МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ГАГАРИНА Ю.А.»
(СГТУ имени Гагарина Ю.А.)
Саратовский колледж машиностроения и энергетики

«СОГЛАСОВАНО»	«УТВЕРЖДАЮ»
Решением П(Ц)МК	Заместитель директора по УР
Протокол №	_____ С.В. Клюквина
от « » _июня_20__г.	«___» _____ 20__ г.

Методические рекомендации для студентов по проведению лабораторных работ по дисциплине «МДК. 01.03. Разработка мобильных приложений»

для специальности 09.02.07 «Информационные системы и программирование»

Саратов 2018

Разработал преподаватель СКМ и Э ФГБОУ ВО «СГТУ имени Гагарина Ю.А» Нечаева Н.М _____

Данные методические рекомендации предназначены для студентов ___ курса, специальности 09.02.07 «ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ»

СОДЕРЖАНИЕ

Введение

Лабораторные работы, как виды учебных занятий, направлены на экспериментальное подтверждение теоретических положений и формирование учебных и профессиональных практических умений и составляют важную часть теоретической и профессиональной практической подготовки. Семинар является видом лабораторных работ.

В процессе лабораторной работы обучающиеся выполняют одну или несколько лабораторных работ под руководством преподавателя в соответствии с изучаемым содержанием учебного материала.

Выполнение обучающимися лабораторных работ проводится с целью:

- формирования практических умений в соответствии с требованиями к уровню подготовки обучающихся, установленными рабочей программой дисциплины по конкретным разделам/ темам дисциплины;
- обобщения, систематизации, углубления, закрепления полученных теоретических знаний;
- совершенствования умений применять полученные знания на практике, реализации единства интеллектуальной и практической деятельности;
- развития интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;
- выработки таких профессионально значимых качеств, как самостоятельность, ответственность, точность, творческая инициатива при решении поставленных задач при освоении общих компетенций.

Цель лабораторных работ

В результате освоения учебной дисциплины «МДК 1.3. Разработка мобильных приложений» обучающийся должен **уметь:**

- осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля;
- осуществлять разработку кода программного модуля на современных языках программирования;
- уметь выполнять оптимизацию и рефакторинг программного кода;
- оформлять документацию на программные средства.

Иметь практический опыт в:

- разработке кода программного продукта на основе готовой спецификации на уровне модуля;
- использовании инструментальных средств на этапе отладки программного продукта;
- проведении тестирования программного модуля по определенному сценарию;
- использовании инструментальных средств на этапе отладки программного продукта;
- разработке мобильных приложений.

В результате освоения учебной дисциплины обучающийся должен **знать:**

- основные этапы разработки программного обеспечения;

- основные принципы технологии структурного и объектно-ориентированного программирования;
- способы оптимизации и приемы рефакторинга;
- основные принципы отладки и тестирования программных продуктов.

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей.

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языках.

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием.

ПК 1.6. Разрабатывать модули программного обеспечения для мобильных платформ.

Организация лабораторной работы

Лабораторная работа должна проводиться в учебных кабинетах или специально оборудованных помещениях (площадках, полигонах и т.п.)- кабинет Лаборатория «Системного и прикладного программирования», кабинет «Стандартизации и сертификации», полигон «Вычислительной техники и учебных бах практик».

В соответствии с требованиями ФГОС СПО 09.02.07 «Информационные системы и программирование» реализация ППССЗ должна обеспечивать выполнение обучающимися и лабораторных работ, включая как обязательный компонент *лабораторные работы с использованием:*

персональных компьютеров с конфигурацией: Core i5, дискретная видеокарта 8GB ОЗУ

лицензионное программное обеспечение: Android Studio; Visual Studio.

Выполнению лабораторных работ предшествует проверка знаний обучающихся - их теоретической готовности к выполнению задания.

Лабораторные работы могут носить репродуктивный, частично-поисковый и поисковый характер.

Работы, носящие *репродуктивный характер*, отличаются тем, что при их проведении обучающиеся пользуются подробными инструкциями, в которых указаны: цель работы, пояснения (теория, основные характеристики), оборудование, аппаратура, материалы и их характеристики, порядок выполнения работы, таблицы, выводы (без формулировки), контрольные вопросы, учебная и специальная литература.

Работы, носящие *частично-поисковый характер*, отличаются тем, что при их проведении обучающиеся не пользуются подробными инструкциями, им не дан порядок выполнения необходимых действий, и они требуют от обучающихся самостоятельного подбора оборудования, выбора способов выполнения работы в инструктивной и справочной литературе и др.

Работы, носящие *поисковый характер*, характеризуются тем, что обучающиеся, опираясь на имеющиеся у них теоретические знания, должны решить новую для них проблему.

При планировании лабораторных работ необходимо находить оптимальное соотношение репродуктивных, частично-поисковых и поисковых работ, чтобы обеспечить высокий уровень интеллектуальной деятельности.

Формы организации обучающихся при проведении лабораторных работах - фронтальная, групповая и индивидуальная.

При *фронтальной форме* организации занятий все обучающиеся выполняют одновременно одну и ту же работу.

При *групповой форме* организации занятий одна и та же работа выполняется бригадами по 2 - 5 человек.

При *индивидуальной форме* организации занятий каждый обучающийся выполняет индивидуальное задание.

Для повышения эффективности проведения лабораторных работ рекомендуется:

- разработка сборников задач, заданий и упражнений;
- разработка контрольно-диагностических материалов для контроля за подготовленностью обучающихся к лабораторным работам, в том числе в форме педагогических тестовых материалов для автоматизированного контроля;
- подчинение методики проведения лабораторных работ ведущим дидактическим целям с соответствующими установками обучающимся;
- применение коллективных и групповых форм работы, максимальное использование индивидуальных форм с целью повышения ответственности каждого обучающегося за самостоятельное выполнение полного объема работ;
- проведение лабораторных работ на повышенном уровне трудности с включением в них заданий, связанных с выбором обучающимися условий выполнения работы, конкретизацией целей, самостоятельным отбором необходимого оборудования;
- подбор дополнительных задач и заданий для обучающихся, работающих в более быстром темпе, для эффективного использования времени, отводимого на лабораторные работы.

Структура лабораторной работы

Лабораторная работа № 1
Тема: Установка и настройка среды разработки мобильных приложений Android Studio

Цели и задачи урока: закрепление теоретических знаний и приобретение практических навыков установки и настройки среды разработки мобильных приложений Android Studio

Тип урока – лабораторная работа.

Методы обучения – репродуктивный

Обеспечение занятия – браузер с выходом в интернет.

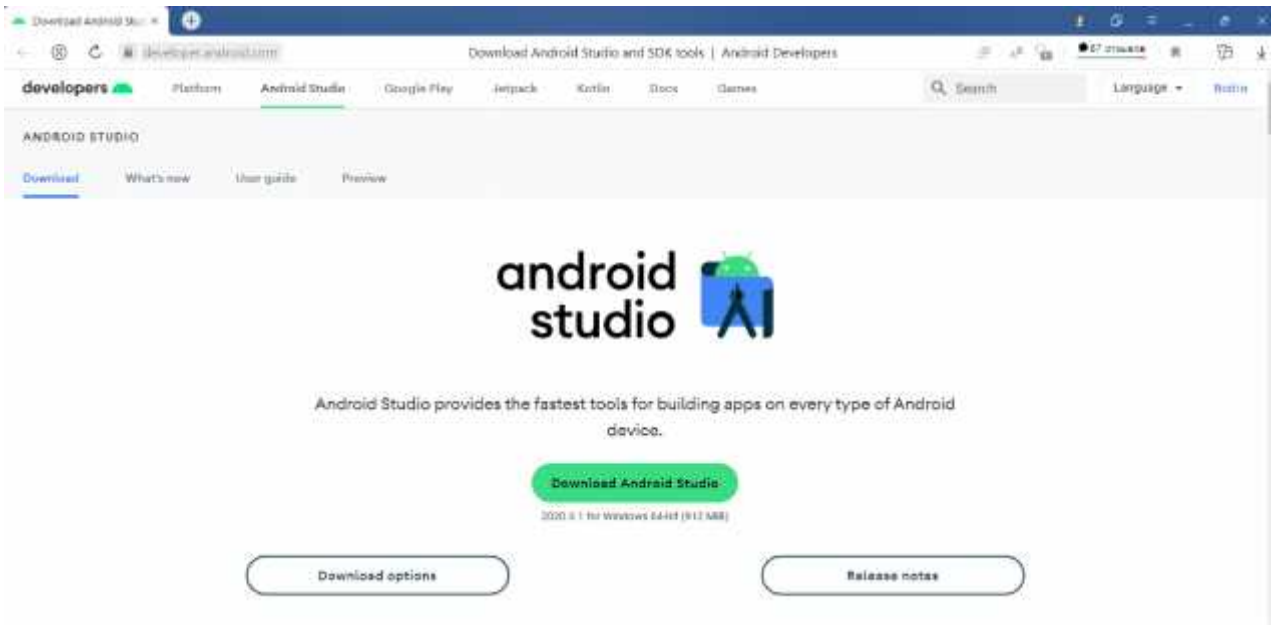
Литература

1. Разработка модулей программного обеспечения для компьютерных систем: учебник для студ. учреждений сред. проф. образования / Г. Н. Федорова. - 4-е изд., перераб. - М.: Издательский центр «Академия», 2020. - 384 с. ISBN 978-5-4468-9443-7
2. ГОСТ 19.201-78 Межгосударственный стандарт ЕСПД Техническое задание. Требования к содержанию и оформлению
3. https://nationalteam.worldskills.ru/skills/razrabotka-mobilnogo-prilozheniya-v-android-studio/

Задание

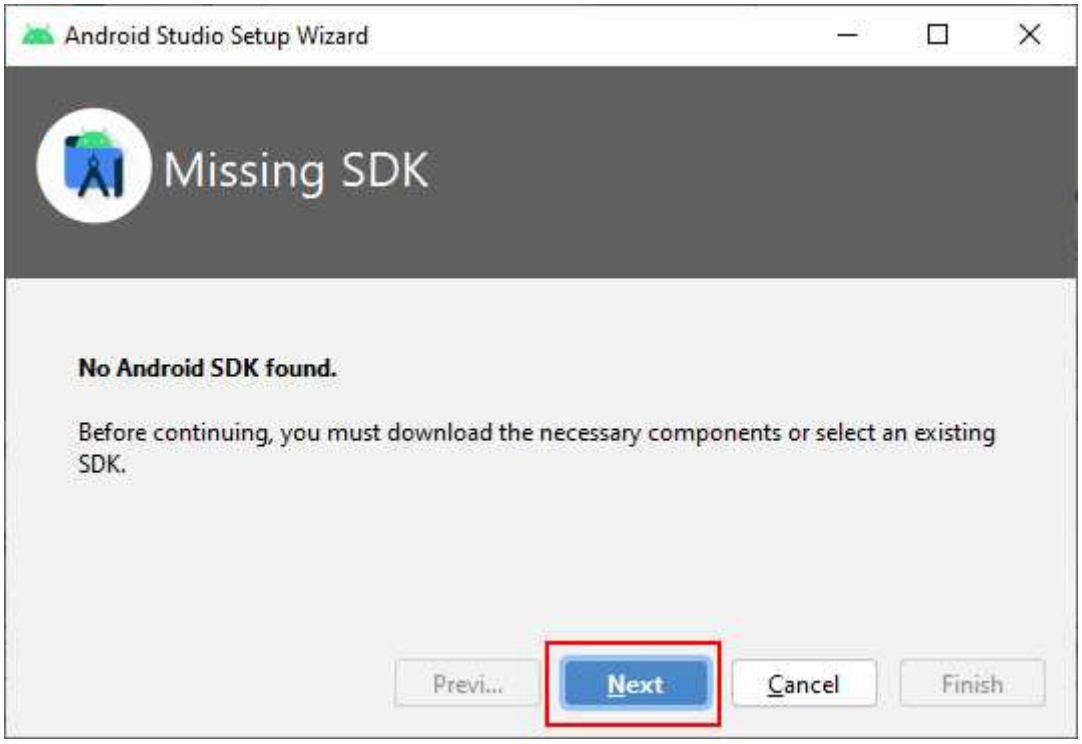
Установка средств разработки

Существуют разные среды разработки для Android. Рекомендуемой средой разработки является **Android Studio**, которая создана специально для разработки под ОС Android. Поэтому мы ее и будем использовать. ЗагLkзрузить файл установщика можно с официального сайта: https://developer.android.com/studio:

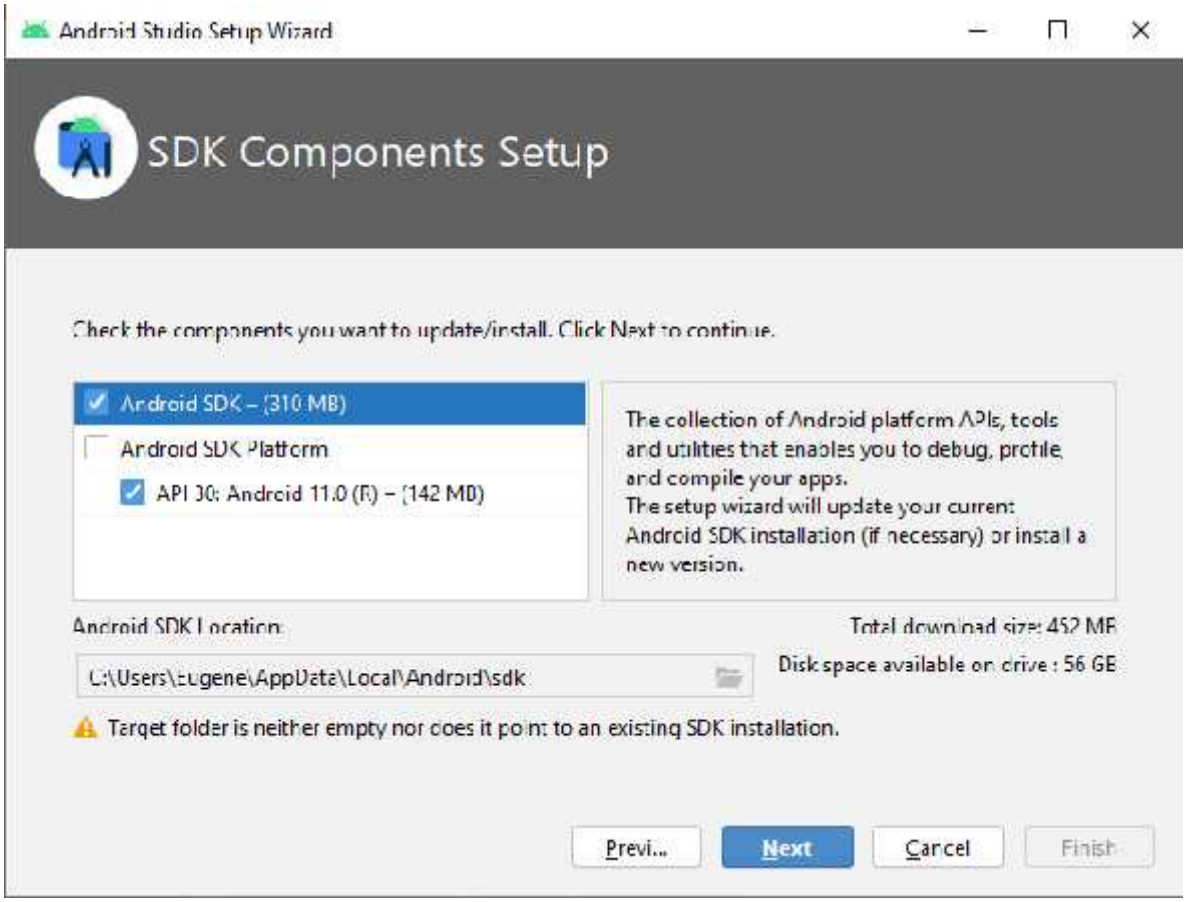


Кроме самой среды Android Studio для разработки также потребуется набор инструментов, который называется Android SDK. Например, если ранее Android SDK еще не было установлено, то при первом обращении к Android Studio она сообщит, что

Android SDK отсутствует.

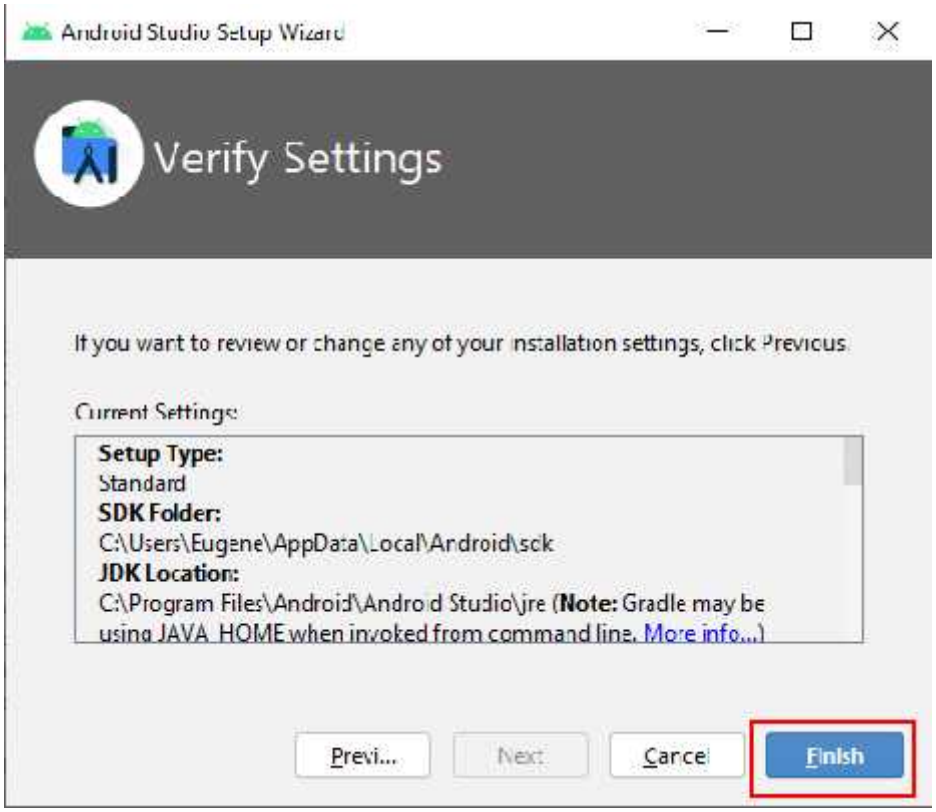


Можно отдельно вручную загрузить Android SDK с официального сайта и установить его. Либо можно сделать это непосредственно из Android Studio. Так, нажать на кнопку Next. И на следующем экране будет предложено загрузить Android SDK для последней версии API (в данном случае для Android 11):

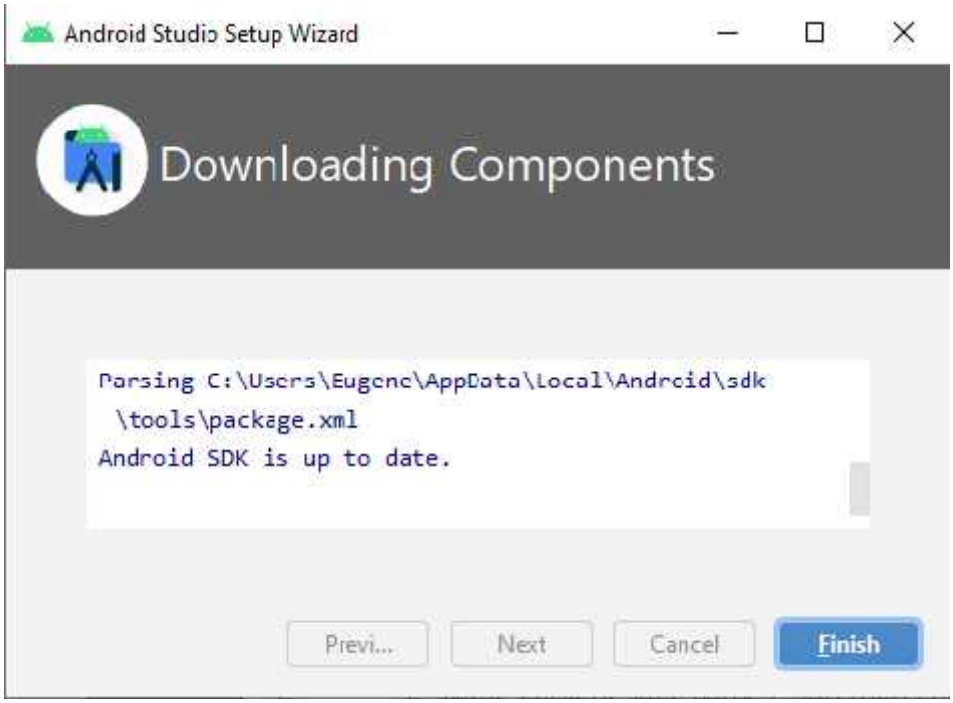


Здесь можно указать место для установки Android SDK, если путь по умолчанию не устраивает.

Нажать на кнопку Next, и далее отобразится окно со сводкой того, что именно будет установлено:

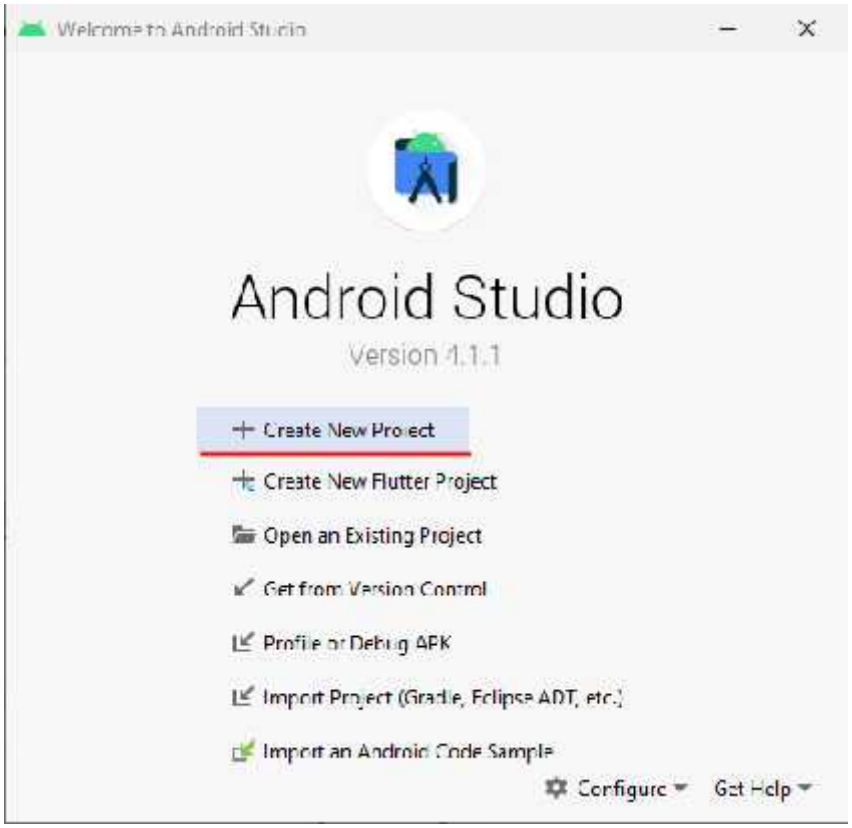


Нажать на кнопку Finish, чтобы, наконец, все это установить.

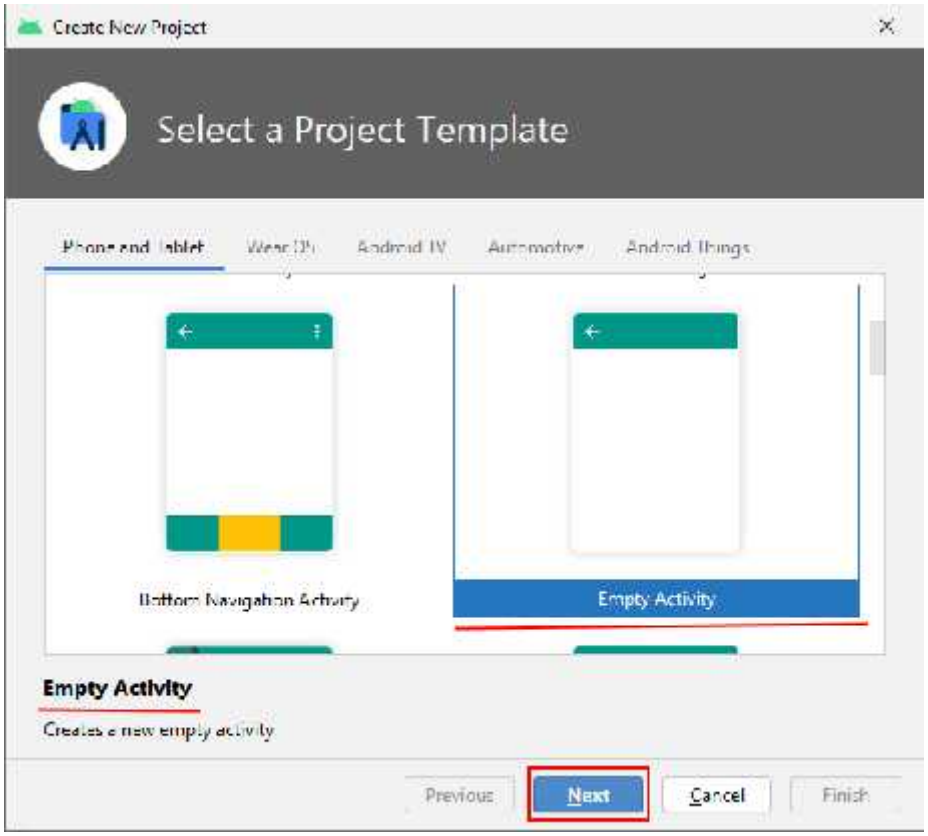


После завершения установки нажать на кнопку Finish. Можем приступать к созданию приложений.

Теперь создадим первое приложение в среде Android Studio для операционной системы Android. Откроем Android Studio и на начальном экране выберем пункт Create New Project:

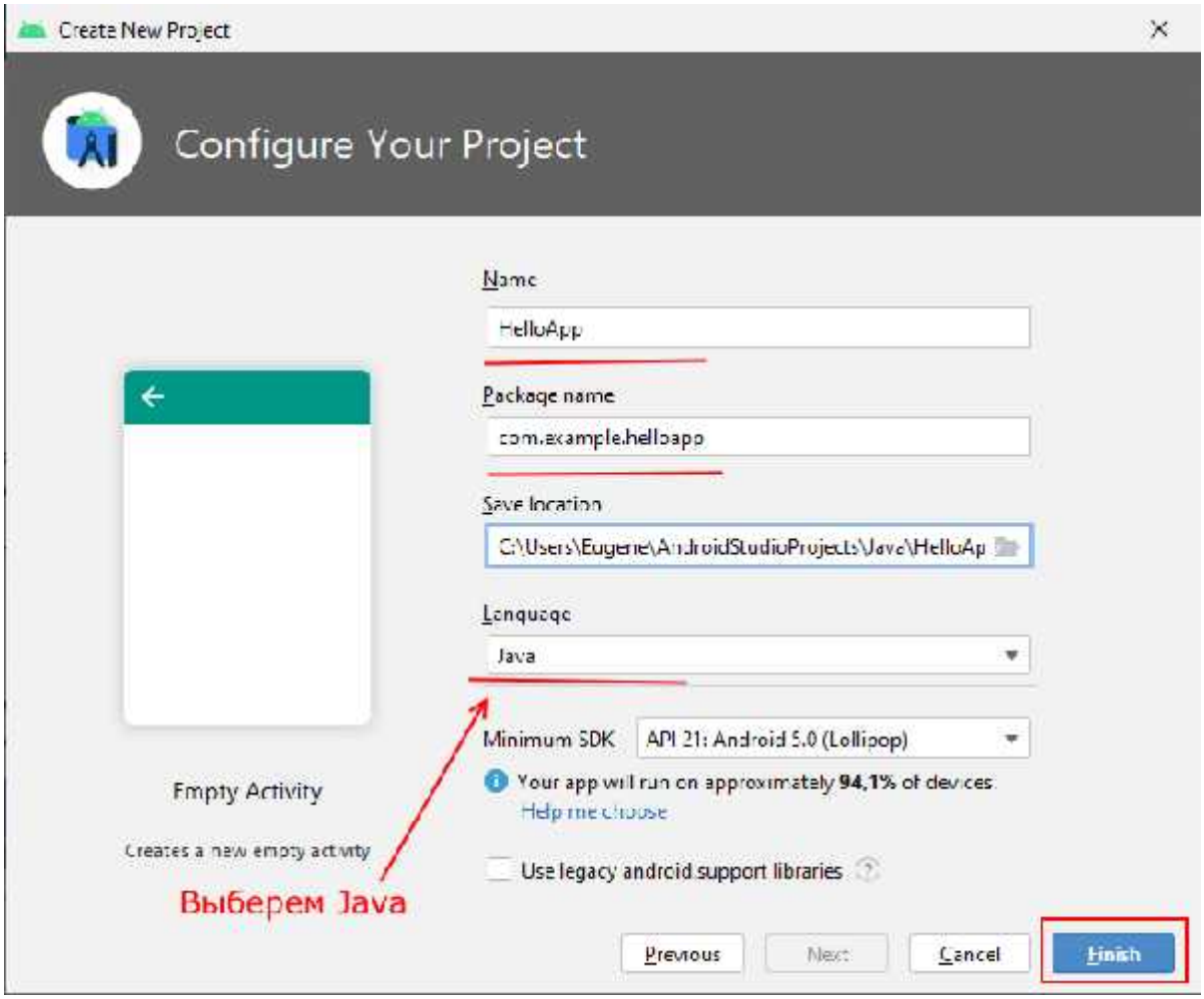


При создании проекта Android Studio вначале предложит нам выбрать шаблон проекта:



Android Studio предоставляет ряд шаблонов для различных ситуаций, но самыми распространенными являются Basic Activity и Empty Activity. Это самые удобные шаблоны для старта для создания большинства приложений. И по умолчанию выбран шаблон Empty Activity (если он не выбран, выберем его) и нажмем на кнопку Next.

После этого отобразится окно настроек нового проекта:



В окне создания нового проекта мы можем установить его начальные настройки:

В поле Name вводится название приложения. Укажем в качестве имени название HelloApp

В поле Package Name указывается имя пакета, где будет размещаться главный класс приложения. В данном случае для тестовых проектов это значение не играет ольшого значения, поэтому установим com.example.helloapp.

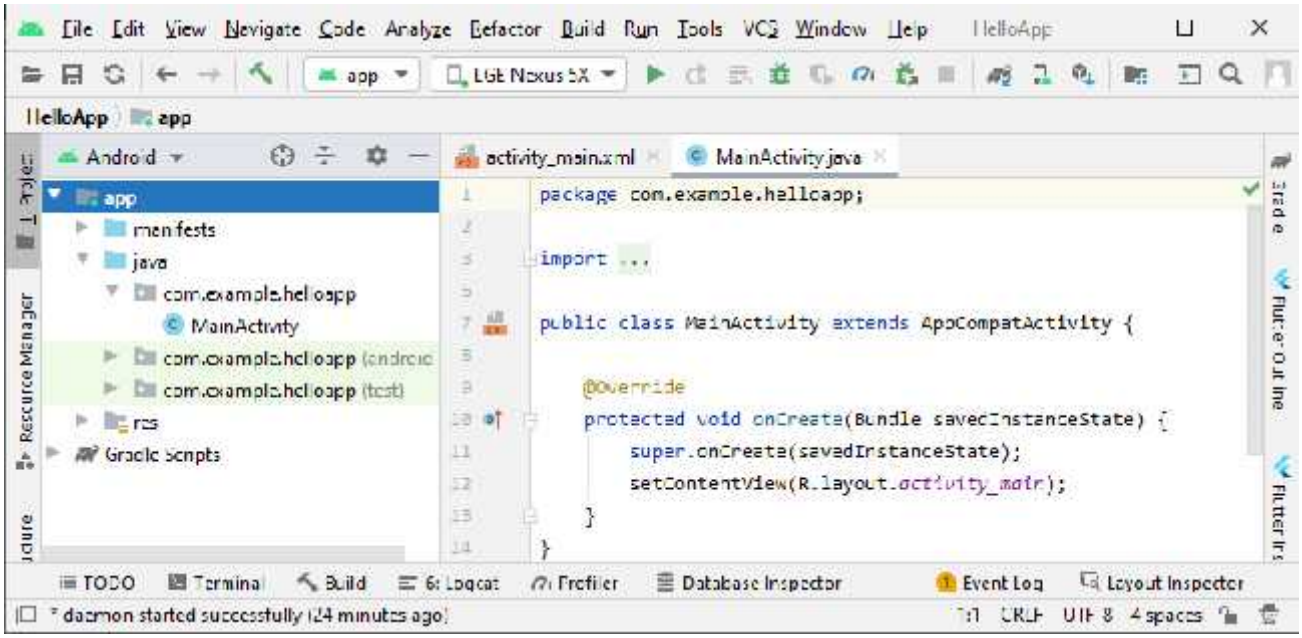
В поле Save Location установливается расположение файлов проекта на жестком диске. Можно оставить значение по умолчанию.

В поле Language в качестве языка программирования укажем Java (будьт внимательны, так как по умолчанию в этом поле стоит Kotlin)

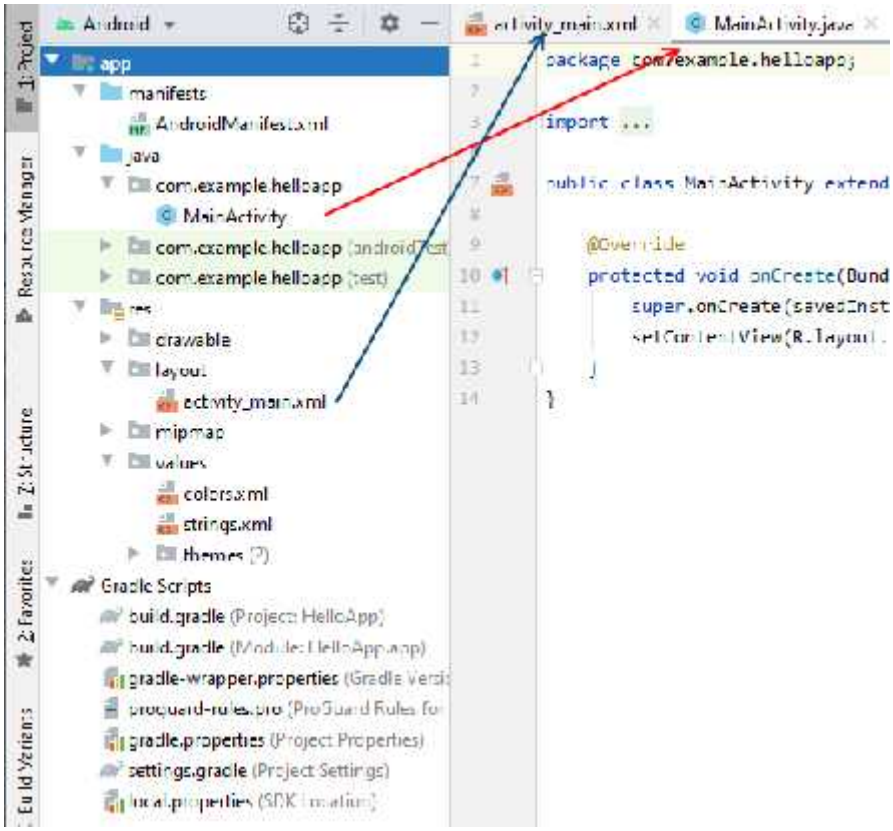
В поле Minimum SDK указывается самая минимальная поддерживаемая версия SDK. Оставим значение по умолчанию - API 21: Android 5.0 (Lollipop), которая означает, что наше приложение можно будет запустить начиная с Android 5.0, а это 94% устройств. На более старых устройствах запустить будет нельзя.

Стоит учитывать, что чем выше версия SDK, тем меньше диапазон поддерживаемых устройств.

Далее нажмем на кнопку Finish, и Android Studio создаст новый проект:



Вначале вкратце рассмотрим структуру проекта, что он уже имеет по умолчанию



Проект Android может состоять из различных модулей. По умолчанию, когда мы создаем проект, создается один модуль - **app**. Модуль имеет три подпапки:

- **manifests**: хранит файл манифеста **AndroidManifest.xml**, который описывает конфигурацию приложения и определяет каждый из компонентов данного приложения.
- **java**: хранит файлы кода на языке java, которые структурированы по отдельным пакетам. Так, в папке com.example.helloapp (название которого было указано на этапе создания проекта) имеется по умолчанию файл **MainActivity.java** с кодом на языке Java, который представляет класс MainActivity, запускаемый по умолчанию при старте приложения
- **res**: содержит используемые в приложении ресурсы. Все ресурсы разбиты на подпапки.
    - папка **drawable** предназначена для хранения изображений, используемых в приложении
    - папка **layout** предназначена для хранения файлов, определяющих графический интерфейс. По умолчанию здесь есть файл **activity_main.xml**, который определяет интерфейс для класса MainActivity в виде xml
    - папки **mipmap** содержат файлы изображений, которые предназначены для создания иконки приложения при различных разрешениях экрана.
    - папка **values** хранит различные xml-файлы, содержащие коллекции ресурсов - различных данных, которые применяются в приложении. По умолчанию здесь есть два файла и одна папка:
        - файл **colors.xml** хранит описание цветов, используемых в приложении
        - файл **strings.xml** содержит строковые ресурсы, используемые в приложении
        - папки **themes** хранит две темы приложения - для светлую (дневную) и темную (ночную)

Отдельный элемент **Gradle Scripts** содержит ряд скриптов, которые используются при построении приложения.

Во всей этой структуре следует выделить файл MainActivity.java, который открыт в Android Studio и который содержит логику приложения и, собственно, с него начинается выполнение приложения. И также выделим файл **activity_main.xml**, который определяет графический интерфейс - по сути то, что увидит пользователь на своем смартфоне после загрузки приложения.

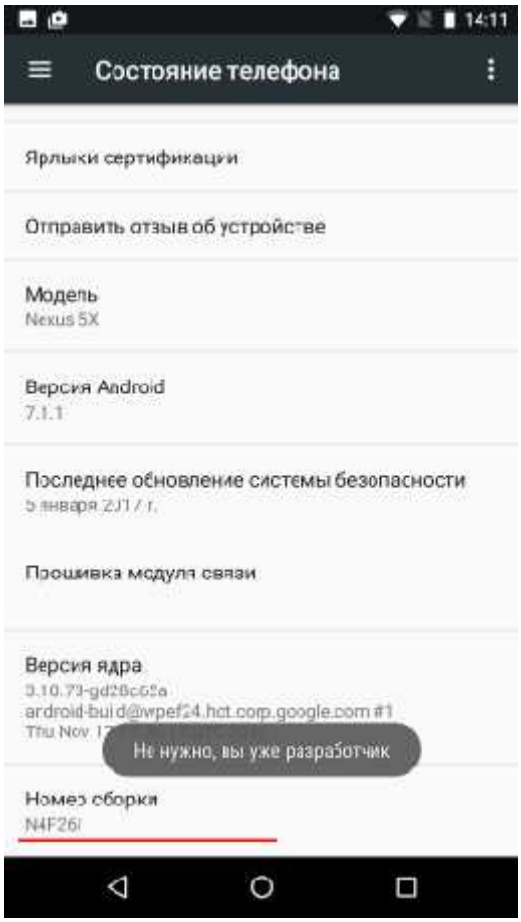
Запуск проекта

Созданный выше проект уже содержит некоторый примитивный функционал. Правда, этот функционал почти ничего не делает, только выводит на экран строку "Hello world!". Тем не менее это уже фактически приложение, которое мы можем запустить.

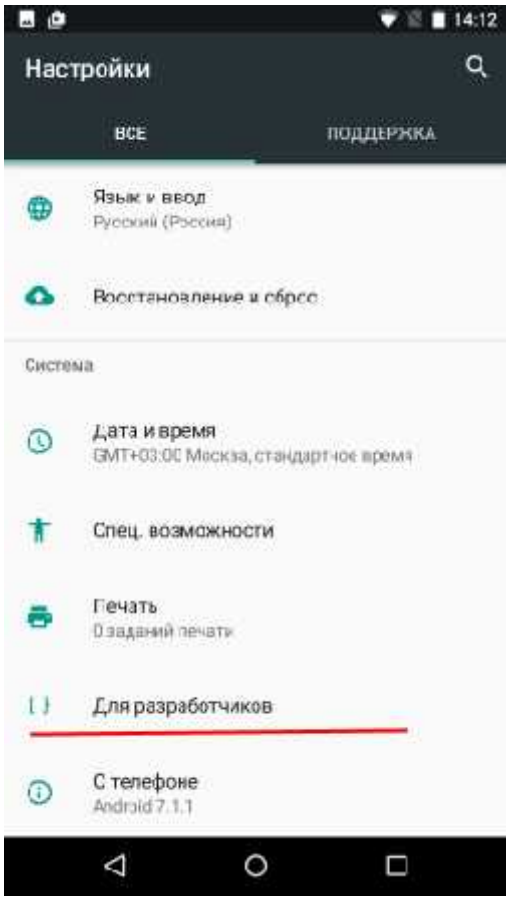
Для запуска и тестирования приложения мы можем использовать эмуляторы или реальные устройства. Но в идеале лучше тестировать на реальных устройствах. К тому же эмуляторы требуют больших аппаратных ресурсов, и не каждый компьютер может потянуть требования эмуляторов. А для использования мобильного устройства для тестирования может потребоваться разве что установить необходимый драйвер.

Режим разработчика на телефоне

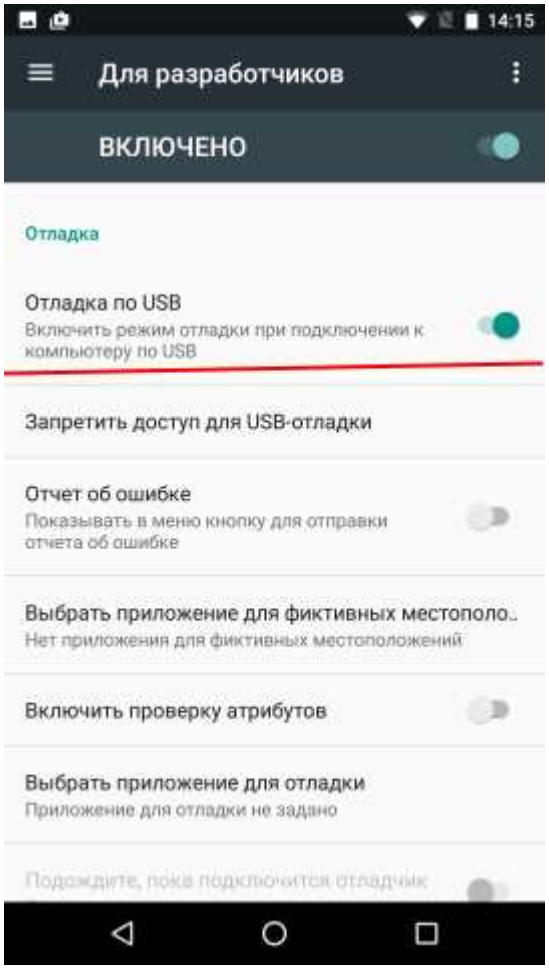
По умолчанию опции разработчика на смартфонах скрыты. Чтобы сделать их доступными, надо зайти в **Settings > About phone (Настройки > О телефоне)** (в Android 8 это в **Settings > System > About phone (Настройки > Система > О телефоне)**) и семь раз нажать **Build Number (Номер сборки)**.



Вернитесь к предыдущему экрану и там вы увидите доступный пункт **Developer options (Для разработчика)**.

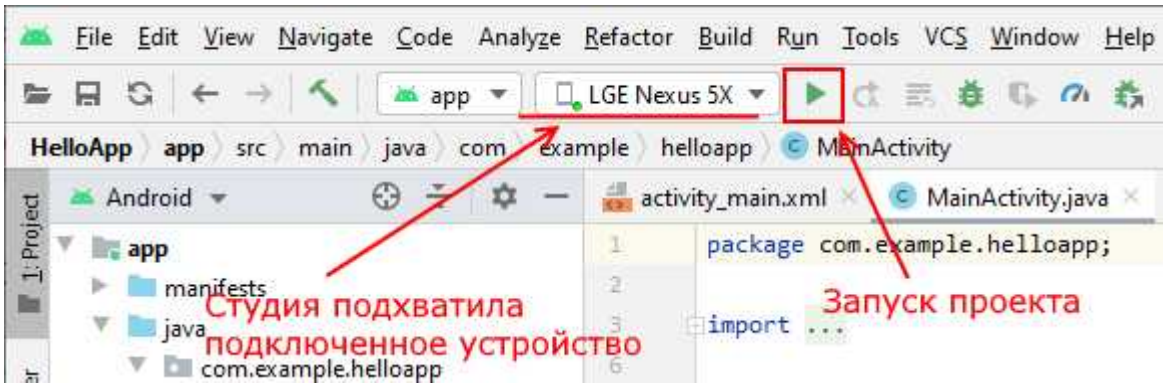


Перейдем к пункту Для разработчиков и включим возможность отладки по USB:



Запуск приложения

Подключим устройство с ОС Android (если мы тестируем на реальном устройстве) и запустим проект, нажав на зеленую стрелочку на панели инструментов.



Выберем устройство и нажмем на кнопку OK. И после запуска мы увидим наше приложение на экране устройства:

Лабораторная работа № 2

Тема: Настройка среды для разработки мобильных приложений Phonegap

Цели и задачи урока: закрепление теоретических знаний и приобретение практических навыков установки и настройки среды разработки мобильных приложений Phonegap

Тип урока – лабораторная работа.

Методы обучения – репродуктивный

Обеспечение занятия – браузер с выходом в интернет.

Литература

1. Разработка модулей программного обеспечения для компьютерных систем: учебник для студ. учреждений сред. проф. образования / Г. Н. Федорова. - 4-е изд., перераб. - М.: Издательский центр «Академия», 2020. - 384 с. ISBN 978-5-4468-9443-7
2. ГОСТ 19.201-78 Межгосударственный стандарт ЕСПД Техническое задание. Требования к содержанию и оформлению
3. https://nationalteam.worldskills.ru/skills/razrabotka-mobilnogo-prilozheniya-v-android-studio/

Задание

Общие шаги по созданию проекта разработки Android с использованием PhoneGap следующие:

1. Установите среду разработки Android и настройте переменные среды.
2. Установите среду Node.js и настройте переменные среды.
3. Используйте npm для установки глобальной среды PhoneGap.
4. Используйте команду PhoneGap для создания проекта PhoneGap.
5. Скомпилируйте PhoneGap в проект Android.
6. Импортируйте вышеупомянутые проекты в ADT для последующей разработки.

Конкретные шаги заключаются в следующем:

1. Установите и настройте среду разработки Android:

идти chttp://developer.android.com/sdk/index.html Загрузите последнюю версию ADT на эту машину и распакуйте ее в соответствующий каталог.

Далее необходимо настроить переменные среды и настроить следующие два пути в системном PATH:

A、{ADT_HOME}\sdk\platform-tools

B、{ADT_HOME}\sdk\tools

На этом настройка среды разработки для Android завершена.

2. Установите и настройте Node.js:

с http://nodejs.org/download/Загрузите и установите версию Node.js. для Windows.

3. Установите среду PhoneGap:

Используя командную строку, выполните:

```
npm install -g phonegap
```

ЭТА КОМАНДА УСТАНОВИТ И НАСТРОИТ ГЛОБАЛЬНУЮ СРЕДУ PHONEGAP ИЗ СЕТИ. УБЕДИТЕСЬ, ЧТО СЕТЬ РАБОТАЕТ ГЛАДКО.

4. Создайте проект phonegap:

Используйте командную строку для выполнения:

```
phonegap create my-app
```

Где my-app — это название проекта, который вы хотите использовать. ПРОЕКТ БУДЕТ СОЗДАН В ТЕКУЩЕМ КАТАЛОГЕ, ПОЭТОМУ ПЕРЕД ЭТИМ ПЕРЕМЕСТИТЕ ТЕКУЩИЙ КАТАЛОГ ПО НУЖНОМУ АДРЕСУ.

5. Скомпилируйте проект в проект Android:

В приведенном выше окне командной строки выполните следующую команду:

```
cd my-app
phonegap build android
```

Это две команды, первая из которых введет текущий каталог в созданный вами каталог. Вторая команда компилирует текущий проект в проект Android (то есть проект, который можно использовать в затмении ADT).

6. Импортируйте проект в затмение ADT:

Запустите eclipse в ADT, затем выберите File-New-Project, выберите Android-> Android Project из существующего кода в открывшемся мастере «New Project» и нажмите Next

На странице навигации следующего шага выберите папку my-app / platform / android, только что созданную в корневом каталоге. В приведенных ниже проектах появятся два проекта, оба из которых отмечены, но не отмечают параметр. Копировать проекты в рабочее пространство.

Выберите Готово, чтобы завершить импорт выше.

На этом этапе среда проекта Android, которая поддерживает телефонный разрыв, готова.

Лабораторная работа № 3

Тема: Установка и настройка конструктора мобильных приложений

Цели и задачи урока: закрепление теоретических знаний и приобретение практических навыков установки и настройки конструктора Apps Tech Global - создание мобильных приложений.

Тип урока – лабораторная работа.

Методы обучения – репродуктивный

Обеспечение занятия – браузер с выходом в интернет.

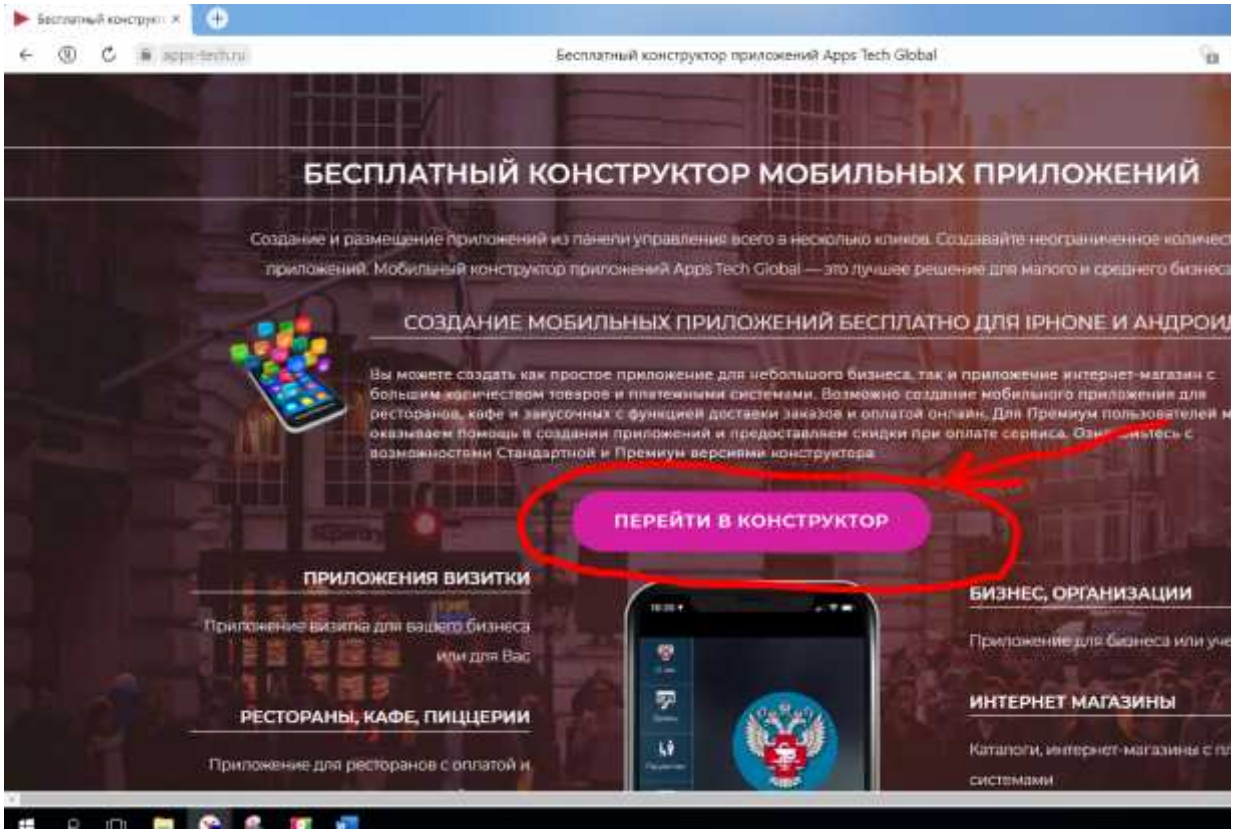
Литература

1. Разработка модулей программного обеспечения для компьютерных систем: учебник для студ. учреждений сред. проф. образования / Г. Н. Федорова. - 4-е изд., перераб. - М.: Издательский центр «Академия», 2020. - 384 с. ISBN 978-5-4468-9443-7
2. ГОСТ 19.201-78 Межгосударственный стандарт ЕСПД Техническое задание. Требования к содержанию и оформлению
3. https://apps-tech.ru/

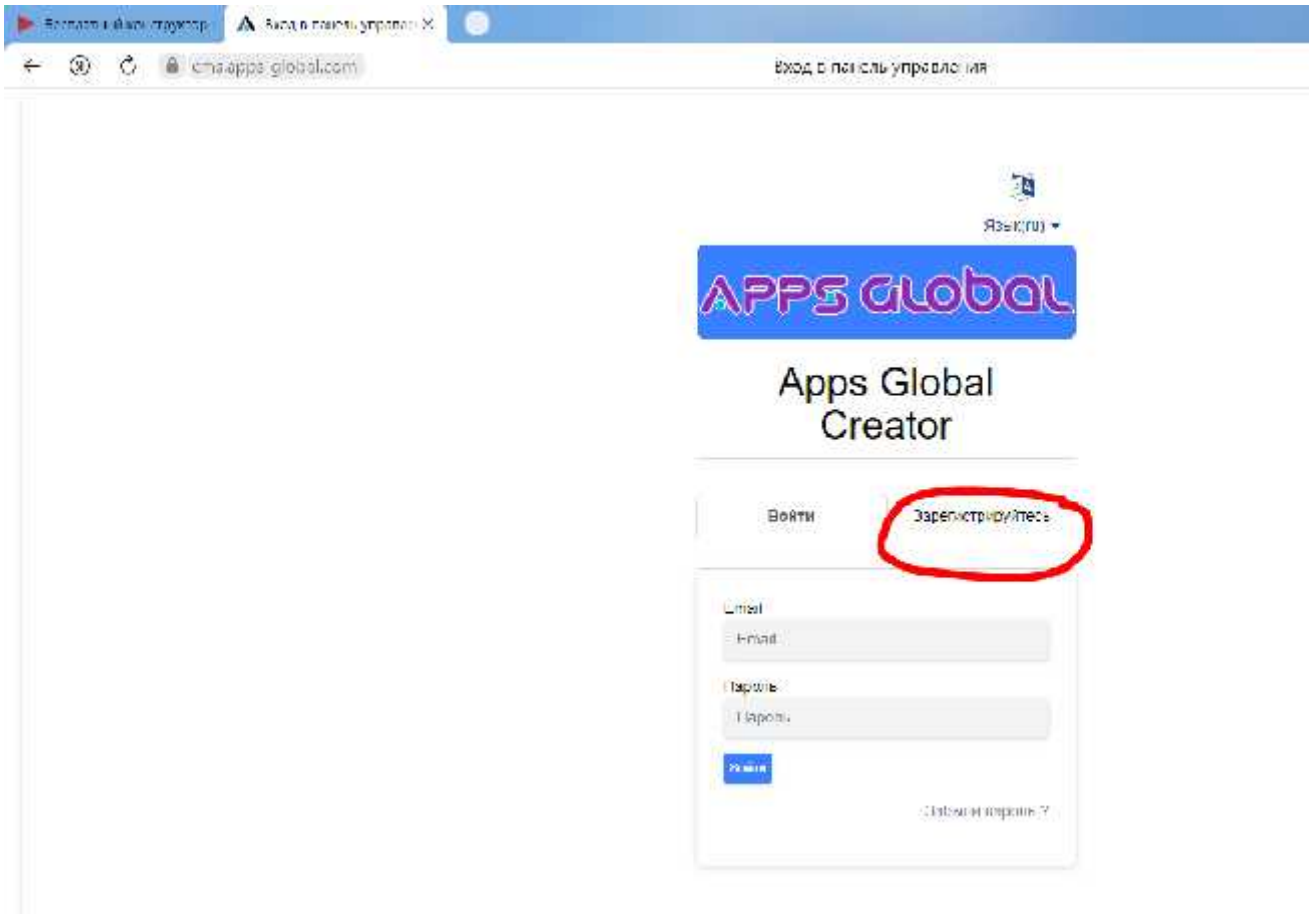
Задание

Для работы с конструктором нужно пройти по ссылке на официальный сайт Apps Tech Global: https://apps-tech.ru/

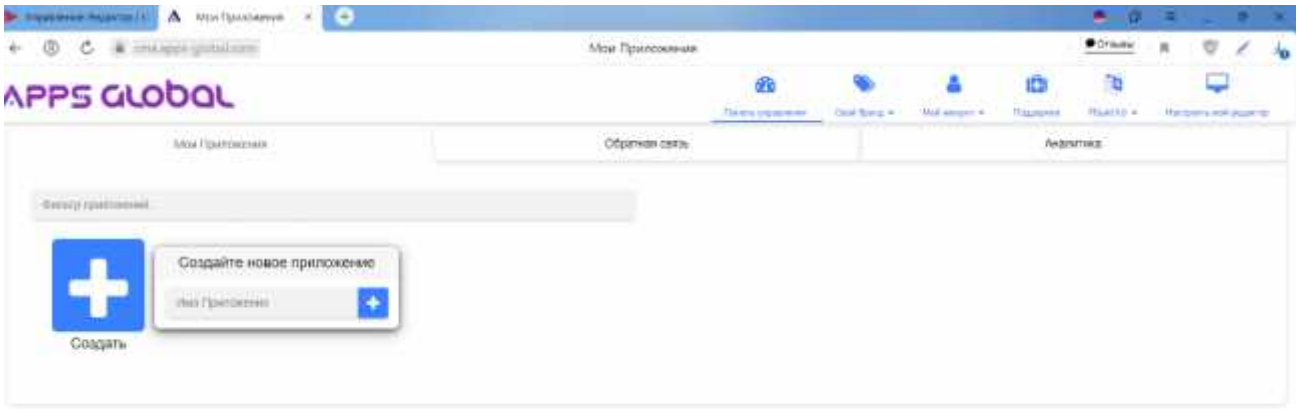
Далее перейти в конструктор



При первом входе нужно зарегистрироваться



Войти в панель управления и создать своё первое приложение:



Полная инструкция по работе в данном конструкторе по ссылке: https://apps-tech.ru/upravlenie-prilozheniem/

Лабораторная работа № 4

Тема: Создание эмуляторов и подключение устройств

Цели и задачи урока: закрепление теоретических знаний и приобретение практических навыков по созданию эмуляторов и подключений устройств в Android Studio

Тип урока – лабораторная работа.

Методы обучения – репродуктивный

Обеспечение занятия – браузер с выходом в интернет.

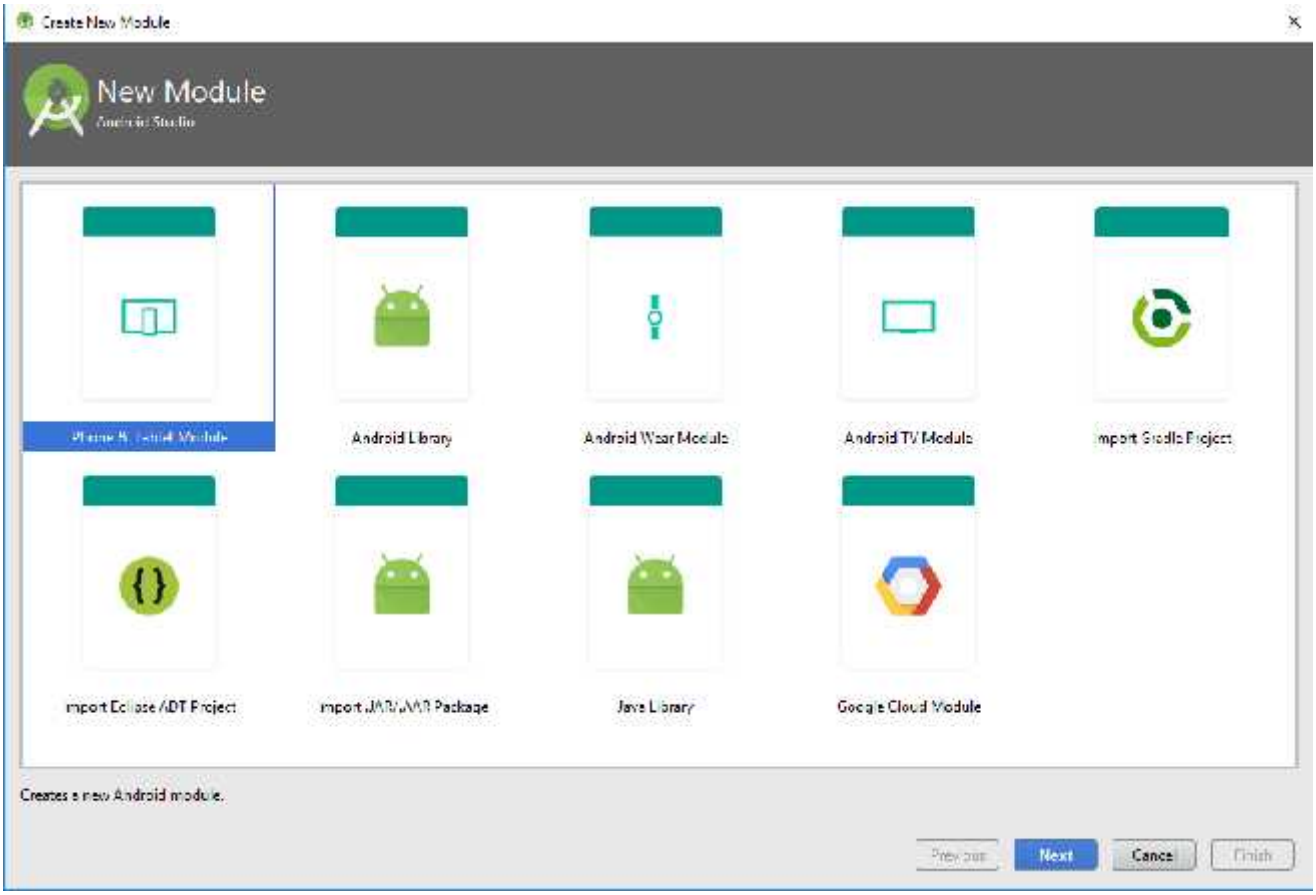
Литература

1. Разработка модулей программного обеспечения для компьютерных систем: учебник для студ. учреждений сред. проф. образования / Г. Н. Федорова. - 4-е изд., перераб. - М.: Издательский центр «Академия», 2020. - 384 с. ISBN 978-5-4468-9443-7
2. ГОСТ 19.201-78 Межгосударственный стандарт ЕСПД Техническое задание. Требования к содержанию и оформлению
3. https://metanit.com/java/android/

Задание

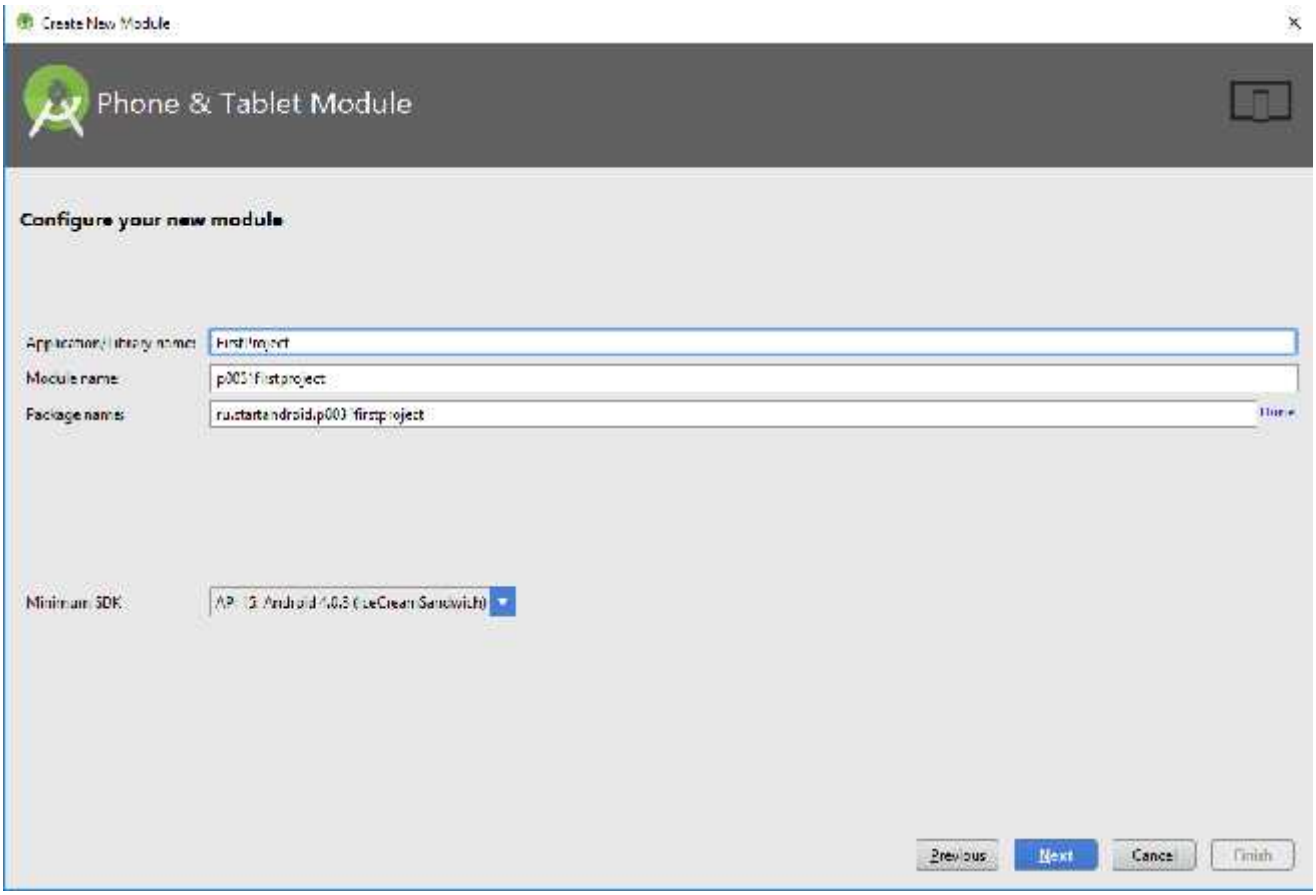
Чтобы создать модуль – в меню выбираем File -> New -> New module

Тип модуля выбираем Phone and Tablet Application



Нажимаем **Next**

Заполняем поля



Application/Library name – непосредственно имя приложения, которое будет отображаться в списке приложений в смартфоне. Пишем тут FirstProject.

Module name – это название модуля. Т.е. это название будет отображаться слева в списке модулей, там, где сейчас есть app. Давайте придумаем шаблон для названия модулей.

Например: p<номер урока(000)><номер проекта в уроке(0)>.

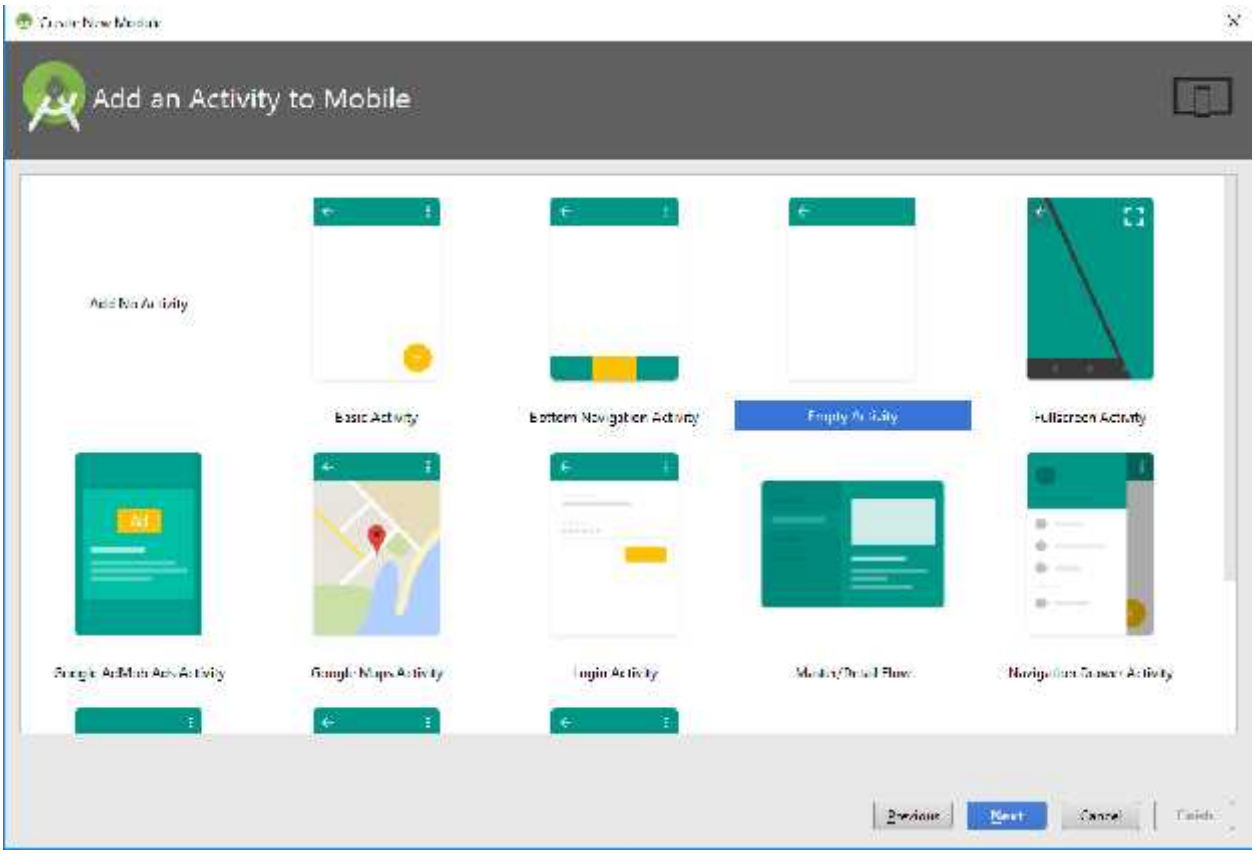
На номер урока выделим три цифры, на номер проекта – одну. Также, будем добавлять название приложения - FirstProject. И все это напишем маленькими буквами и без пробелов. Получится такое имя модуля: p0031firstproject.

Package name – имя пакета отредактируем вручную, нажав edit справа. Оставим там ru.startandroid и добавим точку и имя модуля.

Minimum SDK оставляйте без изменений.

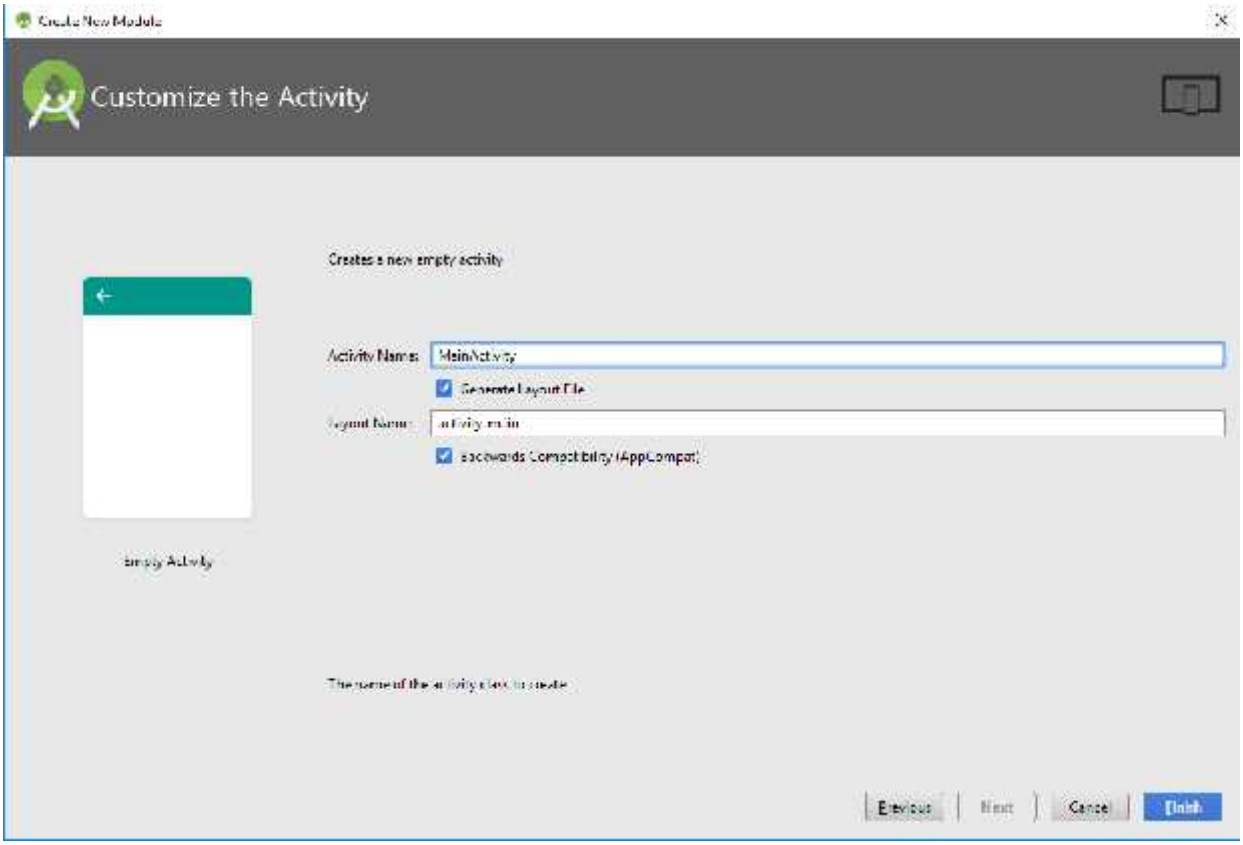
Нажимаем **Next**

Далее выберите **Empty Activity**



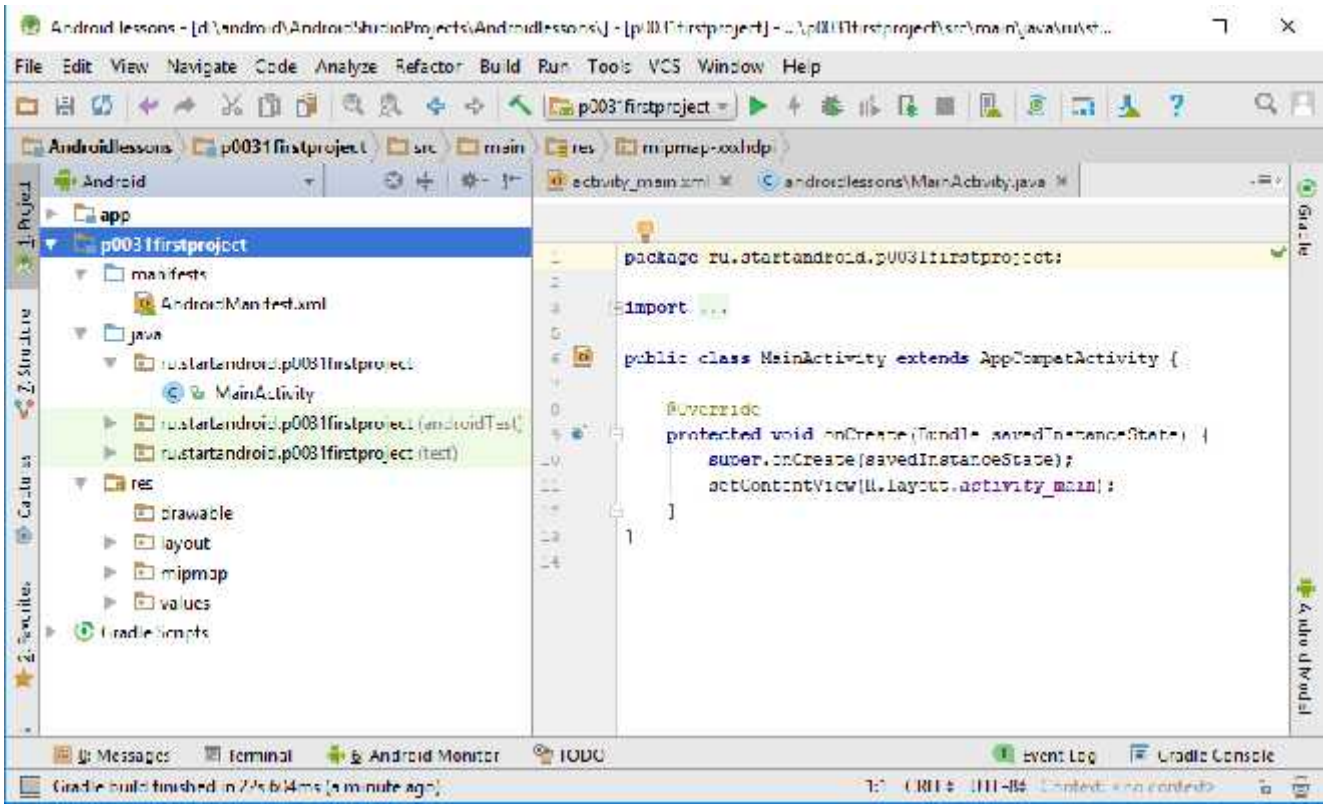
Нажимаем **Next**

Здесь ничего не меняем



Нажимаем **Finish** и ждем.

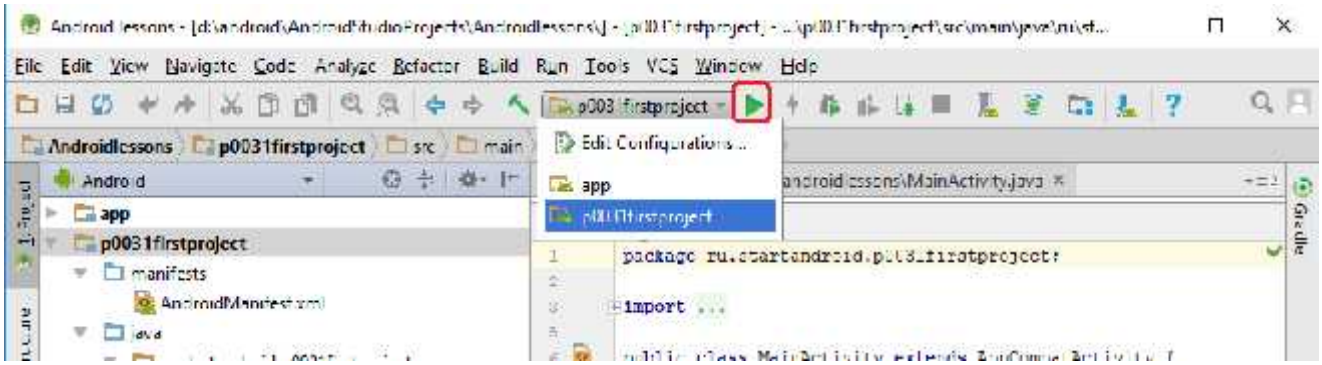
Через какое-то время модуль будет создан и мы увидим его в списке слева. Это p0031firstproject - значение, которое мы указали в поле Module name.



Можно раскрыть этот модуль и посмотреть его содержимое.
Вкратце пройдемся по интересующим нас элементам
Файл **AndroidManifest.xml** – манифест или конфиг-файл приложения
В папке **java** и ее подпапках будет весь, написанный нами, код приложения
Папка **res** используется для файлов-ресурсов различного типа.

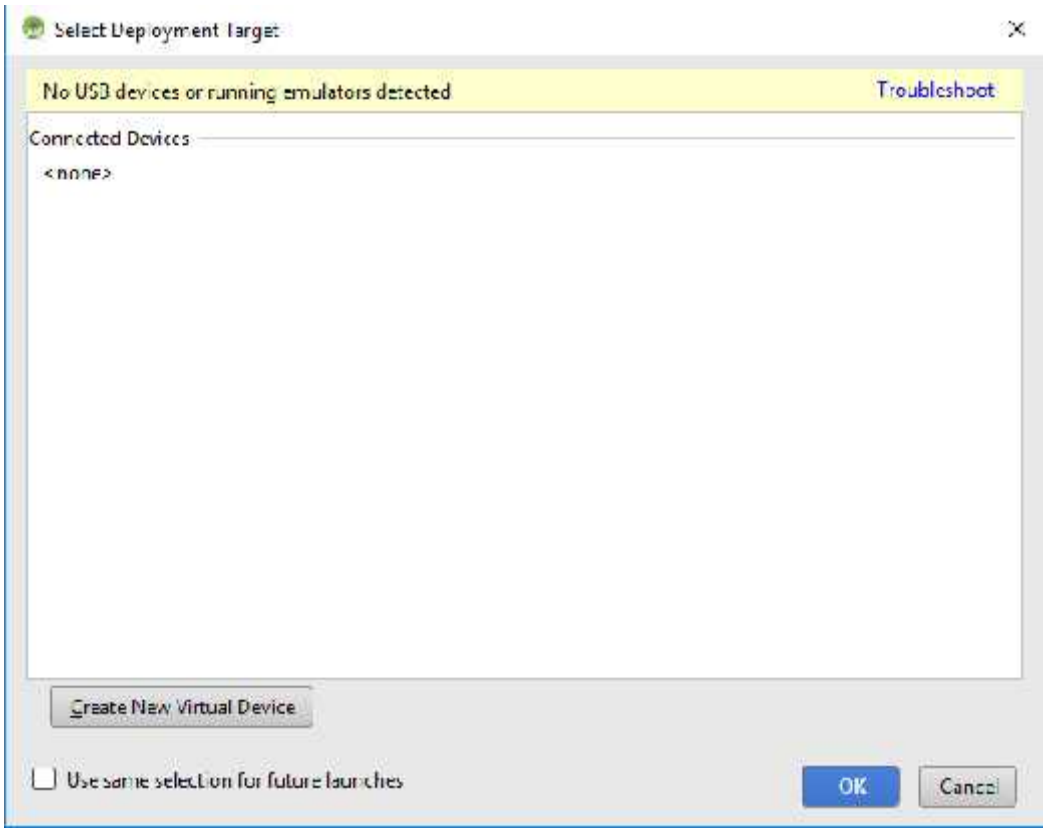
Все это будем в дальнейшем использовать, и станет понятнее, что и зачем нужно.

Запустим наше первое приложение! Для этого надо выбрать соответствующий ему модуль в выпадающем списке сверху



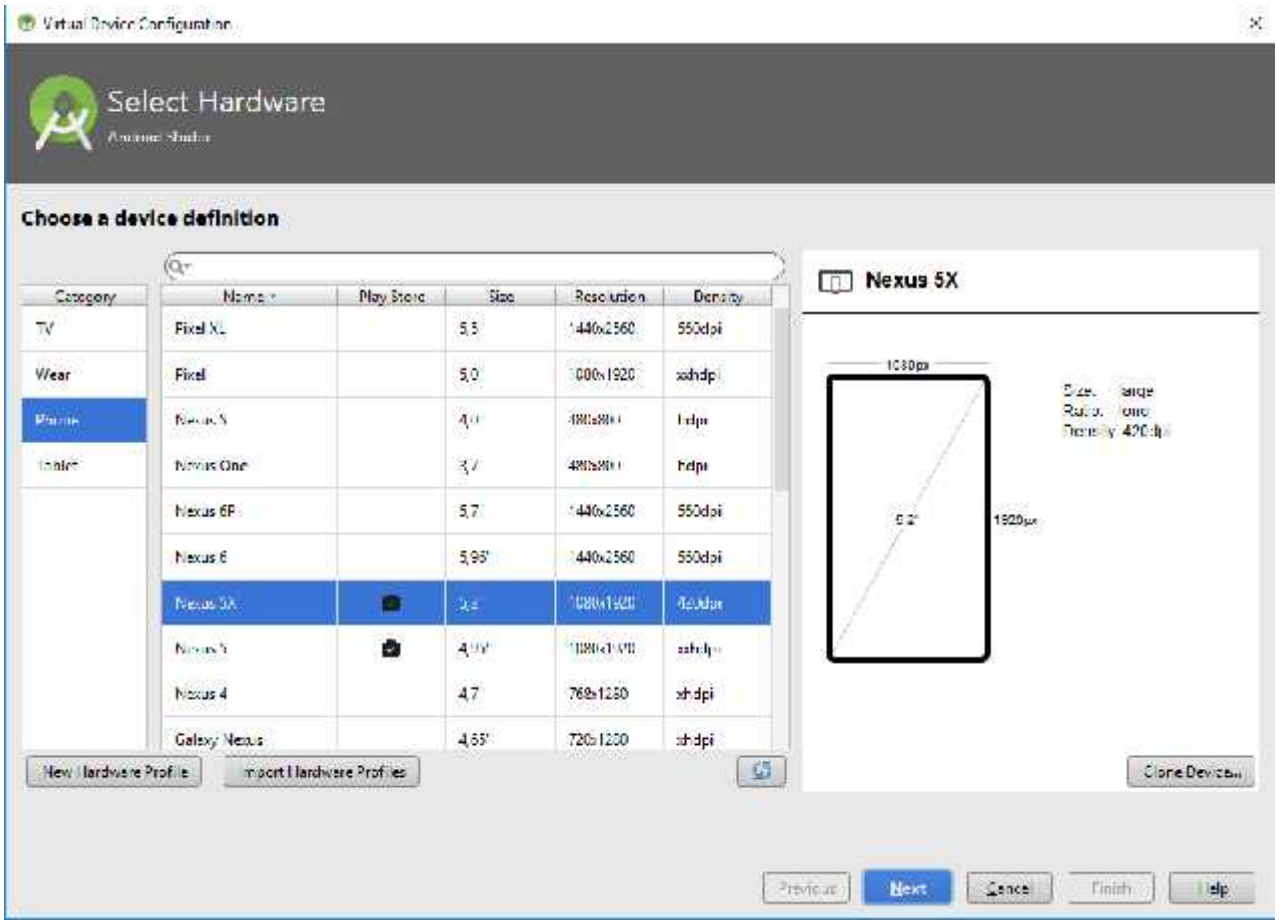
И нажимаем (чуть правее списка) кнопку с зеленым треугольником (либо комбинацию Shift+F10).

Чтобы запустить приложение, нужно какое-нибудь реальное Android-устройство или эмулятор.

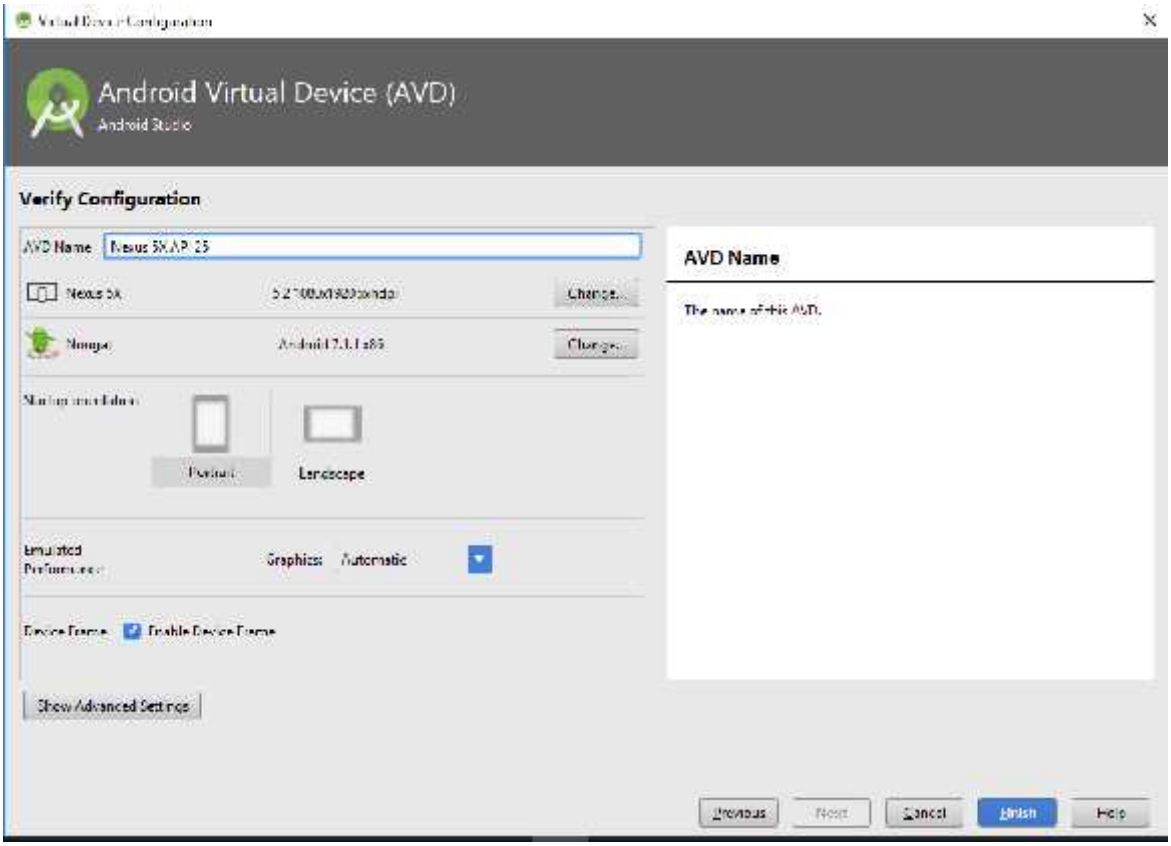


У нас пока не на чем запускать приложение. Можете подключить шнуром реальное устройство, и оно здесь появится (если не возникнет проблем с драйверами или настройками устройства).

Либо можно создать эмулятор. Жмем Create New Virtual Device

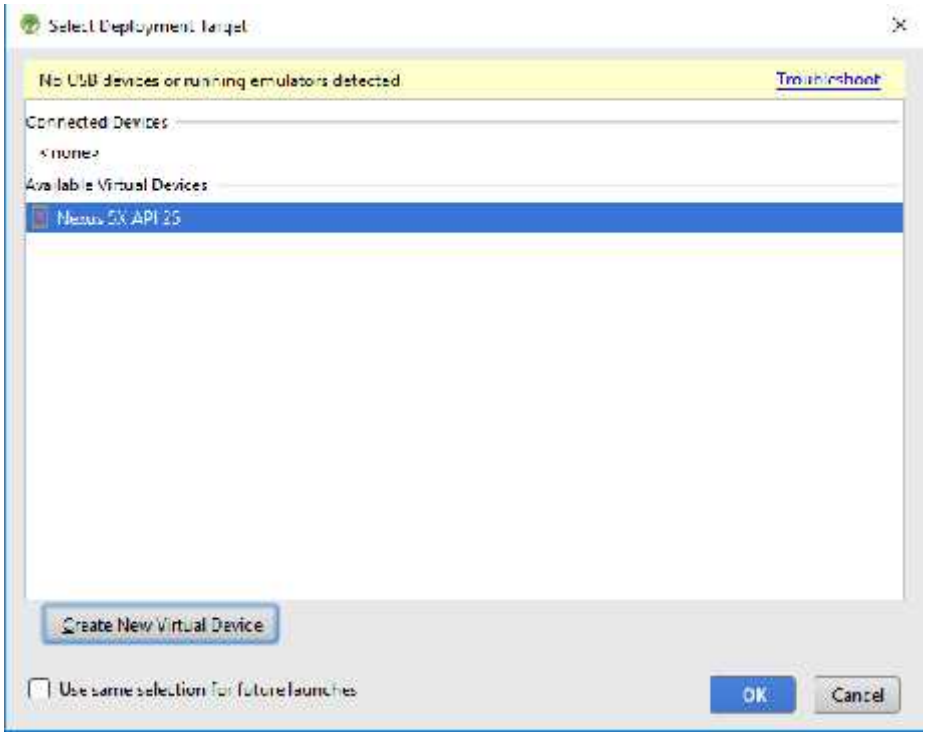


Здесь можно выбрать форм-фактор устройства. Оставляйте то, что выбрано по умолчанию.
Нажимаем **Next**
Далее переходите на вкладку x86 Images и там должен быть образ, в названии которого нет слова Download. Т.е. он уже загружен, и мы можем его использовать.



Нажимаем **Finish**

В итоге в списке устройств появляется только что созданный эмулятор, и мы можем использовать его для запуска приложения.



Нажимаем **Ok**

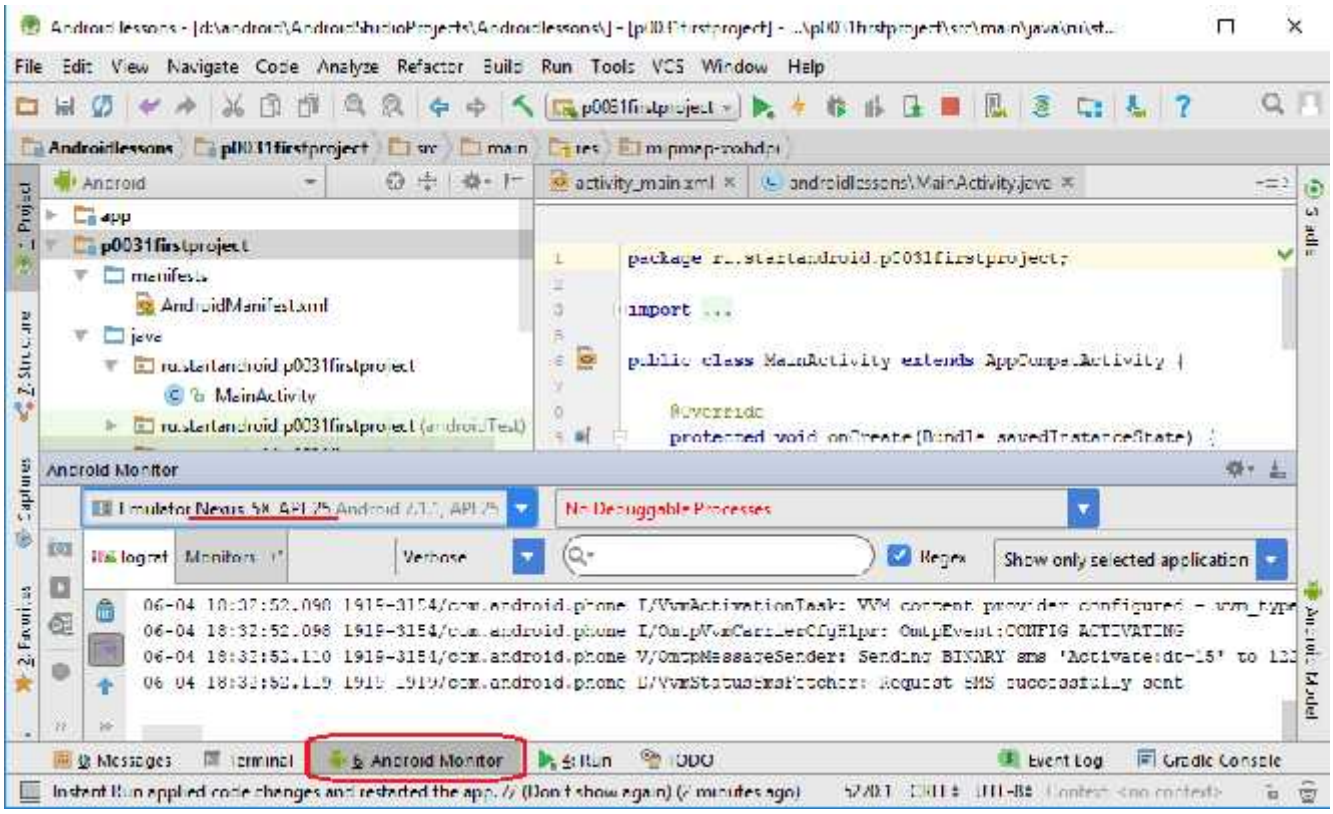
Через какое-то время (вплоть до нескольких минут) появится эмулятор

И в нем начнет запускаться Android

И в итоге запустится наше приложение

Название в заголовке - FirstProject. Именно его мы указывали при создании приложения.
Т.е. вы создали и запустили ваше первое приложение, с чем вас и поздравляю). Впереди сотни таких приложений и запусков.

Если эмулятор не показал ваше приложение, то убедитесь, что Android Studio "видит" этот эмулятор. Для этого снизу слева нажмите вкладку Android Monitor



И в списке устройств чуть выше должен быть виден эмулятор **Nexus_5X_API_25**

Если эмулятор есть в списке, а приложение не отобразилось, то попробуйте снова запустить приложение, нажав зеленый треугольник (Shift+F10).

Если эмулятора в списке нет, то закройте эмулятор и попробуйте снова запустить приложение.

Этот урок был обновлен в июне 2017. Поэтому скриншоты в них могут отличаться от ваших. Это нормально. Но это вовсе не означает, что урок уже безнадежно устарел и читать его смысла нет никакого. Код под 2.3.3 и 7.1.1 в подавляющем большинстве случаев абсолютно один и тот же. В новых версиях Android добавляются новые компоненты, а прошлые обычно остаются без изменений и достаточно редко меняются или объявляются устаревшими.

P.S.

Если у вас открыт проект и вы хотите снова увидеть стартовое окно Android Studio, в меню выберите **File** > **Close Project**.

Вы увидите стартовое окно, слева будет список ваших проектов.

Лабораторная работа № 5
Тема: Настройка режима терминала

Цели и задачи урока: закрепление теоретических знаний и приобретение практических навыков по настройке режима терминала в Android Studio

Тип урока – лабораторная работа.

Методы обучения – репродуктивный

Обеспечение занятия – браузер с выходом в интернет.

Литература

1. Разработка модулей программного обеспечения для компьютерных систем: учебник для студ. учреждений сред. проф. образования / Г. Н. Федорова. - 4-е изд., перераб. - М.: Издательский центр «Академия», 2020. - 384 с. ISBN 978-5-4468-9443-7
2. ГОСТ 19.201-78 Межгосударственный стандарт ЕСПД Техническое задание. Требования к содержанию и оформлению
3. https://metanit.com/java/android/ /

Задание

Android является полностью открытой и кастомизируемой системой. Всё благодаря тому, что в ней используется ядро Linux - самой популярной Open Source системы. На основе Linux создано большое количество ОС для настольных компьютеров и серверов, а также других электронных устройств, в числе которых можно отметить гаджеты на базе Android.

Некоторые команды выполняются только в консольном режиме

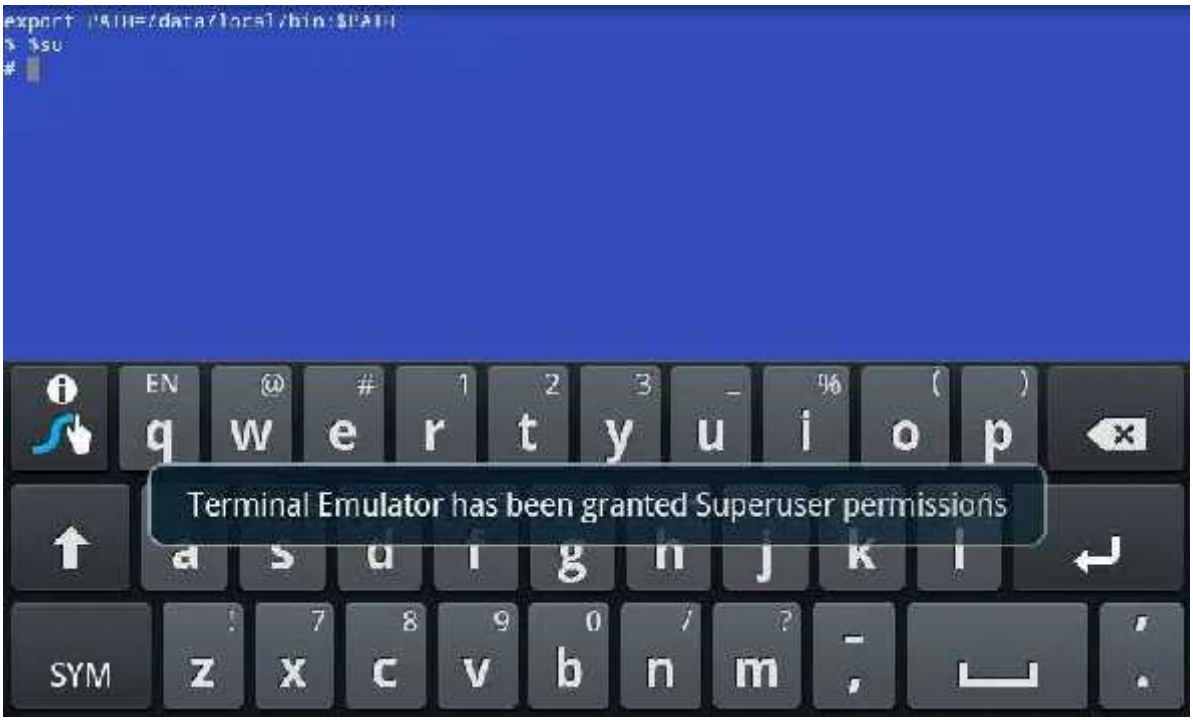
Некоторые операции совершаются при помощи консоли или терминала. Это такая системная утилита без, которая выполняет задания после ручного ввода команды. На Android по умолчанию такая утилита отсутствует, в отличие от настольного Linux или Windows. Благо, что разработчики не едят хлеб даром и ими уже создано множество эмуляторов терминала. Один из них - Android Terminal Emulator. Давайте узнаем о нём подробнее.

Что представляет собой приложение

Приложение является полноценным эмулятором терминала Linux, поддерживает несколько окон, клавиатурные сокращения, понимает кодировку UTF-8. Оно полностью бесплатное, не имеет встроенной рекламы и всплывающих окон.

О чём стоит помнить, работая с этим эмулятором?

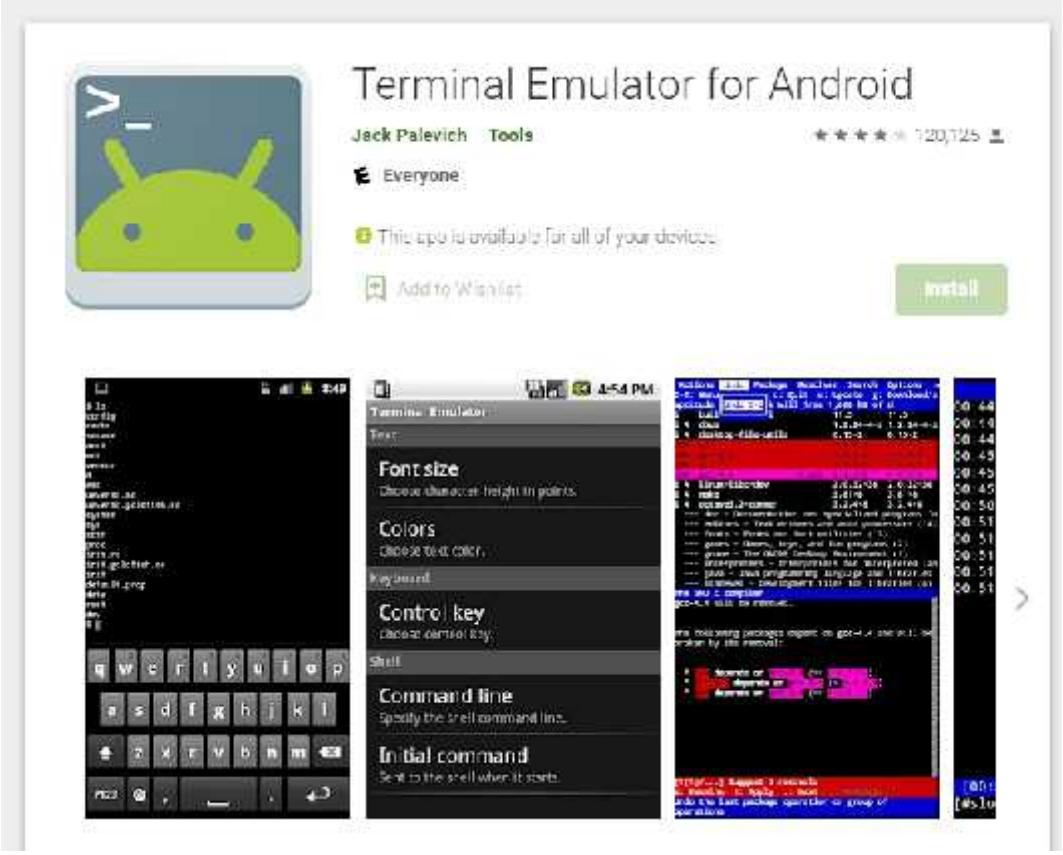
- требуются root-права;
- это не эмулятор игр;
- он не поможет получить root на устройстве;
- нужно знать команды Linux;
- возможно, понадобится установить Busy Box.



Приложение будет полезно для тех пользователей, которые чётко представляют, для чего им нужен терминал и хотя бы немного знают основные команды.

Настройки программы

Программа доступна в Play Market по ссылке https://play.google.com/store/apps/details?id=jackpal.androidterm&hl=ru .



Меню доступно на многих языках, в том числе на русском. После запуска программы, в строке уведомлений вы увидите оповещение о том, что запущен терминальный процесс. Перейдите в настройки, которые разделены на логические группы.

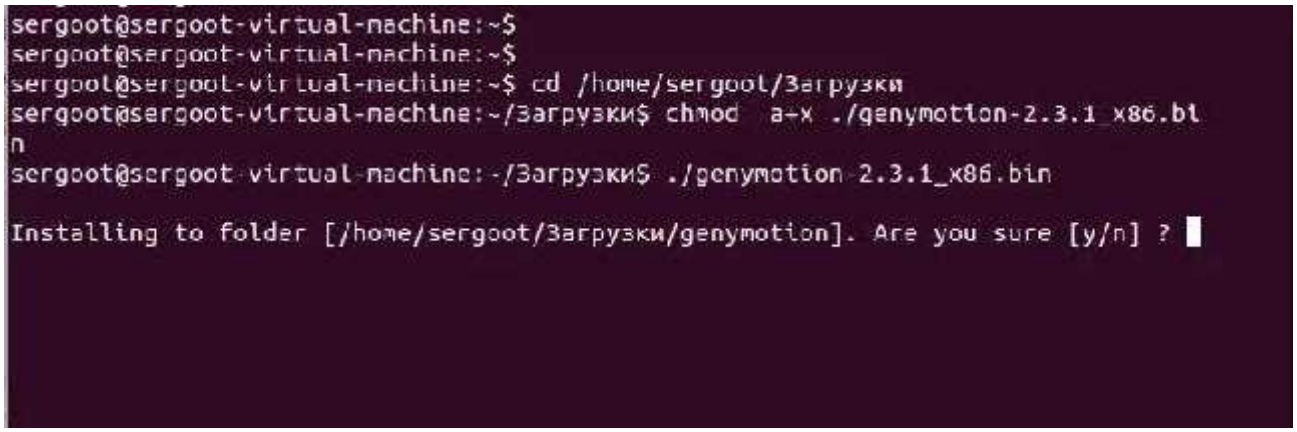
Экран. Можно настроить отображение строки состояния, панели действий и выбрать ориентацию экрана.
Текст. Регулируется размер шрифта, цифровая схема и кодировка текста.
Клавиатура. Установите поведение кнопки назад, настройте сочетание клавиш и выберите аналоги некоторых компьютерных кнопок.
Командная оболочка. Можно указать обработчика команд, предоставить, выбрать тип терминала и домашнюю папку, а также некоторые элементы поведения.

В верхней строке программы содержится всплывающий список окон, поэтому можно быстро переключаться между несколькими открытыми. Новое окно запускается нажатием на значок плюса.

Некоторые команды

adb - Отладчик Android. К мобильным устройствам можно подключать внешние накопители и устройства, эта утилита позволяет управлять ими.
am - Менеджер действий. Можно включить или выключить любое приложение или процесс.
badblocks - проверка карты памяти на наличие битых секторов.
bmgr - резервное копирование Android.
cat - просмотр содержимого файла.
chmod - изменение прав доступа к файлу.

chown - изменение владельца файла.
cmp - сравнение нескольких файлов.
cp - копирование файла.
date - отображение текущей системной даты.
dd - создание образа диска.
dmesg - просмотр лога ядра.
du - просмотр размера файла.
ext4_resize - изменение размера раздела в файловой системе ext4 (требуется root).
fsck_msdos - проверка ошибок на карте памяти.
grep - фильтрация текста.
ifconfig - просмотр сетевых устройств и управление ими (требуется root).
iptables - настройки файервола.
kill - убить процесс по его числовому идентификатору.
log - записать строку в системный лог.
logcut - просмотр системного лога в реальном времени.
ls - просмотр содержимого директории.
lsmod - отображение запущенных модулей ядра.
lsof - отображение открытых файлов.
make_ext4fs - форматирование карты памяти в формат ext4.
md5 - контрольная сумма файла.
mkdir - создание папки в каталоге.
make2fs - форматирование карты памяти в формат ext2.
mount - монтирование диска, образа или папки.
mv - перемещение файла.



netcfg - информация об интернет-соединениях.
notify - слежение за изменениями в файловой системе.
ping - проверка доступности удалённого сервера.
pm - пакетный менеджер Android, можно полностью управлять установленными приложениями.
ps - отображение информации о запущенных процессах.
resize2fs - изменение размера каталога.
rm - удаление файла.
rmdir - удаление папки.
route - управление таблицей маршрутизации.
touch - создание пустого файла.
top - список запущенных процессов.
screenshot - скриншот экрана (требуется root).
shutdown - выключение аппарата.
service - управление сервисами.
Перечисленные команды далеко не все, а лишь основные. Некоторые из них требуют более глубокого изучения.
Android Terminal Emulator - одно из лучших приложений в своём роде. Оно имеет небольшой вес и отличную функциональность.
ВНИМАНИЕ Android Terminal Emulator предназначен не для всех, а направлен на конкретных пользователей, которые свободно используют Linux, если это не ваш случай, скорее всего, вы не сможете наслаждаться этим приложением. Чтобы понять, что такое командная консоль Linux и для чего она предназначается, займитесь изучением официальной документации.

Лабораторная работа № 6
Тема: Создание нового проекта

Цели и задачи урока: закрепление теоретических знаний и приобретение практических навыков по созданию нового проекта в Android Studio

Тип урока – лабораторная работа.

Методы обучения – репродуктивный

Обеспечение занятия – браузер с выходом в интернет.

Литература

1. Разработка модулей программного обеспечения для компьютерных систем: учебник для студ. учреждений сред. проф. образования / Г. Н. Федорова. - 4-е изд., перераб. - М.: Издательский центр «Академия», 2020. - 384 с. ISBN 978-5-4468-9443-7
2. ГОСТ 19.201-78 Межгосударственный стандарт ЕСПД Техническое задание. Требования к содержанию и оформлению
3. https://metanit.com/java/android/

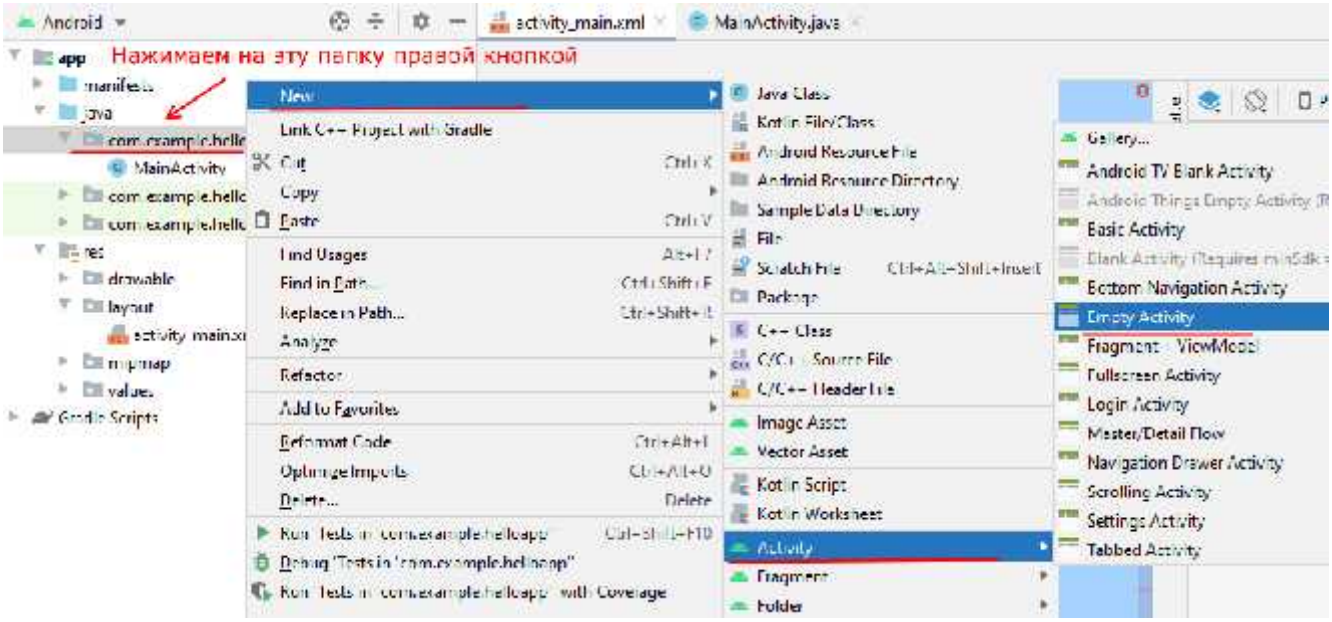
Задание

В лабораторных работах вы создали первый проект, однако весь проект состоял лишь из примитивного кода, добавляемого по умолчанию. Теперь определим само приложение, которое также будет очень простое. Пусть на одной странице приложения мы будем вводить некоторые данные и по нажатию на кнопку будет происходить переход к другой странице приложения, которая будет отображать ранее введенные данные.

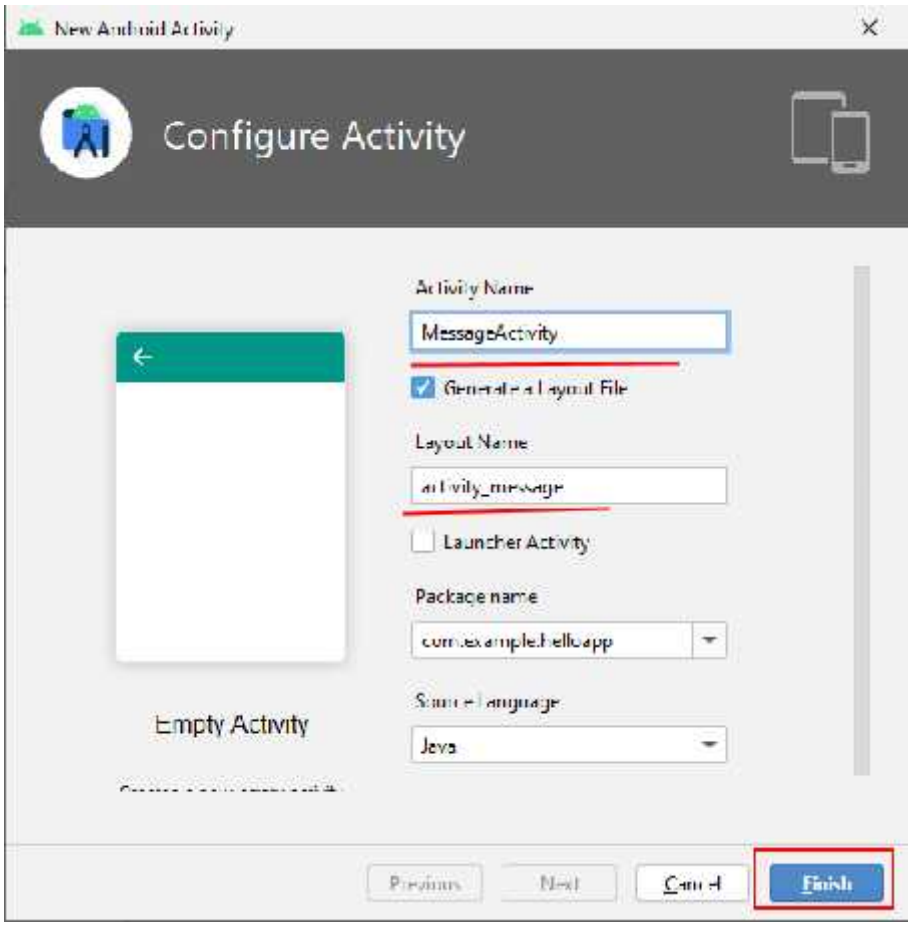
Итак, возьмем ранее созданный проект (или создадим новый).

Добавление новой Activity

По умолчанию у нас уже есть одна activity - класс MainActivity. Теперь добавим еще одну. Для этого нажмем правой кнопкой мыши в структуре проекта на папку, в котором находится класс MainActivity, и затем в контекстном меню выберем **New->Activity->Empty Activity**:



После этого откроется диалоговое окно создания новой activity:

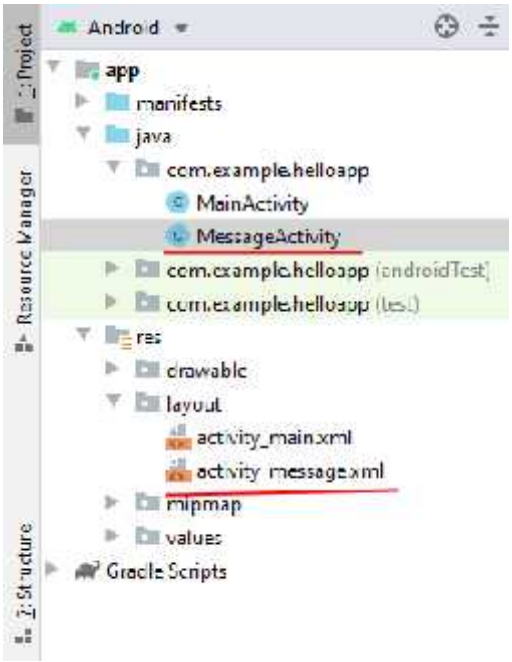


В этом окне в поле **Activity Name** введем **MessageActivity**. После этого в поле **Layout Name** автоматически должно установиться **activity_message** (если не устновилось, то введем в это поле **activity_message**). В остальных полях оставим значения по умолчанию:

- **Package Name** - должен иметь то же название, что и пакет, в котором находится MainActivity
- **Source Language** должен иметь значение **Java**

И затем нажмем Finish.

После этого в папке с MainActivity должен появиться файл с новой activity - MessageActivity:

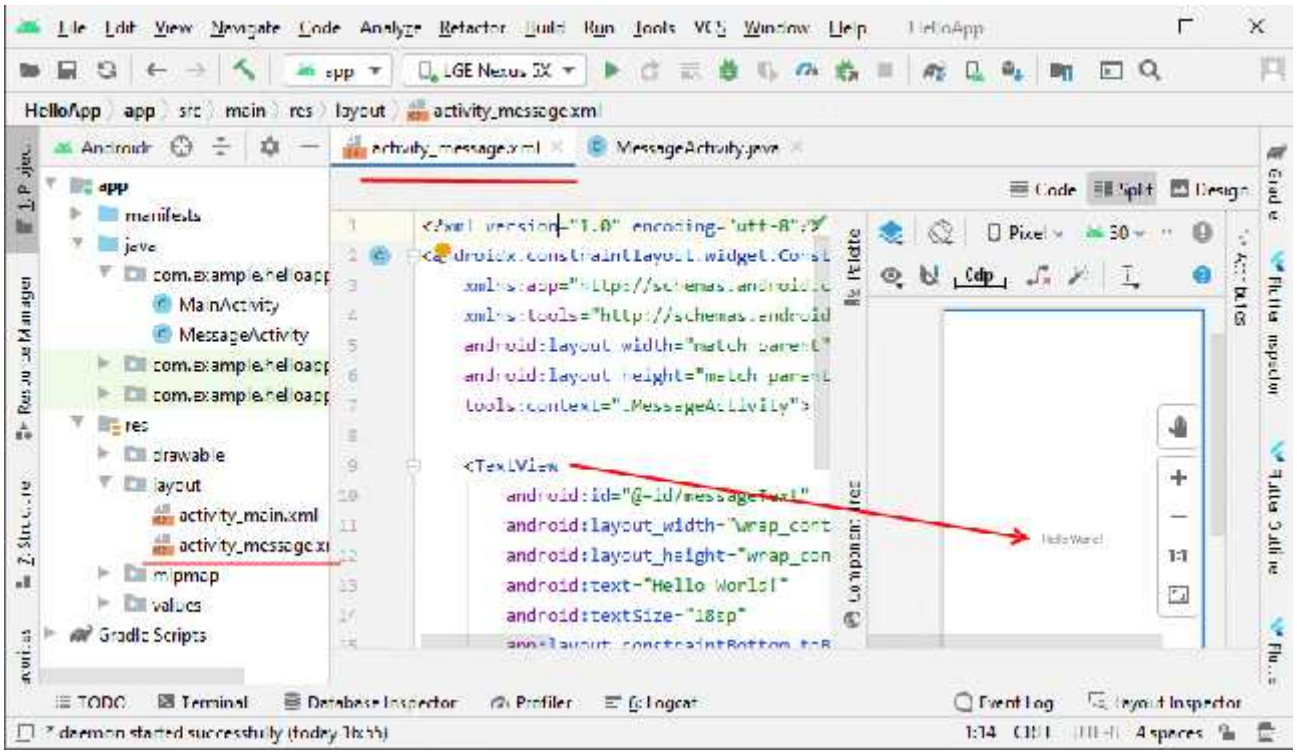


Кроме того, в каталоге *res/layout* должен появиться файл **activity_message.xml**, который представляет описание графического интерфейса для MessageActivity.
Вначале откроем файл **activity_message.xml** и изменим его код следующим образом:

```
<?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=".MessageActivity">
    <TextView
        android:id="@+id/messageText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

Определение интерфейса для MessageActivity содержит только элемент TextView, который выводит некоторую строку на экран. Рассмотрим установленные в нем атрибуты:

- android:id="@+id/messageText" - TextView имеет идентификатор "messageText", через который мы сможем обращаться к TextView в коде java. Символ @ указывает XML-парсеру использовать оставшуюся часть строки атрибута как идентификатор. А знак + означает, что если для элемента не определен id со значением messageText, то его следует определить.
- android:layout_width="wrap_content" - ширина текстового поля будет такой, чтобы вместить все его содержимое на экране
- android:layout_height="wrap_content" - высота TextView будет такой, чтобы вместить все его содержимое на экране
- android:textSize="18sp" - высота текста в TextView составляет 18 единиц (для установки высоты шрифта используется величина sp)
- app:layout_constraintBottom_toBottomOf="parent" - нижний край TextView будет выравниваться по нижней стороне контейнера ConstraintLayout
- app:layout_constraintLeft_toLeftOf="parent" - левая граница TextView будет выравниваться по левой стороне контейнера ConstraintLayout
- app:layout_constraintRight_toRightOf="parent" - правый край TextView будет выравниваться по правой стороне контейнера ConstraintLayout
- app:layout_constraintTop_toTopOf="parent" - верхний край TextView будет выравниваться по верхней стороне контейнера ConstraintLayout



Получение данных

Суть MessageActivity будет заключаться в том, что она будет получать некое текстовое сообщение и выводить его на экран (формально в элемент TextView, который был определен выше). Как мы можем получить в MessageActivity (и в любой другой activity) некоторые данные, которые переданы из другой activity?

Возьмем код класса MessageActivity. По умолчанию он выглядит так:

```
1  package com.example.helloapp;
2
3  import androidx.appcompat.app.AppCompatActivity;
4
5  import android.os.Bundle;
6
7  public class MessageActivity extends AppCompatActivity {
8
9      @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.activity_message);
13     }
14 }
```

И изменим его следующим образом:

```
1  package com.example.helloapp;
2
3  import androidx.appcompat.app.AppCompatActivity;
4  import android.content.Intent;
5  import android.os.Bundle;
6  import android.widget.TextView;
7
8  public class MessageActivity extends AppCompatActivity {
```

```
 9
10      @Override
11      protected void onCreate(Bundle savedInstanceState) {
12          super.onCreate(savedInstanceState);
13
14          setContentView(R.layout.activity_message);
15
16          // Получаем объект Intent, который запустил данную activity
17          Intent intent = getIntent();
18          // Получаем сообщение из объекта intent
19          String message = intent.getStringExtra("message");
20          // Получаем TextView по его id
21          TextView messageText = (TextView) findViewById(R.id.messageText);
22          // устанавливаем текст для TextView
23          messageText.setText(message);
24      }
25  }
```

Также, как и в MainActivity (и в других activity), создание текущей activity происходит в методе onCreate(). Все классы activity должны реализовать метод onCreate, так как система вызывает его при создании новой activity. Именно в этом методе задается компоновка нового объекта activity с помощью метода **setContentView** и именно здесь происходит начальная настройка компонентов.

Каждый объект Activity вызывается объектом Intent. Мы можем в коде Java получить вызывающий объект Intent с помощью метода getIntent:

```
1 Intent intent = getIntent();
```

Но Intent нам важен не сам по себе, а потому что через него в эту activity будут передаваться данные. Эти данные могут представлять различные типы, но в здесь мы предполагаем, что в MainActivity будет передаваться строка. И для получения строки у объекта Intent вызывается метод getStringExtra():

```
1 String message = intent.getStringExtra("message");
```

В метод передается ключ данных. То есть каждый элемент передаваемых данных представлен в формате "ключ-значение". Через ключ мы можем получить значение. И в данном случае ключом будет "message". А значение по этому ключу попадет в переменную String message.

Получив переданные в MessageActivity данные, мы можем передать их в TextView для вывода на экране устройства. Для этого вначале находим виждет TextView по его id и затем вызываем его метод setText(), который устанавливает выводимый в TextView текст:

```
1 TextView messageText = (TextView) findViewById(R.id.messageText);
2 messageText.setText(message);
```

Таким образом, MessageActivity получит переданное ей сообщение и выведет его в TextView. Теперь рассмотрим, как вызвать эту activity из MainActivity и как передать ей это сообщение. Мы определили MessageActivity, которая получает извне некоторые данные. Теперь определим в MainActivity код, который будет запускать MessageActivity и передавать ей некоторые данные.

Определение интерфейса

Изменим файл **activity_main.xml** следующим образом:

```
1 <?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=".MainActivity">

    <EditText
        android:id="@+id/editText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:hint="Введите текст"
        app:layout_constraintRight_toLeftOf="@+id/button"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"  />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:layout_marginStart="16dp"
        android:text="Отправить"
        android:onClick="sendMessage"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/editText" />


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

Здесь мы добавили в ConstraintLayout два элемента: EditText (поле для ввода текста) и Button (кнопка). Рассмотрим по отдельности, что они представляют.

Для ввода текста, который будет передаваться в MessageActivity, добавлен элемент **EditText**:

```
<EditText
        android:id="@+id/editText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:hint="Введите текст"
        app:layout_constraintRight_toLeftOf="@+id/button"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"  />
```

Итак, мы определили следующие атрибуты:

- **android:id**: обеспечивает уникальный идентификатор виджета, по которому мы можем ссылаться на объект
- **android:layout_width** задает ширину элемента. В данном случае значение "0dp". В реальности ширина будет устанавливаться контейнером ConstraintLayout на основании дополнительных атрибутов, которые идут далее.
- **android:layout_height** устанавливает высоту контейнера. Значение wrap_content задает для виджета величину, достаточную для отображения в контейнере
- **android:layout_marginStart**: задает отступ от отступ от левой границы контейнера. В данном случае 16 единиц
- **android:layout_marginTop**: задает отступ от отступ от верхней границы контейнера. В данном случае 16 единиц
- **android:hint**: задает тест-подсказку в текстовом поле
- **app:layout_constraintRight_toLeftOf**: указывает, что EditText располагается слева от элемента, id которого указан в качестве значения. Так, в данном случае значение "@+id/button" представляет идентификатор нашей кнопки. То есть правая сторона элемента EditText будет выравниваться по левой границе элемента Button.
- **app:layout_constraintLeft_toLeftOf="parent"**: указывает, что левая граница элемента будет проходить по левой стороне контейнера ConstraintLayout (с учетом выше установленного отступа)
- **app:layout_constraintTop_toTopOf="parent"**: указывает, что верхняя граница элемента будет проходить по верхней стороне контейнера ConstraintLayout (с учетом выше установленного отступа)

Теперь рассмотрим код кнопки - элемента Button, которая будет запускать MessageActivity:

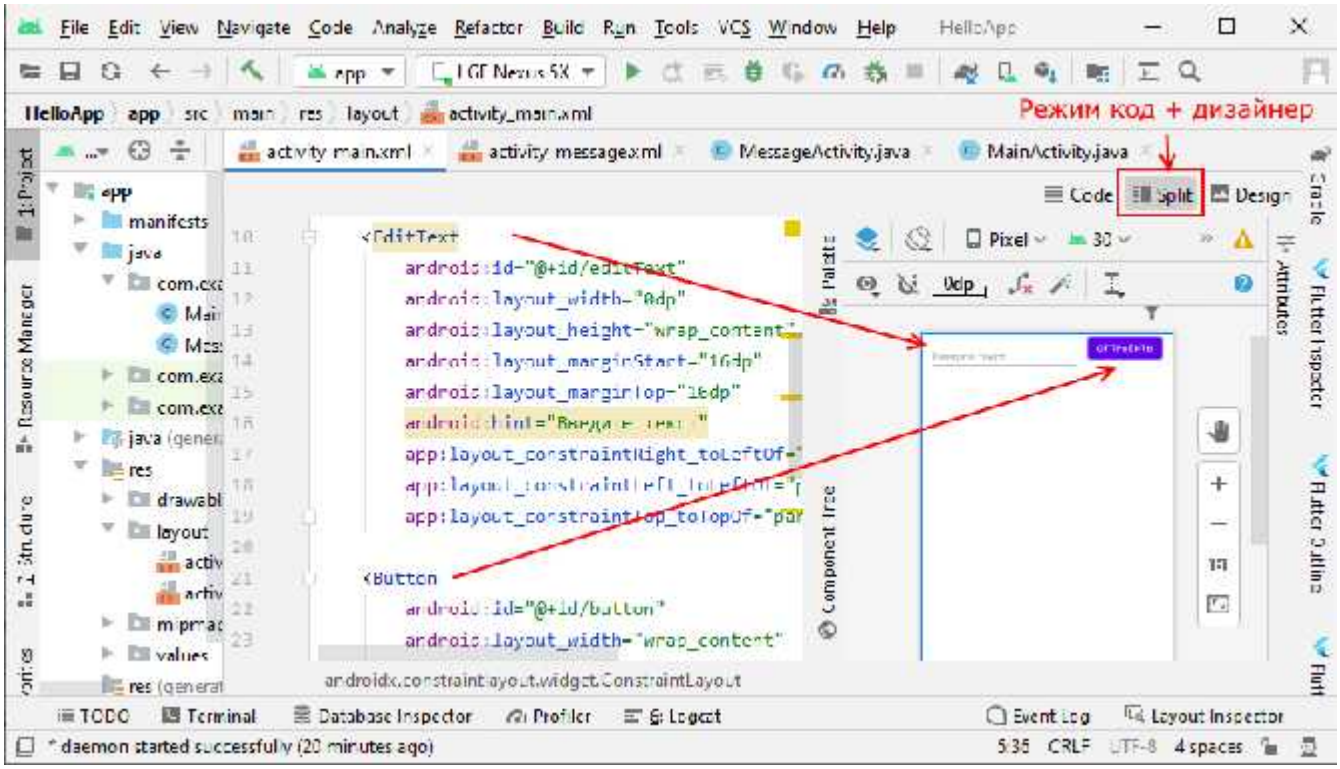
```
<Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:layout_marginStart="16dp"
        android:text="Отправить"
        android:onClick="sendMessage"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/editText" />
```

Элемент Button применяет следующие атрибуты:

- **android:id**: идентификатор кнопки - button.
- **android:layout_width** - в качестве ширины кнопки устнавливается значение wrap_content, поэтому кнопка будет иметь ту ширину, которая достаточна для вывода ее текста.
- **android:layout_height** - значение wrap_content задает для виджета величину, достаточную для отображения текста
- **android:layout_marginStart**: задает отступ от отступ от условной левой границы элемента - в данном случае это правый край текстового поля EditText. В данном случае он равен 16 единиц

- **android:layout_marginTop**: задает отступ от отступ от верхней границы контейнера. В данном случае 16 единиц
- **android:text**: задает тест кнопки
- **android:onClick**: устанавливает обработчик нажатия на кнопку. Значение "sendMessage", присвоенное атрибуту, представляет собой имя метода, определенного в классе связанной activity (в данном случае в MainActivity). То есть при нажатии на кнопку будет вызываться метод sendMessage, который мы далее определим.
- **app:layout_constraintLeft_toRightOf="@+id/editText"**: устанавливает выравнивание левой стороны элемента Button по правой границе EditText. Значение ="@+id/editText" указывает, что нижняя граница кнопки будет выравниваться по нижней границе элемента с id editText, то есть текстового поля.
- **app:layout_constraintTop_toTopOf="parent"**: указывает, что верхняя граница элемента будет проходить по верхней стороне контейнера ConstraintLayout с учетом отступа
- **app:layout_constraintRight_toRightOf**: указывает, что правая граница элемента будет проходить по правой стороне контейнера ConstraintLayout

Если мы перейдем к режиму дизайнера, например, в разделенном режиме (код + дизайнер), то мы увидим следующий макет:



Обработка нажатия кнопки

Итак, выще мы определили для кнопки обработку нажатия: android:onClick="sendMessage". Мы предполагаем, что по нажатию на кнопку будет срабатывать метод sendMessage, в котором будет запускаться MessageActivity. Для добавления этого метода изменим код класса **MainActivity**:

```
1 package com.example.helloapp;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
```

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    // Метод обработки нажатия на кнопку
    public void sendMessage(View view) {
        // действия, совершаемые после нажатия на кнопку
        // Создаем объект Intent для вызова новой Activity
        Intent intent = new Intent(this, MessageActivity.class);
        // Получаем текстовое поле в текущей Activity
        EditText editText = (EditText) findViewById(R.id.editText);
        // Получае текст данного текстового поля
        String message = editText.getText().toString();
        // Добавляем с помощью свойства putExtra объект - первый параметр - ключ,
        // второй параметр - значение этого объекта
        intent.putExtra("message", message);
        // запуск activity
        startActivity(intent);
    }
}
```

Обработчик нажатия кнопки - метод **sendMessage()** должен принимать в качестве параметра объект View, который представляет саму нажатую кнопку:

```
public void sendMessage(View view) {
```

Далее для запуска второй activity необходим объект **Intent**. Объект **Intent** представляет некоторую задачу приложения, которую надо выполнить (например, запуск activity):

```
Intent intent = new Intent(this, MessageActivity.class);
```

Конструктор этого объекта принимает два параметра:

- Первый параметр представляет контекст - объект Context (ключевое слово this употребляется здесь, так как класс MainActivity является подклассом класса Context)
- Вторым параметром идет класс компонента, которому мы передаем объект Intent. В качестве него будет выступать объект класса MessageActivity, который был добавлен в прошлой теме

Внутри метода sendMessage() используем метод **findViewById**, чтобы получить элемент EditText и передать введенный в него текст в объект intent:

```
EditText editText = (EditText) findViewById(R.id.editText);
String message = editText.getText().toString();
```

Затем полученный из текстового поля текст передается в запускаемую activity:

```
intent.putExtra("message", message);
```

Параметр "message" указывает на ключ передаваемых данных. То есть мы можем в новую activity передать множество данных, и чтобы их можно было разграничить, для них устанавливается ключ. Обращаю внимание, что здесь данные передаются по тому же ключу, по которому в MessageActivity мы получаем эти данные:

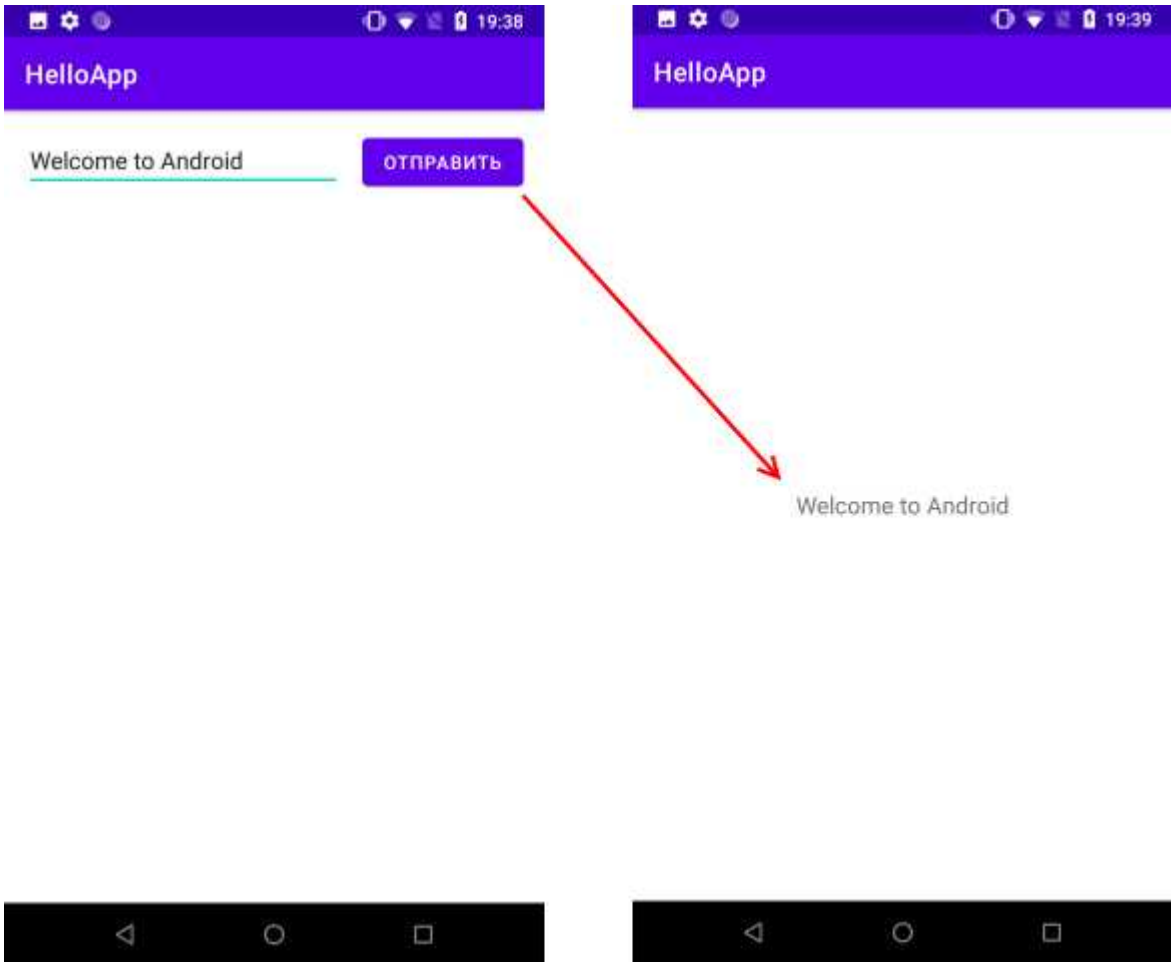
```
String message = intent.getStringExtra("message");
```

Для запуска activity нужно вызвать метод **startActivity()** и передать ему в качестве параметра объект Intent:

```
startActivity(intent);
```

После вызова этого метода система получит сигнал и запустит новый объект Activity, определенный объектом Intent.

Теперь запустим проект и после запуска приложения мы увидим текстовое поле с кнопкой, но после ввода текста и нажатия на кнопку будет запущена новая activity - MessageActivity, которая отобразит ранее введенные данные:



Лабораторная работа № 7

Тема: Изучение и комментирования кода

Цели и задачи урока: закрепление теоретических знаний и приобретение практических навыков изучение и комментирования кода мобильных приложений

Тип урока – лабораторная работа.

Методы обучения – репродуктивный

Обеспечение занятия – браузер с выходом в интернет.

Литература

1. Разработка модулей программного обеспечения для компьютерных систем: учебник для студ. учреждений сред. проф. образования / Г. Н. Федорова. - 4-е изд., перераб. - М.: Издательский центр «Академия», 2020. - 384 с. ISBN 978-5-4468-9443-7
2. ГОСТ 19.201-78 Межгосударственный стандарт ЕСПД Техническое задание. Требования к содержанию и оформлению
3. https://android-study.ru/

Теоретический материал

Комментарии

Комментарии **ОЧЕНЬ!!!** важны. Комментарии – это часть кода, которая не обрабатывается компилятором. Они служат инструментом для программиста, который хочет задокументировать, объяснить или внести ясность в тот код, который он написал и сделать его более понятным для других разработчиков, которые в свою очередь, возможно, захотят этот код усовершенствовать:

```
//это комментарий, объясняющий, что происходит в этой строке
```

Комментарий, как вы могли заметить начинается с двух «//» и заканчивается в конце строки.

```
//могу написать здесь, что угодно
а вот здесь уже не могу
```

Строка без двух «//» уже не будет закомментирована и Android Studio выдаст нам ошибку. Комментарии также можно использовать, если вы захотите отключить какую-нибудь функцию в своём приложении и посмотреть, что же произойдёт? Это происходит гораздо чаще, чем вы себе можете представить:

```
//setContentView(R.layout.activity_main);
```

В такой ситуации меню нашего приложения не загрузится, а будет чистый экран, а может и того хуже, можете поэкспериментировать. Помимо однострочных комментариев можно закомментировать большой кусочек кода. Делается это так – всё что будет заключено между знаками /* и */ будет игнорироваться компилятором:

```
*
 Эта программа была разработана хорошим программистом,
 который пишет много комментариев с целью облегчить
 постижение этого кода
*/
```

Предела в количестве строк нет, какой тип комментирования использовать зависит от ситуации.

Задание

Изучить код из Лабораторной работы № 6, для закрепления материала закомментировать код и показать преподавателю.

Лабораторная работа № 8
Тема: Изменение элементов дизайна

Цели и задачи урока: закрепление теоретических знаний и приобретение практических навыков по изменению элементов дизайна мобильных приложений

Тип урока – лабораторная работа.

Методы обучения – репродуктивный

Обеспечение занятия – браузер с выходом в интернет.

Литература

1. Разработка модулей программного обеспечения для компьютерных систем: учебник для студ. учреждений сред. проф. образования / Г. Н. Федорова. - 4-е изд., перераб. - М.: Издательский центр «Академия», 2020. - 384 с. ISBN 978-5-4468-9443-7
2. ГОСТ 19.201-78 Межгосударственный стандарт ЕСПД Техническое задание. Требования к содержанию и оформлению
3. https://android-study.ru/

Задание

Стили

Настроить элемент можно с помощью различных атрибутов, которые задают высоту, ширину, цвет фона, текста и так далее. Но если несколько элементов используют одни и те же настройки, то можно объединить эти настройки в стили.

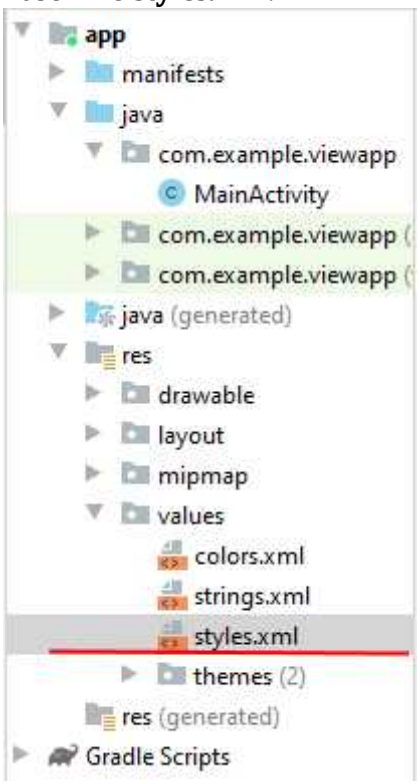
Например, пусть есть несколько элементов TextView:

```
<?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"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textSize="28sp"
        android:textColor="#3f51b5"
        android:text="Android Lollipop"

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textSize="28sp"
        android:textColor="#3f51b5"
        android:text="Android Marshmallow"
```

```
30
31          app:layout_constraintLeft_toLeftOf="parent"
32          app:layout_constraintRight_toRightOf="parent"
33          app:layout_constraintBottom_toTopOf="@+id/textView3"
34          app:layout_constraintTop_toBottomOf="@+id/textView1"
35          />
36      <TextView
37          android:id="@+id/textView3"
38          android:layout_width="0dp"
39          android:layout_height="wrap_content"
40          android:gravity="center"
41          android:textSize="28sp"
42          android:textColor="#3f51b5"
43          android:text="Android Nougat"
44
45          app:layout_constraintLeft_toLeftOf="parent"
46          app:layout_constraintRight_toRightOf="parent"
47          app:layout_constraintBottom_toBottomOf="parent"
48          app:layout_constraintTop_toBottomOf="@+id/textView2"
49          />
50
51  </androidx.constraintlayout.widget.ConstraintLayout>
```

Все эти TextView имеют одинаковый набор свойств, и, к примеру, если захочется изменить цвет текста, то придется менять его у всех трех TextView. Данный подход не оптимален, а

более оптимальный подход представляет использование стилей, которые определяются в проекте в папке **res/values**.
Итак, добавьте в проект в папку **res/values** новый элемент **Value Resourse File**, который назовите **styles.xml**:



Определим в файле **styles.xml** следующее содержимое:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="TextViewStyle">
        <item name="android:layout_width">0dp</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#3f51b5</item>
        <item name="android:textSize">28sp</item>
        <item name="android:gravity">center</item>
    </style>
</resources>
```

Здесь определен новый стиль TextViewStyle, который с помощью элементов item задает значения для атрибутов TextView.
Стиль задается с помощью элемента <style>. Атрибут name указывает на название стиля, через которое потом можно ссылаться на него. Необязательный атрибут parent устанавливает для данного стиля родительский стиль, от которого дочерний стиль будет наследовать все свои характеристики.
С помощью элементов item устанавливаются конкретные свойства виджета, который принимает в качестве значения атрибута name имя устанавливаемого свойства.
Теперь применим стиль, изменим файл activity_main.xml:

```
<?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"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView1"

        style="@style/TextViewStyle"

        android:text="Android Lollipop"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        />
    <TextView
        android:id="@+id/textView2"
        style="@style/TextViewStyle"
        android:text="Android Marshmallow"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/textView3"
        app:layout_constraintTop_toBottomOf="@+id/textView1"
        />
    <TextView
        android:id="@+id/textView3"
        style="@style/TextViewStyle"
        android:text="Android Nougat"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
        />

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

Используя определение style="@style/TextViewStyle" текстовое поле связывается с определением стиля. Итоговый результат буде тот же, что и раньше, только кода становится меньше. А если нужно поменять какие-то характеристики, то достаточно изменить нужный элемент item в определении стиля.

Темы

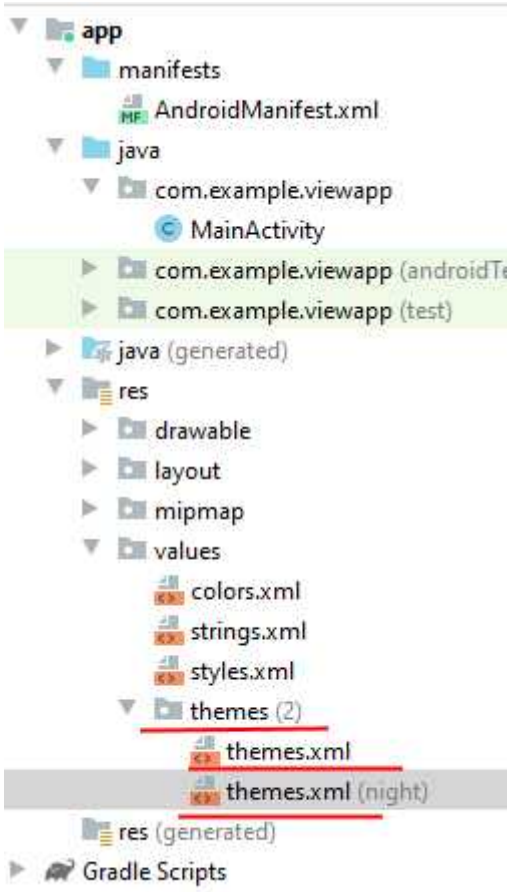
Кроме применение отдельных стилей к отдельным элементам, можно задавать стили для всего приложения или activity в виде тем. Тема представляет коллекцию атрибутов, которые применяются в целом ко всему приложению, классу activity или иерархии виджетов.

Можно самим создать тему. Однако Android уже предоставляет несколько предустановленных тем для стилизации приложения, например, Theme.AppCompat.Light.DarkActionBar и ряд других.

По умолчанию приложение уже применяет темы. Так, откройте файл **AndroidManifest.xml**. В нем можно увидеть следующее определение элемента application, представляющего приложение:

```
<application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ViewApp">
```

Задание темы происходит с помощью атрибута android:theme. В данном случае используется ресурс, который называется в моем случае **Theme.ViewApp**. По умолчанию файлы тем определены в папке **res/values**. В частности, здесь можно найти условный каталог **themes**, в котором по умолчанию есть два элемента: **themes.xml**:



Один файл представляет светлую тему, а другой - темную. Например, откроем файл themes.xml со светлой темой:

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.ViewApp" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
```

```
        <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Здесь можно увидеть, что тема определяется как и стиль с помощью элемента **style**.. Атрибут **parent** указывает на родительскую тему, от которой текущая тема берет все стилевые характеристики. То есть тема "Theme.ViewApp" использует другую тему - "Theme.MaterialComponents.DayNight.DarkActionBar". И кроме того, определяет ряд своих собственных стилей.

Также можно заметить, что здесь определяются не только характеристики для атрибутов, но и семантические имена, например, colorPrimary, которому сопоставлен ресурс "@color/purple_500".

При необходимости мы можем изменить эти характеристики или дополнить тему новыми стилевыми характеристиками. Например, изменим цвет свойства colorPrimary, которое применяется в том числе в качестве фонового цвета заголовка и кнопки:

```
<item name="colorPrimary">#1565C0</item>
```

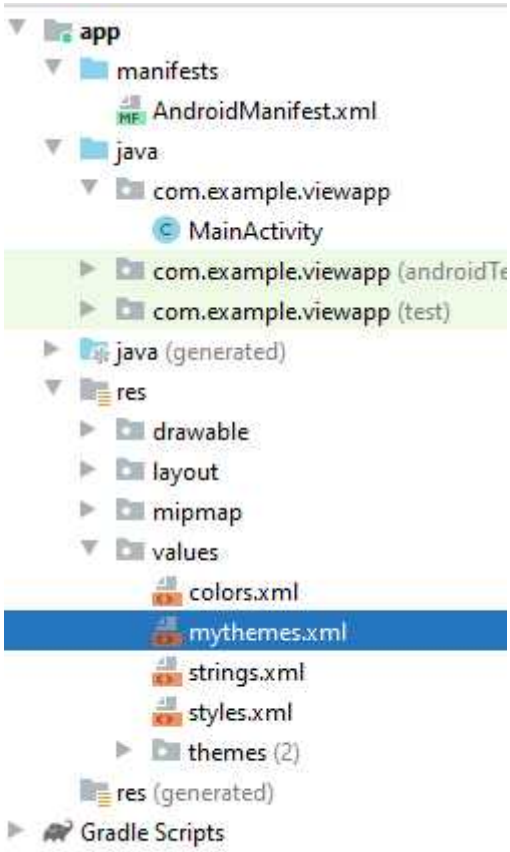
И соответственно изменится цвет по умолчанию для фона заголовка и кнопки:

```
<?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"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello Android"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        />

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

Создание собственной темы

Вместо использования встроенных тем мы можем создать свою. Для этого добавим в папку **res/values** новый файл **mythemes.xml** и определим в нем следующее содержимое:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="MyTheme" parent="Theme.AppCompat.Light">
        <item name="android:textColor">#FF018786</item>
        <item name="android:textSize">28sp</item>
    </style>
</resources>
```



Создали стиль "MyTheme", который унаследован от стиля Theme.AppCompat.Light. В этом стиле переопределили два свойства: высоту шрифта (textSize) - 28sp, а также цвет текста (textColor) - #FF018786.

Теперь определим этот стиль в качестве темы приложения в файле **AndroidManifest.xml**:

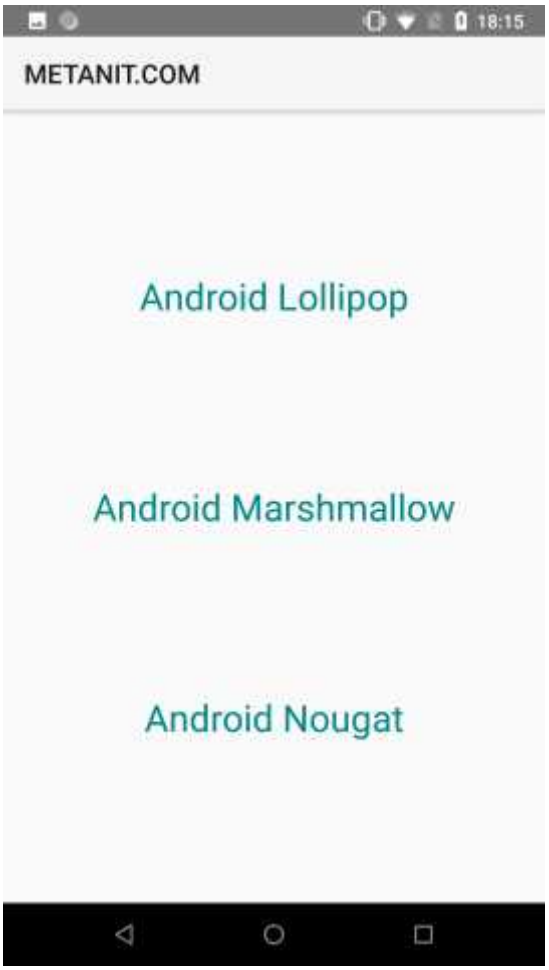
```
<application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"

        android:theme="@style/MyTheme"><!-- применение темы -->
```

Пусть у нас будет следующая разметка в **activity_main.xml**

```
<?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"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Lollipop"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Marshmallow"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/textView3"
        app:layout_constraintTop_toBottomOf="@+id/textView1"
        />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Nougat"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
        />

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

Как видно, для элементов TextView не устанавливается атрибут textSize и textColor, однако поскольку они определены в теме, которая применяется глобально к нашему приложению, то элементы TextView будут подхватывать эти стилевые характеристики:



Применение темы к activity

Выше темы применялись глобально ко всему приложению. Но также можно применить их к отдельному классу Activity. Для этого надо подкорректировать файл манифеста AndroidManifest. Например:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.viewapp"
    android:versionCode="1"
    android:versionName="1.0">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"

        android:theme="@style/Theme.ViewApp">
        <activity android:name=".MainActivity" android:theme="@style/MyTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
        </activity>
    </application>

</manifest>
```

Атрибут **android:theme** элемента <activity> указывает на применяемую к MainActivity тему. То есть глобально к приложению применяется тема "Theme.ViewApp", а к MainActivity - "MyTheme".

Применение темы к иерархии виджетов

Также можно применить тему к иерархии виджетов, установив атрибут **android:theme** у элемента, к которому (включая его вложенные элементы) мы хотим применить тему. Например, примение темы к ConstraintLayout и ее элементам:

```
<?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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    android:theme="@style/MyTheme">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Lollipop"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        />

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

Лабораторная работа № 9
Тема: Создание проекта на Swift

Цели и задачи урока: закрепление теоретических знаний и приобретение практических навыков в создание проекта на Swift

Тип урока – лабораторная работа.

Методы обучения – репродуктивный

Обеспечение занятия – браузер с выходом в интернет.

Литература

1. Разработка модулей программного обеспечения для компьютерных систем: учебник для студ. учреждений сред. проф. образования / Г. Н. Федорова. - 4-е изд., перераб. - М.: Издательский центр «Академия», 2020. - 384 с. ISBN 978-5-4468-9443-7
2. ГОСТ 19.201-78 Межгосударственный стандарт ЕСПД Техническое задание. Требования к содержанию и оформлению
3. https://metanit.com/swift/tutorial/1.1.php

Задание

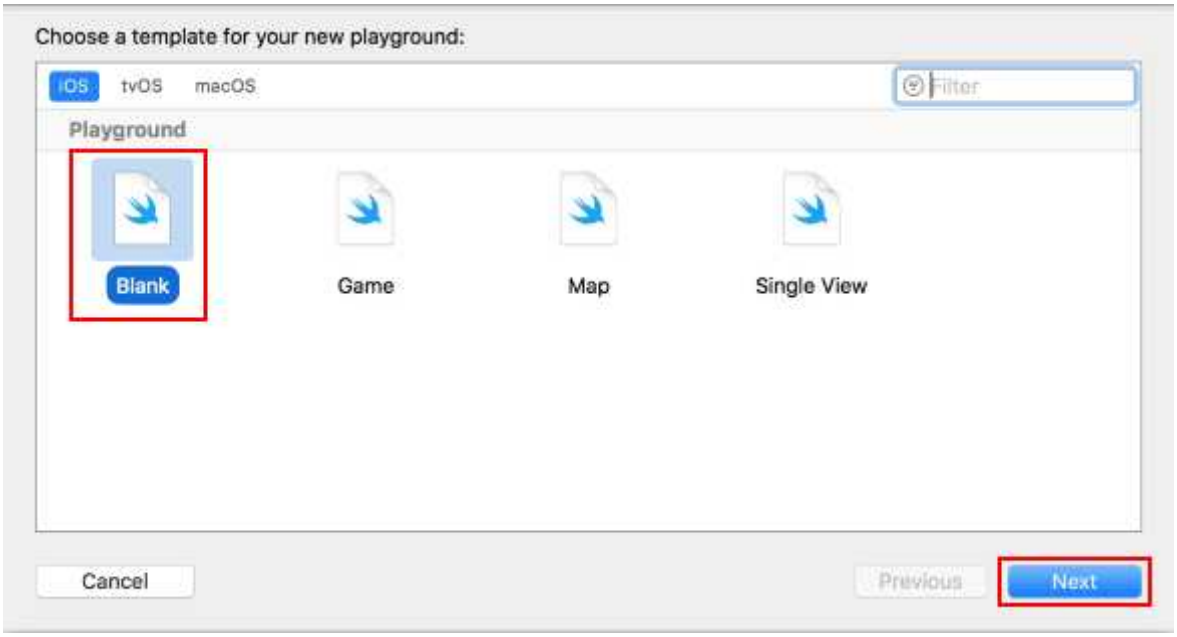
Начало работы с Swift и XCode

Для разработки под iOS потребуется специальная среда программирования, которая называется **XCode**. XCode позволяет использовать языки Swift и Objective-C для создания приложений под iOS и Mac OSX. Она является бесплатной, ее можно установить из App Store:

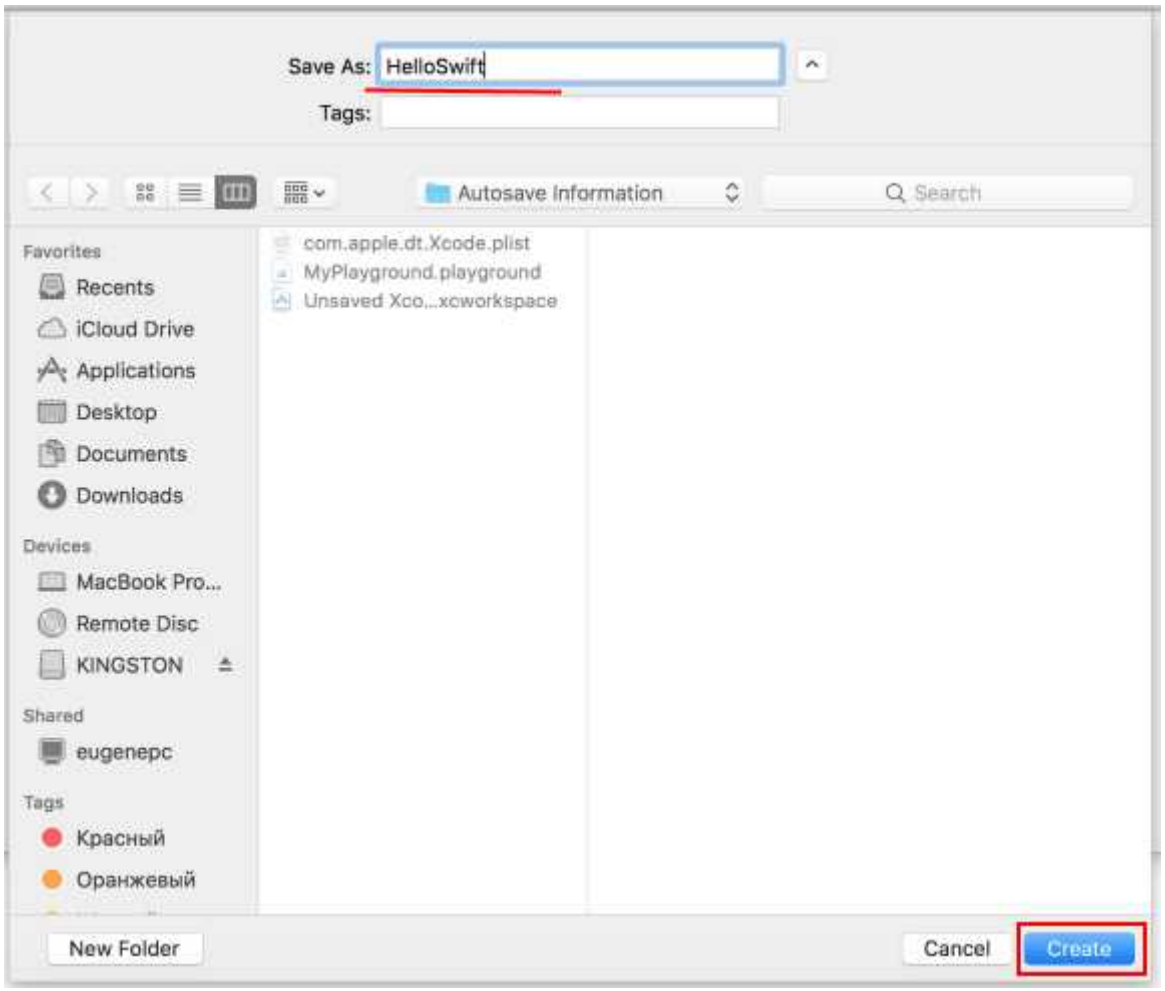
После установки запустим XCode. По умолчанию нам открывается стартовый экран с выбором опций для создания нового проекта, а также список ранее созданных проектов:

В этом окне выберем пункт **Get started with a playground**.

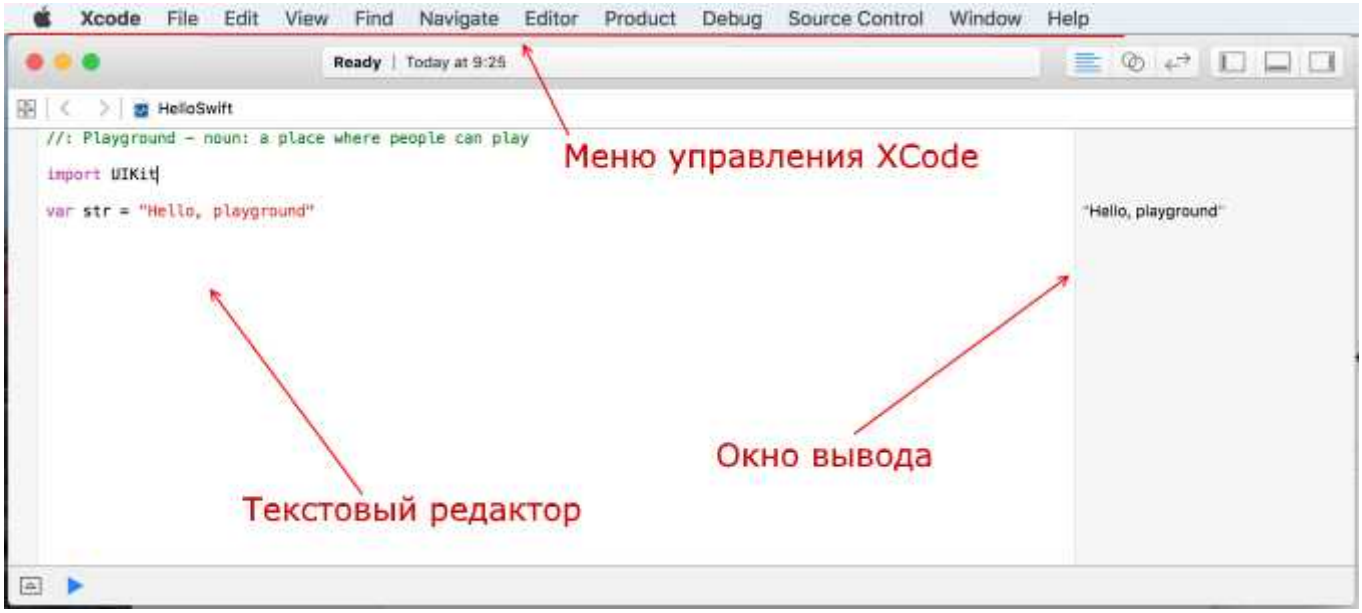
После этого надо будет указать тип проекта. Выберем первый тип проекта - **Blank**:



Далее надо будет указать, под каким именем будет сохраняться проект. Укажем в качестве имени HelloSwift и нажмем на кнопку Create:



И после всех этих настроек нам откроется непосредственно сама среда Playground. По сути она представляет собой текстовый редактор с окном консольного вывода, в котором мы можем потренироваться с выражениями и операциями языка Swift.



В верхней панели MacOS установится меню для управления XCode. Большую часть самого Playground будет занимать текстовый редактор, в который мы будем вводить команды языка Swift. А справа находится окно консольного вывода, где мы сможем увидеть результат введенных команд.

Удалим все из текстового редактора и введем в него следующее простейшее выражение:

```
1 print("Hello world!")
```

print() - это метод, который выводит некоторую строку. В данном случае это строка "Hello world!". И в окне консольного вывода справа мы сможем увидеть это сообщение:

Это довольно простой пример, но в дальнейшем мы часто будем обращаться к среде Playground в XCode при изучении языка Swift.

Структура программы

Вкратце стоит сказать пару вводных слов о структуре программы на Swift. Программа состоит из набора команд, каждая из команд называется **инструкцией** (statement). Например, выше мы использовали следующую инструкцию:

```
1 print("hello world!")
```

Как правило, каждая инструкция помещается на одной строке:

```
1 print("hello world!")
2 print("welcome to swift")
```

При этом надо отметить, что Swift в целом имеет си-подобный синтаксис, то есть родственен таким языкам программирования как C, C++, C#, Java, однако однострочные инструкции не завершаются точкой с запятой. Хотя это можно делать. Но если мы помещаем несколько инструкций на одной строке, например:

```
1 print("hello world!"); print("welcome to swift")
```

То в это случае их следует разделять точкой с запятой.

Как и в других си-подобных языках в Swift для оформления структурных блоков применяются фигурные скобки. Например:

```
1 class Book {    // начало блока класса
2     func print() {  // начало блока функции
3         print("печать книги")
4     }   // конец блока функции
5 }   // конец блока класса
```

Комментарии

В Swift можно определять комментарии к исходному коду. Для создания многострочных комментариев применяется конструкция /* текст комментария */:

```
1 /*
2  Первая программа на Swift
3  функция печатает строку hello world
4 */
5 print("hello world!")
```

Для создания однострочных комментариев применяется двойной слеш:

```
1 print("hello world!") // функция печатает строку hello world
2 print("welcome to swift") // выводит строку welcome to swift
```

При компиляции программы комментарии не учитываются и служат лишь для напоминания разработчику об узловых моментах исходного кода.

Основы Swift

Переменные и константы. Типы данных

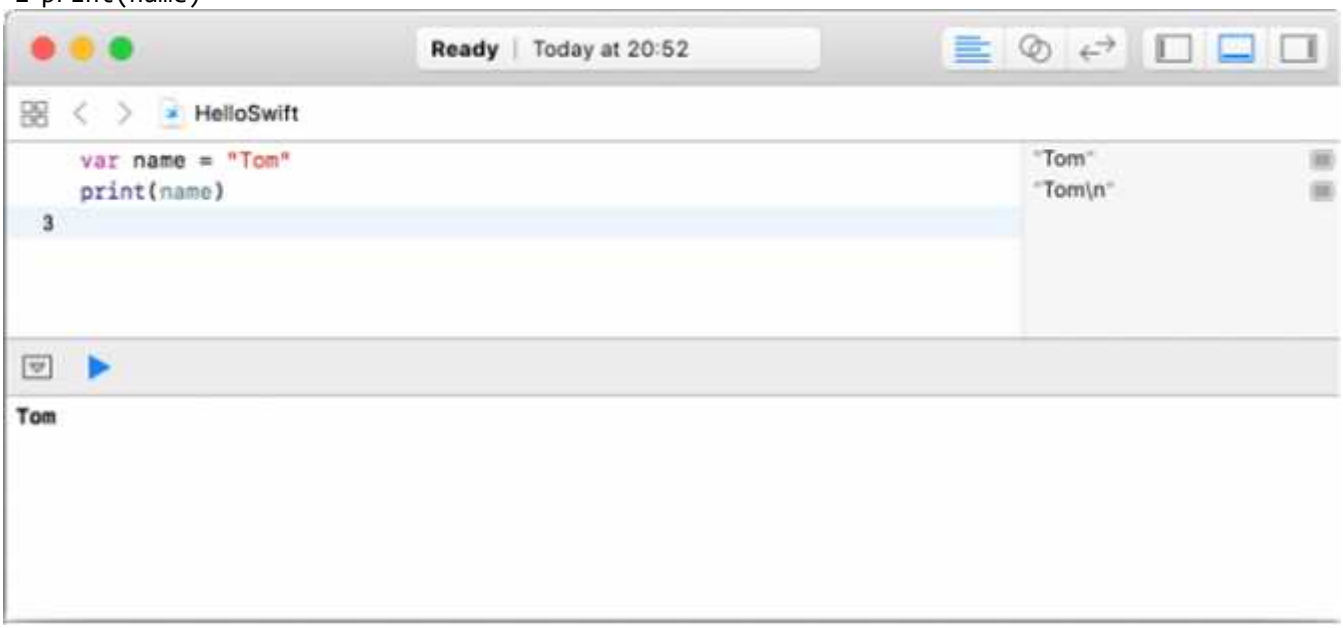
Программа на Swift обладает двумя качествами: она может хранить некоторые данные и может выполнять действия. Для хранения данных в Swift, как и в других языках программирования, используются **переменные**. Переменная представляет именованный участок в памяти, в котором хранится некоторое значение.

Переменные имеют имя и значение. Для определения переменной используется ключевое слово **var**. Например:

```
1 var name = "Tom"
```

В данном случае переменная имеет имя "name" и в качестве значения хранит строку "Tom". После определения переменной мы можем использовать ее, например, передать ее функцию print, чтобы вывести ее значение:

```
1 var name = "Tom"
2 print(name)
```



Отличительной особенностью переменных является то, что мы можем изменять их значение многократно во время работы программы. Например:

```
1 var name = "Tom" // значение Tom
2 name = "Bob" // значение Bob
```

Кроме переменных для хранения данных в программе могут использоваться константы. Константы подобны переменным, они также хранят некоторое значение, за тем

исключением, что определяются с помощью ключевого слова **let**, и мы не можем после их инициализации изменить их значение:

```
let name = "Tom" // значение Tom
// name = "Bob"  - так сделать нельзя, так как name - константа
```

Таким образом, если значение некоторой переменной в течении программы меняться не будет, то вместо этой переменной лучше использовать константу.

Мы можем определить сразу несколько переменных и констант на одой строке. В этом случае они должны разделяться запятой:

```
var name = "Tom", surname = "Smith"
```

Правила именования

Переменные и константы должны иметь уникальные имена. Нельзя использовать в программе несколько переменных и (или) констант с одними и теми же именами.

Причем хорошей практикой является использование названий в так называемом "верблюжем регистре" или CamelCase. То есть названия начинаются с маленькой буквы. Если название состоит из нескольких слов, то только первое из них начинается с маленькой буквы. Например:

```
var personName = "Tom"
var personStreatAddress = "St. Patrick avenue, 20"
```

Типы данных

Каждая переменная или константа хранит в себе значение определенного типа. Например, выше использованная переменная name хранит строку:

```
var name = "Tom"
```

В языке Swift имеются следующие типы данных:

- **Int8**: представляет целые числа со знаком размером не более 8 бит (от -128 до 127)
- **UInt8**: представляет целые положительные числа размером не более 8 бит (от 0 до 255)
- **Int16**: представляет целые числа со знаком размером не более 16 бит (от -32768 до 32767)
- **UInt16**: представляет целые положительные числа размером не более 16 бит (от 0 до 65535)
- **Int32**: представляет целые числа со знаком размером не более 32 бита (от -2147483648 до 2147483647)
- **UInt32**: представляет целые положительные числа размером не более 32 бита (от 0 до 4294967295)
- **Int64**: представляет целые числа со знаком размером не более 64 бита (от -9223372036854775808 до 9223372036854775807)
- **UInt64**: представляет целые положительные числа размером не более 64 бита (от 0 до 18446744073709551615)
- **Int**: представляет целые числа со знаком, например, 1, -30, 458. На 32-разрядных платформах эквивалентен Int32, а на 64-разрядных - Int64
- **UInt**: представляет целые положительные числа, например, 1, 30, 458. На 32-разрядных платформах эквивалентен UInt32, а на 64-разрядных - UInt64
- **Float**: 32-битное число с плавающей точкой, содержит до 6 чисел в дробной части
- **Double**: 64-битное число с плавающей точкой, содержит до 15 чисел в дробной части
- **Float80**: 80-битное число с плавающей точкой

- **Bool**: представляет логическое значение true или false
- **String**: представляет строку
- **Character**: представляет отдельный символ

Тип переменных и констант может определен явно или неявно. Выше тип определялся неявно. Но мы также можем явным образом определить тип:

```
var age: Int = 32
var name: String = "Tom"
```

Определение переменной с типом происходит по шаблону:

```
var название_переменной: тип_переменной = значение_переменной
```

Также мы можем сначала определить переменную, а потом присвоить ей значение:

```
var name: String
name = "Tom"
```

Также можно определить сразу набор однотипных переменных:

```
var height, weight: Double
```

Неявная типизация

Если мы явным образом не указываем тип переменных и констант, то он автоматически выводится системой на основании хранящегося значения. Например:

```
var name = "Tom"
```

Здесь явным образом не указан тип переменной name, однако поскольку она хранит строку, то система будет рассматривать эту переменную как объект типа String. Или, например:

```
var age = 32
```

Здесь также явно не указан тип переменной, поэтому система будет рассматривать эту переменную как объект Int, то есть целое число.

Однако при таком подходе следует учитывать, что Swift не всегда выводит те типы, которые нам могут быть нужны. Например, все целые числа Swift воспринимает как объекты типа Int, а дробные числа - как объекты типа Double. Это надо учитывать, чтобы не попасть в некорректные ситуации. Например:

```
var d = 3.4           // тип Double
var f : Float = 1.2
d = f                   // ! Ошибка - разные типы
```

В данном случае на основании присвоенного значения переменная d будет представлять тип Double, а переменная f представляет тип Float, поэтому при присвоении переменной d значения f мы получим ошибку.

Типобезопасность

Swift является типобезопасным языком со строгой типизацией, поэтому после того, как для переменной будет установлен тип, мы его уже изменить не сможем. Так, в следующем случае мы столкнемся с ошибкой:

```
var age: Int
age = "Tom"
```

Ошибка возникает, так как переменная age ожидает число, а строка "Том" не является числом и не соответствует переменной age по типу.

Мы можем присваивать значение переменной или константы другой переменной:

```
var age: Int = 22
var years = age
```