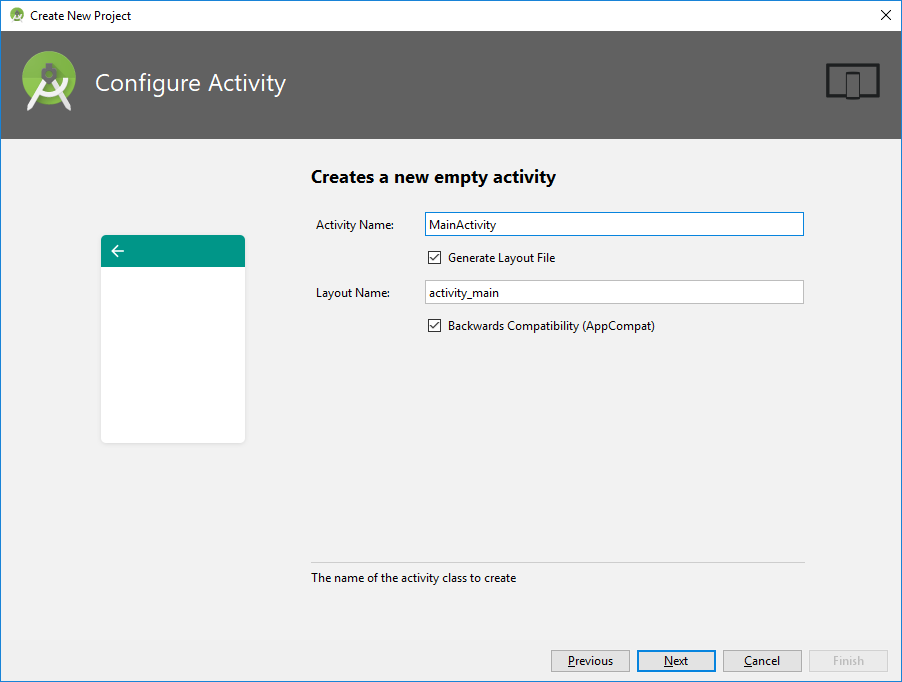


Рисунок 5 – Создание названий файлов

Нажмите «Finish», и через некоторое время, нужное на загрузку необходимых компонентов и построение проекта (рисунок 6), отобразится главное окно среды разработки, которое показано на рисунке 7.

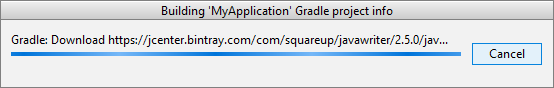


Рисунок 6 – Создание проекта

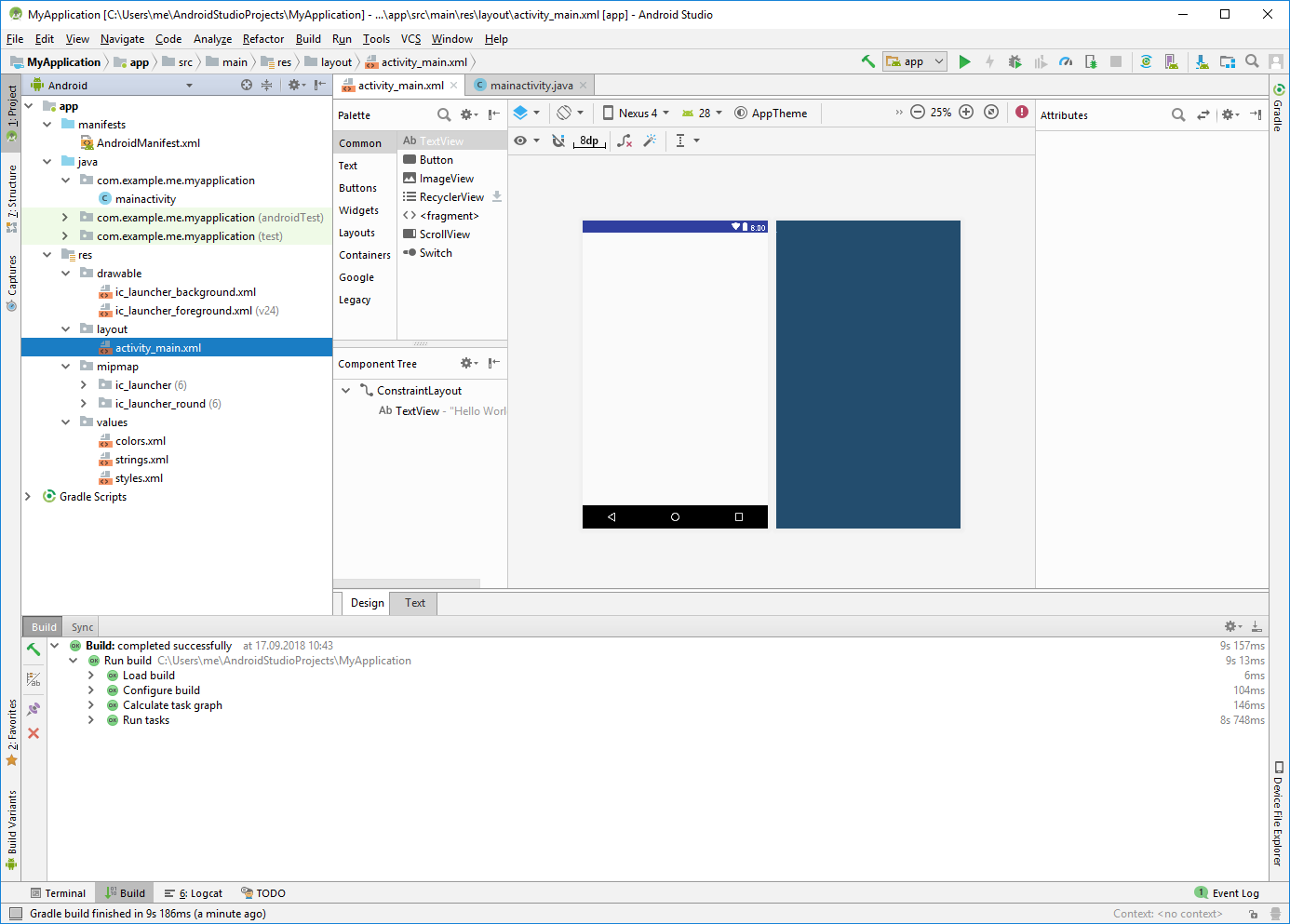


Рисунок 7 – Рабочая область Android Studio

Основные файлы и папки проекта:

* manifests/AndroidManifest.xml - данный файл предоставляет основную информацию о программе системе и является центральным для приложения, в нём перечисляются все используемые Activity, атрибуты и разрешения, которыми обладает приложение, указывается его иконка, название и т.д.
* java/com.example.me.myapplication – содержит файлы с исходным кодом на языке Java. Именно в этой папке размещаются все классы, создаваемые в процессе разработки приложения
* res/ – содержит структуру папок ресурсов приложения, рассмотрим некоторые из них:
  + layout - в данной папке содержатся xml-файлы, которые описывают внешний вид форм и их элементов, пока там находится только activity\_main.xml – интерфейс MainActivity;
  + values - содержит XML файлы, которые определяют простые значения таких ресурсов, как строки, числа, цвета, темы, стили, которые можно использовать в данном проекте.

При создании проекта сразу же открывается файл MainActivity.java, в котором находится исходный код главного класса проекта. Чтобы открыть интерфейс приложения, нужно два раза кликнуть на res/layout/activity\_main.xml файл, содержащий интерфейс первого экрана (Activity) приложения. В зависимости от версии Android Studio для отображения интерфейса приложения может понадобиться дополнительно скачать и установить различные компоненты. Поскольку ранее мы выбрали Empty Activity, в окне интерфейса приложения ничего нет, кроме TextView с надписью «Hello World!». Добавлять интерфейсные элементы можно двумя способами: в графическом (внизу закладка «Design») и текстовом (внизу закладка «Text») представлениях. Интерфейсные элементы расположены в контейнере Palette слева от отображения интерфейса приложения. В контейнере элементы интерфейса сгруппированы по категориям, например Common (наиболее используемые), Text и т.д. В другом контейнере ниже, Component Tree, расположено дерево всех элементов интерфейса, которые есть в данный момент. На экране уже есть компонент TextView со значением «Hello World!»! Если его не видно, значит, у вас возникли проблемы с рендерингом окна дизайна. Справа вверху есть воскликацельный знак в красном кружке, если нажать его, вы увидите суть проблемы. Для того, чтобы исправить эту проблему, нужно перейти в файл Gradle Scripts -> build.gradle (Module: app) и заменить строку

implementation **'com.android.support:appcompat-v7:28.0.0-rc02'**

строкой

implementation **'com.android.support:appcompat-v7:27.1.1'**

т.е. поменять версию библиотеки AppCompat на более старую, но зато более стабильную. Версия 27.1.1 взята для примера как работающая, можно попытаться поменять на другие версии. После этого необходимо нажать ссылку Sync Now справа вверху, чтобы скачалась указанная более стабильная версия библиотеки. В дальнейшем это придётся делать для каждого задания, поэтому как только появится обновление для Android Studio, обновляйтесь до последней версии (Help – Check for Updates…) – возможно, в ней эта проблема будет решена.

Нажмите один раз левой кнопкой мыши по компоненту TextView, и справа появится окно, в котором можно задать текст и id для данного элемента. Поле «text» уже заполнено фразой «Hello world!». Если нажать на многоточие справа от этой фразы, откроется окно ресурсов, показанное на рисунке 8. Дело в том, что в файле res/values/strings.xml хранятся все текстовые данные приложения, что позволяет довольно быстро изменять названия различных элементов, а также с лёгкостью создавать локализацию под различные языки. В это окно выводятся значения ресурсов, в том числе и из этого файла. Если нажать кнопку Add new resource – New string Value справа вверху, можно с помощью диалоговых окон создать новый строковый ресурс (рисунок 9). После нажатия на кнопку OK ссылка на ресурс автоматически добавляется в поле text элемента TextView.

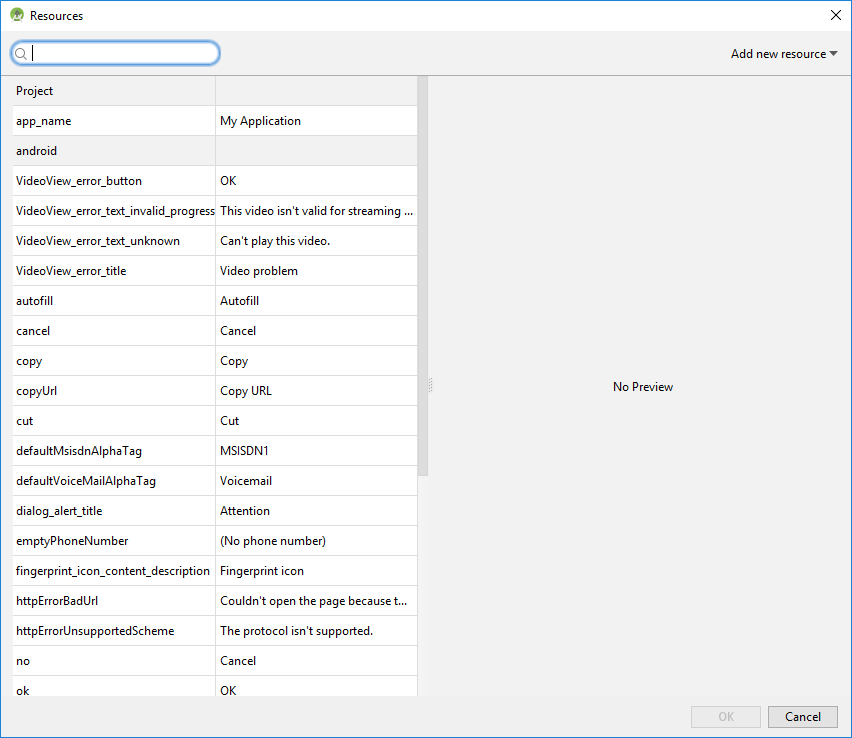


Рисунок 8 – Окно выбора ресурсов

Второй способ создания строкового ресурса – это редактирование непосредственно файла strings.xml. Для этого в файл res/values/strings.xml добавьте новую надпись с именем «fio», для этого необходимо написать следующее:

<**string name="fio"**>Иванов Иван Иванович, гр. БПИ 171</**string**>

После того, как вы создали новую строковую переменную с помощью редактирования файла strings.xml, перейдите в файл activity\_main.xml и измените название надписи на «@string/fio», а в поле id вставьте «fio» и нажмите Enter.

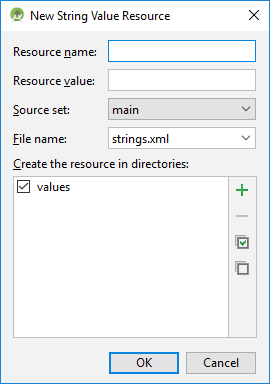


Рисунок 9 – Окно создания нового строкового ресурса

Если всё сделано правильно, то теперь на экране смартфона должны отобразиться ФИО и группа. Осталось запустить приложение на эмуляторе, но для начала его необходимо создать. Нажмите наверху Tools -> AVD Manager (AVD - Android Virtual Device), затем в появившемся окне (рисунок 10) «Create Virtual Device…», выберите модель телефона (Phone), которую хотите эмулировать, например, Nexus 5X, нажмите «Next», выберите образ и скачайте его, нажав на Download. Лучше во избежание излишней "тормозухи" выбрать API 16 (Android 4.1, armeabi-v7a) на закладке Other Images (рисунок 11), однако если у вас мощный компьютер, можно выбрать образы из вкладки Recommended или тот, который соответствует вашему реальному устройству под управлением Android. Закладка x86 Images служит для счастливых обладателей комьютеров на базе процессоров Intel – для этих эмуляторов существует встроенное средство для десятикратного ускорения работы эмулятора (Intel HAXM – Hardware Accelerated Execution Manager). Вариант образа, у которого в скобках написано Google API, служит для прогона запуска приложений, использующих Google Play Services; нам это пока не требуется.

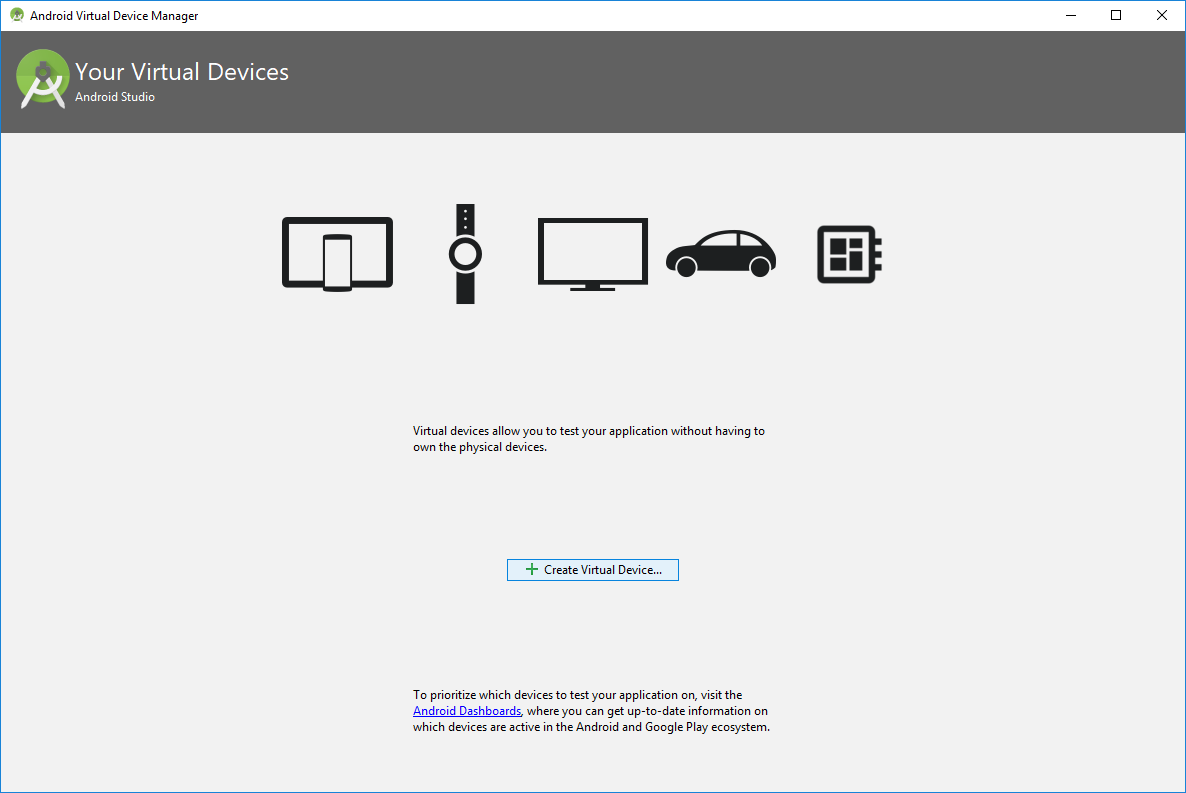


Рисунок 10 – Окно создания эмулятора (AVD)

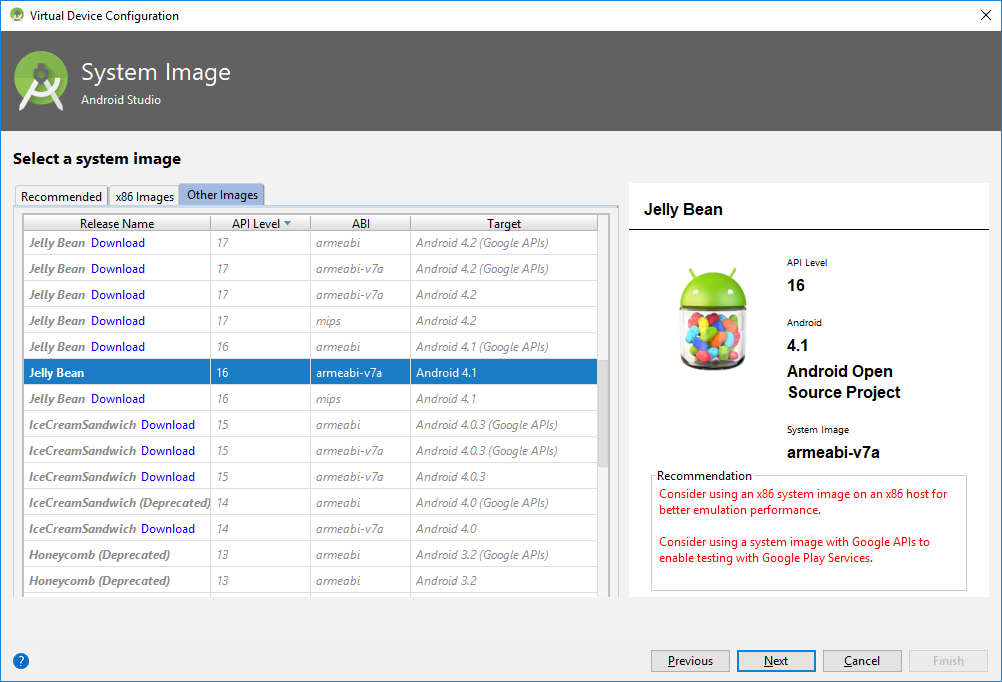


Рисунок 11 – Окно выбора и скачивания образа эмулятора

После скачивания образа устройства и нажатия кнопки Finish в окне скачивания выберите из списка скачанный образ, который теперь выделяется жирным шрифтом (см. рисунок 11). Нажмите «Next», в следующем окне нажмите «Show Advanced Settings» в левом нижнем углу и в поле камера выберите оба поля «Emulated» (рисунок 12). Остальное менять нет необходимости, нажмите «Finish» и закройте окно AVD Manager. Осталось нажать зелёную кнопку «Run ‘app’» вверху справа, выбрать и запустить эмулятор (рисунок 13), нажав кнопку OK, и ждать его загрузки. После того, как эмулятор загрузится, разблокируйте экран смартфона (или нажмите кнопку OK), и сразу же должно запуститься приложение, как показано на рисунке 14. Справа располагается панель управления эмулятором, где можно менять его ориентацию в пространстве, управлять звуком, нажимать аппаратные кнопки, вводить координаты локации и многое другое.

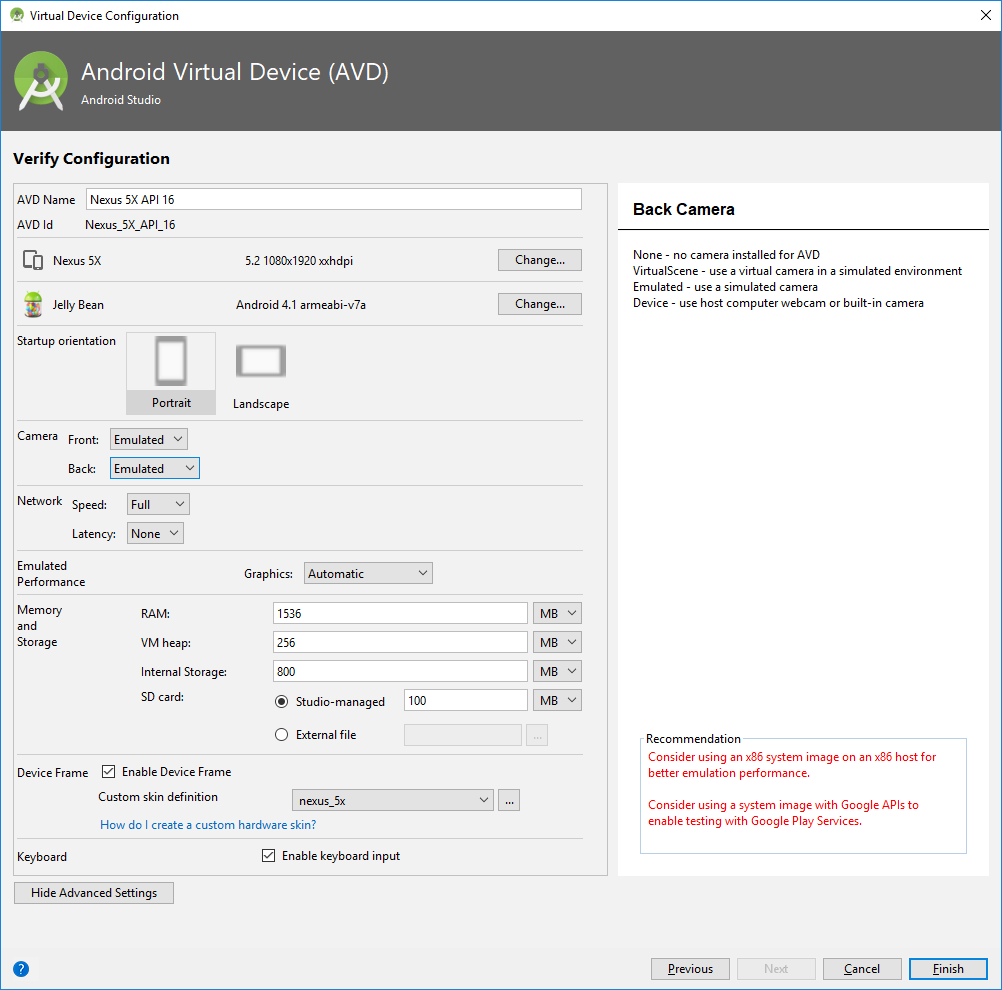


Рисунок 12 – Окно дополнительных свойств эмулятора

Если вы хотите использовать опцию Instant Run (для Instant Apps), необходимо сконфигурировать эмулятор с версией API не ниже 21.

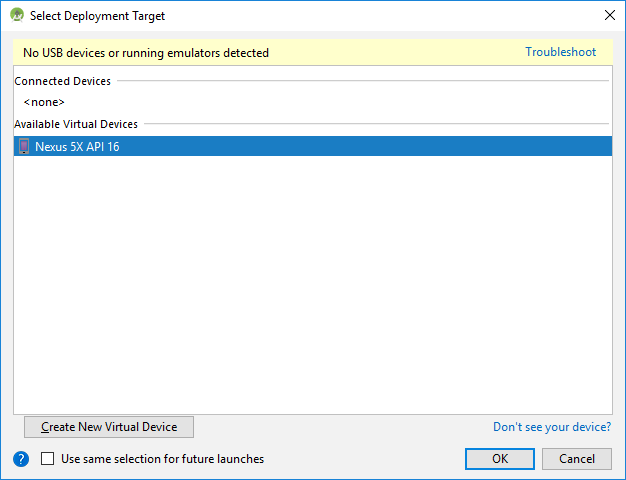


Рисунок 13 – Окно выбора эмулятора для запуска



Рисунок 14 – Эмулятор Android с запущенным приложением

Теперь необходимо проверить установленные компоненты SDK. Для этого перейдите Tools -> SDK Manager и сверьтесь со следующим списком. Если каких-то компонентов не хватает, то следует их установить.

* SDK Tools
  + Android SDK Tools
  + Android SDK Platform-tools
  + Android SDK Build-tools (последней версии)
  + Google Play services
  + Android Support Repository
  + Google Repository
  + Intel x86 Emulator Accelerator (HAXM installer)
  + Google USB Driver
  + Google Play Instant Development SDK (если хотите использовать Android Instant Apps)
* SDK Platforms
  + Android API 16 (или той версии, которая у вас на реальном устройстве)

### **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 Activity, удалите 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 15. Не забудьте про элемент с ФИО и группой.

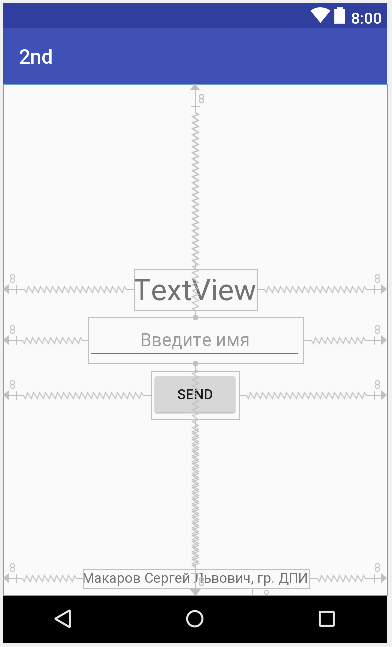


Рисунок 15 – Расположение элементов на экране приложения

Что касается интерфейса, нужно помнить, что если вы работаете с ConstraintLayout, вам нужно в окне дизайна привязать используемые 3 элемента интерфейса к границам экрана и связать их между собой, иначе весь интерфейс будет в одной куче в левом верхнему углу экрана, как будто вы используете FrameLayout.

В поле EditText мы будем вводить текст и после нажатия на кнопку этот текст будет отображаться в элементе TextView. Дополнительно в поле EditText можно указать какой-нибудь фоновый текст, чтобы пользователь видел, куда же ему вводить информацию. Чтобы добавить фоновый текст в EditText необходимо выделить этот элемент и в правой нижней панели его свойств найти пункт hint, туда можно вписать любую фразу. Можно написать текст просто словами, либо же через файл strings.xml. В нём достаточно добавить строчку:

<**string name="hint\_editText"**>Введите имя</**string**>

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

@string/hint\_editText

Можете поступать так, как вам удобнее, но всё же правильнее писать все текстовые поля в файле strings.xml. Затем также переименуйте кнопку, и переходим к написанию кода в файл MainActivity.java.

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 пишем следующее:

**final** TextView textView = (TextView) findViewById(R.id.***textView***);  
**final** EditText editText = (EditText) findViewById(R.id.***editText***);

Button button = (Button) findViewById(R.id.***button***);

textView.setText(**""**); // чтобы при старте не был виден

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:

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

На самом деле среда разработки может написать большую часть кода за вас, просто начните писать button.setO, а дальше можно нажать Enter, так как у вас в подсказке будет нужная вам конструкция, получится следующее:

button.setOnClickListener();

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

button.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
   
 }  
});

Осталось в метод onClick написать нужный нам код:

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

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 рисунках 16 и 17.

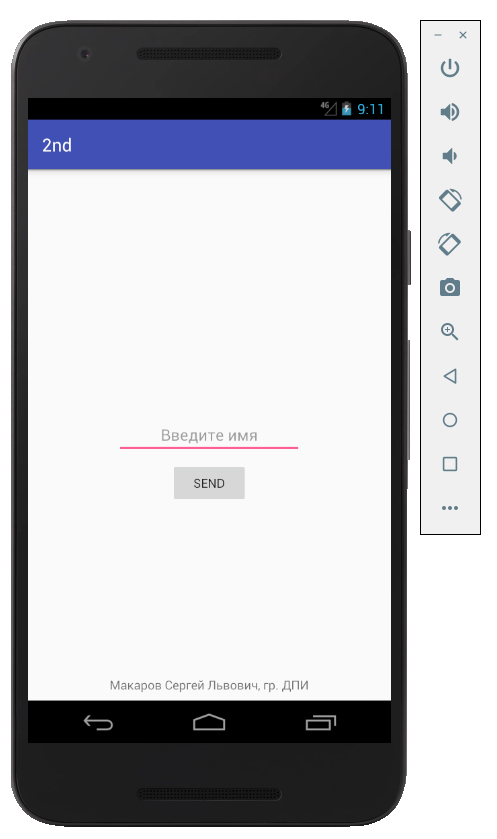
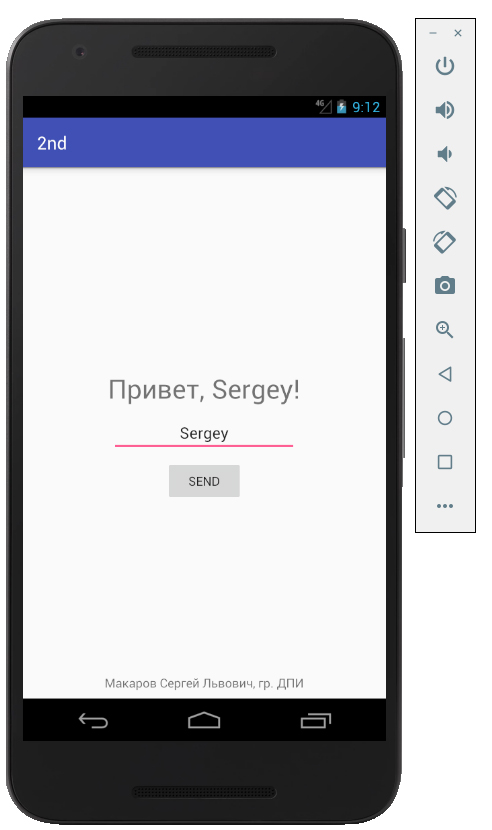
 

Рисунок 16 – Вид до ввода текста Рисунок 17 – Вид после ввода текста

### **Лабораторная работа №3. Работа с экранами**

Задание: создать приложение, которое состоит из нескольких activities. Первое 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 второе 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 первого 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.

Пока обновлялась эта методичка, вышла новая версия 3.2 Android Studio, поэтому первое, что нужно сделать, это обновить среду (Help – Check for Updates…), включая системы сборки Gradle, Android SDK Platrofm-Tools и т.д. После апдейта уже нет проблем с интерфейсом, описанных в начале лабораторной работы №1.

Создайте новый проект с Empty Activity и удалите стандартный TextView с фразой «Hello World!». Теперь в первом activity (файл main\_activity.xml) расставьте элементы TextView с текстом «Activity 1», EditText с фоновым текстом «Введите имя» и кнопку Button с текстом «Go». Не забудьте свои ФИО и группу. Пример расстановки элементов показан на рисунке 18.

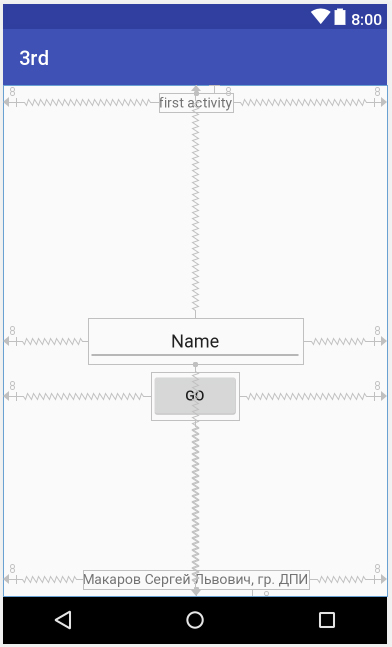


Рисунок 18 – Пример расстановки элементов в файле activity\_main.xml

Начнём с интерфейса. Добавим второе activity, для этого слева наверху нажмите по папке app правой кнопкой мыши и выберите пункт New -> Activity -> Empty Activity, как показано на рисунке 19.

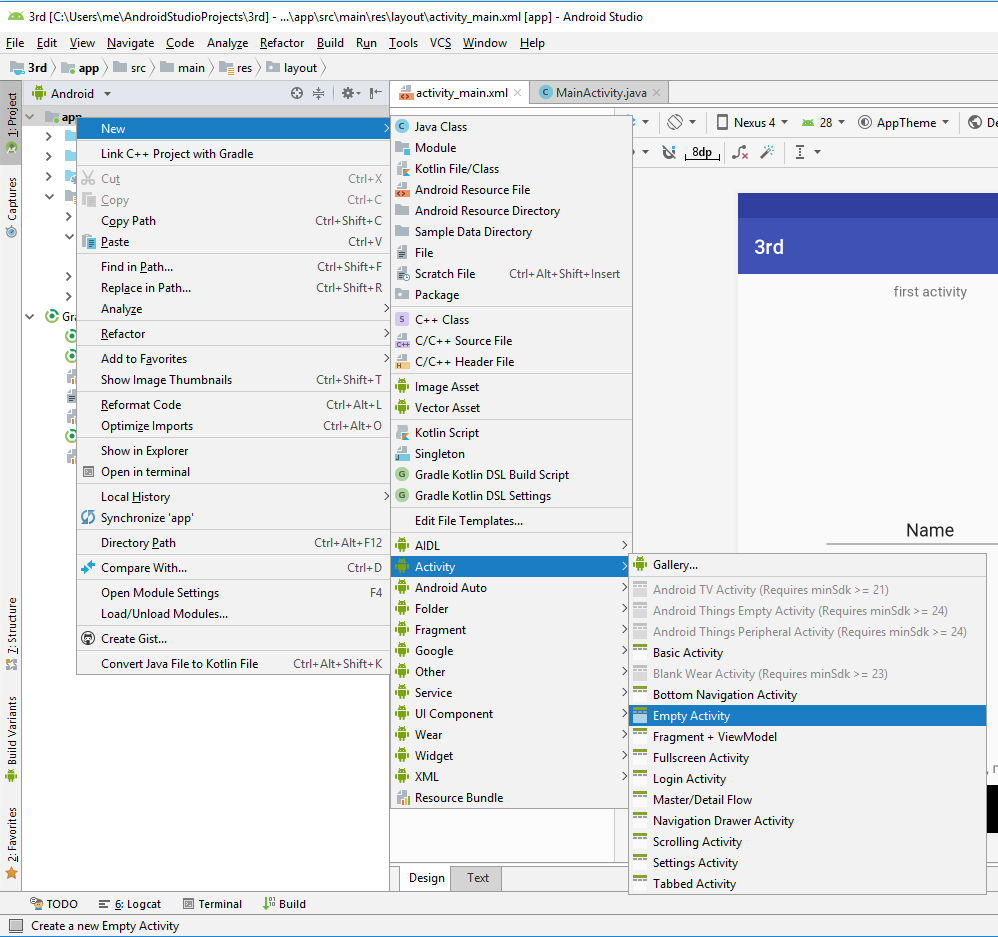


Рисунок 19 – Добавление нового Activity

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 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 первое activity. Пример расстановки элементов показан на рисунке 20.

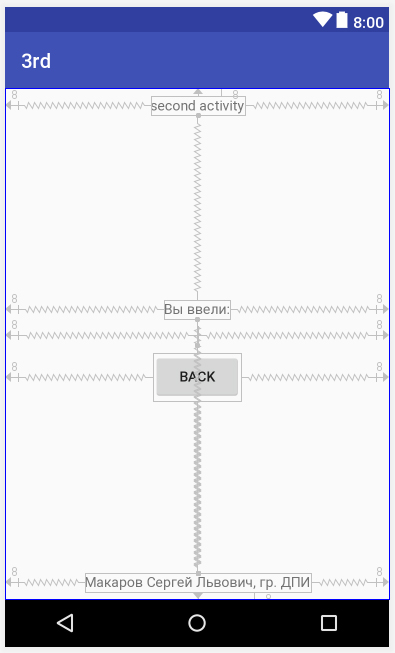


Рисунок 20 – Пример расстановки элементов в файле activity\_second.xml

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

**public static** String *text2remember*;

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

**final** EditText editText = findViewById(R.id.***editText***);  
Button button = findViewById(R.id.***button***);

Заметьте, что если раньше каст (cast, или (EditText), (Button) и т.д.) перед методом findViewById нужно было указывать обязательно, как мы сделали в лабораторной работе №2, в новой версии Android Studio это необязательно. Теперь присвойте текст полю editText текст из переменной text2remember:

editText.setText(*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 второе activity. Теперь создайте для кнопки listener и метод onClick (по подсказкам, показано во второй лабораторной работе) и внутри метода onClick напишите следующее:

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

Здесь переменной text2remember присваивается текст из поля EditText и определяется понятие Intent для запуска второго activity.

Теперь перейдите в файл SecondActivity.java и в методе onCreate определите элементы и присвойте полю TextView текст из переменной text2remember:

TextView textView = findViewById(R.id.***textView6***);  
textView.setText(MainActivity.*text2remember*);  
Button button = findViewById(R.id.***button2***);

Осталось только создать для кнопки listener и метод onClick для возвращения на первое activity. Создайте их и внутри метода onClick напишите:

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

Не забывайте про импорт зависимостей, среда разработки вам об этом напомнит и поможет это сделать.

Заметьте, при создании второго activity в файл манифеста AndroidManifest.xml автоматически добавилась строка

<**activity android:name=".SecondActivity"**></**activity**>

Каждое activity приложения должно быть упомянуто в файле AndroidManifest.xml.

Теперь запустите приложение и протестируйте, всё должно работать. Пример работы приложения показан на рисунках 21 и 22.

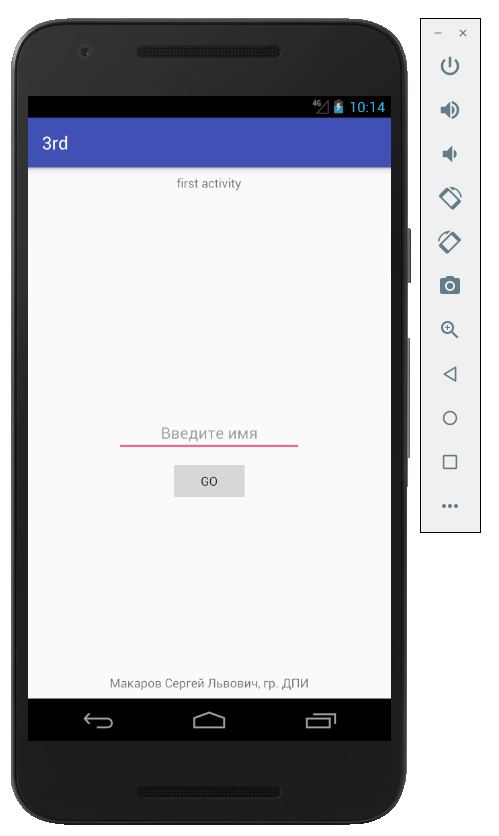
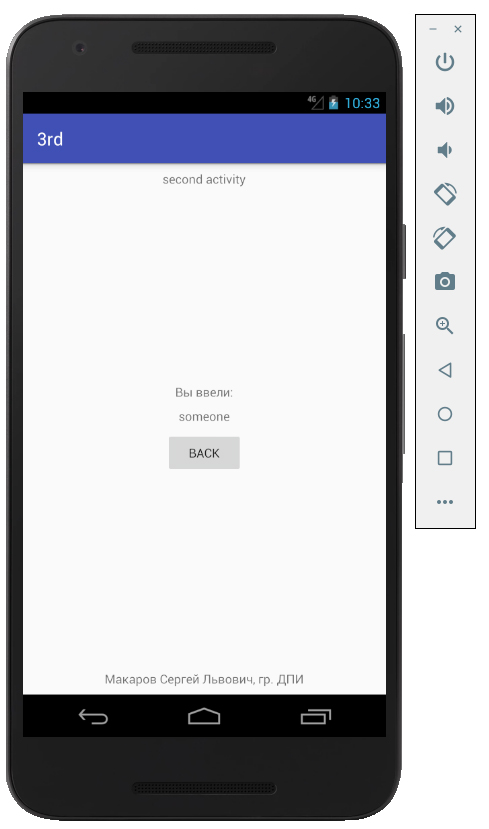
 

Рисунок 21 – Вид первого activity Рисунок 22 – Вид второго activity

### **Лабораторная работа №4. Стили и темы**

Задание: написать приложение, работающее с разными темами/стилями. Сначала создать свой стиль и применить его к какому-нибудь интерфейсному элементу, затем - свою тему, которая применяется ко всем интерфейсным элементам. Приложение при этом должно выглядеть нестандартно. Запустить на эмуляторе и убедиться, что всё работает.

Создайте новый проект с Empty Activity, перейдите в activity\_main.java, удалите стандартное текстовое поле с фразой «Hello world!» и поместите друг под друга 4 TextView, а затем поменяйте у них размер шрифта ( свойство textSize) на 10, 14, 28 и 36 sp соответственно (не забудьте про ФИО и группу). Как мы видим, все они отличаются друг от друга. Бывают ситуации, когда всем элементам необходимо иметь один стиль (один размер шрифта, один цвет текста и т.д.). В данном случае можно поменять параметры вручную, а что делать, есть у вас 100 таких элементов? В таком случае можно создать стиль и применить его ко всем нужным элементам. Переходим в файл values/styles.xml и ниже существующего стиля под названием AppTheme создаём следующий стиль:

<**style name="style1" parent="@android:style/TextAppearance"**>  
 <**item name="android:textColor"**>#000000</**item**>  
 <**item name="android:textSize"**>30sp</**item**>  
 <**item name="android:typeface"**>monospace</**item**>  
</**style**>

TextAppearance – это стиль для текста в Android по умолчанию. Далее мы устанавливаем цвет текста чёрным, делаем текст 30 размера и тип указываем monospace. Теперь нужно применить этот стиль к созданным четырём элементам. Выделите все четыре TextView и в панели Attributes (cправа) найдите свойство style, где пропишите стиль:

@style/style1

Также стиль можно выбрать из списка, нажав на 3 точки справа от текстового поля (в появившемся окне он будет последним в списке). Примените стиль. Вы увидите, что текстовые поля, кроме одного (у которого и так было 14sp по умолчанию, поэтому вы не стали ничего вводить в полу textSize), не поменялись. Это происходит потому, что поле textSize непустое и поэтому имеет приоритет над стилем. Если очистить все значения textSize у всех полей, то и все выделенные текстовые поля приведутся к единому виду.

Создадим подстиль, который будет соответствовать предыдущему стилю во всём, кроме размера шрифта. В файле styles.xml ниже стиля style1 напишите:

<**style name="style1.bigfont"**>  
 <**item name="android:textSize"**>50sp</**item**>  
</**style**>

Применяем данный подстиль к любому текстовому полю и видим, что размер шрифта увеличился.

Со стилями разобрались, теперь переходим к созданию собственной темы для приложения. В файле styles.xml уже создана базовая тема под названием AppTheme:

<**style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar"**>  
 *<!-- Customize your theme here. -->* <**item name="colorPrimary"**>@color/colorPrimary</**item**>  
 <**item name="colorPrimaryDark"**>@color/colorPrimaryDark</**item**>  
 <**item name="colorAccent"**>@color/colorAccent</**item**>  
</**style**>

Мы можем кастомизировать данную тему. Изменим родительскую тему приложения на следующую:

<**style name="AppTheme" parent="Base.Theme.AppCompat.Light.Dialog.Alert"**>  
 *<!-- Customize your theme here. -->* <**item name="colorPrimary"**>@color/colorPrimary</**item**>  
 <**item name="colorPrimaryDark"**>@color/colorPrimaryDark</**item**>  
 <**item name="colorAccent"**>@color/colorAccent</**item**>  
</**style**>

Теперь приложение будет иметь вид диалогового окна. Добавим какие-нибудь параметры, чтобы тема отличалась от стандартной:

<**item name="android:background"**>#ff0e6647</**item**>  
<**item name="android:textColor"**>#ffffff</**item**>  
<**item name="android:textSize"**>32sp</**item**>  
<**item name="android:textAllCaps"**>true</**item**>  
<**item name="android:typeface"**>monospace</**item**>

Свойства довольно понятны из названий, поэтому расписывать их не имеет смысла. Удалите из текстовых полей предыдущие стили и увидите, что текстовые поля сразу подстроились под параметры из темы. Дело в том, что тема AppTheme уже прописана как базовая в приложении, поэтому любые изменения сразу же отображаются на всех элементах. В поле с ФИО можно поставить свойство gravity в значение center\_horizontal.

Ну и в заключение переходим в файл MainActivity.java и в объявлении класса оставим следующее:

**public class** MainActivity **extends** Activity {

Делается это для того, чтобы в приложении не выводилась так называемая «шапка» с названием приложения.

Запустите проект на эмуляторе, он должен иметь вид, как показано на рисунке 23.

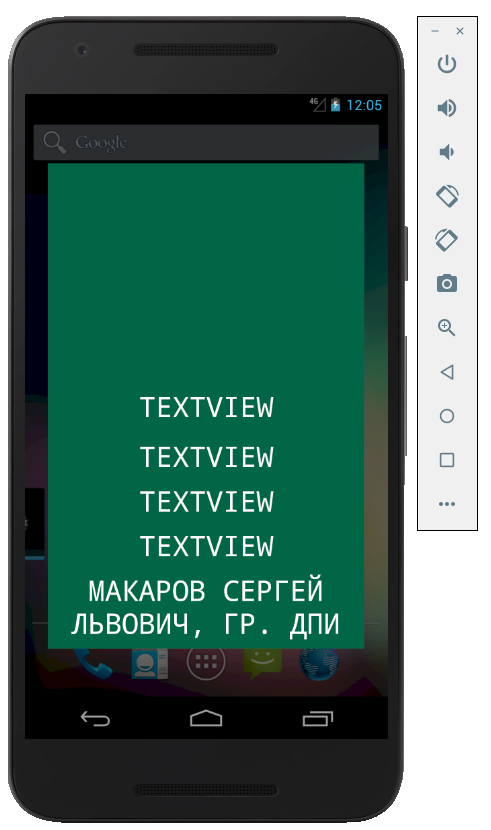


Рисунок 23 – Результат выполнения лабораторной работы №4

Приложение запустилось без «шапки» и в виде уведомления. Теперь придумайте собственную тему, чтобы всем не сдавать одно и то же приложение.

### **Лабораторная работа №5. Работа со списками**

Задание: создать пользовательский список. Например, создать свой список в виде твиттера (картинка и текст), элементы которого просто статически задать в массиве (как и картинки).

Создайте новый проект с Empty Activity, перейдите в файл activity\_main.java и удалите стандартное текстовое поле с фразой «Hello World!». Для начала необходимо создать xml-файл, который будет содержать список. Нажмите правой кнопкой мыши по папке values и создайте файл так, как показано на рисунке 24.

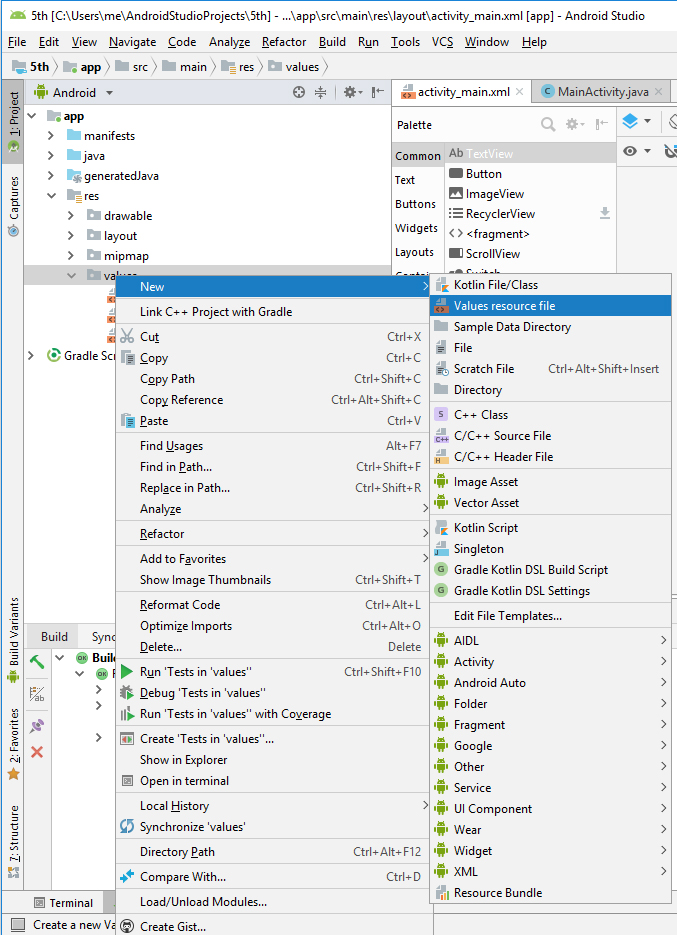


Рисунок 24 – Создание нового файла ресурсов

В появившемся окне оставьте все параметры по умолчанию и введите название файла, например, «images», после чего нажмите OK. После проделанных действий в папке values появился файл images.xml, в котором нужно будет создать список. Переходим в этот файл и внутри тэга resources пишем:

<**string-array name="images"**>  
 <**item**>Image 1</**item**>  
 <**item**>Image 2</**item**>  
 <**item**>Image 3</**item**>  
 <**item**>Image 4</**item**>  
</**string-array**>

В item можно использовать любые слова и фразы на любом языке. Теперь перейдите в файл MainActivity.java и исправьте объявление класса:

**public class** MainActivity **extends** ListActivity {

Не забудьте импортировать нужные классы, среда разработки подскажет об этом. Затем добавьте в метод onCreate следующий код:

setListAdapter(**new** ArrayAdapter<String>(**this**, android.R.layout.***simple\_list\_item\_1***, getResources().getStringArray(R.array.***images***)));

Теперь перейдите в файл activity\_main.xml и из раздела Legacy панели Palette перенесите объект ListView на экран. Обратите внимание, что ListView необходимо присвоить id «@android:id/list». Сделать это можно в текстовом представлении, добавив в тэг ListView следующую строчку:

**android:id="@android:id/list"**

Запустите приложение и увидите список, который был написан в файле images.xml. Однако, чтобы список был с картинками, нужно воспользоваться Custom Lists. Для этого в папке layout необходимо добавить ещё один файл. Нажмите правой кнопкой мыши по папке layout и создайте новый Layout resource file. В появившемся окне оставьте все параметры по умолчанию и введите название файла «list\_item». Переходим в этот файл и помещаем элементы ImageView и TextView на своё усмотрение. Пример расположения элементов в данном файле показан на рисунке 25. Так будет выглядеть один элемент нашего списка.

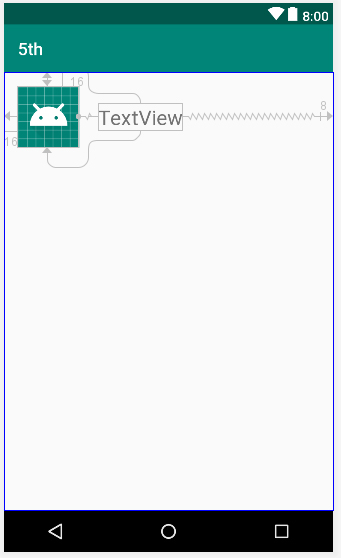


Рисунок 25 – Пример расположения элементов в файле list\_item.xml

Теперь вернитесь в файл MainActivity.java и измените код в методе onCreate на следующий:

setListAdapter(**new** MyAdapter(**this**, android.R.layout.***simple\_list\_item\_1***, R.id.***textView***, getResources().getStringArray(R.array.***images***)));

Далее создайте класс MyAdapter внутри класса MainActivity:

**public class** MyAdapter **extends** ArrayAdapter<String> {  
 **public** MyAdapter(Context context, **int** resource, **int** textViewResourceId, String[] string){  
 **super**(context, resource, textViewResourceId, string);  
 }  
  
 @Override  
 **public** View getView(**int** position, View convertView, ViewGroup parent){  
 LayoutInflater inflater = (LayoutInflater) getSystemService(Context.***LAYOUT\_INFLATER\_SERVICE***);  
 View row = inflater.inflate(R.layout.***list\_item***, parent, **false**);  
 String[] items = getResources().getStringArray(R.array.***images***);  
 ImageView image = (ImageView) row.findViewById(R.id.***imageView***);  
 TextView text = (TextView) row.findViewById(R.id.***textView***);  
 text.setText(items[position]);  
 **return** row;  
 }  
}

Но это ешё не всё. Необходимо создать конструкцию switch-case, чтобы в каждой строке была соответствующая ей картинка. Перед return row вставьте следующий код:

**switch** (items[position]) {  
 **case "Image 1"**:  
 image.setImageResource(R.drawable.image1);  
 **break**;  
 **case "Image 2"**:  
 image.setImageResource(R.drawable.image2);  
 **break**;  
 **case "Image 3"**:  
 image.setImageResource(R.drawable.image3);  
 **break**;  
 **case "Image 4"**:  
 image.setImageResource(R.drawable.image4);  
 **break**;  
}

Теперь осталось поместить любые изображения в папку drawable. Изображения либо должны называться image1.jpeg, image2.jpeg и т.д., либо в конструкции switch-case необходимо поменять путь до картинок в строке R.drawable.image1 и т.д. Не забудьте про текстовое поле с ФИО. Запустите приложение и увидите собственный список с картинкой и текстом. Пример показан на рисунке 26.

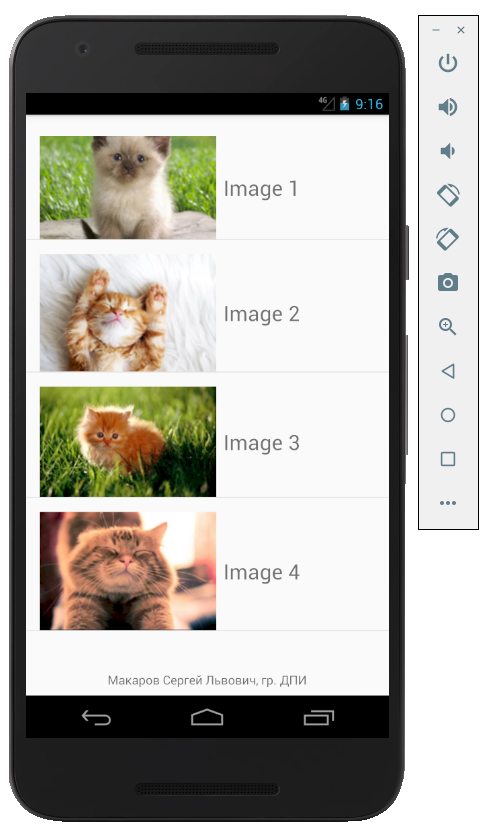


Рисунок 26 – Пример выполнения лабораторной работы №5

### **Лабораторная работа №6. Работа с анимацией**

Задание: создать приложение, содержащее анимированные интерфейсные элементы (например, увеличивающиеся при клике на них кнопки, вращающиеся TextView и т.д.).

Самым простым видом анимации является покадровая анимация, когда картинки (кадры) сменяют друг друга, создавая эффект движения или изменения изображения. Это слишком простой, но утомительный процесс, поэтому рассмотрим анимацию другого типа, так называемую tween-анимацию. Создайте новое приложение с Empty Activity, перейдите в файл activity\_main.xml, удалите текстовое поле и поместите в центр экрана кнопку, а внизу – как обычно, текстовое поле с ФИО. Затем необходимо создать новый xml-файл для описания анимации. Для этого нажмите правой кнопкой мыши по папке res и создайте новый файл Android resource с параметрами, которые изображены на рисунке 27.

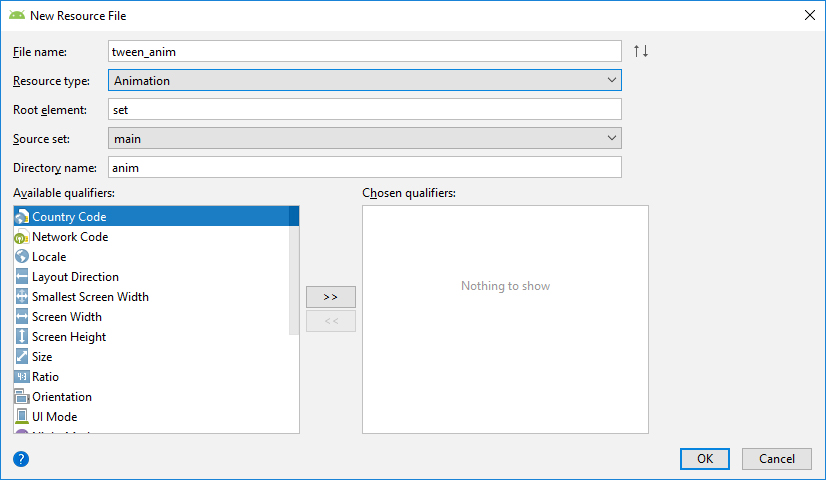


Рисунок 27 – Создание файла анимации

После данных действий в папке res появится новая папка anim с файлом tween\_anim.xml внутри. Переходим в этот файл и добавляем следующий код внутрь тэга set:

<**scale  
 android:fromXScale="2.0"  
 android:toXScale="0.5"  
 android:fromYScale="2.0"  
 android:toYScale="0.5"  
 android:pivotX="50%"  
 android:pivotY="50%"  
 android:duration="2000"** />

Здесь описывается уменьшение объекта с длительность анимации в 2 секунды (**android:duration**). Так как анимация будет отображаться на кнопке, нужно будет вернуть её в исходный размер, для этого добавляем ещё немного кода:

<**scale  
 android:fromXScale="0.5"  
 android:toXScale="2.0"  
 android:fromYScale="0.5"  
 android:toYScale="2.0"  
 android:pivotX="50%"  
 android:pivotY="50%"  
 android:duration="2000"  
 android:startOffset="2000"** />

Теперь кнопка сможет обратно увеличиться до своих размеров, анимация увеличения также будет длиться 2 секунды, но она будет отрабатывать не сразу, а с задержкой в 2 секунды (**android:startOffset**). То есть сначала идёт анимация уменьшения объекта, а затем анимация увеличения. Ну и в заключение добавим анимацию вращения кнопки:

<**rotate  
 android:pivotX="50%"  
 android:pivotY="50%"  
 android:fromDegrees="0"  
 android:toDegrees="360"  
 android:duration="2000"  
 android:startOffset="4000"** />

Анимация вращения будет ждать 4 секунды (пока отработают первые две анимации), а затем за 2 секунды провернёт кнопку на 360 градусов по часовой стрелке.

Ещё одним важным свойством анимации является параметр interpolator. Интерполятор позволяет ускорить или замедлить выполнение анимации на устройстве, а также добавить некоторые эффекты. Добавим следующий код в конец открывающего тэга set:

**android:interpolator="@android:anim/bounce\_interpolator"**

Набрав **@android:anim** в кавычках, можно увидеть, какие типы интерполяторов бывают, см. рисунок 28. Можно поэкспериментировать с разными типами интерполяторов.

Рисунок 28 – Варианты интерполяторов

Итак, в нашем случае открывающий тэг set должен выглядеть следующим образом:

<**set xmlns:android="http://schemas.android.com/apk/res/android" android:interpolator="@android:anim/bounce\_interpolator"**>

С описанием анимации закончили, теперь необходимо перейти в файл MainActivity.java и выполнить анимацию после нажатия на кнопку. Для этого в методе onCreate создаём ссылку на кнопку:

**final** Button button = findViewById(R.id.***button***);

И создаём для неё listener и onClick. В onClick добавляем анимацию:

Animation animation = AnimationUtils.*loadAnimation*(MainActivity.**this**, R.anim.***tween\_anim***);  
button.startAnimation(animation);

Импортируем все зависимости и исправляем ошибки, после чего запускаем проект на эмуляторе. Теперь необходимо изменить параметры в файле tween\_anim.xml на собственные, чтобы все не сдавали одну и ту же анимацию. Поэкспериментируйте с различными параметрами, можете также производить анимацию не с кнопкой, а с каким-нибудь другим элементом. Результат выполнения работы показан на рисунке 29.

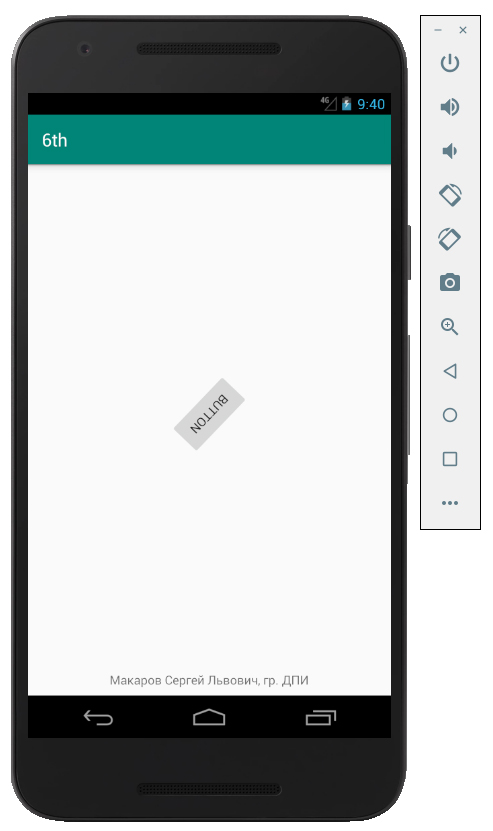


Рисунок 29 – Результат анимации

### **Лабораторная работа №7. Работа с картами**

Задание: создать приложение, отображающее после запуска карты Google или какие-нибудь другие карты.

Создайте новый проект, но на этот раз выберите Google Maps Activity вместо Empty Activity. Для того, чтобы пользоваться картами Google, необходимо получить персональный ключ для приложения. Переходим в файл res/values/google\_maps\_api.xml (открывается по умолчанию), и в комментариях видим инструкцию на английском языке о том, как получить ключ для приложения. Если коротко, то необходимо перейти по персональной ссылке из комментария, залогиниться в свой Google аккаунт, нажать кнопку Continue и затем Create API key, после чего перед вами появится окно API key created с ключом, начинающимся на "AIza". Копируем значение ключа из поля и вставляем в файл res/values/google\_maps\_api.xml вместо фразы YOUR\_KEY\_HERE.

Теперь переходим в файл MapsActivity.java, и в самом последнем методе onMapReady добавляем следующее в начало, после строки **mMap** = googleMap;:

LocationManager locationManager = (LocationManager) **this**.getSystemService(Context.***LOCATION\_SERVICE***);

Затем строчкой ниже начинаем писать LocationListener locationListener = **new** Loc и дальше выбираем первую подсказку. Появится 4 метода, в первом методе onLocationChanged пишем следующее:

LatLng latlng = **new** LatLng(location.getLatitude(), location.getLongitude());  
**mMap**.clear();  
**mMap**.moveCamera(CameraUpdateFactory.*newLatLng*(latlng));  
**mMap**.animateCamera(CameraUpdateFactory.*zoomTo*(15));  
**mMap**.addMarker(**new** MarkerOptions().position(latlng).title(**"I am here"**));

После этих четырёх методов (после символа }; - не забудьте поставить точку с запятой) пишем следующее:

locationManager.requestLocationUpdates(LocationManager.***NETWORK\_PROVIDER***, 0, 0, locationListener);  
locationManager.requestLocationUpdates(LocationManager.***GPS\_PROVIDER***, 0, 0, locationListener);

Здесь мы запрашиваем обновление местоположения по вышкам сотовой связи и по GPS. Код просит добавить проверку разрешений для пользователя, добавляем эту проверку (выделяем строку с ошибкой, слева видим красную лампочку с восклицательным знаком внутри, нажимаем на неё и выбираем Add permission check) и одновременно в файл манифеста добавляем соответствующие разрешения. Разрешение **ACCESS\_FINE\_LOCATION** уже есть, добавим недостающее:

<**uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"** />

Далее вернёмся в MapsActivity.java и в методе onMapReady закомментируем всё, что идёт в конце и относится к Сиднею (после *// Add a marker in Sydney and move the camera*), и после этого вставим две строчки:

**mMap**.setMyLocationEnabled(**true**);  
**mMap**.getUiSettings().setZoomControlsEnabled(**true**);

**mMap**.setMapType(2);

Таким образом, на карте отобразится кнопка текущего местоположения и кнопки зума.

Прежде чем запускать приложение, необходимо создать новый эмулятор с поддержкой Google Play services (Google APIs), если у вас ещё нет такого эмулятора, так как на обычном эмуляторе карту вы не увидите, а увидите вместо неё надпись "Your\_app\_name won't run without Google Play services, which are not supported by your device.". Затем, возможно, после запуска приложения вы можете получить сообщение о том, что "Your\_app\_name won't run unless you update Google Play services", так как для работы приложения нужна версия 2 этих сервисов – нажмите кнопку Update в этом случае (кнопка будет работать только в том случае, если ваш эмулятор поддреживает функциональность Google Play). Лучше всего сконфигурировать эмулятор с одной из последних версий API (начиная с Lollipop) – тогда не надо будет обновлять Google Play services до последней версии. Если после запуска приложения появляется предложение установить Instant Run, надо согласиться.

Также помните о добавлении вашего ФИО в приложение – в этот раз это будет сделать сложнее, чем обычно, так как надо самостоятельно создать layout в файле res -> layout -> activity\_maps.xml, обернуть в него fragment с картой и добавить элемент TextView.

Запускаем приложение, однако эмулятор не имеет данных о местоположении, но их можно ему передать! Для этого в панели инструментов справа от эмулятора нажмите на три точки внизу, после этого откроется окно, показанное на рисунке 30. Справа в окне видно, что можно ввести значения широты и долготы и передать их приложению с помощью кнопки SEND. Введите 55.769794 в поле Longitude и 37.655640 в поле Latitude (координаты Москвы) и нажмите кнопку SEND.