

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Саратовский государственный технический университет
имени Гагарина Ю.А.»

Филиал федерального государственного бюджетного образовательного
учреждения высшего образования
«Саратовский государственный технический университет
имени Гагарина Ю.А.» в г. Петровске

УТВЕРЖДАЮ

Директор филиала СГТУ

имени Гагарина Ю.А. в г. Петровске

А.А. Бесшапошникова

«05» июля 2024 г.



МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ


по дисциплине

МДК. 01.03 «Разработка мобильных приложений»

направление подготовки

09.02.07 «Информационные системы и программирование»

Методические указания рассмотрены
на заседании предметной (цикловой) комиссии
общепрофессиональных дисциплин,
профессиональных модулей специальностей
технического профиля
«14» июня 2024 года, протокол №12

Председатель ПЦК  /Ю.А. Табарова/

Петровск 2024

Пояснительная записка

Методические указания по выполнению практических работ подготовлены на основе рабочей программы учебной дисциплины МДК. 01.03 «Разработка мобильных приложений», разработанной на основе ФГОС СПО по специальности 09.02.07 «Информационные системы и программирование» и соответствующих общих (ОК) и профессиональных (ПК) компетенций:

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

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях.

ОК 04. Эффективно взаимодействовать и работать в коллективе и команде.

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

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, 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. Пользоваться профессиональной документацией на государственном и иностранном языках.

ОК 10. Использовать знания по финансовой грамотности, планировать предпринимательскую деятельность в профессиональной сфере

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

ПК 1.6. Разрабатывать модули программного обеспечения для мобильных платформ

При выполнении практических работ студент 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

При выполнении практических работ студент 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

Содержание практических занятий определено рабочей программой и тематическим планированием, соответствует теоретическому материалу изучаемых разделов учебной дисциплины.

Объем практических занятий по дисциплине определяется учебным планом по данной специальности.

Продолжительность практического занятия – 2 академических часа. Перед проведением практического занятия преподавателем организуется инструктаж, а по его окончании – обсуждение итогов.

Комплект методических указаний по выполнению практических занятий по дисциплине МДК. 01.03 «Разработка мобильных приложений» содержит 26 практических занятий.

**Перечень практических работ
по дисциплине МДК. 01.03 «Разработка мобильных приложений»**

ПРАКТИЧЕСКАЯ РАБОТА № 1

Тема: Создание эмуляторов и подключение устройств

ПРАКТИЧЕСКАЯ РАБОТА № 2

Тема: Создание эмуляторов и подключение устройств

ПРАКТИЧЕСКАЯ РАБОТА № 3

Тема: Настройка режима терминала

ПРАКТИЧЕСКАЯ РАБОТА № 4

Тема: Создание нового проекта

ПРАКТИЧЕСКАЯ РАБОТА № 5

Тема: Создание нового проекта

ПРАКТИЧЕСКАЯ РАБОТА № 6

Тема: Создание нового проекта

ПРАКТИЧЕСКАЯ РАБОТА № 7

Тема: Создание нового проекта

ПРАКТИЧЕСКАЯ РАБОТА № 8

Тема: Изучение и комментирование кода

ПРАКТИЧЕСКАЯ РАБОТА № 9

Тема: Изучение и комментирование кода

ПРАКТИЧЕСКАЯ РАБОТА № 10

Тема: Изменение элементов дизайна

ПРАКТИЧЕСКАЯ РАБОТА № 11

Тема: Изменение элементов дизайна

ПРАКТИЧЕСКАЯ РАБОТА № 12

Тема: Изменение элементов дизайна

ПРАКТИЧЕСКАЯ РАБОТА № 13

Тема: Обработка событий: цветовая индикация

ПРАКТИЧЕСКАЯ РАБОТА № 14

Тема: Обработка событий: цветовая индикация

ПРАКТИЧЕСКАЯ РАБОТА № 15

Тема: Обработка событий: цветовая индикация

ПРАКТИЧЕСКАЯ РАБОТА № 16

Тема: Обработка событий: подсказки

ПРАКТИЧЕСКАЯ РАБОТА № 17

Тема: Обработка событий: подсказки

ПРАКТИЧЕСКАЯ РАБОТА № 18

Тема: Обработка событий: подсказки

ПРАКТИЧЕСКАЯ РАБОТА № 19

Тема: Подготовка стандартных модулей

ПРАКТИЧЕСКАЯ РАБОТА № 20

Тема: Подготовка стандартных модулей

ПРАКТИЧЕСКАЯ РАБОТА № 21

Тема: Обработка событий: переключение между экранами

ПРАКТИЧЕСКАЯ РАБОТА № 22

Тема: Обработка событий: переключение между экранами

ПРАКТИЧЕСКАЯ РАБОТА № 23

Тема: Передача данных между модулями

ПРАКТИЧЕСКАЯ РАБОТА № 24

Тема: Тестирование и оптимизация мобильного приложения

ПРАКТИЧЕСКАЯ РАБОТА № 25

Тема: Тестирование и оптимизация мобильного приложения

ПРАКТИЧЕСКАЯ РАБОТА № 26

Тема: Тестирование и оптимизация мобильного приложения

ИНСТРУКЦИИ ДЛЯ ОБУЧАЮЩИХСЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

Прежде чем приступить к выполнению заданий, внимательно прочитайте данные рекомендации.

В ходе выполнения практических работ студент должен:

- выполнять требования по охране труда
- соблюдать инструкцию по правилам и мерам безопасности в кабинете информационных технологий
- строго выполнять весь объем работы, указанный в задании
- соблюдать требования эксплуатации компьютерной техники (правила включения и выключения)
- предоставить отчет о проделанной работе по окончании выполненной работы, который должен содержать:

1. Название работы.
2. Цель работы.
3. Задание и его решение.
4. Вывод о проделанной работе.

Текст отчета по практической работе должен быть набран на компьютере шрифтом Times New Roman размером 14 пт. (при оформлении текста используется текстовый редактор MS Word). Шрифт, используемый в иллюстративном материале (таблицы и рисунки), рекомендуется уменьшить до 12 пт. Межстрочный интервал в основном тексте - полуторный. В иллюстративном материале межстрочный интервал рекомендуется сделать одинарным. Поля страницы должны быть: левое поле - 30 мм; правое поле – 15 мм; верхнее и нижнее поле - 20 мм.

Каждый абзац должен начинаться с красной строки. Отступ абзаца – 1,25 см от левой границы текста.

Студент должен выполнить практическую работу самостоятельно (или в группе, если это предусмотрено заданием). Практическая работа выполняется согласно заданию и методическим рекомендациям. Результат работы представляется преподавателю в виде файла (файлов) в личном каталоге, защищается обучающимися.

По ходу выполнения работы при возникновении вопросов обучающийся может получить консультацию у преподавателя или самостоятельно воспользоваться лекционным материалом, рекомендуемой литературой.

1. Составление программы на языке программирования

Правила оформления кода:

1. Используйте разумные имена для переменных и функций

Программа должна быть хорошо понятна человеку при чтении. Если при чтении программы приходится понимать назначение переменных и функций по тому, как они используются, то читать код становится гораздо сложнее. Неудачно выбранные имена могут привести к тому, что смысл программы может быть неправильно понят. Выбирайте такие имена, которые бы объясняли смысл переменных и функций, тогда код станет гораздо понятнее, и не потребуется писать множество комментариев.

При написании составных слов, например в именах переменных, пишите их слитно без пробелов, при этом каждое новое слово пишется с большой буквы.

2. Не дублируйте код. Если в программе есть одинаковые выражения или фрагмента кода, вынесите этот код в отдельную функцию. Верным признаком необходимости создания новой функции является желание скопировать фрагмент кода из одного места программы в другое. В таком случае сразу перенесите этот фрагмент в отдельную функцию.

Если фрагменты кода похожи, но не идентичны, подумайте, не получится ли и их вынести в одну функцию, возможно добавив параметры или условия.

3. Не используйте «магические константы». Использование неименованных «магических» констант в коде нежелательно:

- при чтении кода может быть не понятно, что это за число, и почему оно именно такое;
- чаще всего одно и то же число потребуется написать в нескольких местах кода. Если его придётся изменять, можно пропустить одно из использований, что приведёт к ошибке.

Если в коде нужно использовать константу, дайте ей имя, используя `const`.

4. Расставляйте пробелы вокруг бинарных операторов. Это улучшает читаемость формул.

5. Всегда выделяйте блоки условных операторов и циклов скобками. В любой блок условия или цикла может захотеться добавить новое выражение. При этом можно забыть добавить скобки. Лучше сразу добавить скобки, чтобы потом не было с этим проблем.

6. Расставляйте скобки одинаково. Выберите и используйте для себя один из стилей расстановки скобок. Это улучшает читаемость структуры программы.

7. Не делайте строки слишком длинными. Строка программы должна помещаться на экране. Обычно рекомендуют ограничить максимальную ширину строки в 80 символов. Если определение или вызов функции получается слишком широким поместите по одному параметру на каждой строке. Длинные математические выражения разбивайте на несколько строк, разбивая по границам логических блоков выражения.

8. Объявляйте переменные непосредственно перед использованием. Обязательно указывайте начальное значение для переменных. Значение неинициализированной переменной может быть любым. Использование (чтение) такого значения приведёт к недетерминированной работе программы, а в некоторых случаях является неопределённым поведением. Чтобы избежать проблем всегда инициализируйте переменные прямо в момент их создания.

9. Единый стиль оформления кода во всем проекте;

10. Визуальное выделение наиболее значимых частей — используя *вертикальное форматирование*, мы выделяем объявление переменных, цикл заполнения массива случайными числами и цикл обработки по формуле. Если ваша функция выполняет несколько действий — то разумно разделить соответствующие блоки кода пустыми строками.

ПРАКТИЧЕСКАЯ РАБОТА № 1

Тема: Создание эмуляторов и подключение устройств

Цель работы: научиться создавать и подключать устройства для мобильных приложений

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

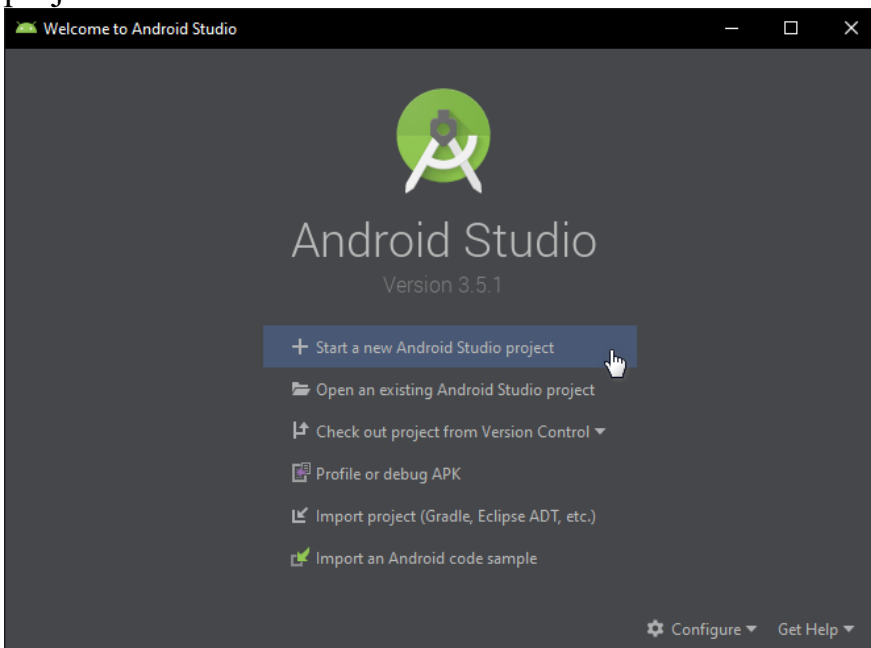
Справочный материал:

Эмуляция (англ. emulation) в вычислительной технике – комплекс программных, аппаратных средств или их сочетание, предназначенное для копирования (или эмулирования) функций одной вычислительной системы (гостя) на другой, отличной от первой, вычислительной системе (хосте) таким образом, чтобы эмулированное поведение как можно ближе соответствовало поведению оригинальной системы (гостя). Целью является максимально точное воспроизведение поведения в отличие от разных форм компьютерного моделирования, в которых имитируется поведение некоторой абстрактной модели

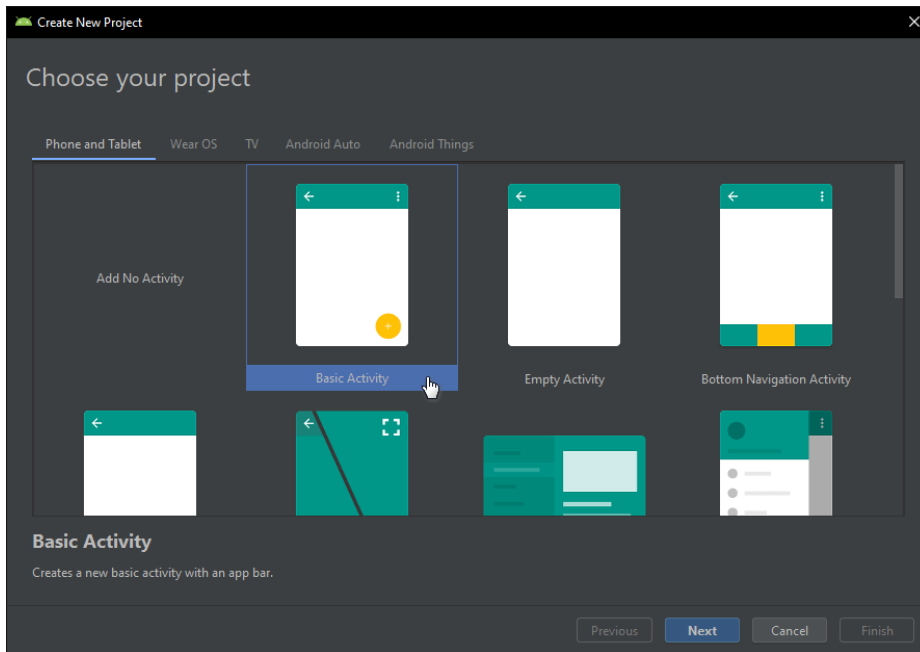
Содержание работы:

Задание 1. Подготовить эмулятор Android Studio

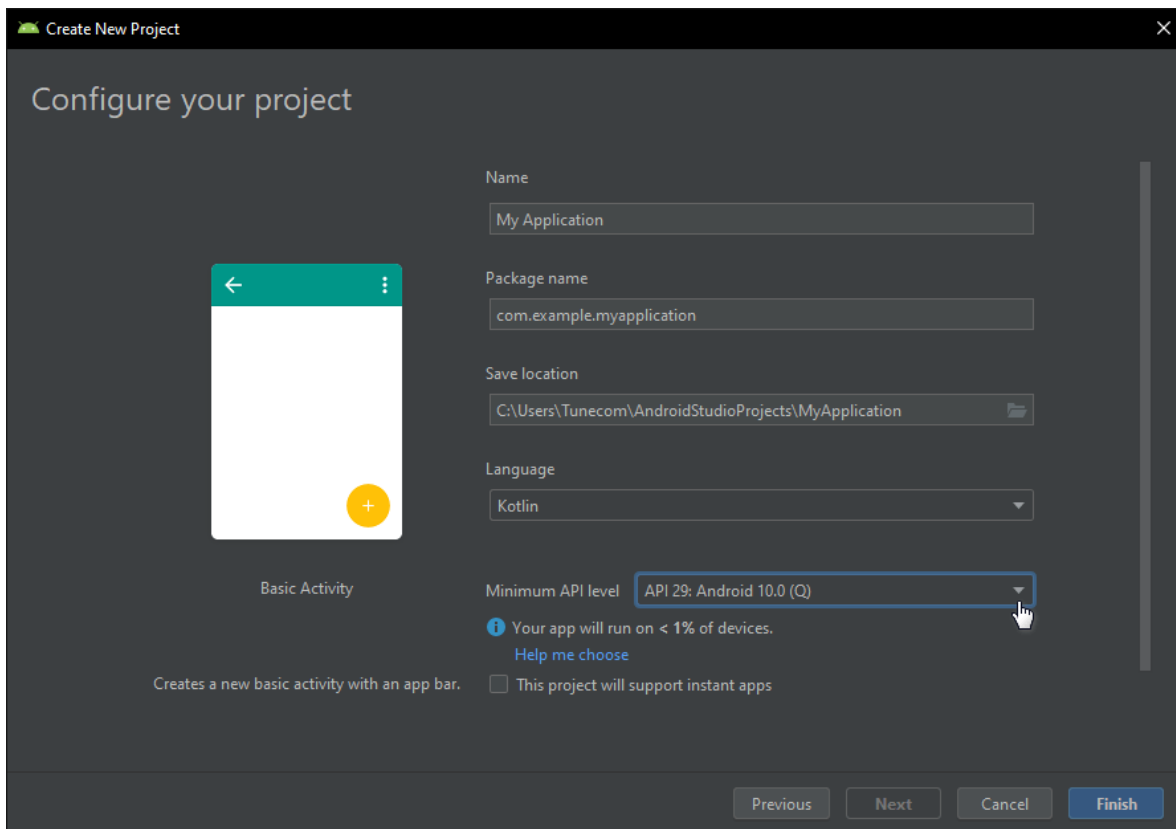
1. В окне "Welcome to Android Studio" нажмите "Start a new Android Studio project".



2. Выберите "Basic Activity".



3. Задайте "Версию Android", мы выбрали последнюю Android 10 (Q) и нажимайте "Finish".



ПРАКТИЧЕСКАЯ РАБОТА № 2

Тема: Создание эмуляторов и подключение устройств

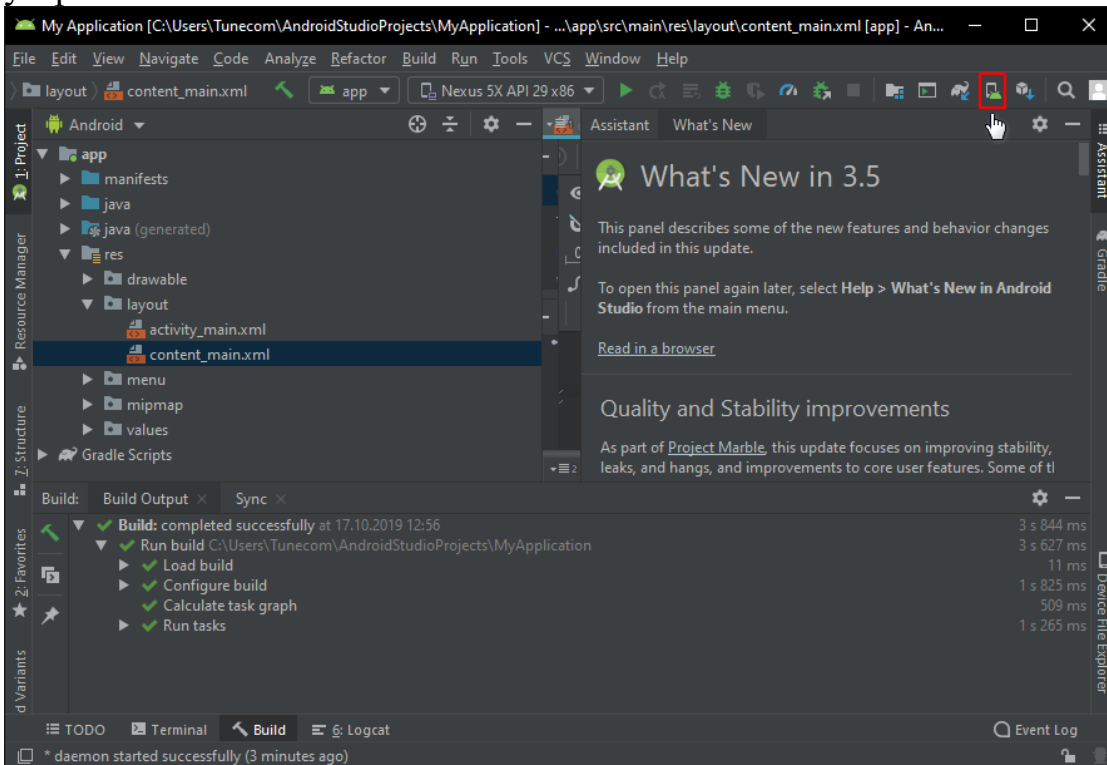
Цель работы: научиться настраивать и подключать эмулятор Android Studio

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

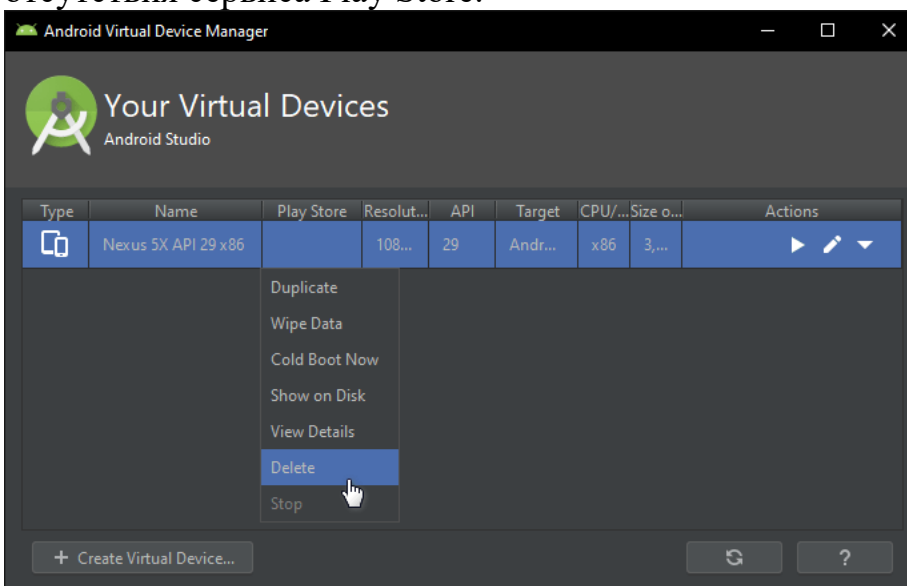
Содержание работы:

Задание 1. Настроить и запустить эмулятор Android Studio

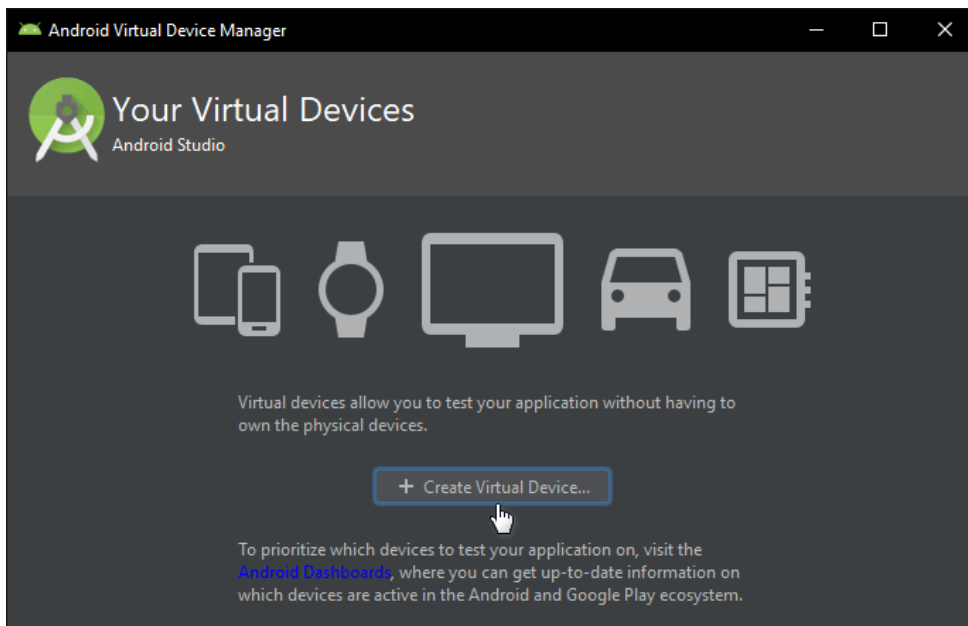
1. Запустите Android Studio и нажмите кнопку менеджера виртуального Android-устройства.



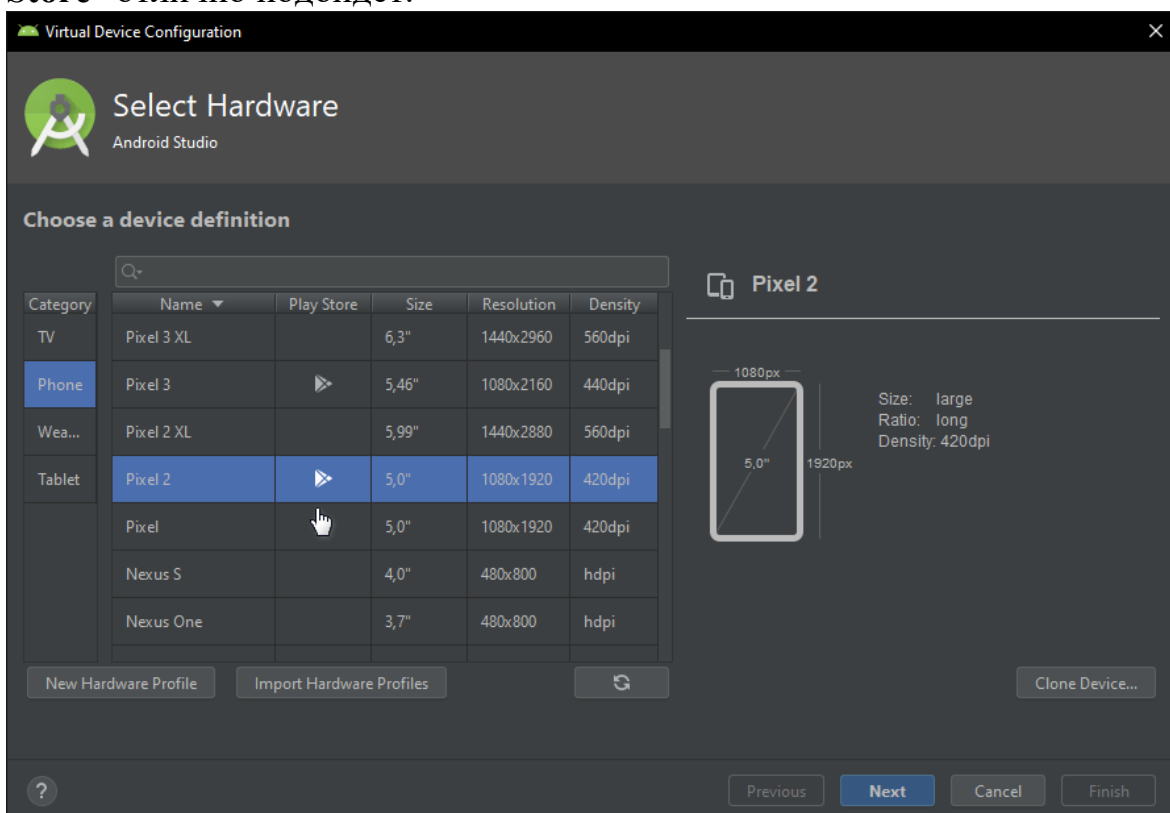
2. В нем будет готовое устройство "Nexus 5 X" которое можно удалить из-за отсутствия сервиса Play Store.



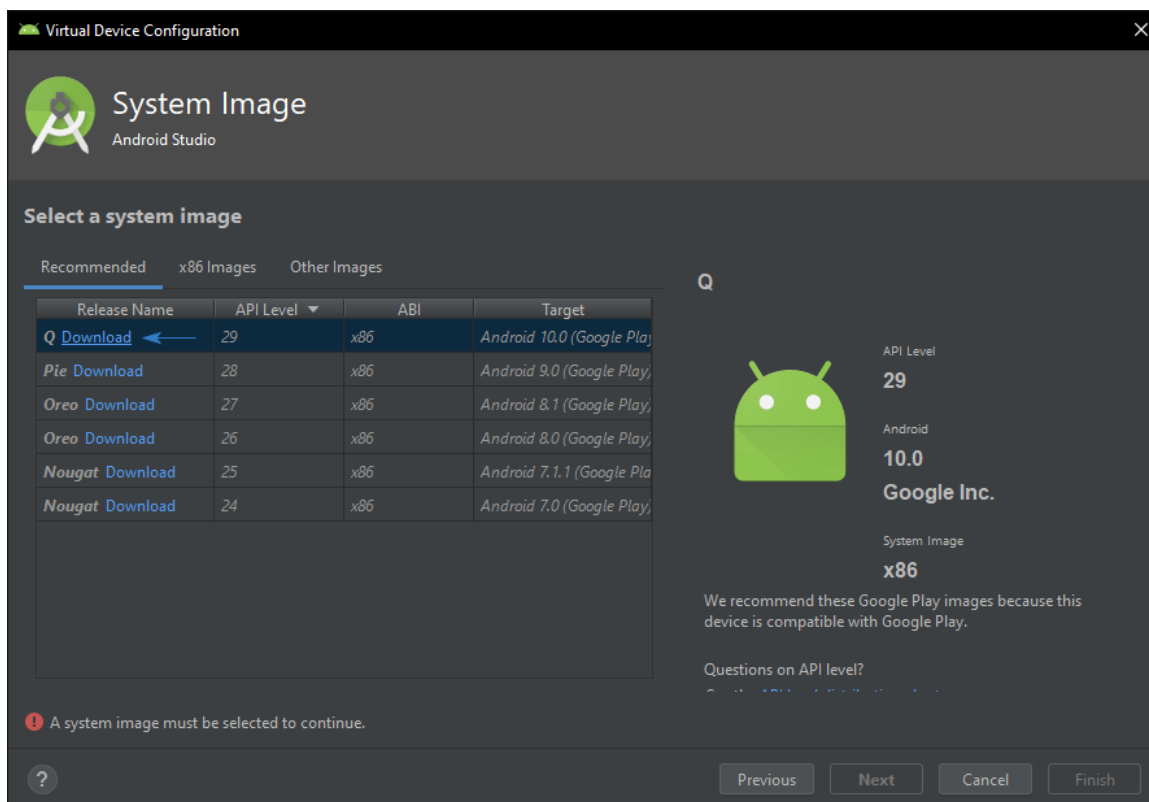
3. После чего, создадим своё нажав "Create Virtual Device".



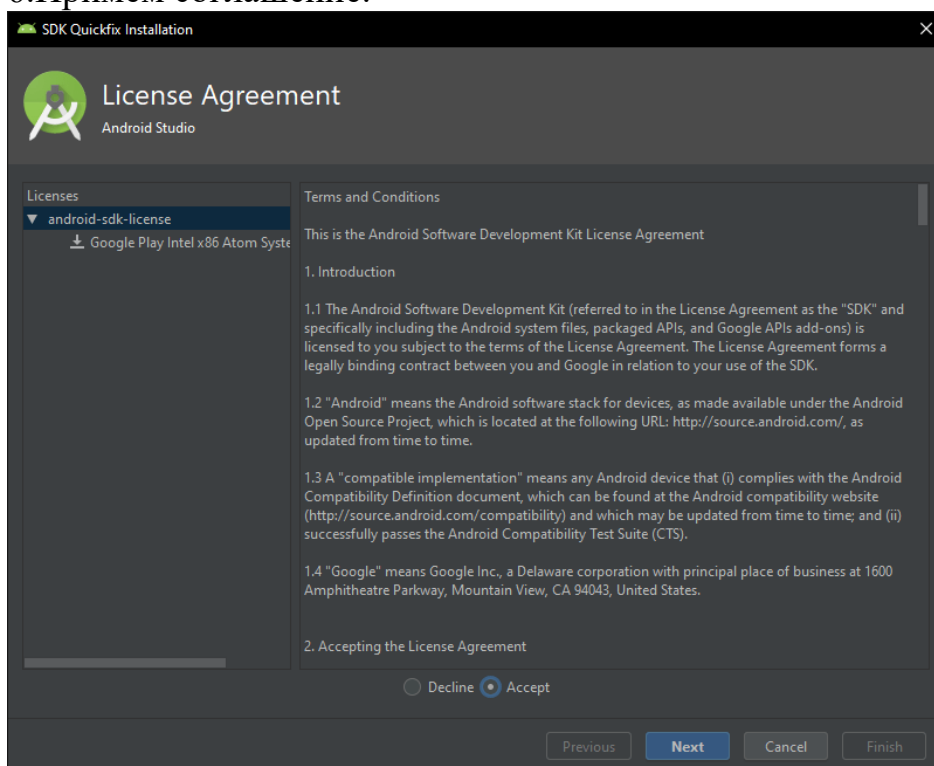
4. Отметим категорию устройства, это может быть смартфон, планшет либо другое. Мы выберем **"Phone"** и эмулируем смартфон, **"Pixel 2"** со значком **"Play Store"** отлично подойдет.



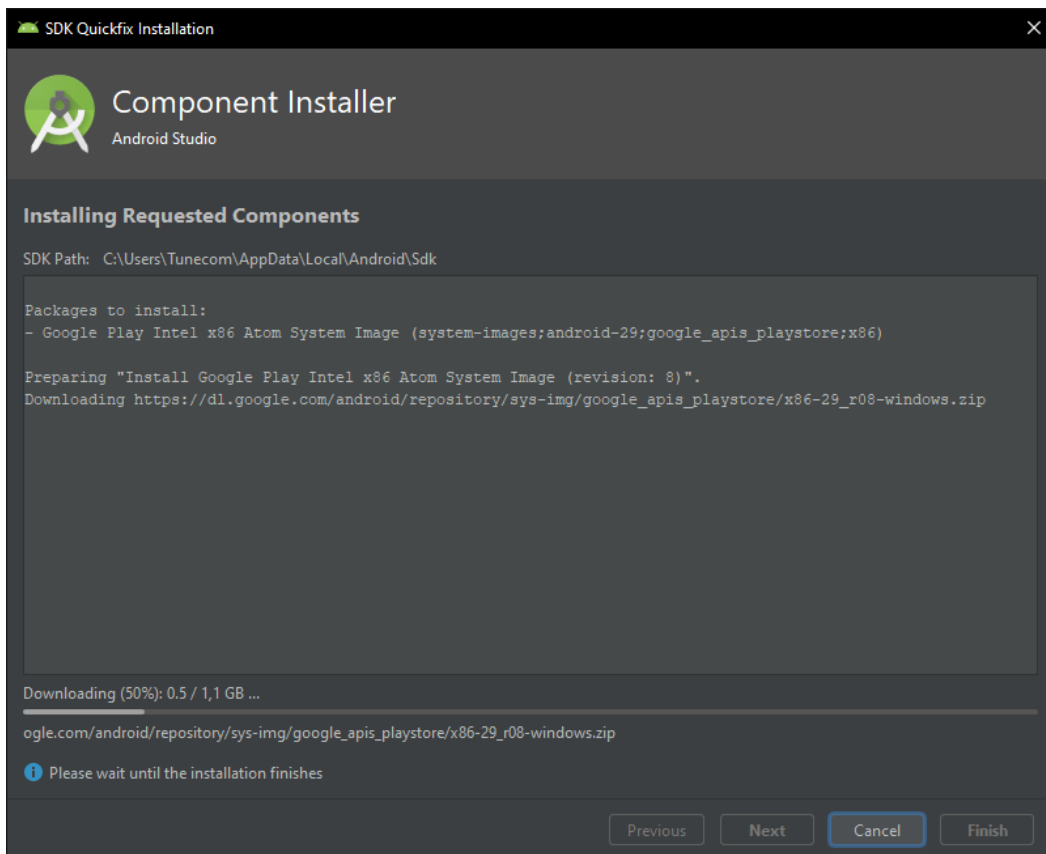
5. Теперь скачаем операционную систему Android 10 с сервисом Google Play нажав **"Download"**.



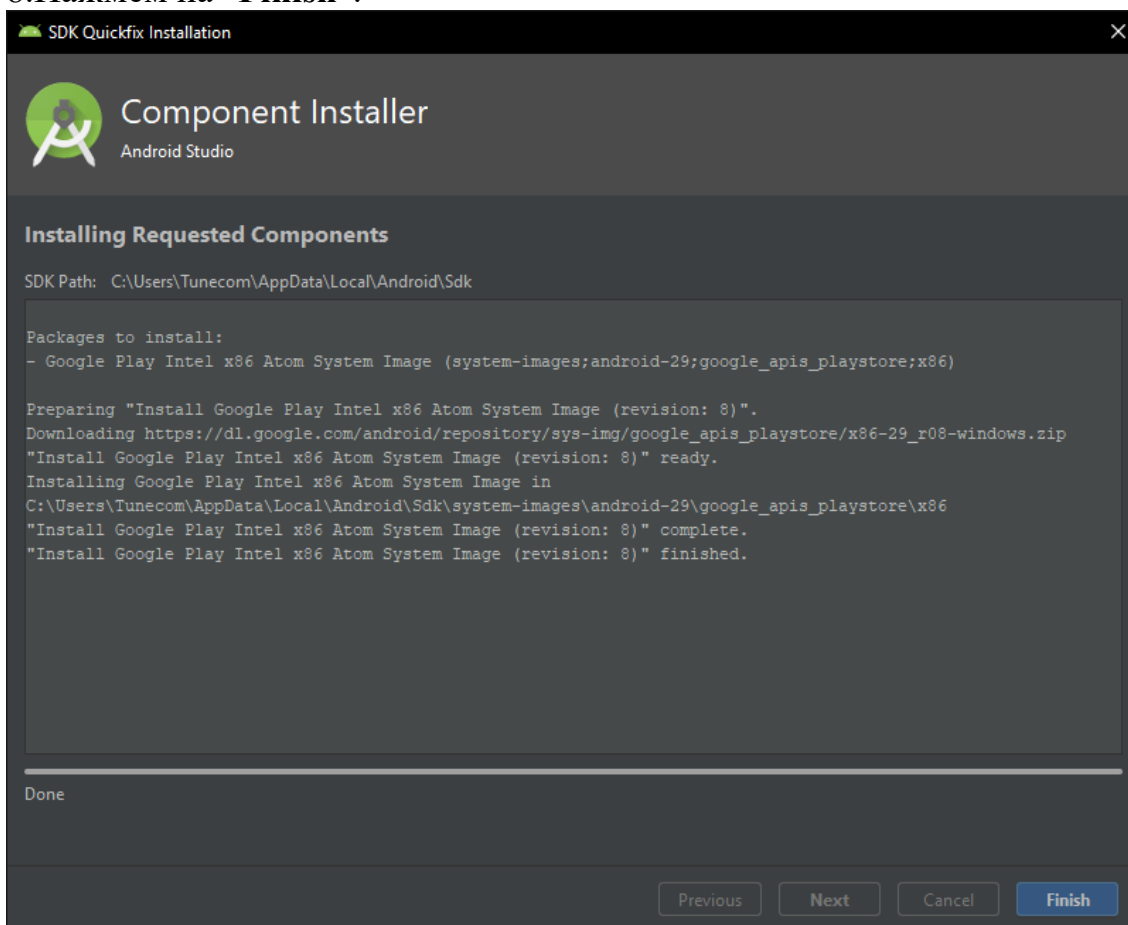
6.Примем соглашение.



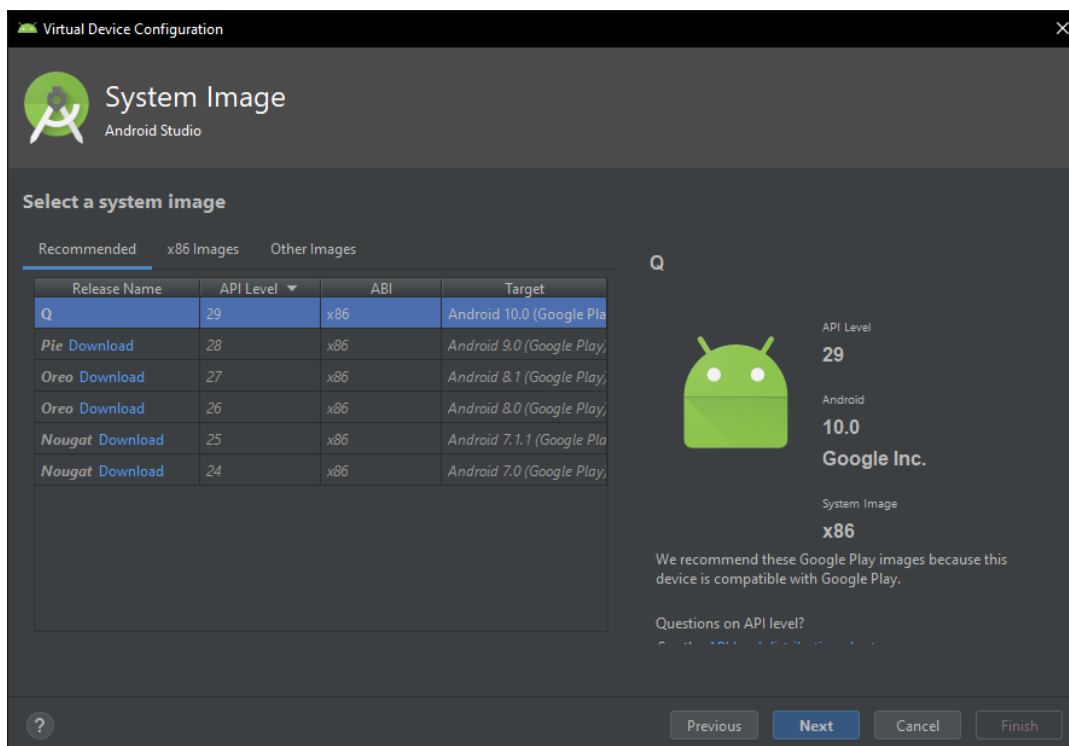
7.Дождемся скачивания и распаковки системы.



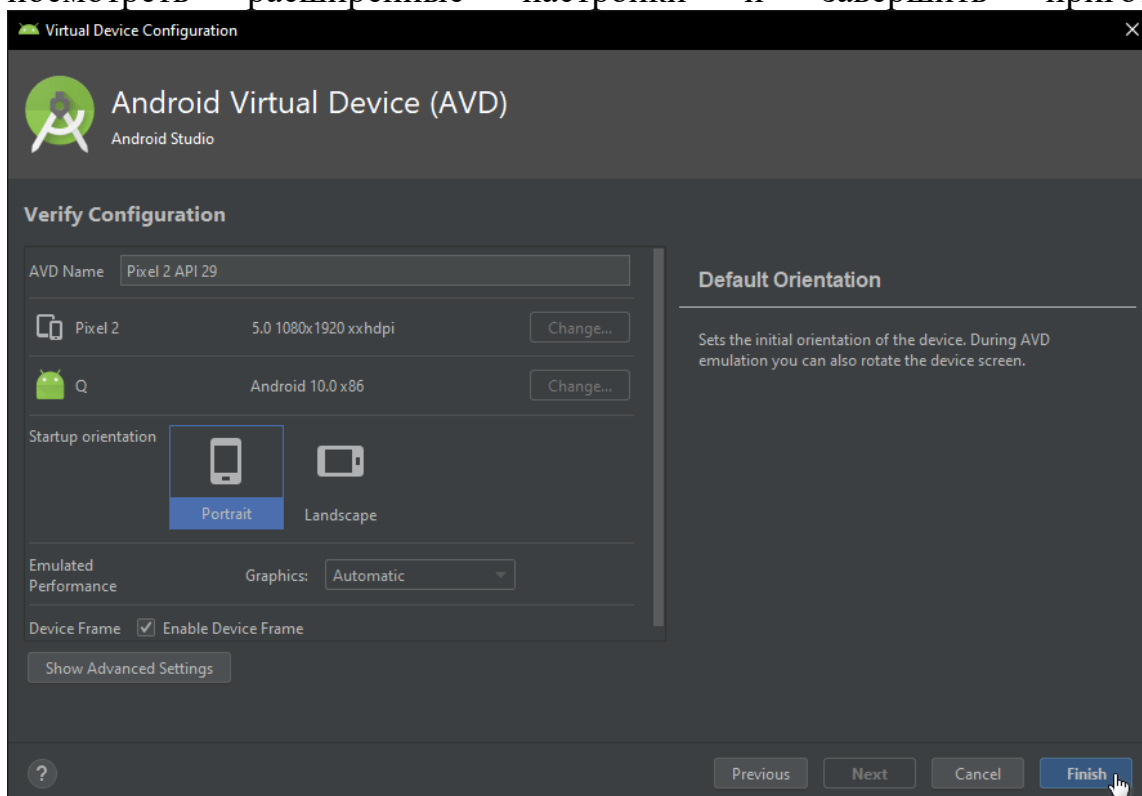
8.Нажмем на **"Finish"**.



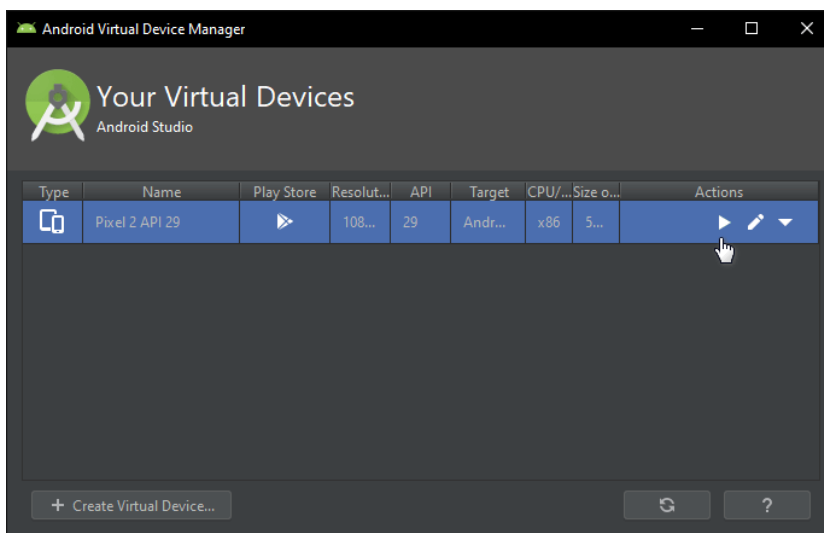
9.И пойдём дальше.



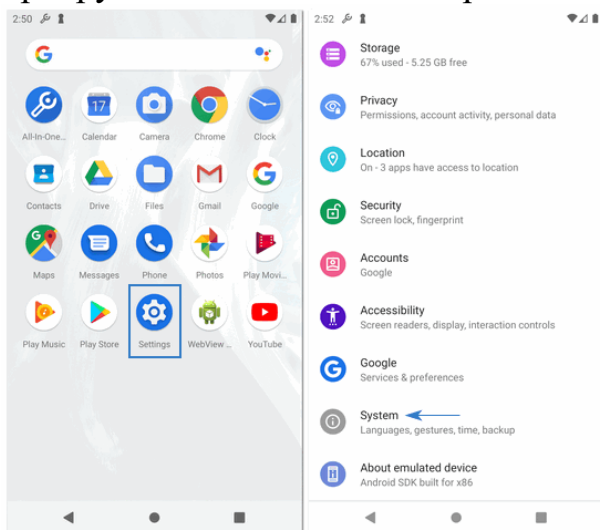
10.3. Здесь можно задать новое имя устройству, выбрать положение экрана, посмотреть расширенные настройки и завершить приготовления.



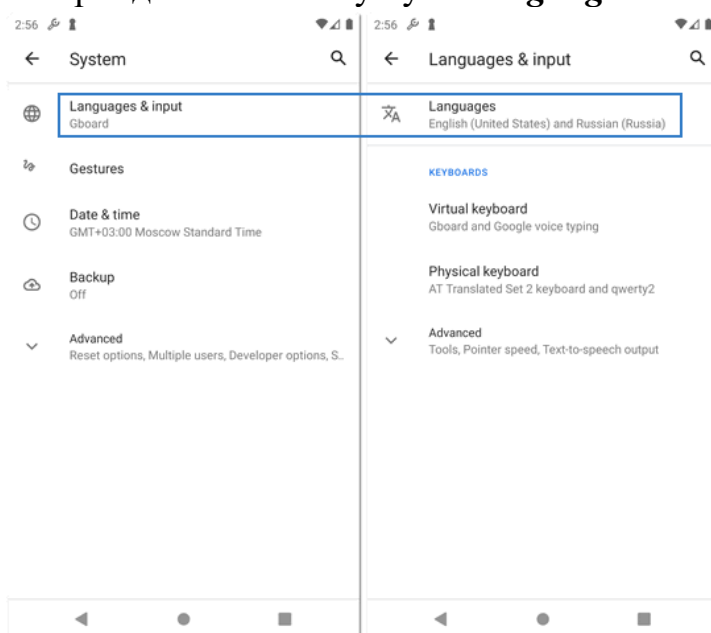
11. После чего, нажать кнопку запуска эмулятора Android и дождаться загрузки системы.



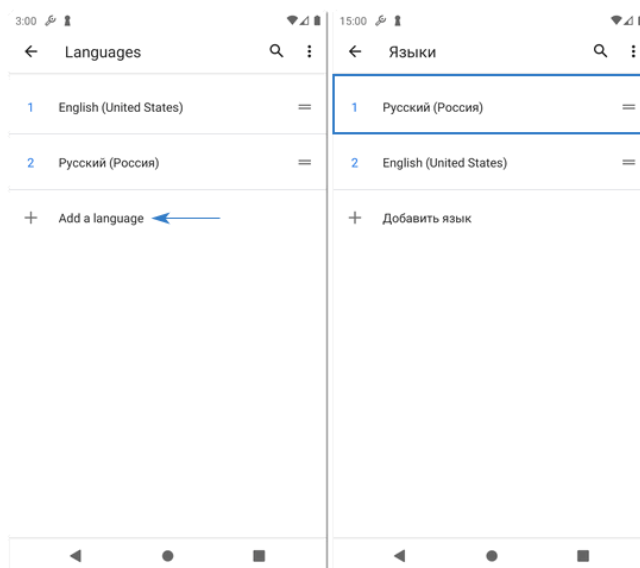
12. Когда устройство будет загружено, перейдите в настройки Android, прокрутите вниз и зайдите в раздел "System".



13. Пройдите по такому пути *Languages & input* > *Languages*.



14.Добавьте "Русский язык" через кнопку "Add a Language", поставьте "Русский" первым языком.



ПРАКТИЧЕСКАЯ РАБОТА № 3

Тема: Настройка режима терминала

Цель работы: получить практические навыки по настройке режима терминала.

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Изучить команды эмулятора терминала в системе Android

1. Достаточно скачать эмулятор терминала (Terminal Emulator for Android) и начать вводить команды, а если вам мало покажется встроенных команд Android скачайте приложение Busy Box, которое добавляет набор команд для контакта с файловой системой Android.

2. Для некоторых из команд нужны права Root.

3. Все описанные здесь команды были протестированы и полностью выполняют свои функции.

adb (Android Debug Bridge) - утилита для отладки Андроид устройств с ПК

pm - менеджер пакетов

pm list packages (смотрим список установленных пакетов)

am – менеджер для запуска и остановки приложений

ls - выводит содержимое текущей папки

ls -l (флаг -l выводит расширенную информацию о файлах)

cd - переход к нужной директории

cd /sdcard/Download (Переход в папку Download)

date - выводит дату

service - управление сервисами

service list (выводит список работающих сервисов)

df - показывает размер объектов в указанной папке

mkdir - создаем папку

mkdir test (создаем папку test)

rmdir - удаляем папку

rmdir test (удаляем папку test)

touch - создаем файл

touch file_test (создаем файл file_test)

rm - удаляем файл

rm file_test (удаляем файл file_test)

cp - копируем файл

cp /sdcard/Download/file_test /sdcard/ (копирует файл file_test из каталога Download в sdcard)

mv - перемещаем файл

du - показывает объем файла

netcfg - просмотр информации о сетевых соединениях

netstat - статистика сетевых подключений

iftop - предоставляет информацию об активных сетевых соединениях

ping - просмотр доступности сетевого узла

iptables - управляет работой межсетевого экрана

ps - просмотр всех запущенных процессов
su - команда позволит перейти в разряд суперпользователя (Root)
mv - переименовывает файл
cal - выводит календарь текущего месяца
uptime - сколько времени работает операционная система после последней перезагрузки
free - показывает использование памяти
pwd - выводит текущий путь
vi - текстовый редактор
history - показывает историю введенных команд
kill - уничтожить процесс
id - выводит идентификатор пользователя и группы
top - список запущенных процессов
reboot - перезагрузка устройства (права Root)
clear - очищает окно терминала
uname - информация о системе
uname -a (флаг -a позволяет узнать версию ядра Linux)
hostname - изменить или вывести имя текущего хоста
dd - утилита для копирования данных
wget - команда для скачивания файлов
iostat - показывает нагрузку процессора
mpstat - статистика использования процессоров в системе
ip - настройка сети
ip address show (показывает все ip адреса и интерфейсы)
lsmod - выводит список загруженных модулей ядра
ed - текстовый редактор
iw - настройка WiFi сети

ПРАКТИЧЕСКАЯ РАБОТА № 4

Тема: Создание нового проекта

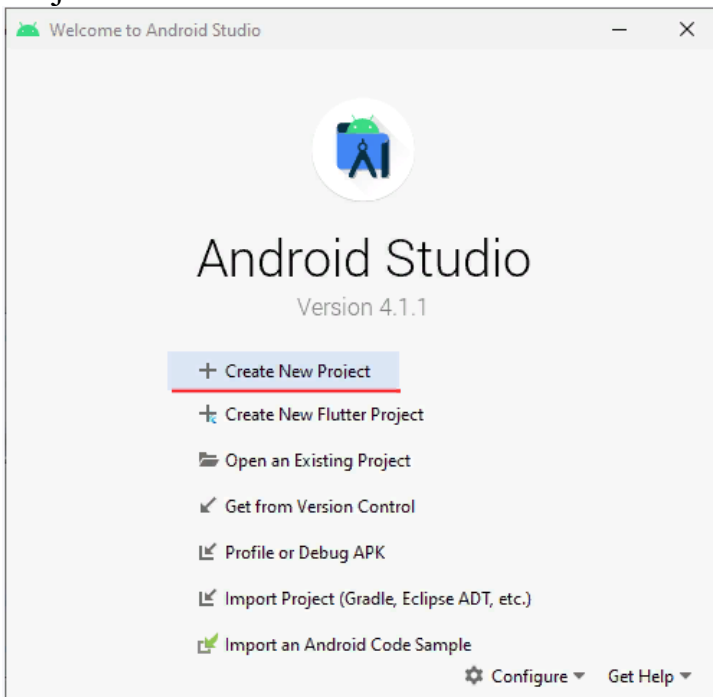
Цель работы: получить практические навыки по созданию приложения в Android Studio.

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

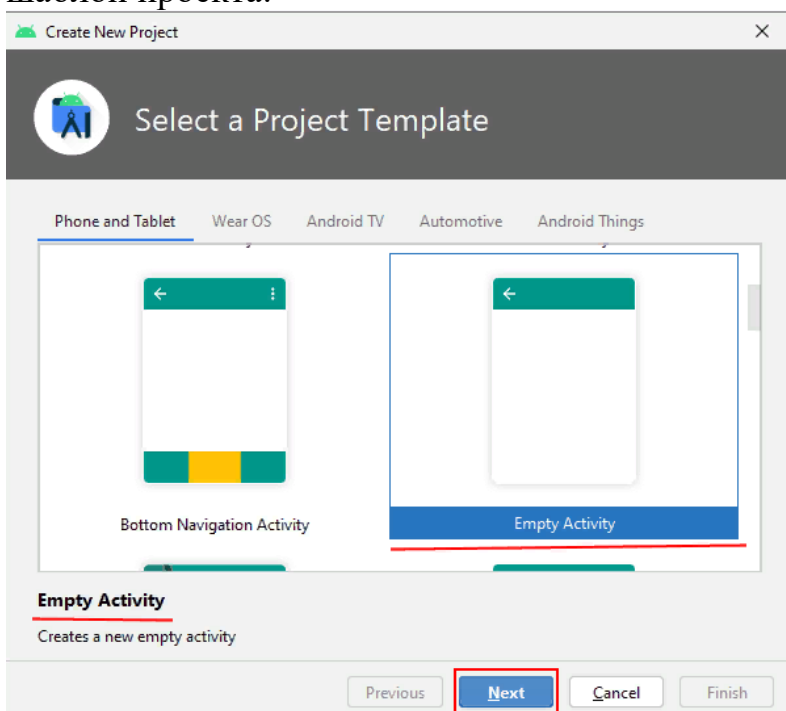
Содержание работы:

Задание 1. Создать новый проект в среде Android Studio.

1. Откроем Android Studio и на начальном экране выберем пункт Create New Project:

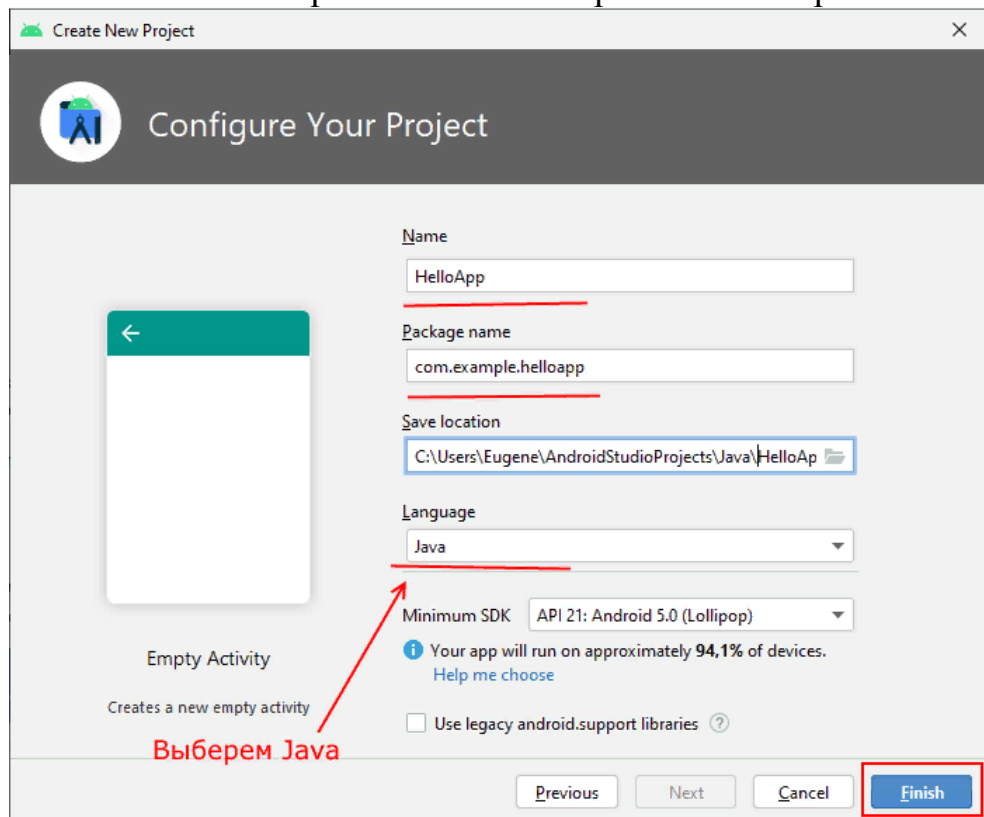


2. При создании проекта Android Studio вначале предложит нам выбрать шаблон проекта:



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.

3. После этого отобразится окно настроек нового проекта:

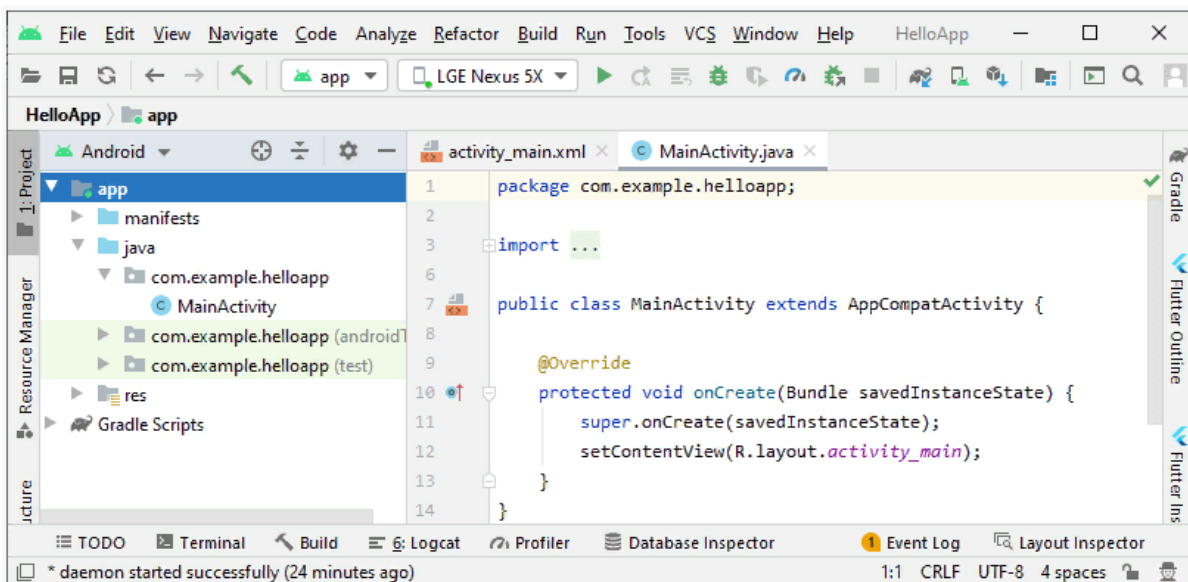


4. В окне создания нового проекта мы можем установить его начальные настройки:

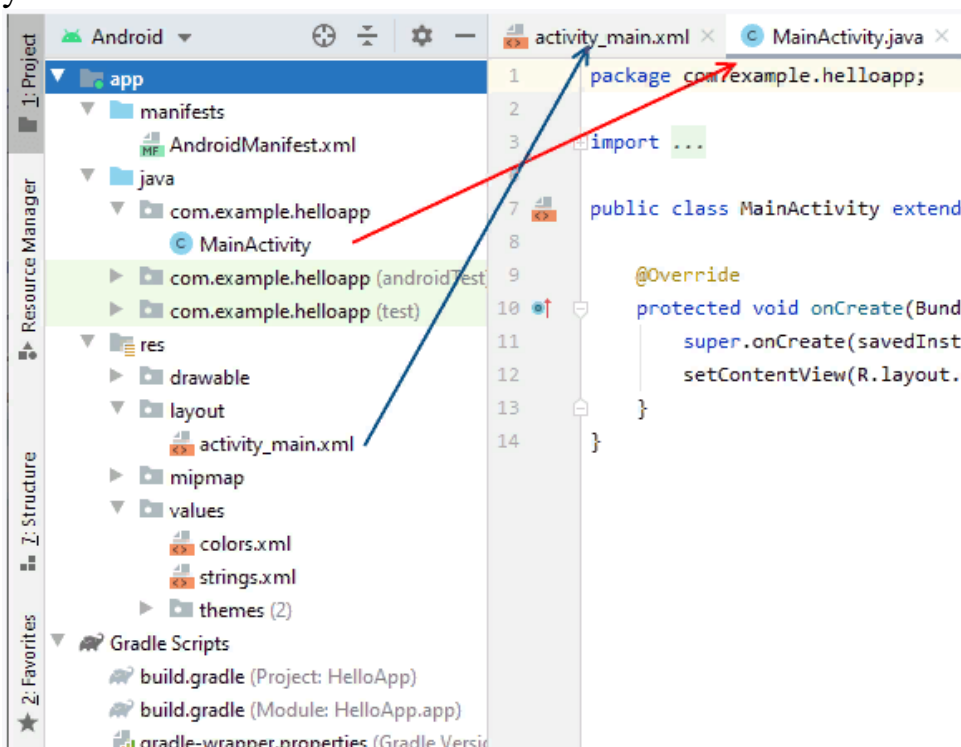
- В поле Name вводится название приложения. Укажем в качестве имени название HelloApp
- В поле Package Name указывается имя пакета, где будет размещаться главный класс приложения. В данном случае для тестовых проектов это значение не играет большого значения, поэтому установим com.example.helloapp.
- В поле Save Location устанавливается расположение файлов проекта на жестком диске. Можно оставить значение по умолчанию.
- В поле Language в качестве языка программирования укажем Java (будьте 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.

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



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



7. 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.android.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 `miramar` 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.

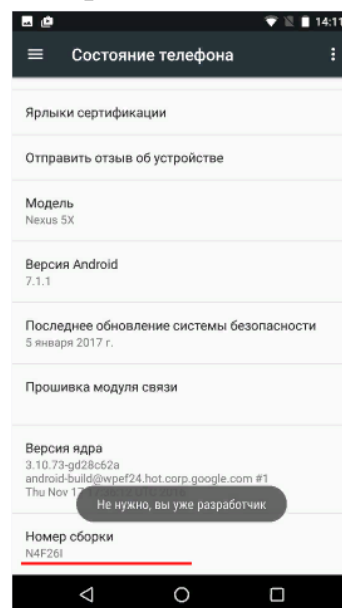
8. Запуск проекта

Стандартный проект, который был создан в прошлой теме, который уже содержит некоторый примитивный функционал. Правда, этот функционал почти ничего не делает, только выводит на экран строку "Hello world!".

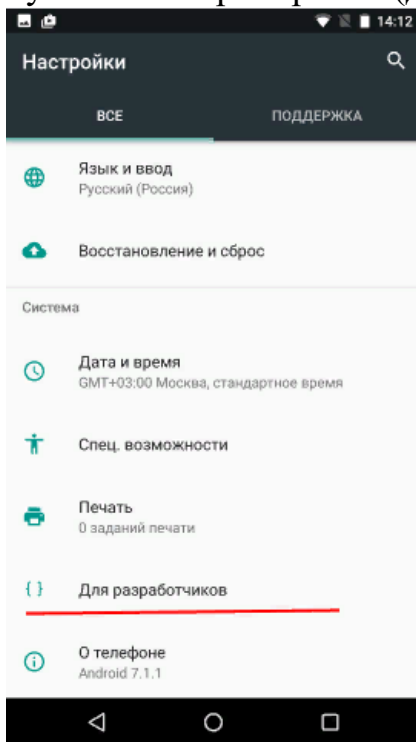
Для запуска и тестирования приложения мы можем использовать эмуляторы или реальные устройства. Но в идеале лучше тестировать на реальных устройствах. К тому же эмуляторы требуют больших аппаратных ресурсов, и не каждый компьютер может потянуть требования эмуляторов. А для использования мобильного устройства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

9. Режим разработчика на телефоне

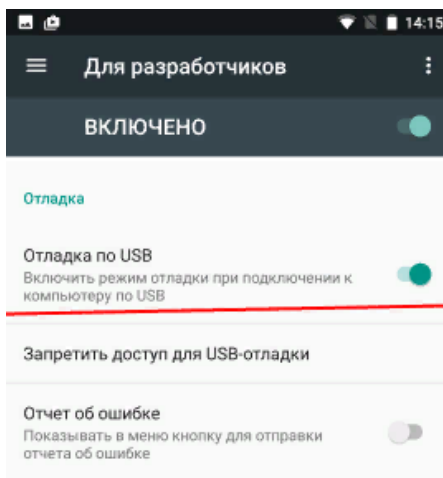
По умолчанию опции разработчика на смартфонах скрыты. Чтобы сделать их доступными, надо зайти в `Settings > About phone` (Настройки > О телефоне) (в `Android 8` это в `Settings > System > About phone` (Настройки > Система > О телефоне)) и семь раз нажать `Build Number` (Номер сборки).



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

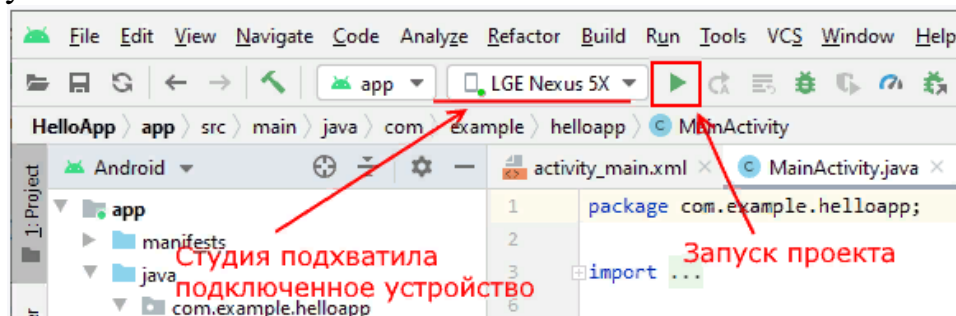


11. Перейдем к пункту Для разработчиков и включим возможность отладки по USB:



12. Запуск приложения

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



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



Hello World!

Задание 2. Разработать мобильное приложение по индивидуальным заданиям.

ПРАКТИЧЕСКАЯ РАБОТА № 5

Тема: Создание нового проекта

Цель работы: получить практические навыки при разработке графического интерфейса мобильного приложения.

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Разработать графический интерфейс, созданного в Практической работе № 4 мобильного приложения.

1. Запустить приложение, созданное в Практической работе № 4. Выполнение приложения Android по умолчанию начинается с класса MainActivity, который по умолчанию открыт в Android Studio:

```
package com.android.helloapp;  
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);    } }
```

2. Каждый отдельный экран или страница в приложении описывается таким понятием как activity. Так вот, если мы запустим приложение на устройстве, то на экране по сути увидим определенную activity, которая представляет данный интерфейс.

3. Класс MainActivity по сути представляет обычный класс java, в начале которого идет определение пакета данного класса:

```
package com.android.helloapp;
```

4. Далее идет импорт классов из других пакетов, функциональность которых используется в MainActivity:

```
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;
```

5. Затем идет собственно определение класса:

```
public class MainActivity extends AppCompatActivity
```

6. По умолчанию MainActivity наследуется от класса AppCompatActivity, который выше подключен с помощью директивы импорта. Класс AppCompatActivity по сути представляет отдельный экран (страницу) приложения или его визуальный интерфейс. И MainActivity наследует весь этот функционал.

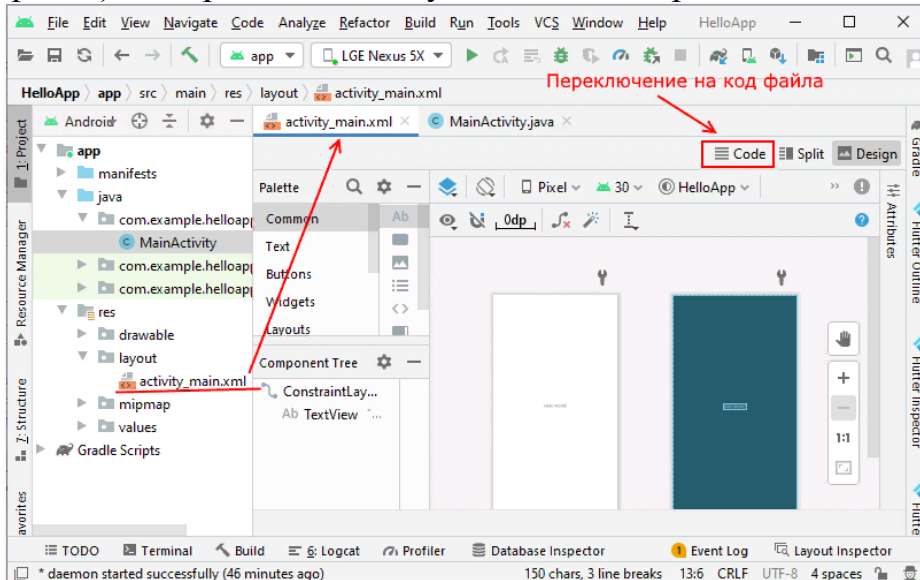
7. По умолчанию MainActivity содержит только один метод onCreate(), в котором фактически и создается весь интерфейс приложения:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);    }
```

8. В метод setContentView() передается ресурс разметки графического интерфейса:

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

9. Именно здесь и решается, какой именно визуальный интерфейс будет иметь MainActivity. Но что в данном случае представляет ресурс R.layout.activity_main? Это файл activity_main.xml из папки res/layout (в принципе можно заметить, что название ресурса соответствует названию файла), который также по умолчанию открыт в Android Studio:

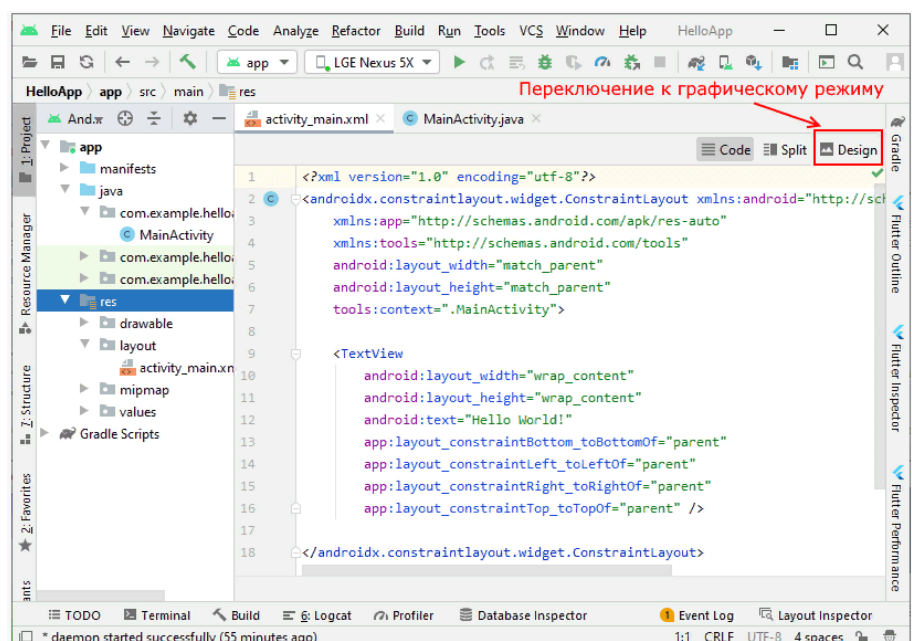


10. Файл activity_main.xml

Android Studio позволяет работать с визуальным интерфейсом как в режиме кода, так и в графическом режиме. Так, по умолчанию файл открыт в графическом режиме, и мы наглядно можем увидеть, как у нас примерно будет выглядеть экран приложения. И даже набросать с панели инструментов какие-нибудь элементы управления, например, кнопки или текстовые поля.

Но также мы можем работать с файлом в режиме кода, поскольку activity_main.xml - это обычный текстовый файл с разметкой xml. Для переключения к коду нажмем на кнопку Code над графическим представлением.

(Дополнительно с помощью кнопки Split можно переключиться на комбинированное представление код + графический дизайнер)



11. Здесь мы увидим, что на уровне кода файл activity_main.xml содержит следующую разметку:

```
<?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">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

12. Весь интерфейс представлен элементом-контейнером androidx.constraintlayout.widget.ConstraintLayout:

```
<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">
```

13. ConstraintLayout позволяет расположить вложенные элементы в определенных местах экрана. Вначале элемента ConstraintLayout идет определение пространств имен XML:

```
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"
```

14. Каждое пространство имен задается следующим образом: xmlns:префикс="название_ресурса". Например, в

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

15. Название ресурса (или URI - Uniform Resource Indicator) - "http://schemas.android.com/apk/res/android". И этот ресурс сопоставляется с префиксом android (xmlns:android).

Каждый ресурс или URI определяет некоторую функциональность, которая используется в приложении, например, предоставляют теги и атрибуты, которые необходимы для построения приложения.

– xmlns:android="http://schemas.android.com/apk/res/android": содержит основные атрибуты, которые предоставляются платформой Android,

применяются в элементах управления и определяют их визуальные свойства (например, размер, позиционирование)

- `xmlns:app="http://schemas.android.com/apk/res-auto"`: содержит атрибуты, которые определены в рамках приложения

- `xmlns:tools="http://schemas.android.com/tools"`: применяется для работы с режиме дизайнера в Android Studio

16. И чтобы упростить работу с этими ресурсами, применяются префиксы. Например, дальше мы видим:

```
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
```

`android:layout_width` определяет ширину контейнера. Этот атрибут (`layout_width`) расположен в ресурсе `"http://schemas.android.com/apk/res/android"`. И поскольку этот ресурс сопоставляется с префиксом `android`, то для обращения к атрибуту перед ним через двоеточие указывается префикс данного ресурса.

Значением атрибута `android:layout_weight` является `"match_parent"`. Это значит, что элемент (`ConstraintLayout`) будет растягиваться по всей ширине контейнера (экрана устройства).

Атрибут `android:layout_height="match_parent"` определяет высоту контейнера и также определен в `"http://schemas.android.com/apk/res/android"`. Значение `"match_parent"` указывает, что `ConstraintLayout` будет растягиваться по всей длине контейнера (экрана устройства).

Атрибут `tools:context` определяет, какой класс `activity` (экрана приложения) связан с текущим определением интерфейса. В данном случае это класс `MainActivity`. Это позволяет использовать в Android Studio различные возможности в режиме дизайнера, которые зависят от класса `activity`.

17. `TextView`. Текстовое поле устанавливает текст с помощью атрибута `android:text`.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

`android:layout_width` устанавливает ширину виджета. Значение `wrap_content` задает для виджета величину, достаточную для отображения в контейнере.

`android:layout_height` устанавливает высоту виджета. Значение `wrap_content` аналогично установке ширины задает для виджета высоту, достаточную для отображения в контейнере

`android:text` устанавливает текст, который будет выводиться в `TextView` (в данном случае это строка `"Hello World!"`)

`app:layout_constraintLeft_toLeftOf="parent"`: указывает, что левая граница элемента будет выравниваться по левой стороне контейнера `ConstraintLayout`. Обратите внимание, что этот атрибут определен в пространстве имен с префиксом `app`, то есть в `"http://schemas.android.com/apk/res-auto"`.

`app:layout_constraintTop_toTopOf="parent"`: указывает, что верхняя граница элемента будет выравниваться по верхней стороне контейнера `ConstraintLayout`.

`app:layout_constraintRight_toRightOf="parent"`: указывает, что правая граница элемента будет выравниваться по правой стороне контейнера `ConstraintLayout`.

`app:layout_constraintBottom_toBottomOf="parent"`: указывает, что нижняя граница элемента будет выравниваться по нижней стороне контейнера `ConstraintLayout`.

Стоит отметить, что последние четыре атрибута вместе будут приводить к расположению `TextView` по центру экрана.

Таким образом, при запуске приложения сначала запускается класс `MainActivity`, который в качестве графического интерфейса устанавливает разметку из файла `activity_main.xml`. И поскольку в этой разметке прописан элемент `TextView`, который представляет некоторый текст, то мы и увидим его текст на экране смартфона.

Задание 2. Разработать графический интерфейс мобильных приложений, разработанных по индивидуальным заданиям.

ПРАКТИЧЕСКАЯ РАБОТА № 6

Тема: Создание нового проекта

Цель работы: получить практические навыки по разработке мобильных приложений

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

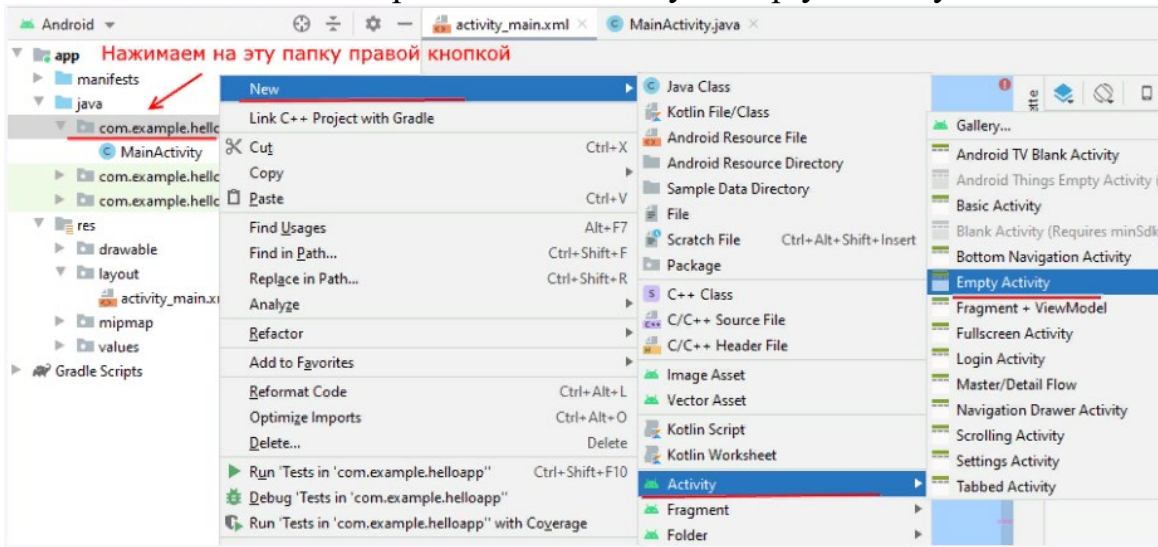
Содержание работы:

Задание 1. Разработать мобильное приложение, в котором 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.

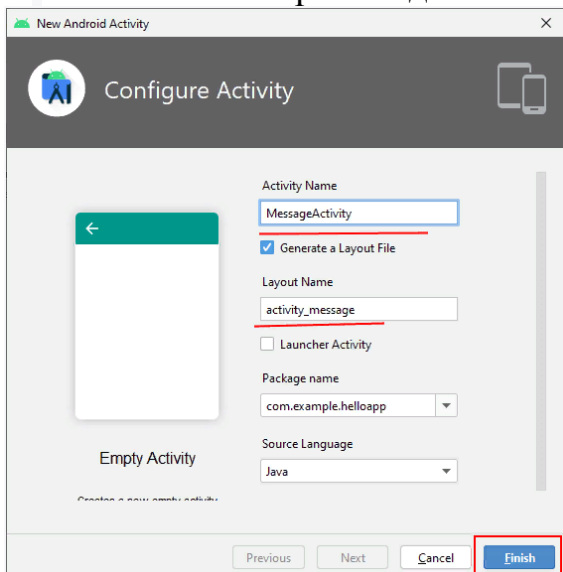
1. Возьмем ранее созданный проект (или создадим новый)

2. Добавление новой Activity

И 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 которой находится класс MainActivity, и затем в контекстном меню выберем New->Activity->Empty Activity:



3. После этого откроется диалоговое окно создания новой activity:



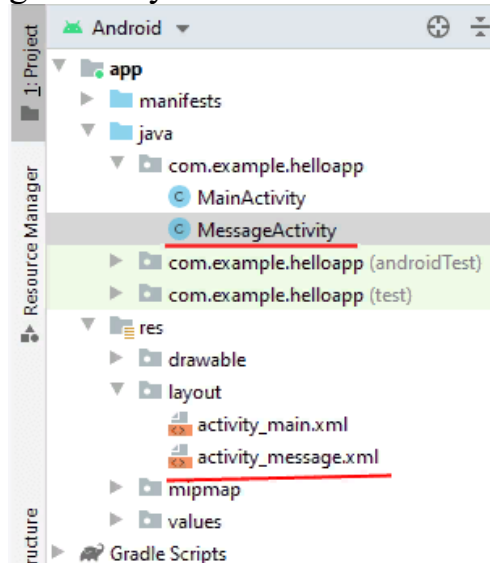
В этом окне в поле Activity Name введем MessageActivity. После этого в поле Layout Name автоматически должно установиться activity_message (если не установилось, то введем в это поле activity_message). В остальных полях оставим значения по умолчанию:

Package Name - должен иметь то же название, что и пакет, в котором находится MainActivity

Source Language должен иметь значение Java

И затем нажмем Finish.

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



5. 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:

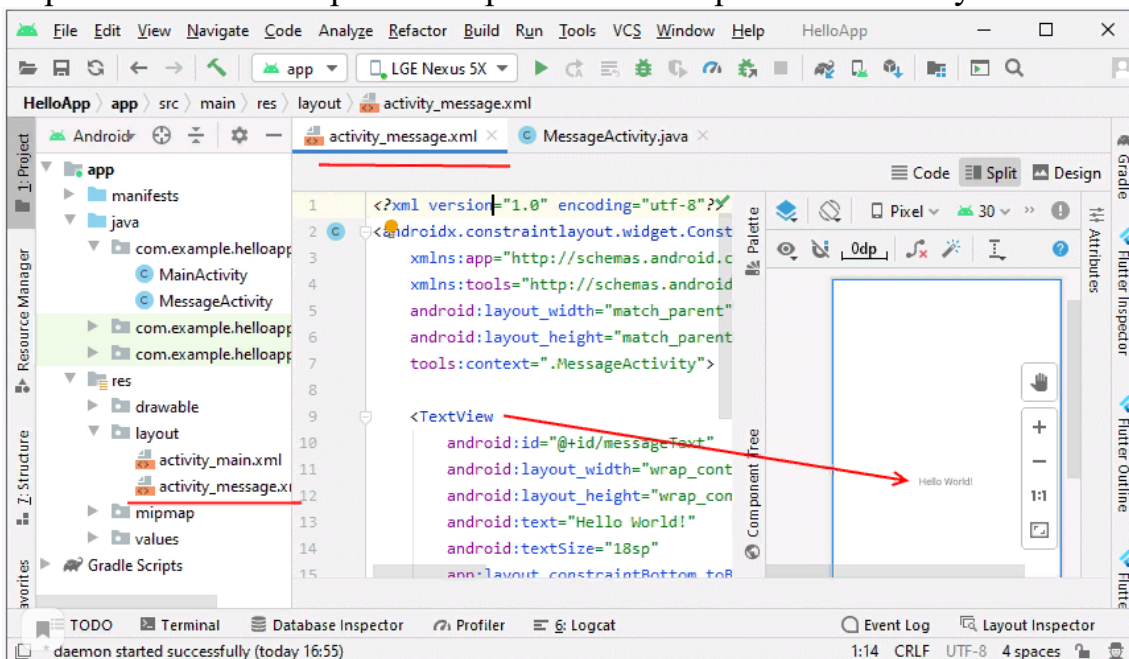
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    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>
```

6. 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



7. 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:

```
package com.example.helloapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MessageActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_message);    }    }
```

8. И изменим его следующим образом:

```
package com.example.helloapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;
public class MessageActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_message);
        // Получаем объект Intent, который запустил данную activity
        Intent intent = getIntent();
        // Получаем сообщение из объекта intent
        String message = intent.getStringExtra("message");
        // Получаем TextView по его id
        TextView messageText = (TextView) findViewById(R.id.messageText);
        // устанавливаем текст для TextView
        messageText.setText(message);    }    }
```

9. 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:

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

10. 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():

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

11. 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 виджет TextView по его id и затем вызываем его метод setText(), который устанавливает выводимый в TextView текст:

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

12. 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.

ПРАКТИЧЕСКАЯ РАБОТА № 7

Тема: Создание нового проекта

Цель работы: получить практические навыки по разработке мобильных приложений

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. В Практической работе № 6 определили MessageActivity, которая получает извне некоторые данные. Теперь определим в MainActivity код, который будет запускать MessageActivity и передавать ей некоторые данные.

1. Определение интерфейса. Изменим файл activity_main.xml 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=".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.

2. 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" />
```

3. 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)

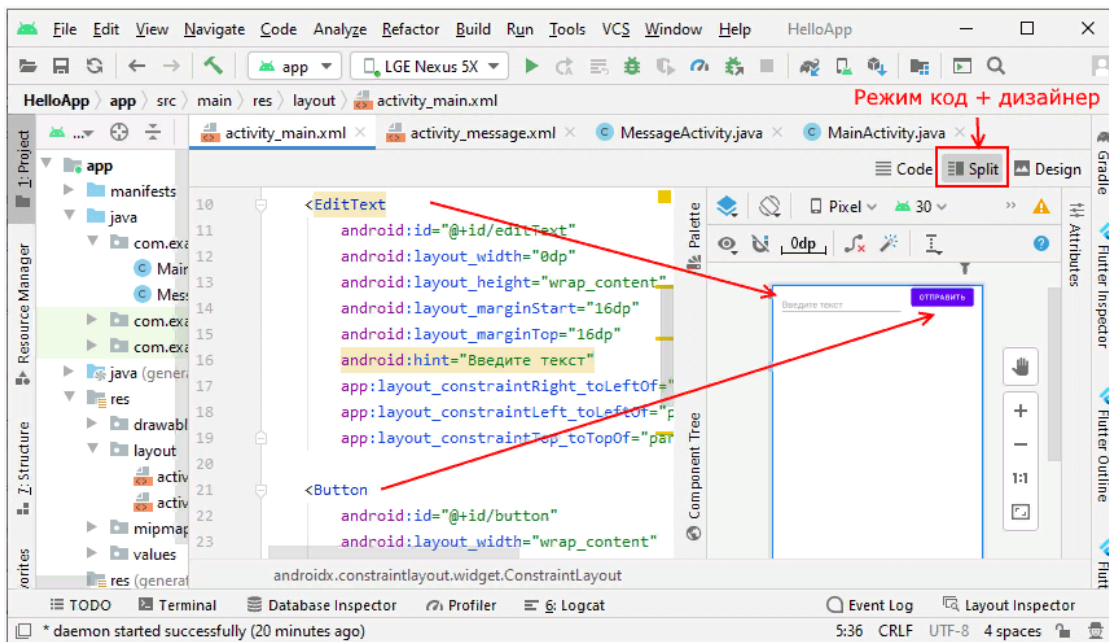
4. Теперь рассмотрим код кнопки - элемента Button, которая будет запускать MessageActivity:

```
<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 устанавливается 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 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 верхней границы контейнера. В данном случае 16 единиц
 - android:text: задает текст 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
5. Если мы перейдем к режиму дизайнера, например, в разделенном режиме (код + дизайнер), то мы увидим следующий макет:



6. Обработка нажатия кнопки. Итак, выше 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`:

```
package com.example.helloapp;
import androidx.appcompat.app.AppCompatActivity;
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);
        // Получаем 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); } }
```

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

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

8. 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);
```

9. 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

10. 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();
```

11. Затем полученный из текстового поля текст передается в запускаемую `activity`:

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

12. 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:

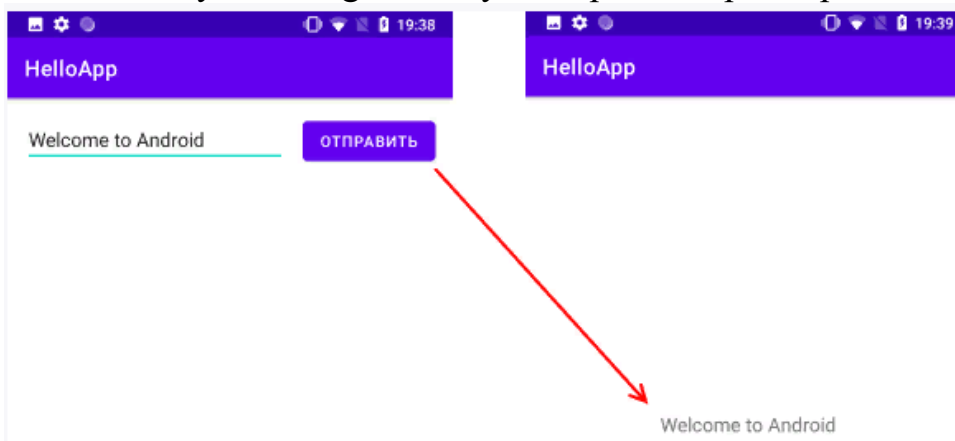
```
String message = intent.getStringExtra("message");
```

13. 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`.

14. 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:



ПРАКТИЧЕСКАЯ РАБОТА № 8

Тема: Изучение и комментирование кода

Цель работы: получить практические навыки по изучению и комментированию кода мобильных приложений

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Справочный материал:

Неявные намерения используются для запуска Активностей для выполнения заказанных действий в условиях, когда неизвестно (или безразлично), какая именно Активность (и из какого приложения) будет использоваться.

При создании Намерения, которое в дальнейшем будет передано методу `startActivity`, необходимо назначить действие (action), которое нужно выполнить, и, возможно, указать URI данных, которые нужно обработать. Также можно передать дополнительную информацию с помощью свойства `extras` Намерения. Android сам найдет подходящую Активность (основываясь на характеристиках Намерения) и запустит ее. Пример неявного вызова телефонной «звонилки»:

```
Intent intent = new Intent(Intent.ACTION_DIAL,  
Uri.parse("tel:(495)502-99-11"));  
startActivity(intent);
```

Для определения того, какой именно компонент должен быть запущен для выполнения действий, указанных в Намерениях, Android использует Фильтры Намерений (Intent Filters). Используя Фильтры Намерений, приложения сообщают системе, что они могут выполнять определенные действия (action) с определенными данными (data) при определенных условиях (category) по заказу других компонентов системы. Для регистрации компонента приложения (Активности или Сервиса) в качестве потенциального обработчика Намерений, требуется добавить элемент `<intent-filter>` в качестве дочернего элемента для нужного компонента в Манифесте приложения. У элемента `<intent-filter>` могут быть указаны следующие дочерние элементы (и соответствующие атрибуты у них):

- `<action>`. Атрибут `android:name` данного элемента используется для указания названия действия, которое может обслуживаться. Каждый Фильтр Намерений должен содержать не менее одного вложенного элемента `<action>`. Если не указать действие, ни одно Намерение не будет «проходить» через этот Фильтр. У главной Активности приложения в Манифесте должен быть указан Фильтр Намерений с действием `android.intent.action.MAIN`
- `<category>`. Сообщает системе, при каких обстоятельствах должно обслуживаться действие (с помощью атрибута `android:name`). Внутри `<intent-filter>` может быть указано несколько категорий. Категория `android.intent.category.LAUNCHER` требуется Активности, которая желает иметь «иконку» для запуска. Активности, запускаемые с помощью метода `startActivity`, обязаны иметь категорию `android.intent.category.DEFAULT`

- `<data>`. Дает возможность указать тип данных, которые может обрабатывать компонент. `<intent-filter>` может содержать несколько элементов `<data>`. В этом элементе могут использоваться следующие атрибуты:
 - `android:host` : имя хоста (например, `www.specialist.ru`)
 - `android:mimeType` : обрабатываемый тип данных (например, `text/html`)
 - `android:path` : «путь» внутри URI (например, `/course/android`)
 - `android:port` : порт сервера (например, `80`)
 - `android:scheme` : схема URI (например, `http`)

Пример указания Фильтра Намерений:

```
<activity android:name=".MyActivity"
  android:label="@string/app_name" >
  <intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="text/plain" />
  </intent-filter>
</activity>
```

При запуске Активности с помощью метода `startActivity` неявное Намерение обычно подходит только одной Активности. Если для данного Намерения подходят несколько Активностей, пользователю предлагается список вариантов. Определение того, какие Активности подходят для Намерения, называется `Intent Resolution`. Его задача – определить наиболее подходящие Фильтры Намерений, принадлежащие компонентам установленных приложений. Для этого используются следующие проверки в указанном порядке:

- Проверка действий. После этого шага остаются только компоненты приложений, у которых в Фильтрах Намерений указано действие Намерения. В случае, если действие в Намерении отсутствует, совпадение происходит для всех Фильтров Намерений, у которых указано хотя бы одно действие.
- Проверка категорий. Все категории, имеющиеся у Намерения, должны присутствовать в Фильтре Намерений. Если у Намерения нет категорий, то на данном этапе ему соответствуют все Фильтры Намерений, за одним исключением, упоминавшимся выше: Активности, запускаемые с помощью метода `startActivity`, обязаны иметь категорию `android.intent.category.DEFAULT`, так как Намерению, использованному в этом случае, по умолчанию присваивается данная категория, даже если разработчик не указал ничего явно. Из этого исключения, в свою очередь, есть исключение: если у Активности присутствуют действие `android.intent.action.MAIN` и категория `android.intent.category.LAUNCHER`, ему не требуется иметь категорию `android.intent.category.DEFAULT`.
- Проверка данных. Здесь применяются следующие правила:
 - Намерение, не содержащее ни URI, ни типа данных, проходит через Фильтр, если он тоже ничего перечисленного не содержит.
 - Намерение, которое имеет URI, но не содержит тип данных (и тип данных

невозможно определить по URI), проходит через Фильтр, если URI Намерения совпадает с URI Фильтра. Это справедливо только в случае таких URI, как mailto: или tel:, которые не ссылаются на реальные данные.

- Намерение, содержащее тип данных, но не содержащее URI подходит только для аналогичных Фильтров Намерений.
- Намерение, содержащее и тип данных, и URI (или если тип данных может быть вычислен из URI), проходит этот этап проверки, только если его тип данных присутствует в Фильтре. В этом случае URI должен совпадать, либо(!) у Намерения указан URI вида content: или file:, а у Фильтра URI не указан. То есть, предполагается, что если у компонента в Фильтре указан только тип данных, то он поддерживает URI вида content: или file:.

В случае, если после всех проверок остается несколько приложений, пользователю предлагается выбрать приложение самому. Если подходящих приложений не найдено, в выпустившей Намерение Активности возникает Исключение.

Содержание работы:

Задание 1. Создайте новый проект MetroPicker. Проект отредактировать, добавив кнопку

1. Запустить Android Studio. Создайте проект.
2. Добавьте вспомогательную Активность ListViewActivity для отображения и выбора станций метро.
3. Отредактируйте файл разметки res/layout/main.xml: добавьте кнопку выбора станции метро, присвоив идентификаторы виджетам TextView и Button для того, чтобы на них можно было ссылаться в коде.
4. Установите обработчик нажатия на кнопку в главной Активности для вызова списка станции и выбора нужной станции.
5. Напишите нужный обработчик для установки выбранной станции метро в виджет TextView родительской Активности (метод setText виджета TextView позволяет установить отображаемый текст). Не забудьте обработать ситуацию, когда пользователь нажимает кнопку «Назад» (в этом случае «никакой станции не выбрано» и главная Активность должна известить об этом пользователя).
6. Убедитесь в работоспособности созданного приложения, проверив реакцию различные действия потенциальных пользователей.

ПРАКТИЧЕСКАЯ РАБОТА № 9

Тема: Изучение и комментирование кода

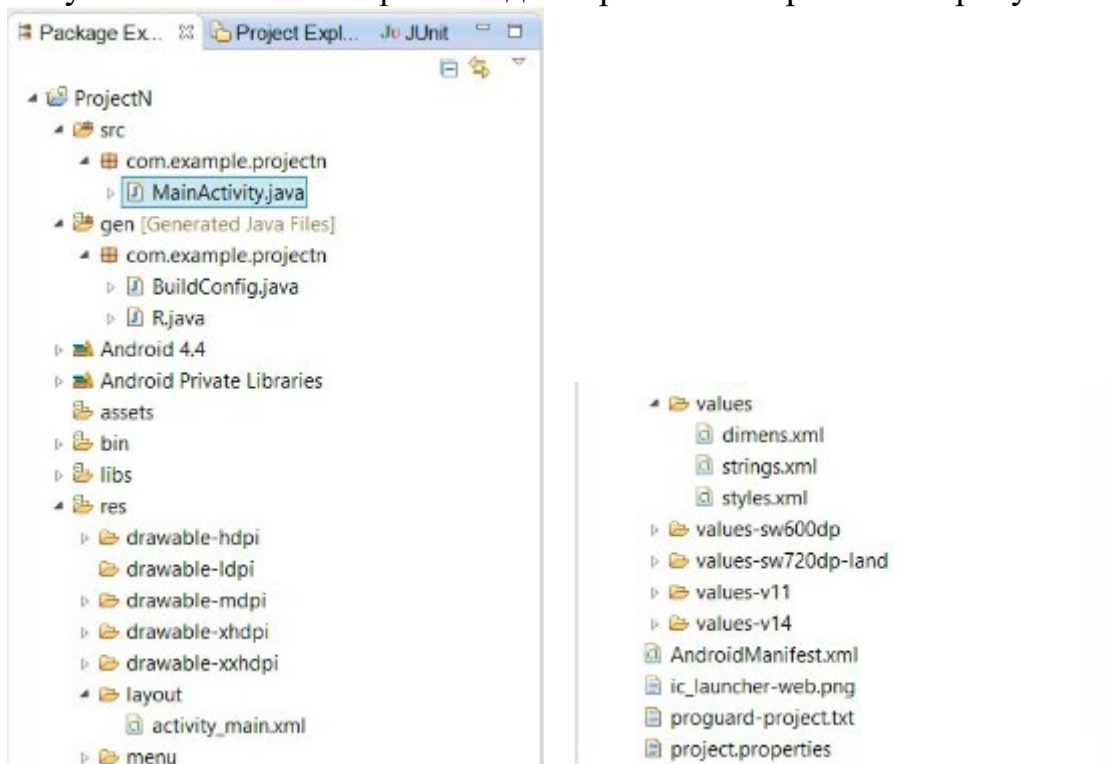
Цель работы: получить практические навыки по изучению и комментированию кода мобильных приложений

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Создать проект в Android Studio изучить файл AndroidManifest.xml.

1. Создайте проект и назовите его ProjectN, среда разработки подготавливает необходимые папки и файлы. Полный иерархический список обязательных элементов проекта можно увидеть на вкладке Package Explorer, иерархия полученных папок и файлов для проекта изображена на рисунке



2. Рассмотрим папки:

А) папка src - содержит файлы с исходным кодом на языке Java. Именно в этой папке размещаются все классы, создаваемые в процессе разработки приложения. Сейчас в этой папке в пакете com.example.projectn размещается единственный класс.

Комментарий 1: Имя пакету присваивается в процессе создания приложения в поле Package Name, использовать com.example не рекомендуется, т. к. пакет с таким именем нельзя загрузить в Google Play.

Комментарий 2: Имя файлу присваивается в процессе создания приложения на этапе настройки активности. Имя определяется в поле Activity Name.

Комментарий 3: Package Explorer отображает структуру папок, которая создается в каталоге, выбранном в качестве рабочего (Workspace) при запуске Eclipse.

Б) папка gen - содержит java-файлы, которые не требуется изменять и лучше вообще не трогать. Эти файлы генерируются автоматически. Нас может заинтересовать файл R.java он содержит идентификаторы (ID) для всех ресурсов приложения.

В) папка 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;

- values - содержит XML файлы, которые определяют простые значения, таких ресурсов как, строки, числа, цвета, темы, стили, которые можно использовать в данном проекте;

- menu - содержит XML файлы, которые определяют все меню приложения.

3. Файл манифеста **AndroidManifest.xml** предоставляет основную информацию о программе системе. Каждое приложение должно иметь свой файл **AndroidManifest.xml**. Редактировать файл манифеста можно вручную, изменяя XML-код или через визуальный редактор Manifest Editor (Редактор файла манифеста), который позволяет осуществлять визуальное и текстовое редактирование файла манифеста приложения.

Корневым элементом манифеста является **<manifest>**. Помимо данного элемента обязательными элементами являются теги **<application>** и **<uses-sdk>**. Элемент **<application>** является основным элементом манифеста и содержит множество дочерних элементов, определяющих структуру и работу приложения. Порядок расположения элементов, находящихся на одном уровне, произвольный. Все значения устанавливаются через атрибуты элементов. Кроме обязательных элементов, в манифесте по мере необходимости используются другие элементы.

Описание файла:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest />
```

```
<uses-permission />
```

```
<permission />
```

```
<permission-tree />
```

```
<permission-group />
```

```
<instrumentation />
```

```
<uses-sdk />
```

```
<uses-configuration />
```

```
<uses-feature />
```

```
<supports-screens />
```

```
<application>
```

```
<activity>
```

```

<intent-filter>
  <action />
  <category />
  <data />
</intent-filter>
<meta-data />
</activity>
<activity-alias>
  <intent-filter>
    <action />
    <category />
    <data />
  </intent-filter>
  <meta-data />
</activity-alias>
<service>
  <intent-filter>
    <action />
    <category />
    <data />
  </intent-filter>
  <meta-data />
</service>
<receiver>
  <intent-filter>
    <action />
    <category />
    <data />
  </intent-filter>
  <meta-data />
</receiver>
<provider>
  <grant-uri-permission />
  <path-permission />
  <meta-data />
</provider>
<uses-library />
</application>
</manifest>

```

4.Элемент *<manifest>* является корневым элементом манифеста. По умолчанию Eclipse создает элемент с четырьмя атрибутами:

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="ru.alexanderklimov.helloandroid"
android:versionCode="1"
android:versionName="1.0">

```

Атрибуты:

`xmlns:android` - определяет пространство имен Android. Оно всегда одно и то же

`package` - определяет уникальное имя пакета приложения, которое вы задали при создании проекта. Android Marketplace проверяет уникальность при приеме приложения, поэтому рекомендуется использовать свое имя для избежания конфликтов с другими разработчиками.

`android:versionCode` - указывает на внутренний номер версии, используемый для сравнения версий программы. «versionCode» должен быть целым.

`android:versionName` - указывает номер пользовательской версии. Можно использовать строку или строковый ресурс. Этот номер видит пользователь.

Элемент `<permission>` - объявляет разрешение, которое используется для ограничения доступа к определенным компонентам или функциональности данного приложения. В этой секции описываются права, которые должны запросить другие приложения для получения доступа к вашему приложению.

`android:name` - название разрешения

`android:label` - имя разрешения, отображаемое пользователю

`android:description` - описание разрешения

`android:icon` - значок разрешения

`android:permissionGroup` - определяет принадлежность к группе разрешений

`android:protectionLevel` - уровень защиты

Элемент `<uses-permission>` - запрашивает разрешение, которые приложению должны быть предоставлены системой для его нормального функционирования. Разрешения предоставляются во время установки приложения, а не во время его работы.

`android:name` - `<uses-permission>` имеет единственный атрибут с именем разрешения `android:name`. Это может быть разрешение, определенное в элементе `<permission>` данного приложения, разрешение, определенное в другом приложении или одно из стандартных системных разрешений, например: `android:name="android.permission.CAMERA"` или `android:name=""android.permission.READ_CONTACTS"`

Наиболее распространенные разрешения

- `INTERNET` - доступ к интернету
- `READ_CONTACTS` - чтение (но не запись) данных из адресной книги пользователя
- `WRITE_CONTACTS` - запись (но не чтение) данных из адресной книги пользователя
- `RECEIVE_SMS` - обработка входящих SMS
- `ACCESS_COARSE_LOCATION` - использование приблизительного определения местонахождения при помощи вышек сотовой связи или точек доступа Wi-Fi
- `ACCESS_FINE_LOCATION` - точное определение местонахождения при помощи GPS

Элемент `<permission-tree>` - объявляет базовое имя для дерева разрешений. Этот элемент объявляет не само разрешение, а только пространство имен, в которое могут быть помещены дальнейшие разрешения.

Элемент `<permission-group>` - определяет имя для набора логически связанных разрешений. Это могут быть как объявленные в этом же манифесте с элементом `<permission>` разрешения, так и объявленные в другом месте. Этот элемент не объявляет разрешение непосредственно, только категорию, в которую могут быть помещены разрешения. Разрешение можно поместить в группу, назначив имя группы в атрибуте `permissionGroup` элемента `<permission>`.

Элемент `<instrumentation>` - объявляет объект `instrumentation`, который дает возможность контролировать взаимодействие приложения с системой. Обычно используется при отладке и тестировании приложения и удаляется из `release`-версии приложения.

Элемент `<uses-sdk>` - позволяет объявлять совместимость приложения с указанной версией (или более новыми версиями API) платформы Android. Уровень API, объявленный приложением, сравнивается с уровнем API системы мобильного устройства, на который устанавливается данное приложение.

Атрибуты

`android:minSdkVersion` - определяет минимальный уровень API, требуемый для работы приложения. Система Android будет препятствовать тому, чтобы пользователь установил приложение, если уровень API системы будет ниже, чем значение, определенное в этом атрибуте.

`android:maxSdkVersion` - позволяет определить самую позднюю версию, которую готова поддерживать ваша программа. Ваше приложение будет невидимым в Google Play для устройств с более свежей версией.

`targetSdkVersion` - позволяет указать платформу, для которой вы разрабатывали и тестировали приложение.

Элемент `<uses-configuration>` - указывает требуемую для приложения аппаратную и программную конфигурацию мобильного устройства. Спецификация используется, чтобы избежать установки приложения на устройствах, которые не поддерживают требуемую конфигурацию.

- `reqFiveWayNav` - используйте значение *true*, если приложению требуется устройство ввода, поддерживающее навигацию вверх, вниз, влево, вправо, а также нажатие выделенного элемента. К таким устройствам относятся трекболы и D-pad. В принципе устарело

- `reqHardKeyboard` - используйте значение *true*, если приложению нужна аппаратная клавиатура.

- `reqKeyboardType` - позволяет задать тип клавиатуры: `nokeys`, `qwerty`, `twelvekey`, `undefined`

- `reqNavigation` - укажите одно из значений: `nonav`, `dpad`, `trackball`, `wheel` или `undefined`, если требуется устройство для навигации

- `reqTouchScreen` - если требуется сенсорный экран, то используйте нужное значение из возможных вариантов: `notouch`, `stylus`, `finger`, `undefined`.

Приложение не будет устанавливаться на устройстве, которое не соответствует заданной вами конфигурации. В идеале, вы должны разработать такое приложение, которое будет работать с любым сочетанием устройств ввода. В этом случае **<uses-configuration>** не нужен.

Элемент *<uses-feature>* - объявляет определенную функциональность, требующуюся для работы приложения. Таким образом, приложение не будет установлено на устройствах, которые не имеют требуемую функциональность.

Элемент *<supports-screens>* - определяет разрешение экрана, требуемое для функционирования устройства. Данный тег позволяет указать размеры экран, для которого был спроектировано приложение. Система будет масштабировать ваше приложение на основе ваших макетов на тех устройствах, которые поддерживают указанные вами разрешения экран.

Возможные значения:

smallScreen - экраны QVGA

normalScreen - стандартные экраны HVGA и WQVGA

largeScreen - большие экраны

xlargeScreen - очень большие экраны, которые превосходят размеры планшетов

anyDensity - установите значение *true*, если ваше приложение способно масштабироваться для отображения на экране с любым разрешением.

По умолчанию, для каждого атрибута установлено значение *true*. Вы можете указать, какие размеры экранов ваше приложение не поддерживает.

<supports-screens

android:smallScreen=["false"]

android:normalScreen=["true"]

android:largeScreen=["true"]

android:anyDensity=["false"] />

Начиная с API 13 (Android 3), у тега появились новые атрибуты:

- **requiresSmallestWidthDp** - указываем минимальную поддерживаемую ширину экрана (наименьшая сторона устройства) в аппаратно-независимых пикселях. С его помощью можно отфильтровать устройства при размещении приложения в Google Play

- **compatibleWidthLimitDp** - задаёт верхнюю границу масштабирования для вашего приложения. Если экран устройства выходит за указанную границу, система включит режим совместимости.

- **largestWidthLimitDp** - задаёт абсолютную верхнюю границу, за пределами которой ваше приложение точно не может быть масштабировано. В этом случае приложение запускается в режиме совместимости, которую нельзя отключить. Следует избегать подобных ситуаций и разрабатывать макеты для любых экранов.

<supports-screens android:smallScreens="false"

android:normalScreens="true"

android:largeScreens="true"

android:requiresSmallestWidthDp="480"

android:compatibleWidthLimitDp="600"

android:largestWidthLimitDp="720" />

<application>

Элемент *<application>* один из основных элементов манифеста, содержащий описание компонентов приложения, доступных в пакете: стили, значок и др. Содержит дочерние элементы, которые объявляют каждый из компонентов, входящих в состав приложения. В манифесте может быть только один элемент *<application>*.

Элемент *<activity>* - объявляет активность. Если приложение содержит несколько активностей, не забывайте объявлять их в манифесте, создавая для каждой из них свой элемент *<activity>*. Если активность не объявлена в манифесте, она не будет видна системе и не будет запущена при выполнении приложения или будет выводиться сообщение об ошибке.

Атрибуты

android:name - имя класса. Имя должно включать полное обозначение пакета, но т. к. имя пакета уже определено в корневом элементе *<manifest>*, имя класса, реализующего деятельность, можно записывать в сокращенном виде, опуская имя пакета

android:label - текстовая метка, отображаемая пользователю

android:launchMode - управление стеком

android:parentActivityName - В приложениях с API 16 и выше используется эта строка. Она сообщает, какая активность является родительской. Для старых устройств используйте метаданные

Элемент *<activity>* содержит множество других атрибутов, определяющих разрешения, ориентацию экрана и т. д.

ПРАКТИЧЕСКАЯ РАБОТА № 10

Тема: Изменение элементов дизайна

Цель работы: изучение основ разработки интерфейсов мобильных приложений

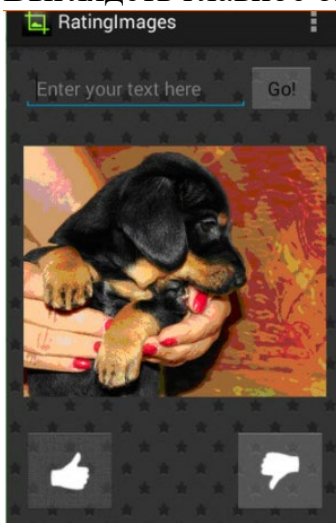
Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Разработать интерфейс приложения, которое ищет в сети Интернет изображения по запросу пользователя, позволяет оценивать их, скачивать, и посещать интернет-страницы сайтов, на которых было найдено изображение.

1. Создание заготовки для приложения

Выглядеть главное окно будет примерно так:

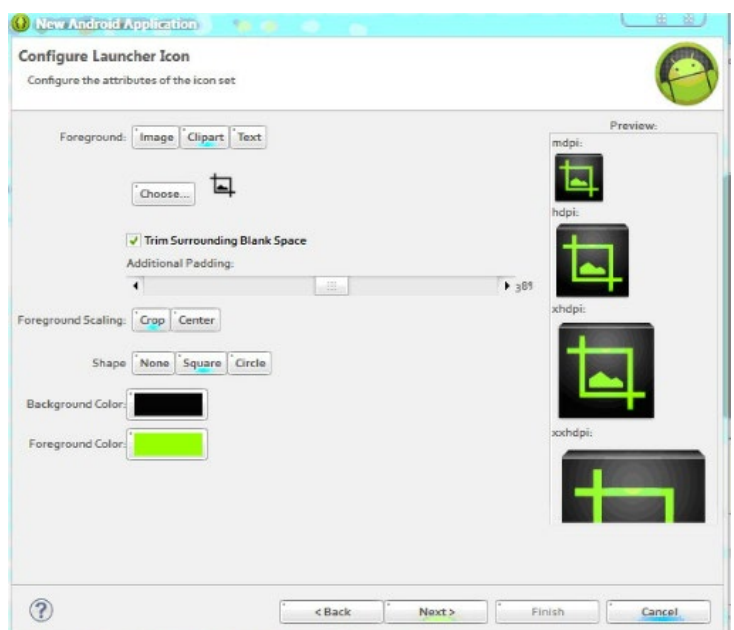


На нём присутствуют поле ввода текста для запроса пользователя и кнопка, начинающая поиск изображений. Внизу экрана две кнопки: "like" и "dislike", с их помощью пользователь сможет оценить изображение. После того, как пользователь сделает оценку изображения, текущее изображение закрывается и загружается следующее.

2. Назовём новый проект "RatingImages" ("Рейтинг изображений"). Создайте иконку на свой вкус.

3. После создания проекта откройте activity_main.xml из каталога res/layout/. Когда вы откроете файл activity_main.xml, вы увидите графический редактор макета.

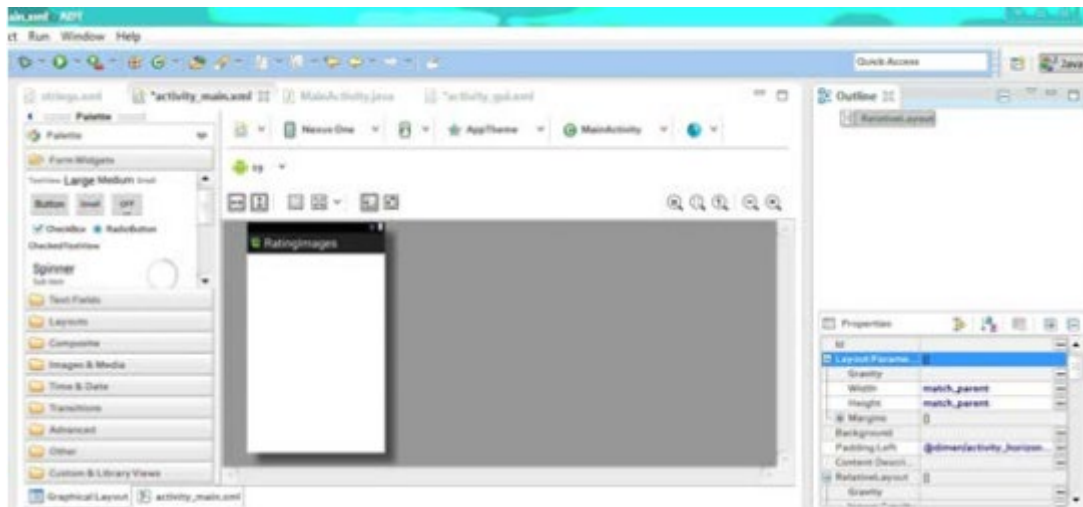
4. Теперь щелкните по вкладке activity_main.xml в нижней части экрана. Открылся XML-редактор



кода. Этот способ редактирования

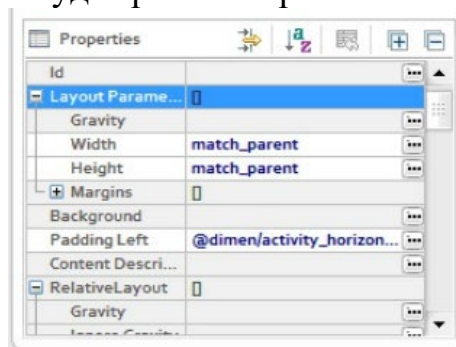
стандартный, но все изменения, вносимые в этот документ, можно также ощутить визуально, перейдя на графический редактор.

5.Вернёмся на вкладку с графическим редактором. Во-первых, подготовим документ к началу работы, для этого удалите <TextView>.Результат выглядит так:

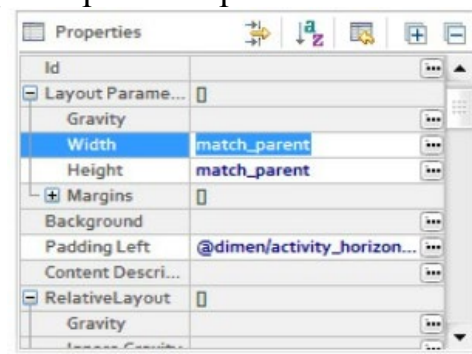


6. На рабочей области экрана остался один элемент. Это макет <RelativeLayout>. В нём позиция дочерних элементов может быть описана по отношению друг к другу или к родителю.

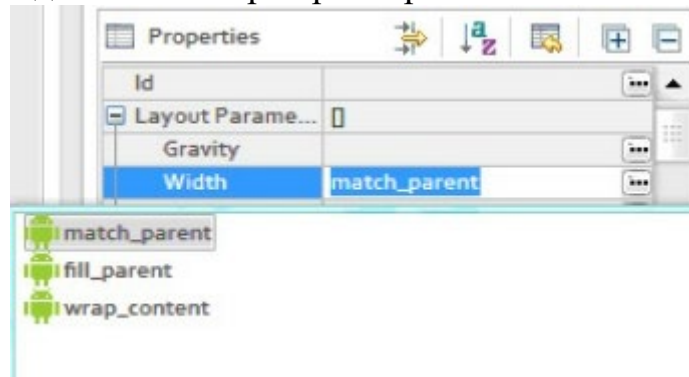
7.Два атрибута, ширина и высота (android:layout_width и android:layout_height), требуются для всех элементов для того, чтобы указать их размер. Так как <RelativeLayout> - это корень в макете, то нужно, чтобы он заполнял всю область экрана. Это достигается при помощи установки параметра "match_parent" для ширины и высоты. Это значение указывает, что ширина и высота элемента будет равна ширине и высоте родителя.



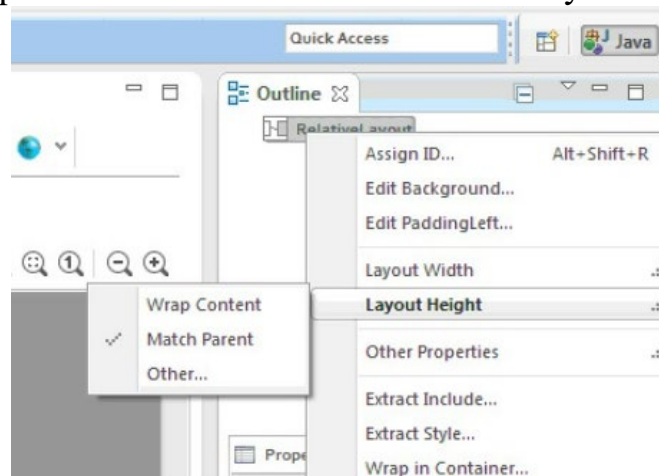
8. Однократным щелчком левой кнопкой мыши по надписи "Width" активируйте строку с параметрами ширины



9. Щелчком левой кнопки мыши по области ввода вызовите диалоговое окно, и двойным щелчком сделайте выбор параметра:



10. Или щелчком правой кнопки мыши по <RelativeLayout> в Outline:



При выполнении первого способа вы увидите еще два возможных параметра: "fill_parent" и "wrap_content". На самом деле, match_parent = fill_parent, но "fill_parent" считается устаревшим, и к использованию в новых проектах предлагается "match_parent". Параметр "wrap_content" указывает, что представление будет увеличиваться при необходимости, чтобы поддерживать соответствие содержанию экрана.

ПРАКТИЧЕСКАЯ РАБОТА № 11

Тема: Изменение элементов дизайна

Цель работы: научиться размещать элементы и менять их свойства

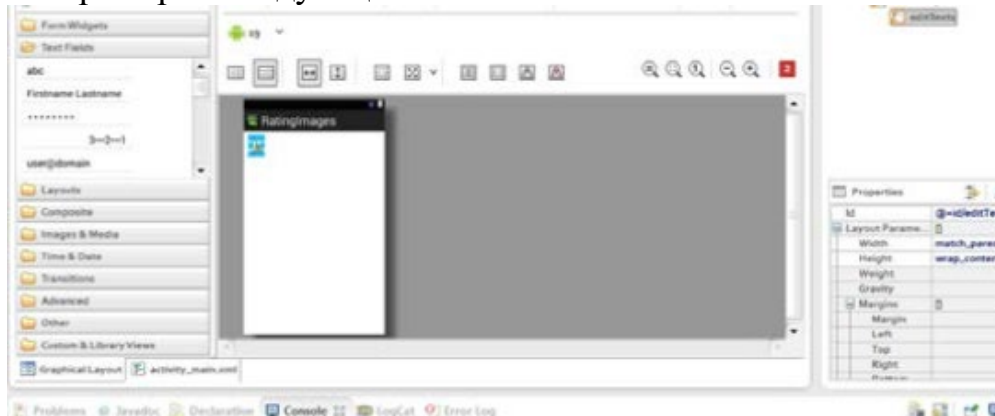
Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Добавление текстового поля в мобильное приложение, созданное в Практической работе № 10.

1. Откройте приложение, созданное в предыдущей работе.

2. Для начала добавьте элемент `<LinearLayout>` с горизонтальной ориентацией в `<RelativeLayout>`, и укажите для ширины и высоты параметр `"wrap_content"`. Теперь, для создания пользовательского редактируемого текстового поля, добавьте элемент `<EditText>` с параметром `"wrap_content"` для ширины и высоты в `<LinearLayout>`. Сейчас должно получиться примерно следующее:



Возможно, появился желтый предупреждающий знак, но сейчас это неважно, со временем он исчезнет. Наличие таких предупреждений никак не влияет на компилируемость проекта.

3. Теперь переходим к настройке добавленных нами элементов. Для многих элементов нужно назначать `id`, он обеспечивает уникальный идентификатор, который можно использовать как ссылку на объект из кода вашего приложения для управления им. Откроем редактор кода XML-файла и обратим внимание на элемент

`<EditText>`:

`<EditText`

`android:id="@+id/editText1"`

`android:layout_width="wrap_content"`

`android:layout_height="wrap_content">`

`<requestFocus/>`

`</EditText>`

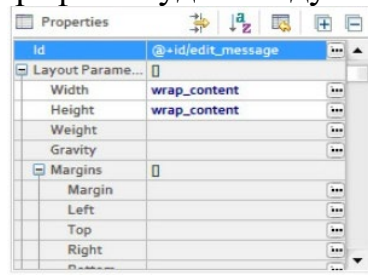
Строка `<requestFocus/>` появляется добавлением элемента `requestFocus` (папка `Advanced`), и позволяет установить фокус на нужном компоненте. Важно использовать этот элемент, когда у вас имеется, к примеру, три текстовых поля, и нужно, чтобы фокус был на втором из них.

При указании id, знак (@) требуется в том случае, если вы имеете ввиду любой ресурс объекта из XML-файла. За ним следуют тип ресурса (в данном случае ID), косая черта (слеш) и имя ресурса (editText1).

Знак плюс (+) перед типом ресурсов необходим только тогда, когда вы впервые определяете идентификатор ресурса.

По сути, id, который создается автоматически, уже уникален, но грамотнее переименовывать id в соответствии со назначением элемента.

4. Зададим id для текстового поля. Для этого прямо в коде строку `android:id="@+id/editText1"` заменяем на `android:id="@+id/edit_message"`, жмём CTRL+S и открываем графический редактор. Если всё хорошо, то в свойствах текстового поля в графе id будет следующее:



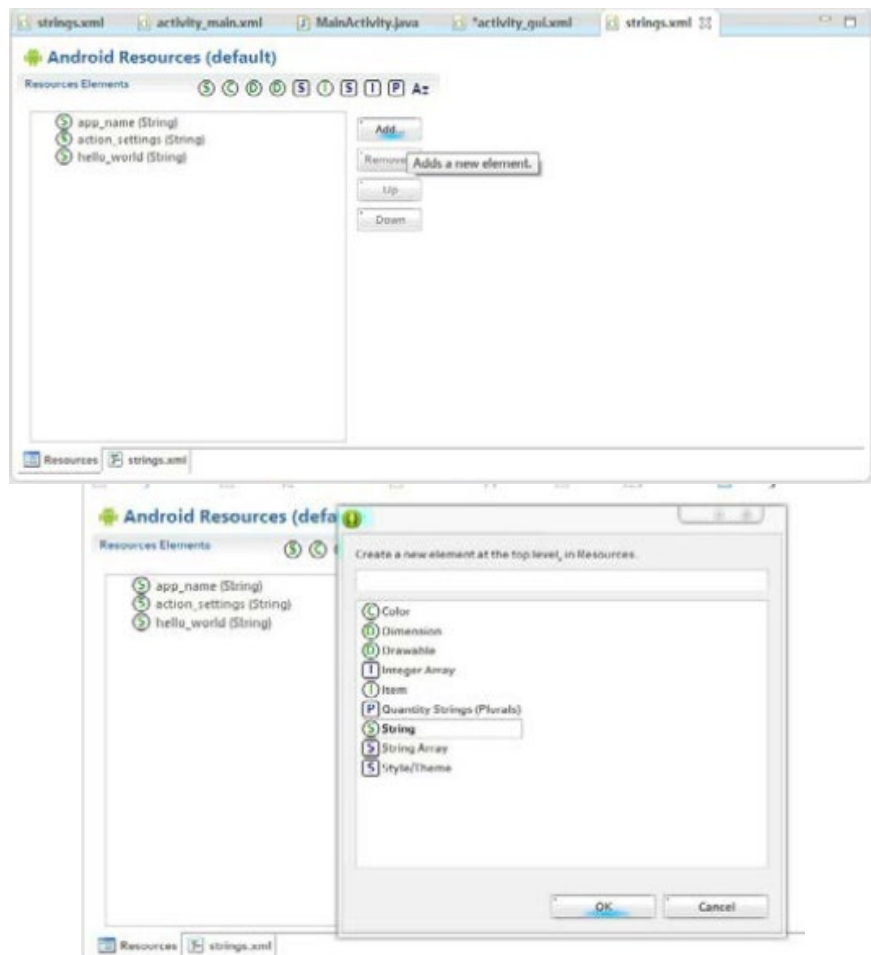
5. Добавим в код ещё две строки:

`android:ems="10"` - задаёт соответствия для симметричного отображения шрифтов, `android:hint="@string/edit_message"` - содержание тестового поля "по умолчанию", т.е. пока пользователь не начал вводить в поле текст. Вместо того, чтобы использовать просто слово (например `android:hint="message"`), что крайне не удобно при изменении основного языка приложения, используется ссылка на значение, хранящееся в файле `strings.xml`. Поскольку это относится к конкретному ресурсу (а не только к id), знак плюс не нужен.

6. Так как ещё не определили строку ресурсов файле `strings.xml`, и потому вы получите следующее:



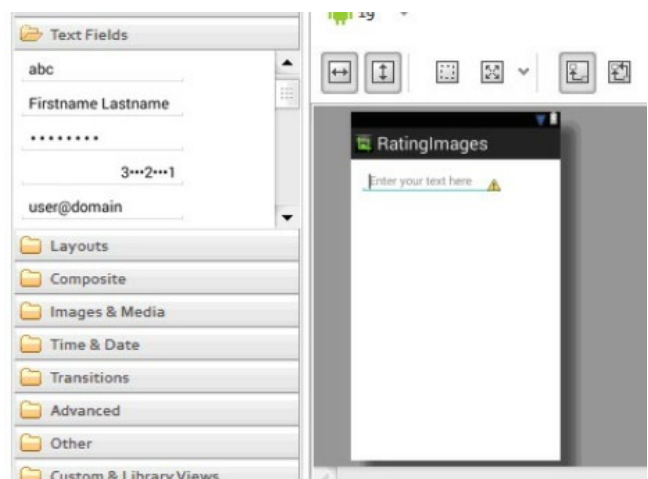
7. Для того, чтобы ссылка на ресурс начала работать, нужно этот ресурс создать. Откройте файл `res/values/strings.xml`. Очевидно, что его тоже можно редактировать двумя способами: графически и вручную. Выберите тот способ, который больше нравится



Заполняем поля "Имя" и "Значение":



8. Сохраняем и получаем результат:



ПРАКТИЧЕСКАЯ РАБОТА № 12

Тема: Изменение элементов дизайна

Цель работы: научиться размещать элементы и менять их свойства

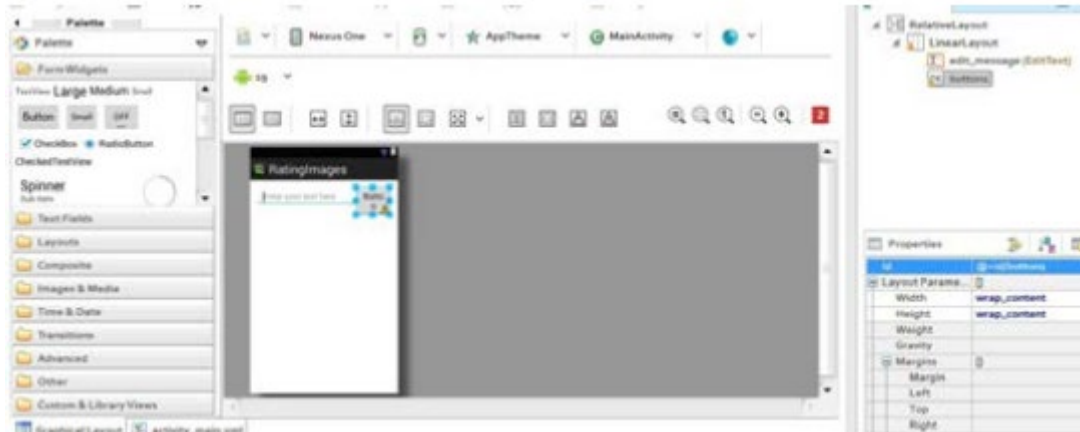
Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Добавить кнопки в созданное в Практической работе № 11 приложение.

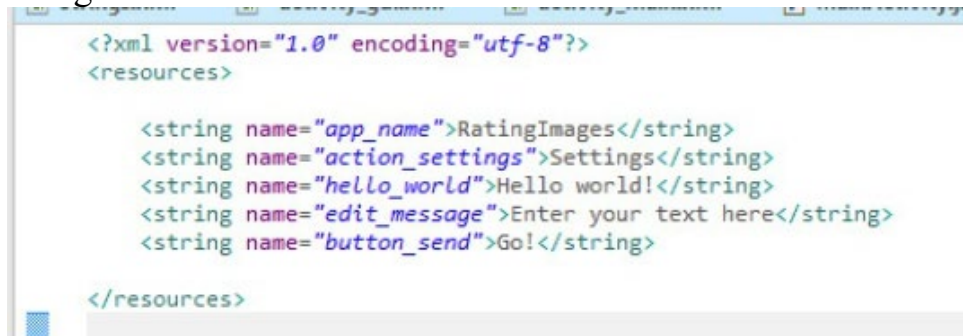
1. Откройте созданное в предыдущей работе приложение.

2. Добавьте `<Button>` в макет после элемента `<EditText>`:



Чтобы кнопка трансформировалась в соответствии с текстом кнопки, ширина и высота должны быть установлены во "wrap_content".

3. Теперь поменяем надпись на кнопке на "Go!" с помощью ссылки на ресурс в XML-коде главной активности и добавления одного ресурса в файл strings.xml:



4. Сохраните. На графическом редакторе главной активности будут изменения:

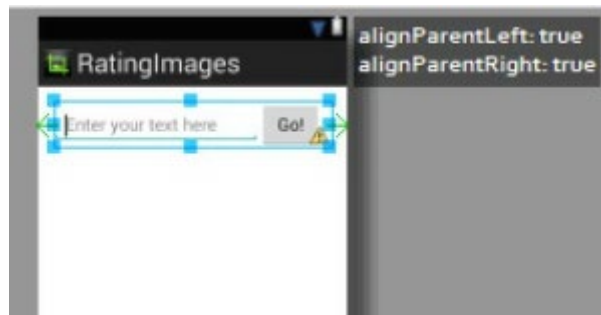


5. Теперь, когда мы поместили два главных представления на `<LinearLayout>` элемент, настало время добавить ещё два параметра для

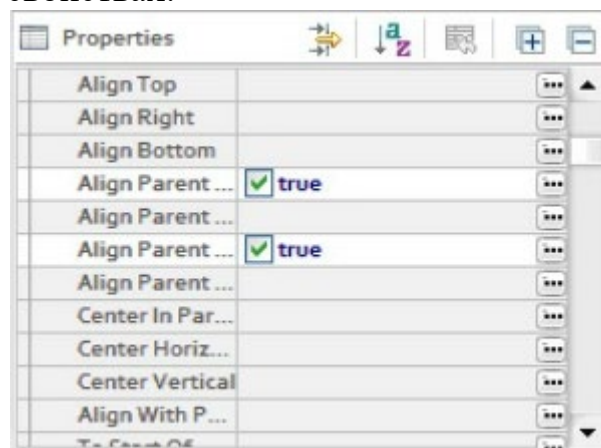
этого элемента. Речь идет о "приращении" правого и левого краёв лейаута к правому и левому краям <RelativeLayout> элемента соответственно. А сделать это проще простого - просто потяните мышкой один край к другому!



6. Зелёные стрелки по краям дают понять, к какому элементу удалось прицепить край



7. Это отобразится в свойствах:



ПРАКТИЧЕСКАЯ РАБОТА № 13

Тема: Обработка событий: цветовая индикация

Цель работы: научиться работать с цветом в Android Studio

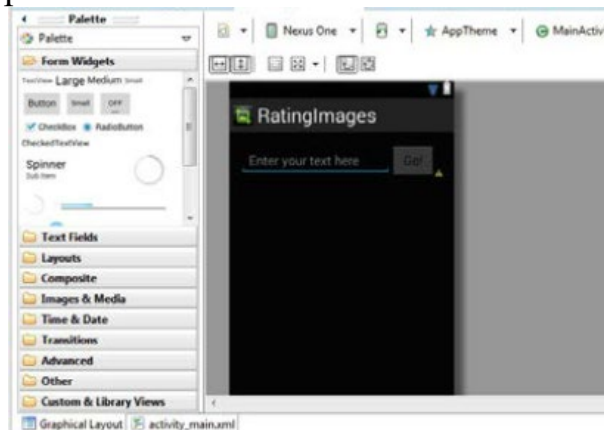
Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

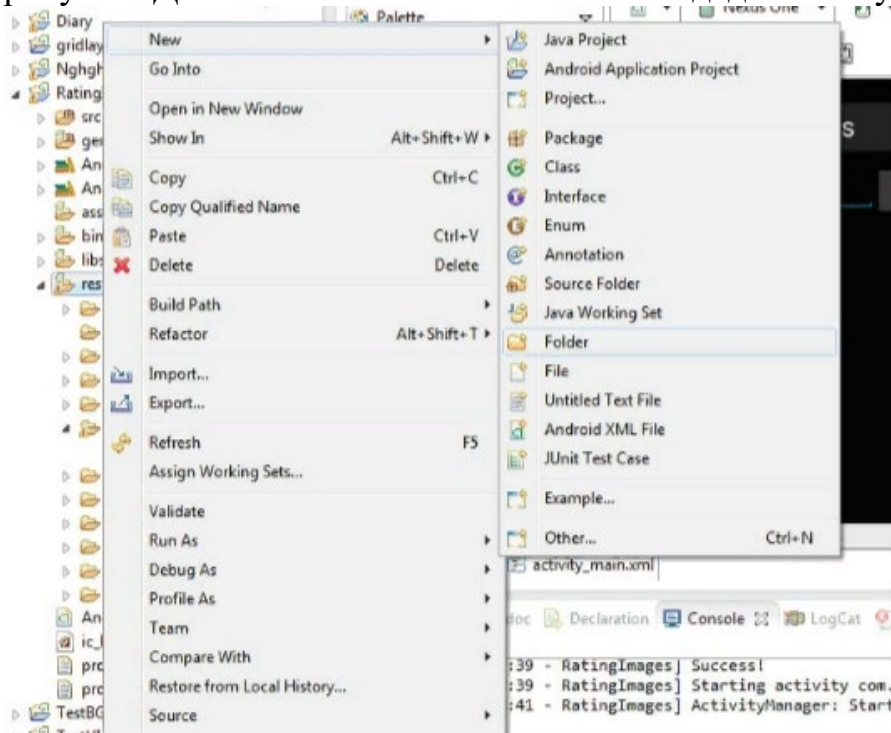
Задание 1. Смена фона приложения, разработанного в Практической работе № 12.

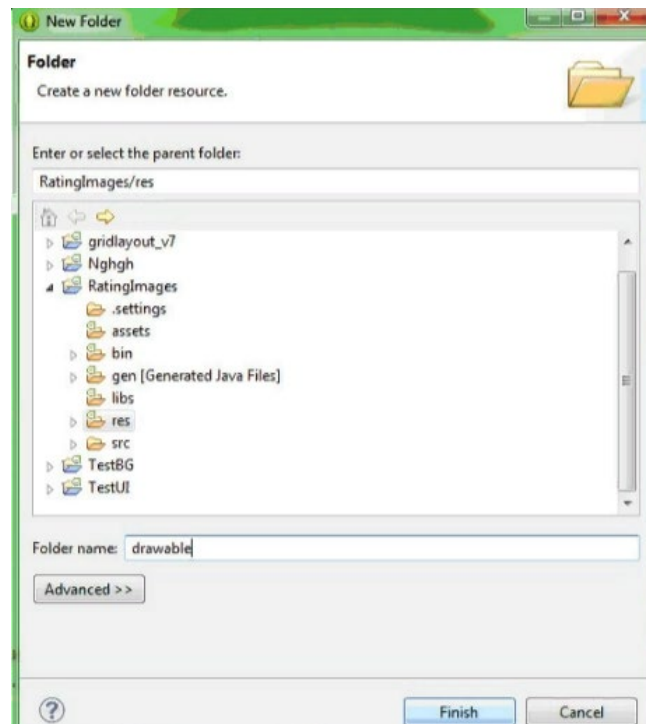
1. Запустить приложение, созданное в предыдущей работе

2. Чтобы изменить цвет фона на чёрный, нужно в XML-коде главной активности написать одну строку в блоке `<RelativeLayout>` элемента: `android:background="#000000"`. Сохраните и проверьте результат, открыв графический редактор.

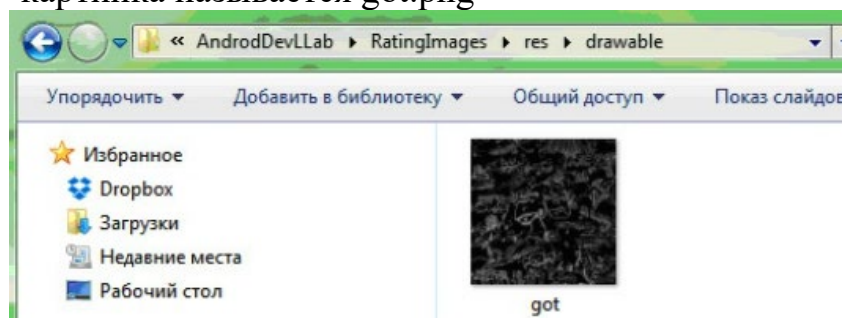


3. Поместим рисунок. Для этого сначала в папке `res/` создадим папку `drawable/`

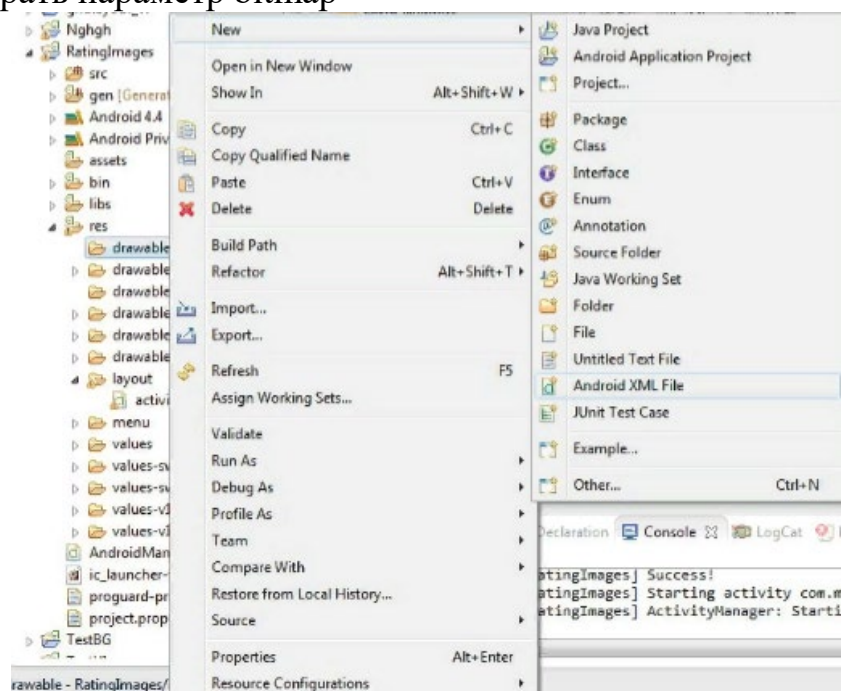


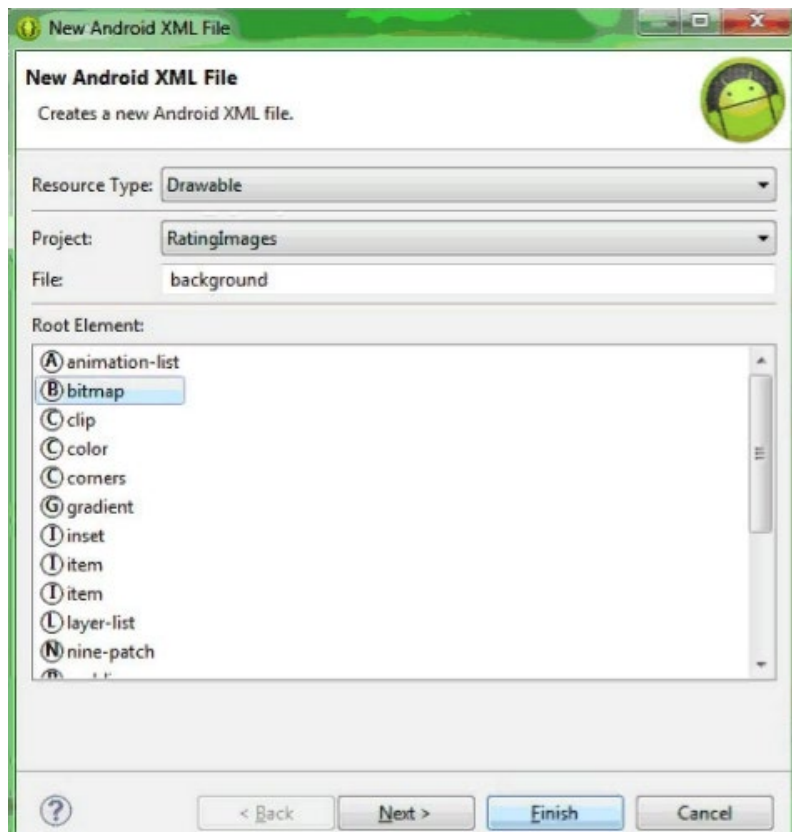


4. После того, как папка создана, нужно положить в эту папку изображение - картинка называется got.png



5. После этого в папке drawable/ нужно создать файл background.xml, важно при создании выбрать параметр bitmap





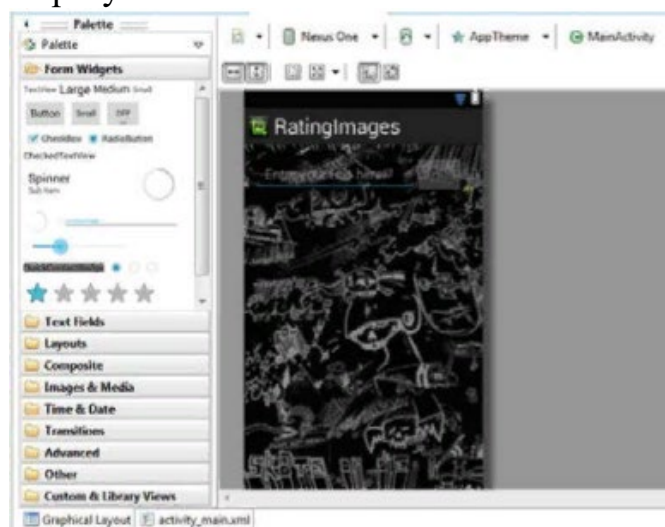
6. Как только новый файл открылся, пропишем в него одну строчку, с указанием на то, откуда и какой файл использовать:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/got">
</bitmap>
```

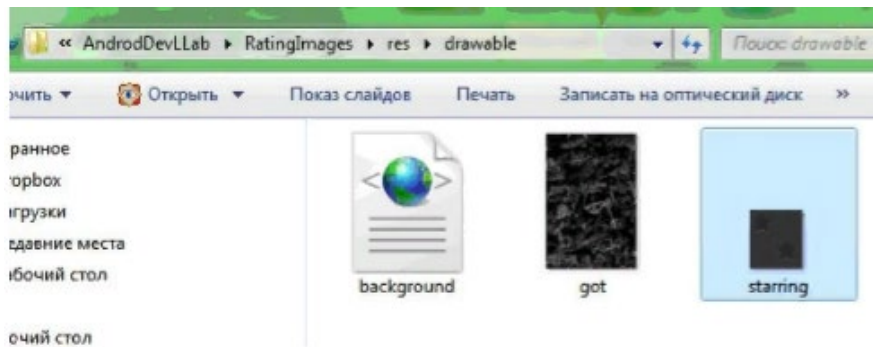
7.Вернемся в редактор XML-кода, туда, где прописывали цвет фона. Вместо строки `android:background="#000000"` напомним ссылку на XML-файл `android:background="@drawable/background"`.

8.Сохраняем и видим результат



9. Фон смотрится хорошо, но очевидно, что кнопка и поле ввода просто затерялись, а это значит, что для этого приложения такой фон не подходит.

Подберите другой рисунок, скачайте его и скопируйте изображение в папку drawable/.



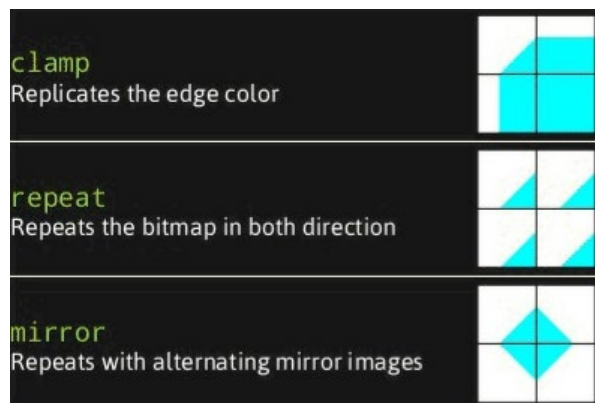
10. Теперь немного изменим файл background.xml. Во-первых, нужно изменить имя изображения со старого на новое.

`android:src="@drawable/starring"`

Во-вторых, добавим такую строку: `android:tileMode="repeat"`

11. Сохраняем.

12. Атрибут `android:tileMode` задает тип заполнения, в данном случае простое повторение исходного изображения. Кроме `repeat` возможны варианты `clamp` и `mirror`. Помните, что данный приём применим только к `bitmap`, к фигурам, созданным при помощи XML, применить данную операцию нельзя.



13. Получаем результат:



14. В `<RelativeLayout>` стоит добавлять `android:background` только в том случае, если вы хотите неподвижный фон, а в `<ScrollView>` чтобы фон прокручивался вместе с контентом. Если вы выбрали тёмный фон, то стоит поменять цвет текста, вводимого в поле ввода, например на белый. Для этого в блок `<EditText>` добавим строку `android:textColor="#ffffff"`

ПРАКТИЧЕСКАЯ РАБОТА № 14

Тема: Обработка событий: цветовая индикация

Цель работы: научиться применять обработчики событий.

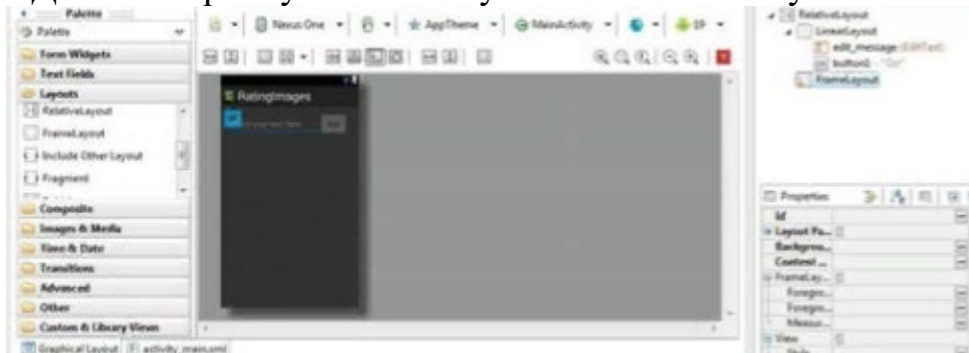
Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Добавить область просмотра изображений в приложение, созданное в Практической работе № 13

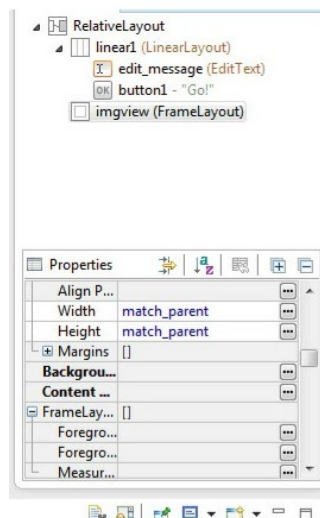
1. Откройте приложение из предыдущей работы

2. Создадим область отображения изображений, которые пользователь будет оценивать. Добавьте "рамку" <FrameLayout> в <RelativeLayout>:

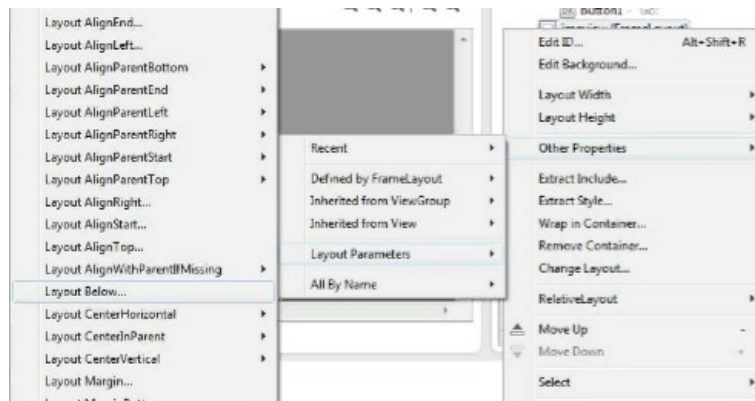


Обратите внимание, что предупреждающий жёлтый треугольник исчез, но появился у нового элемента, и это значит, что всё идёт по плану.

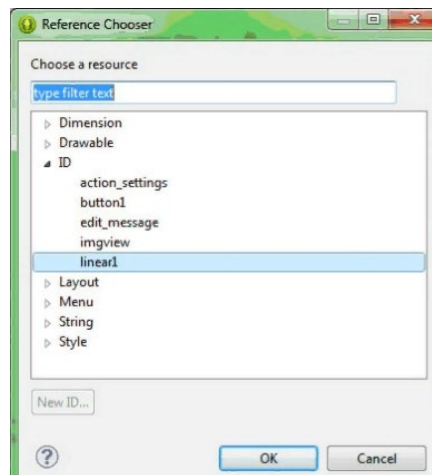
3. Укажем этому элементу ширину, высоту и id. Предупреждение должно пропасть.



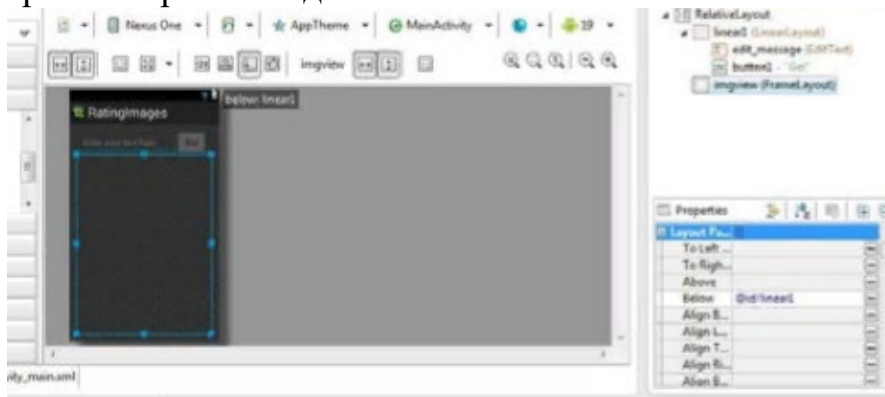
4. Сейчас FrameLayout "заезжает" на поле ввода и кнопку. Чтобы исправить это, нужно задать для "рамки" параметр, который будет следить, чтобы "рамка" находилась ниже, чем поле ввода и кнопка. Выберите из списка Outline "рамку" и кликом правой кнопки мыши вызовите контекстное меню: Other Properties -> Layout Parametrs -> LayoutBelow...



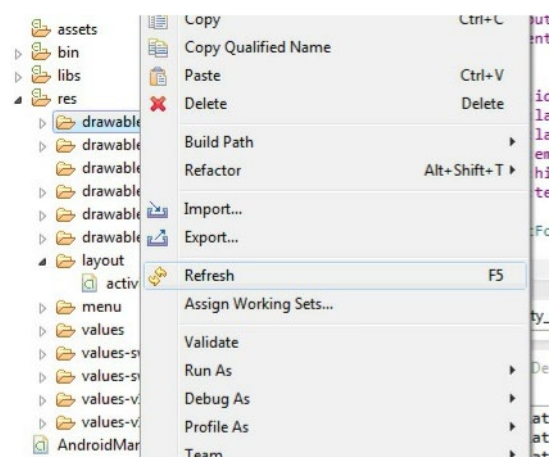
5. Появится окно. В нём выбираем из списка "ID" имя лейаута, на котором находятся поле ввода и кнопка:



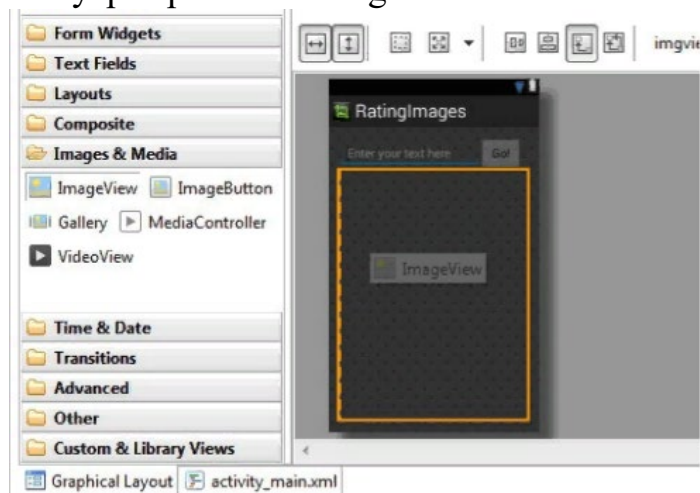
Жмём ОК и "рамка" примет вид.



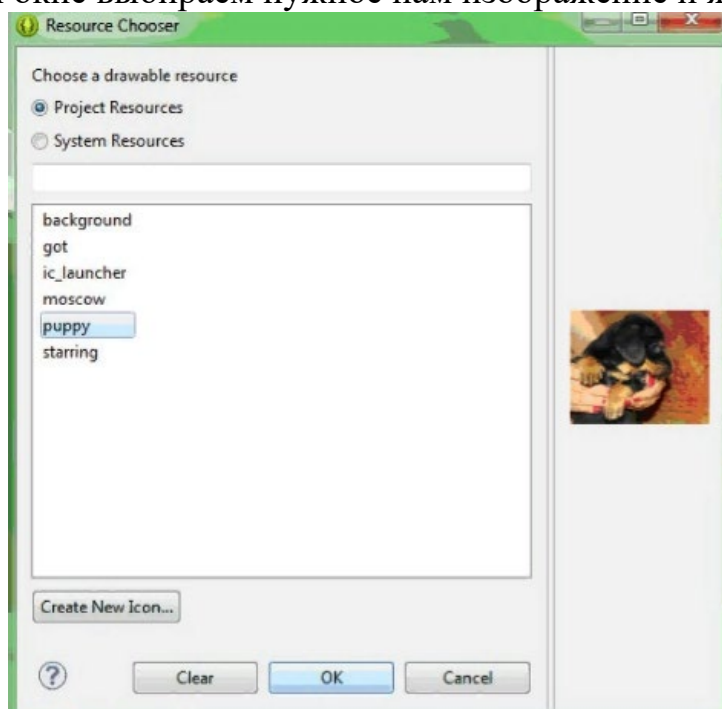
6. Теперь для наглядности поместим туда изображение. Сначала поместим само изображение в папку res/drawable/ и обновим её. Это нужно для того, чтобы новые данные загрузились в проект.



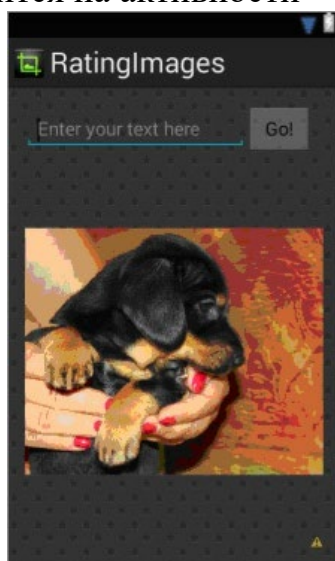
7. Теперь поместим внутрь "рамки" <ImageView>



8. В появившемся окне выбираем нужное нам изображение и жмём ОК

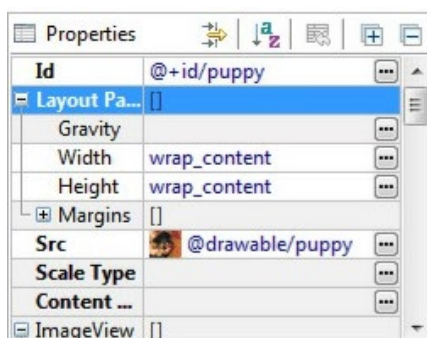


9. Изображение должно появиться на активности



10. Вместе с изображением появился новый предупреждающий знак. В описании предупреждения сказано: "[Accessibility] MissingcontentDescription attribute on image". Это означает: Атрибут android:contentDescription используется в программах, которые поддерживают специальные возможности для людей с нарушениями зрения и слуха. То есть, текст, прописанный в этом атрибуте, не будет показан на экране, но при включенной в "Специальных возможностях" услуге "звуковые подсказки" этот текст будет проговариваться, когда пользователь переходит к этому элементу управления.

Этот атрибут можно задать для ImageButton, ImageView и CheckBox, и задавать его или нет - дело каждого. Свойства у изображения должны быть следующими.



11. Сохраните работу.

ПРАКТИЧЕСКАЯ РАБОТА № 15

Тема: Обработка событий: цветовая индикация

Цель работы: научиться использовать обработчики событий

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Разработать мобильное приложение с добавлением разных элементов. В качестве фона выбрать изображение. При нажатии на кнопку, должно происходить смена фона приложения.

ПРАКТИЧЕСКАЯ РАБОТА № 16

Тема: Обработка событий: подсказки

Цель работы: научиться использовать обработчики событий.

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

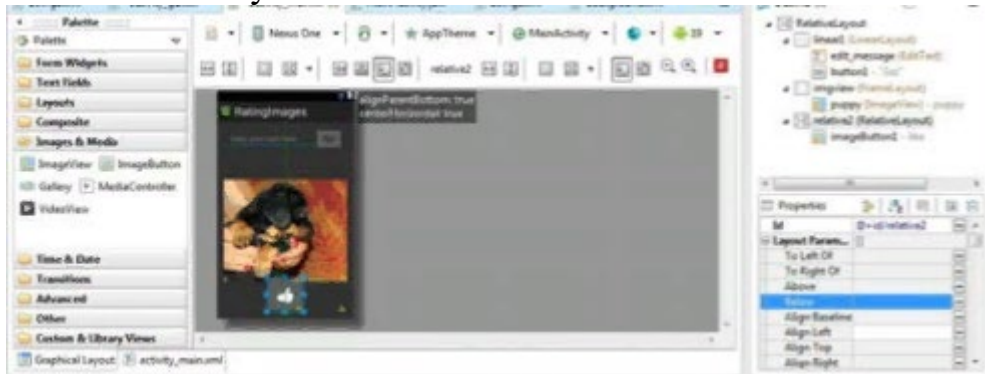
Содержание работы:

Задание 1. Создать кнопки оценивания в приложении, созданном в Практической работе № 14.

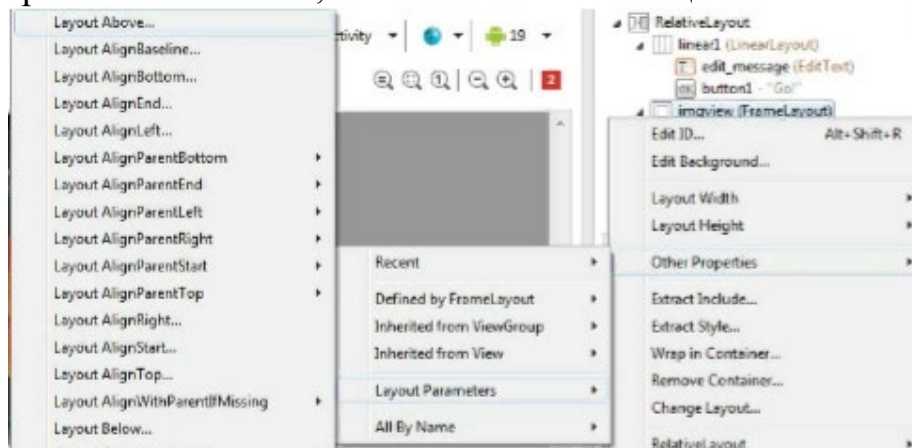
1. Откройте приложение.

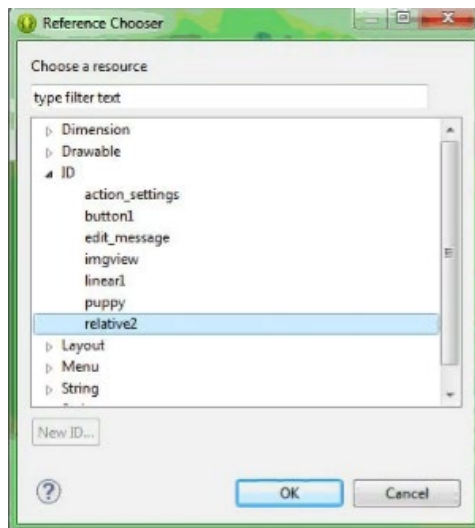
2. Добавьте на форму <RelativeLayout>, задайте для него ширину и высоту wrap_content, и укажите id. Снова в папку res/drawable/ нужно добавить файлы. Найдите изображения "Палец вверх" и "Палец вниз", и поместите их в эту папку, после чего обновите её.

3. Добавьте <ImageButton>, выберите изображение "Палец вверх", и переместите <RelativeLayout> в такое положение

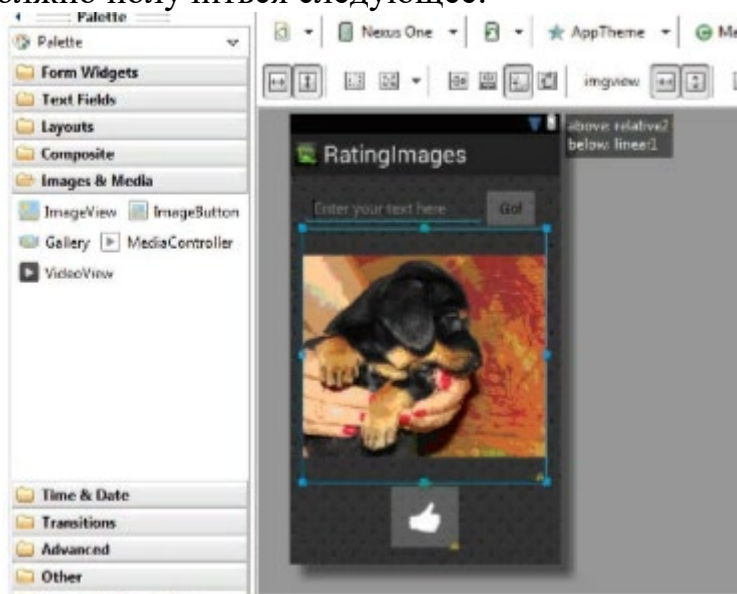


4. Укажите "рамке" быть выше, чем поле с кнопкой оценки:





5. В результате должно получиться следующее:



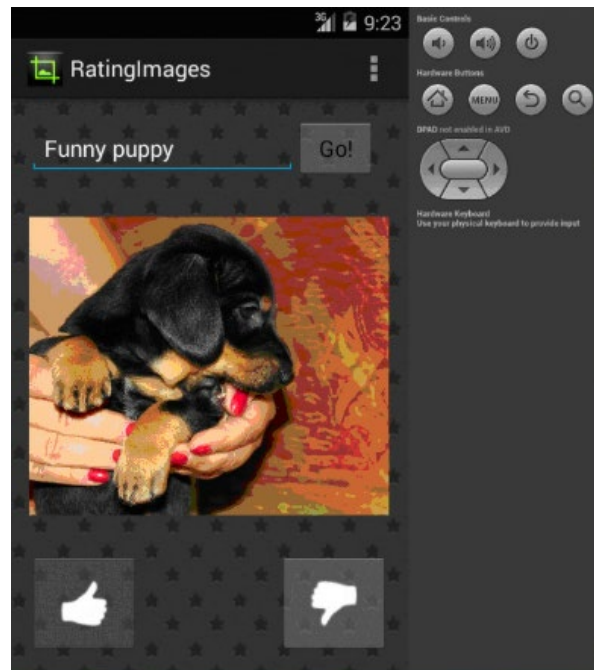
6. Добавьте еще одну кнопку - кнопку "Палец вниз". Она "наложилась" на первую кнопку. Чтобы это исправить, сделайте с `<RelativeLayout>` то же самое, что и с `<LinearLayout>`: растяните элемент влево и вправо, до получения такого результата



7. Теперь расставьте кнопки по краям так, чтобы они "прикрепились" к краям



8. Сохраните. Теперь можно запустить эмулятор и посмотреть, что получилось.



ПРАКТИЧЕСКАЯ РАБОТА № 17

Тема: Обработка событий: подсказки

Цель работы: научиться использовать обработчики событий – добавление подсказок.

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Справочный материал:

EditText Floating Labels (всплывающие подсказки для EditText) в Android.

Содержание работы:

Задание 1. Создать приложение в Android Studio, добавить всплывающие подсказки.

1.Создайте приложение

2. Для начала нужно прописать следующие зависимости в файле **build.gradle**, который лежит на уровне приложения:

```
dependencies {  
    compile 'com.android.support:design:23.2.+'  
}
```

3.Для отображения всплывающих подсказок над полем EditText используется TextInputLayout, который позволяет обернуть находящийся в нем view-элемент, в нашем случае это поле EditText. Такая компоновка также позволяет показать сообщения об ошибках, расположенные ниже поля EditText. Когда поле EditText в фокусе, назначенная подсказка-«поплавок» будет показана на верхней левой стороне.

4. Теперь нам нужно внести изменения в основной макет приложения:

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingLeft="12dp"  
    android:paddingRight="12dp"  
    android:paddingTop="8dp"  
    android:paddingBottom="8dp">  
    <android.support.design.widget.TextInputLayout  
        android:id="@+id/login_layout"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:layout_marginTop="8dp">  
        <EditText  
            android:id="@+id/edit_text_email"  
            android:layout_width="match_parent"  
            android:layout_height="wrap_content"
```

```

android:inputType="textEmailAddress"
android:hint="@string/username" />
</android.support.design.widget.TextInputLayout>
<android.support.design.widget.TextInputLayout
android:id="@+id/password_layout"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="8dp">
<EditText
android:id="@+id/edit_text_password"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:inputType="textPassword"
android:hint="@string/password" />
</android.support.design.widget.TextInputLayout>
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/sign_in"
android:id="@+id/sign_in_button"
android:layout_gravity="center_horizontal" />
</LinearLayout>

```

5. Ну и соответственно унаследоваться от AppCompatActivity, чтобы изменения стали отображаться:

```

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity); } }

```

6. Также не забываем прописать строковые ресурсы:

```

<resources>
<string name="app_name">EditText Floating Labels Demo</string>
<string name="username">Username</string>
<string name="password">Password</string>
<string name="sign_in">Sign In</string>
<string name="login_error">Username can not be empty</string>
</resources>

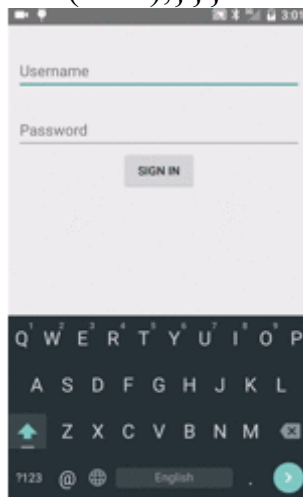
```

7. Кроме подсказки, мы можем показать ошибку в TextInputLayout. Для этого нужно воспользоваться методом setErrorEnabled (...) и setError (...). В качестве примера можно проверить поле входа в систему во время фокусировки на

другой view-элемент. Для этого достаточно обновить наш метод onCreate(...) и реализовать интерфейс onFocusChange(...) с его методами.

```
public class MainActivity extends AppCompatActivity implements
View.OnFocusChangeListener {
    TextInputLayout mUsernameLayout;
    EditText mUsername;
    EditText mPassword;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
        mUsernameLayout = (TextInputLayout) findViewById(R.id.login_layout);
        mUsername = (EditText) findViewById(R.id.edit_text_email);
        mPassword = (EditText) findViewById(R.id.edit_text_password);
        mUsername.setOnFocusChangeListener(this);
        mPassword.setOnFocusChangeListener(this); }
    @Override
    public void onFocusChange(View v, boolean hasFocus) {
        if (v != mUsername && mUsername.getText().toString().isEmpty()) {
            mUsernameLayout.setErrorEnabled(true);
            mUsernameLayout.setError(getResources().getString(R.string.login_error));
        } else {
            mUsernameLayout.setErrorEnabled(false);}}}

```



*Вот так в итоге
это будет
выглядеть.*

ПРАКТИЧЕСКАЯ РАБОТА № 18

Тема: Обработка событий: подсказки

Цель работы: применение практических навыков по обработке событий.

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Добавить систему текстовых подсказок в мобильное приложение, созданное в Практической работе № 15.

ПРАКТИЧЕСКАЯ РАБОТА № 19

Тема: Подготовка стандартных модулей

Цель работы: научиться создавать приложения, состоящие из нескольких активностей, и диалоговые окна, а также познакомиться с элементами тач-интерфейса

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Разработать многоэкранное приложение со списком

1. Создайте проект MultiScreen. В java-файле замените Activity на ListActivity. Класс ListActivity разработан таким образом, что на экране есть только прокручиваемый список и ему не нужна дополнительная разметка. Поэтому файл activity_main.xml можно удалить. Также следует удалить следующую строчку из класса MultiScreen: setContentView(R.layout.activity_main); т.к. layout-файл мы только что удалили.

2. Теперь нам нужен посредник, который свяжет список и названия элементов этого списка. Для этого в Android есть интерфейс Adapter. Нам потребуется наследник этого класса ArrayAdapter:

```
new ArrayAdapter(Context context, int textViewResourceId, String[] objects)
```

В качестве аргументов ArrayAdapter потребует текущий контекст, идентификатор ресурса с разметкой для каждой строки, массив строк. Мы можем указать в качестве текущего контекста ListActivity (можно использовать ключевое слово this), готовый системный идентификатор ресурса (android.R.layout.simple_list_item_1) и созданный массив строк (String[] islands = {"Канары", "Курилы", "Мальдивы", "Филиппины"};). А выглядеть это будет так:

```
private ArrayAdapter<String> adapter;  
adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,  
islands);
```

Обратите внимание на строчку android.R.layout.simple_list_item_1. В ней уже содержится необходимая разметка для элементов списка. Если вас не устраивает системная разметка, то можете создать собственную разметку в xml-файле и подключить её. Осталось только подключить адаптер: setListAdapter(adapter);

3. Теперь нам нужно подключить обработку нажатия. Для этого необходимо знать, на какой пункт списка осуществляется нажатие. Существует специальный интерфейс OnItemClickListener с методом onItemClick(). Подключаем обработчик:

```
getListView().setOnClickListener(itemListener);
```

```
Набираем все в том же onCreate  
OnItemClickListener itemListener = new  
OnItemClickListener(){  
    }  
}
```

Если подчеркнётся первое слово, то импортируем нужный класс. Далее подведём курсор ко второму слову new OnItemClickListener. Нам будет

предложено добавить метод (Add unimplementedmethod). Соглашаемся и получаем заготовку:

```
OnItemClickListener itemListener = new OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,  
long arg3) {  
        // TODO Auto-generated method stub    }  
};
```

Меняем имена переменных arg на более привычные и понятные public void onItemClick(AdapterView<?> parent, View v, int position, long id);

4. Осталось описать, что будет происходить при нажатии на элемент. По нашей задумке каждый элемент будет открывать новое окно с соответствующим содержимым. Для начала следует создать еще 4 класса: Canari, Curili, Maldivi, Philippini, и 4 xml-оболочки к ним. Можно просто создать и копировать по одному файлу в обеих директориях, меняя только название и содержимое. Например, создадим файлы Canari.java и canari.xml типа Activity

Обратите внимание, что как только мы создали ещё один java-файл, нужно записать его в файл манифеста, иначе приложение не будет видеть этот новый класс. Файл AndroidManifest.xml находится сразу подпапкой res. Добавьте код с именем класса в тегах <activity> </activity>, сразу под главной активностью.

```
<activity android:name="com.mypackage.multiscreen.Canari"  
    android:label="@string/can" >  
</activity>
```

Строку can нужно создавать в файле strings.xml, также как и другие строки.

Перейдем в файл canari.xml и создадим на экране TextView. Экраны будут отличаться друг от друга картинками. Возьмите 4 любые картинки с вашего компьютера (можете также найти их в интернете) и перетащите из проводника Windows в папку res\drawable. Теперь вы можете поместить элемент ImageView на экран и, выбрав нужную картинку из ресурсов проекта, подключить ее. Остальные экраны создаются аналогично.

5. Теперь перейдем в главный класс и опишем обработку события onItemClick(). Создадим переключатель, который зависит от номера элемента.

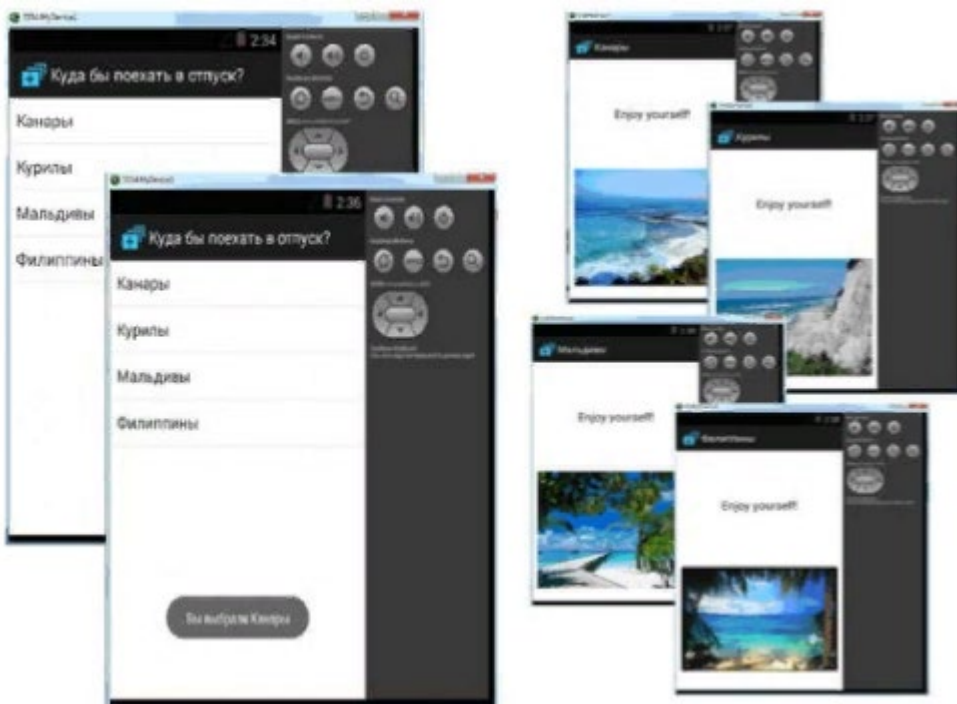
```
switch (position) {  
case 0:  
break;  
case 1:  
break;  
case 2:  
break;  
case 3:  
break;}
```

Для запуска нового экрана необходимо создать экземпляр класса Intent и указать класс, на который будем переходить (у нас их 4, поэтому для каждого случая выбираем свою). После этого вызывается метод startActivity(), который и запускает новый экран. `Intent intent = new Intent(MultiScreen.MainActivity.this, Canari.class); startActivity(intent);`

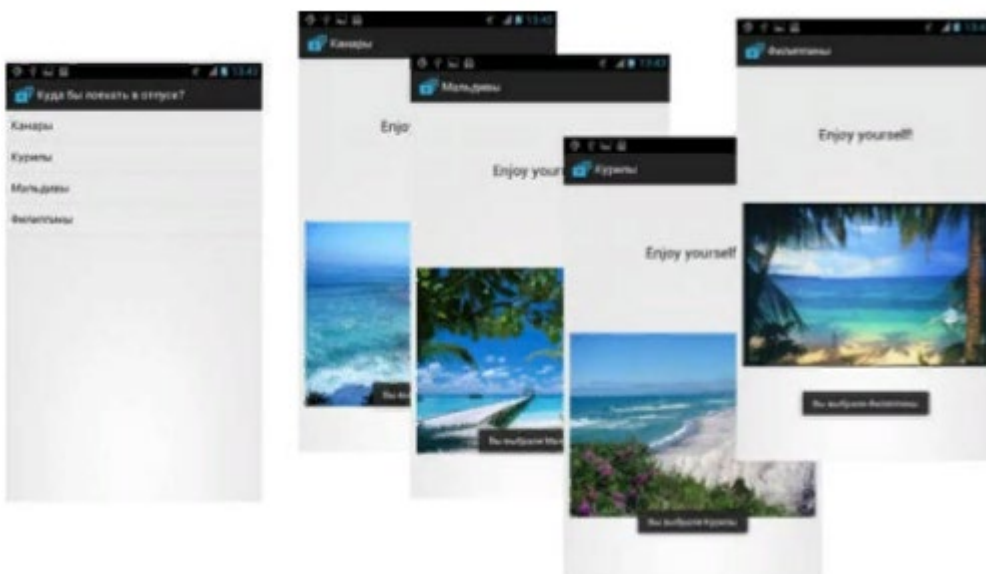
6. Теперь осталось добавить всплывающее окно Toast, которое будет показывать, какой элемент мы выбрали. Этот виджет можно импортировать так же, как и предыдущие. Нам потребуется метод makeText(), у которого есть три параметра: контекст приложения, текстовое сообщение и продолжительность времени показа уведомления.

`Toast.makeText(getApplicationContext(), "Вы выбрали " + parent.getItemAtPo`

7. Сохраните и запустите на эмуляторе приложение



8. Запустите на устройстве



ПРАКТИЧЕСКАЯ РАБОТА № 20

Тема: Подготовка стандартных модулей

Цель работы: научиться создавать приложения, состоящие из нескольких активностей, и диалоговые окна, а также познакомиться с элементами тач-интерфейса

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Создание диалогового окна

1. Создайте новый проект Dialog
2. Создайте кнопку на вашей активности и назовите ее "Выбрать фон" (для этого нужно в файле strings создать строку соответствующего содержания). Присвойте активности и кнопке id.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/relativeLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <Button
        android:id="@+id/background_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="174dp"    android:text="@string/bg"/>
</RelativeLayout>
```

3. Наше приложение меняет фон на выбранный. Значит, нам нужно создать цвета и их названия в файле strings.xml. Как вы помните, этот файл находится в папке values, которая в свою очередь находится в папке res. Также создадим строку messages, которая нам понадобится для диалогового окна.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Dialog</string>
    <string name="action_settings">Settings</string>
    <string name="bg">Выбрать фон</string>
    <string name="message">Хотите поменять фон?</string>
    <color name="redColor">#FFFF0000</color>
    <color name="yellowColor">#FFFF00</color>
    <color name="greenColor">#FF00FF00</color>
    <string name="red">Красный</string>
    <string name="yellow">Жёлтый</string>
    <string name="green">Зелёный</string>
</resources>
```

4. Перейдем в файл MainActivity.java. Создайте следующие переменные:
private Button bgButton;
public RelativeLayout relativeLayout;

Context context;

Если компилятор подчеркивает тип и сообщает об ошибке, например, подчеркивает Context, наведите курсор на подчеркнутое слово: должно появиться контекстное меню, предлагающее варианты, как можно исправить ошибку. Выберите Import 'Context', чтобы импортировать библиотеку.

5. Теперь нужно описать, что будет происходить при нажатии на нашу кнопку. Для начала свяжем объекты из activity_main.xml и переменные в MainActivity.java через id (в onCreate):

```
bgButton = (Button)findViewById(R.id.background_button);
```

```
relativeLayout = (RelativeLayout)findViewById(R.id.relativeLayout);
```

Context - это объект, который предоставляет доступ к базовым функциям приложения.

Добавляем в код context = MainActivity.this;

6. Теперь нужно добавить событие onClick и навесить на кнопку OnClickListener. Добавляем в заголовок класса MainActivity implements OnClickListener. Чтобы связать кнопку и Listener в onCreate пишем bgButton.setOnClickListener(this);

Создадим теперь событие onClick

@Override

```
public void onClick(View v){ }
```

В этом объекте создаем собственно диалог и называем его: AlertDialog.Builder builder = new AlertDialog.Builder(this);

```
builder.setTitle(R.string.message);
```

```
AlertDialog alert = builder.create(); alert.show();
```

Мы будем создавать диалоговое окно, предоставляющее пользователю выбор из списка. Для этого потребуется ещё одна переменная, которая сформирует список из имеющихся строк.

```
final CharSequence[] items = {getText(R.string.red), getText(R.string.yellow),  
getText(R.string.green)};
```

7. Сформируем собственно наш список и зададим еще один Listener, который будет менять цвет фона на выбранный:

```
builder.setItems(items, new DialogInterface.OnClickListener() {
```

```
public void onClick(DialogInterface dialog, int item) {
```

```
switch (item) {
```

```
case 0: { relativeLayout.setBackgroundResource(R.color.redColor);
```

```
break;}
```

```
case 1: {relativeLayout.setBackgroundResource(R.color.yellowColor);
```

```
break;}
```

```
case 2: {relativeLayout.setBackgroundResource(R.color.greenColor);
```

```
break;} }
```

8. Осталось добавить в каждый case всплывающие окна Toast, и приложение полностью готово!

```
Toast.makeText(context, R.string.green, Toast.LENGTH_LONG).show();
```

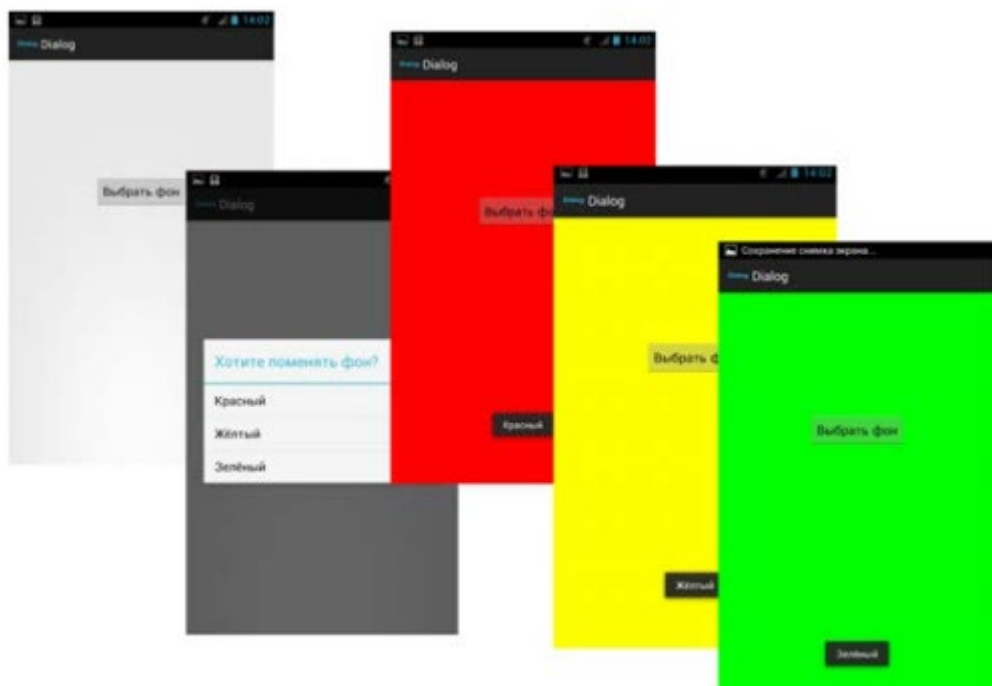
```
package com.mypackage.dialog;
```

```

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;import android.widget.RelativeLayout;
import android.widget.Toast;
public class MainActivity extends Activity implements OnClickListener {
private Button bgButton;
    public RelativeLayout relativeLayout;
Context context;
@Override    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
bgButton = (Button) findViewById(R.id.background_button);
bgButton.setOnClickListener(this);
context = MainActivity.this;
relativeLayout = (RelativeLayout)findViewById(R.id.relativeLayout);    }
@Override
public void onClick(View v) {
final CharSequence[] items = {getText(R.string.red),
getText(R.string.yellow),getText(R.string.green)    };
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle(R.string.message);
builder.setItems(items, new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int item) {
switch (item) {
case 0: { relativeLayout.setBackgroundResource(R.color.redColor);
Toast.makeText(context, R.string.red, Toast.LENGTH_LONG).show();
break;}
case 1: {relativeLayout.setBackgroundResource(R.color.yellowColor);
Toast.makeText(context, R.string.yellow, Toast.LENGTH_LONG).show();
break;}
case 2: {relativeLayout.setBackgroundResource(R.color.greenColor);
Toast.makeText(context, R.string.green, Toast.LENGTH_LONG).show();
break;}    }    }    });
AlertDialog alert = builder.create();
    alert.show();    }}

```

9.Сохраните и запустите приложение



ПРАКТИЧЕСКАЯ РАБОТА № 21

Тема: Обработка событий: переключение между экранами

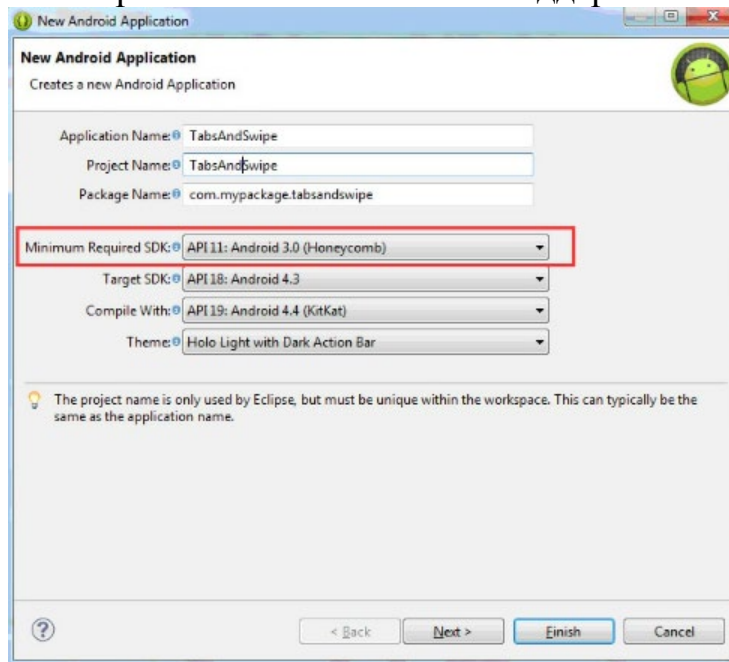
Цель работы: применение практических навыков по созданию многооконного приложения.

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

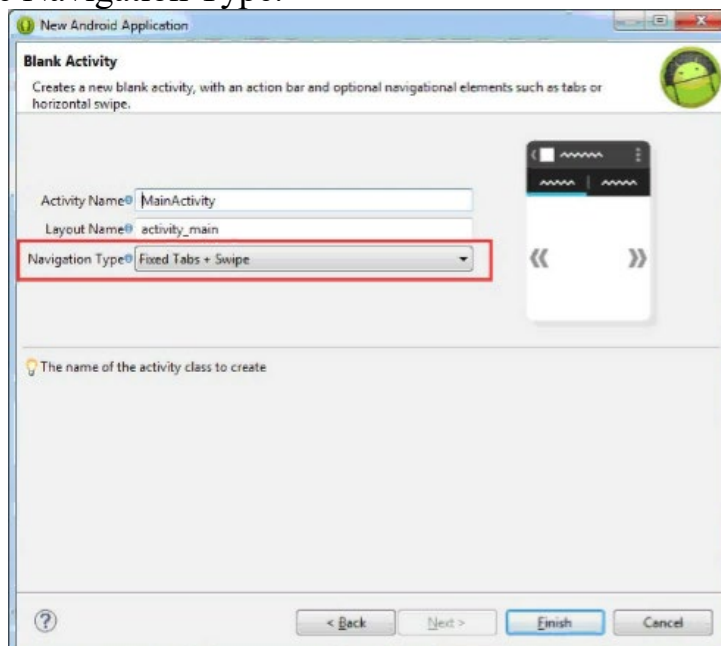
Содержание работы:

Задание 1. Создание приложения со слайдингом из шаблона

1. Создайте проект TabsAndSwipe. Обратите внимание: чтобы использовать стандартный шаблон активности Fixed Tabs + Swipe, вам необходимо при создании проекта указать Minimum Required SDK не меньше, чем API11, т.к. в более ранних версиях этот шаблон не поддерживается.



Следующие три окна оставляем без изменений. Далее в окне Blank Activity выбираем в графе Navigation Type.



2. Посмотрите на структуру проекта: у вас появились два xml-файла в папке res\layout - activity_main.xml и fragment_main_dummy.xml.

3. Запустите приложение, чтобы убедиться, что все компилируется правильно.

4. Создайте три копии файла fragment_main_dummy.xml. Назовите их соответственно first_tab, second_tab и third_tab. Присвойте элементу TextView в каждом файле уникальный id. Можете поместить на экраны какие-нибудь элементы, например картинки или надписи.

5. Теперь переходим к файлу MainActivity.java. Нас интересует класс

```
public static class DummySectionFragment extends Fragment
```

Делаем три копии данного класса со всем содержимым, называя их соответственно FirstActivity, SecondActivity, ThirdActivity.

6. Поменяйте в них следующие строки

```
View rootView = inflater.inflate(R.layout.fragment_main_dummy, container, false);
TextView dummyTextView = (TextView)
rootView.findViewById(R.id.section_label);
```

Замените R.layout.fragment_main_dummy на R.layout.first и R.id.section_label на R.id.section_label1 соответственно.

7. Поменяйте строки-названия секций в файле strings.xml.

```
<string name="title_section1">Лента</string>
```

```
<string name="title_section2">Фото</string>
```

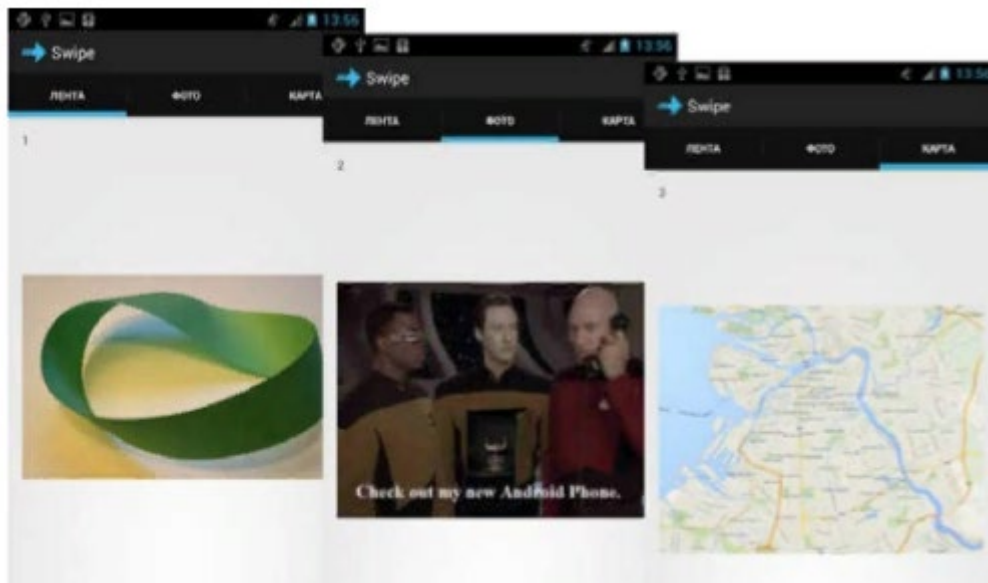
```
<string name="title_section3">Карта</string>
```

8. Теперь переходим к классу SectionsPagerAdapter(). Нас интересует его метод Fragment getItem(), именно его мы будем изменять. Чтобы при перелистывании менялось не только содержимое TextView, но и всего фрагмента, замените код этого метода наследующий:

```
public Fragment getItem(int position) {
// getItem is called to instantiate the fragment for the given page.
// Return a DummySectionFragment (defined as a static inner class
// below) with the page number as its lone argument.
Fragment fragment=null;
Bundle args;
switch (position) {
case 0:
fragment = new FirstFragment();
args = new Bundle();
args.putInt(FirstFragment.ARG_SECTION_NUMBER, position +
ragment.setArguments(args);
break;
case 1:
fragment = new SecondFragment();
args = new Bundle();
args.putInt(SecondFragment.ARG_SECTION_NUMBER, position + 1);
fragment.setArguments(args);
break;
case 2:
fragment = new ThirdFragment();
args = new Bundle();
```

```
args.putInt(ThirdFragment.ARG_SECTION_NUMBER, position + 1);  
fragment.setArguments(args);  
break;    }  
return fragment;
```

9. Приложение готово, можно запускать



ПРАКТИЧЕСКАЯ РАБОТА № 22

Тема: Обработка событий: переключение между экранами

Цель работы: научиться применять обработчики событий для переключения между экранами.

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Подумайте над собственным приложением, сочетающим различные возможности проектирования многооконных приложений. Создайте прототип этого приложения и настройте его пользовательский интерфейс.

ПРАКТИЧЕСКАЯ РАБОТА № 23

Тема: Передача данных между модулями

Цель работы: научиться передавать данные между модулями мобильного приложения.

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Разработать приложения, получающего координаты устройства и отслеживающего их изменение

1.Создадим приложение. Далее будем править java-файл, определяющий класс активности приложения. Внесем в этот класс следующие дополнения:

1.1.Нам потребуется экземпляр класса LocationManager, поэтому в методе onCreate() запишем следующую конструкцию:

```
LocationManager mlocManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
```

экземпляр класса LocationManager, как и большинство системных сервисов, создается с помощью вызова метода getSystemService().

1.2.Укажем, что в приложении необходимо получать обновления координат, сделаем это с помощью вызова метода requestLocationUpdates() класса LocationManager:

```
mlocManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
0, 0, mlocListener);
```

первый параметр метода указывает способ получения координат, для этого используются константы класса LocationManager:

GPS_PROVIDER - сообщает, что координаты определяются с помощью GPS;

NETWORK_PROVIDER - сообщает, что координаты определяются с использованием сигналов сотовых вышек и Wi-Fi.

В случае, когда необходимо получать координаты и с GPS, и от сотовых вышек необходимо вызвать метод requestLocationUpdates() дважды: один раз первый параметр должен быть равен GPS_PROVIDER, второй раз -NETWORK_PROVIDER. Второй и третий параметры метода управляют частотой обновлений. Второй определяет минимальное время между извещениями об обновлениях, третий - минимальное изменение расстояния между извещениями об обновлениях. Если оба эти параметра имеют нулевое значение, то извещения об обновлениях появляются настолько часто, насколько это возможно. Последний параметр указывает на слушателя, который получает вызовы при обновлениях координат. В нашем случае слушателем является переменная mlocListener - реализация интерфейса LocationListener.

1.3. Теперь необходимо объявить переменную mlocListener:

```
LocationListener mlocListener = new LocationListener(){
public void onLocationChanged(Location location) {
tvLat.append(" "+location.getLatitude());
tvLon.append(" "+location.getLongitude()); }
```

```

public void onStatusChanged(String provider, int status, Bundle extras) {}
public void onProviderEnabled(String provider) {}
public void onProviderDisabled(String provider) {}
};

```

Переменная `mlocListener` инициализируется реализацией интерфейса `LocationListener`. Этот интерфейс содержит 4 метода, каждый из которых должен быть определен в реализации интерфейса. Нас интересует метод `onLocationChanged()`, который вызывается каждый раз при изменении показаний GPS-датчика, в этом методе всего лишь выполняется вывод полученных координат в информационные поля.

1.4. Чтобы разрешить получать обновления координат с помощью сигналов от сотовых вышек (`NETWORK_PROVIDER`) или с GPS-датчика (`GPS_PROVIDER`), необходимо в файле манифеста добавить строку:

```

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" /> (для работы с сигналами сотовых вышек)
или <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" /> (для работы GPS датчиком).

```

Без этих полномочий приложение "сломается" во время выполнения, когда попытается запросить обновление координат. Если в приложении в качестве источника координат используются и `NETWORK_PROVIDER`, и `GPS_PROVIDER`, в манифесте достаточно запросить только полномочия на `ACCESS_FINE_LOCATION`. Тогда как `ACCESS_COARSE_LOCATION` позволяет работать только с `NETWORK_PROVIDER`.

1.5. После создания приложения, разумеется, необходимо протестировать его работу. Проще всего это сделать, используя реальное устройство под управлением Android, но если устройства под рукой нет, можно использовать виртуальное устройство. Однако при этом необходимо решить вопрос, каким образом имитировать передачу данных о местоположении на эмулятор. Проще всего воспользоваться для этого DDMS.

1.6. Код программы:

```

package com.example.lab5_4_geolocation;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.widget.TextView;
public class MainActivity extends Activity {
    TextView tvOut;
    TextView tvLon;
    TextView tvLat;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

```

```

setContentView(R.layout.activity_main);
tvOut = (TextView)findViewById(R.id.textView1);
tvLon = (TextView)findViewById(R.id.longitude);
tvLat = (TextView)findViewById(R.id.latitude);
//Получаем сервис
LocationManager mlocManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
LocationListener mlocListener = new LocationListener() {
public void onLocationChanged(Location location) {
//Called when a new location is found by the network location provider.
tvLat.append(" "+location.getLatitude());
tvLon.append(" "+location.getLongitude());    }
public void onStatusChanged(String provider, int status, Bundle extras) {}
public void onProviderEnabled(String provider) {}
public void onProviderDisabled(String provider) {}
};
//Подписываемся на изменения в показаниях датчика
mlocManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
mlocListener);
//Если gps включен, то ... , иначе вывести "GPS is not turned on..."
if (mlocManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
tvOut.setText("GPS is turned on...");
} else {
tvOut.setText("GPS is not turned on...");    } }}

```

ПРАКТИЧЕСКАЯ РАБОТА № 24

Тема: Тестирование и оптимизация мобильного приложения

Цель работы: научиться создавать тестирующее приложение

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Справочный материал:

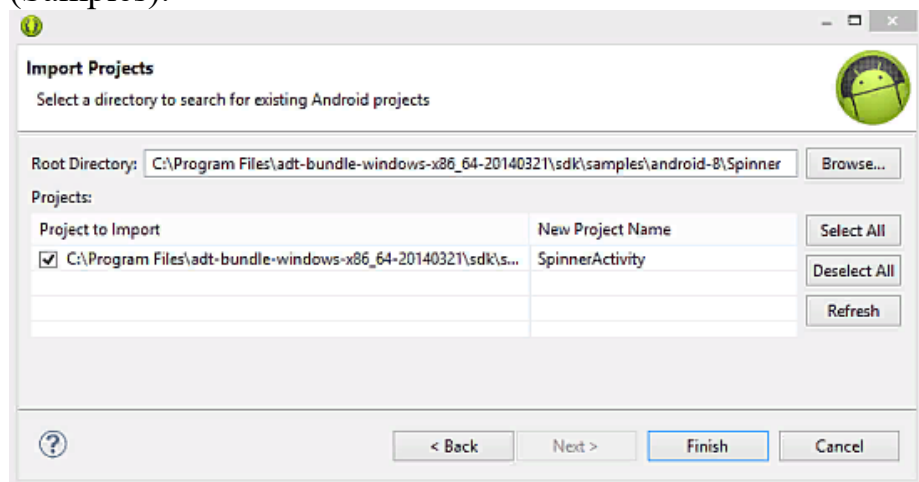
Android SDK включает мощные инструменты для тестирования приложений. Инструменты расширяют JUnit дополнительными возможностями; предоставляют готовые к использованию классы для объектов, имитирующих Android систему; дают контроль над главным приложением во время его тестирования.

В работе используем простое Android приложение из комплекта Android SDK, для которого создадим тестирующее приложение, тем самым продемонстрируем инструменты Android тестирования, включенные в Android IDE.

Содержание работы:

Задание 1. Создать тестирующее приложение

1. В качестве приложения для тестирования будем использовать уже готовое приложение, входящее в набор созданных разработчиками Android SDK примеров, приложение называется Spinner и содержится в наборе примеров (Samples).

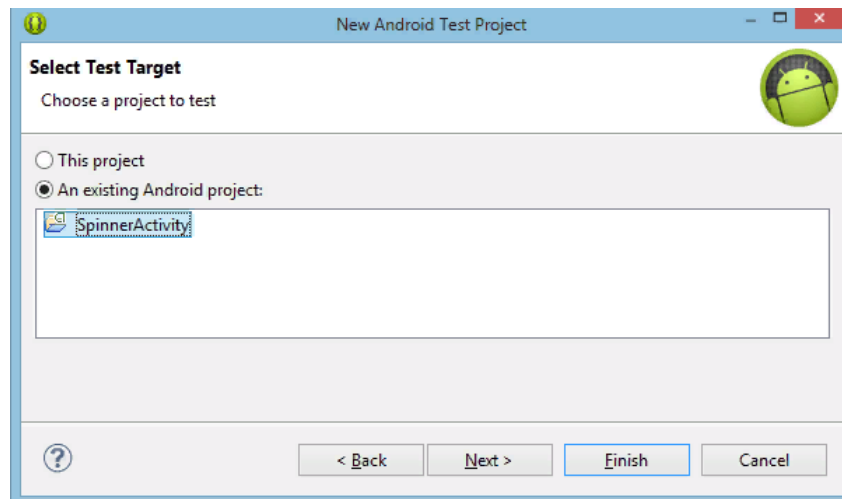


2. Запустите Eclipse, создайте новое Android приложение, используя уже существующее: New->Project...->Android->Android Project from Existing Code, в открывшемся диалоге выберите расположение проекта. Убедитесь, что приложение запускается и что-то выполняет.

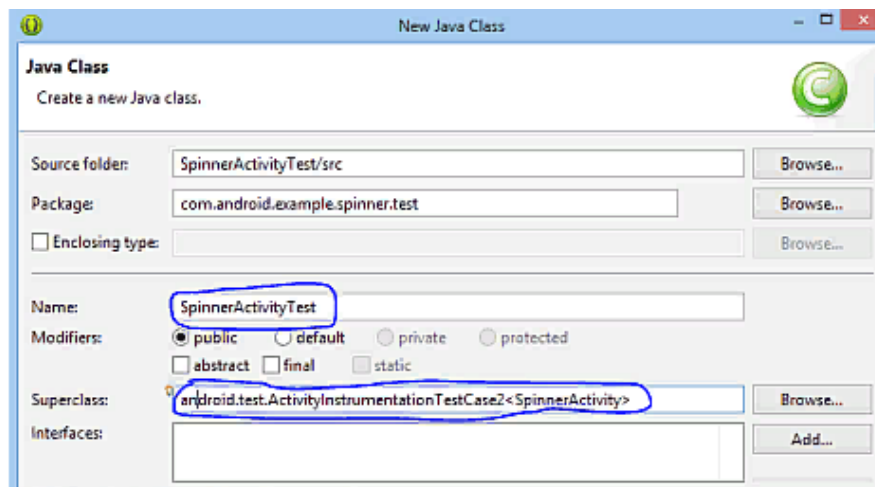
3. Создадим тестирующее приложение.

В Eclipse New->Project...->Android->Android Test Project

Задайте имя проекта, обычно имя тестирующего проекта собирается следующим образом: имя тестируемого проекта + слово Test, в нашем случае получится SpinnerActivityTest. Далее необходимо указать тестируемый проект



4. Создадим класс тестов



В пакете, расположенном в папке src/ проекта SpinnerActivityTest, создадим новый класс. Имя класса: SpinnerActivityTest, суперкласс: android.test.ActivityInstrumentationTestCase2<SpinnerActivity>

Класс ActivityInstrumentationTestCase2 спроектирован для выполнения функционального тестирования одной или нескольких активностей приложения.

Чтобы была возможность обращаться к SpinnerActivity необходимо в файл SpinnerActivityTest.java добавить следующую строчку:
import com.android.example.spinner.SpinnerActivity;

5. Добавим конструктор класса тестов:

```
public SpinnerActivityTest(){
    super("com.android.example.spinner", SpinnerActivity.class);
}
```

6. Добавим метод начальных установок

Метод setUp() вызывается перед каждым тестом, используется для инициализации переменных и очистки значений после предыдущих тестов. Также можно использовать метод tearDown(), который вызывается после каждого теста.

Код метода:

```
@Override
protected void setUp() throws Exception {
```

```

        super.setUp();
        setActivityInitialTouchMode(false);
        mActivity = getActivity();
        mSpinner = (Spinner)mActivity.findViewById
(com.android.example.spinner.R.id.Spinner01);
        mPlanetData = mSpinner.getAdapter();
    }

```

Рассмотрим метод:

- `super.setUp()` — вызывает конструктор суперкласса для `setUp()`, как этого требует JUnit;
- `setActivityInitialTouchMode(false)` — выключает режим касаний на эмуляторе и устройстве, если какой-то из тестов передает события нажатия кнопок в приложение, необходимо отключать режим касаний перед запуском любой активности, иначе вызовы будут игнорироваться;
- `getActivity()` — получает ссылку на тестируемую активность, этот вызов также запускает активность, если это до сих пор не сделано;
- `findViewById(int)` — получает ссылку на виджет `Spinner` в тестируемом приложении;
- `getAdapter()` — получает ссылку на адаптер (массив строк) соответствующий `Spinner`.

В класс тестов необходимо добавить следующие элементы:

```

private SpinnerActivity mActivity;
private Spinner mSpinner;
private SpinnerAdapter mPlanetData;

```

И импортировать следующие пакеты:

```

import android.widget.Spinner;
import android.widget.SpinnerAdapter;

```

7. Добавим тест начальных условий

Этот тест проверяет, что тестируемое приложение инициализировано корректно. Метод будет проверять следующее:

- инициализирован слушатель события выбора элемента из списка, этот слушатель вызывается когда в спинере выбирается какой-то элемент;
- инициализирован адаптер, предоставляющий значения для спинера;
- адаптер содержит правильное количество записей.

Код метода:

```

public void testPreConditions() {
    assertTrue(mSpinner.getItemSelectedListener() != null);
    assertTrue(mPlanetData != null);
    assertEquals(mPlanetData.getCount(), ADAPTER_COUNT);}

```

В класс тестов необходимо добавить элемент:

```

public static final int ADAPTER_COUNT = 9;

```

8. Добавим тест интерфейса пользователя

Создадим UI тест, который выбирает элементы из виджета спинер. Тест посылает события нажатия клавиш и проверяет, что выбор соответствует ожидаемым результатам.

Для работы со спинером тест должен запросить фокус и затем установить его в известную позицию, для этого используются методы `requestFocus()` и `setSelection()`. Оба эти метода взаимодействуют с представлениями тестируемого приложения, поэтому должны вызываться специальным образом.

Код для получения фокуса и установки выделения выглядит следующим образом:

```
mActivity.runOnUiThread(  
    new Runnable() {  
        public void run() {  
            mSpinner.requestFocus();  
            mSpinner.setSelection(INITIAL_POSITION);        }    } );
```

Необходимо добавить его в метод:

```
public void testSpinnerUI() {  
    ...}
```

Необходимо добавить в класс тестов элемент:

```
public static final int INITIAL_POSITION = 0;
```

Далее необходимо сделать выбор элемента спинера для этого передадим событие нажатия кнопки в спинер:

```
this.sendKeys(KeyEvent.KEYCODE_DPAD_CENTER);  
for (int i = 1; i <= TEST_POSITION; i++) {  
    this.sendKeys(KeyEvent.KEYCODE_DPAD_DOWN);        }  
this.sendKeys(KeyEvent.KEYCODE_DPAD_CENTER);
```

Необходимо добавить в класс тестов элемент:

```
public static final int TEST_POSITION = 5;
```

Необходимо импортировать:

```
import android.view.KeyEvent;
```

Проверка результатов. Запросить текущее состояние спинера и сравнить его с ожидаемым значением. Вызов метода `getSelectedItemPosition()` позволит получить текущую выбранную позицию, а метод `getItemAtPosition()` возвращает элемент, соответствующий этой позиции (представленный в виде строки). Метод `assertEquals()`; проверяет совпадает ли полученное значение с ожидаемым "Saturn".

Добавьте в метод `public void testSpinnerUI()` следующий код:

```
mPos = mSpinner.getSelectedItemPosition();  
mSelection = (String)mSpinner.getItemAtPosition(mPos);  
TextView resultView = (TextView)mActivity.findViewById(  
    com.android.example.spinner.R.id.SpinnerResult);  
String resultText = (String) resultView.getText();  
assertEquals(resultText,mSelection);
```

Необходимо добавить в класс тестов следующие элементы:

```
private String mSelection;  
private int mPos;
```

И импортировать:

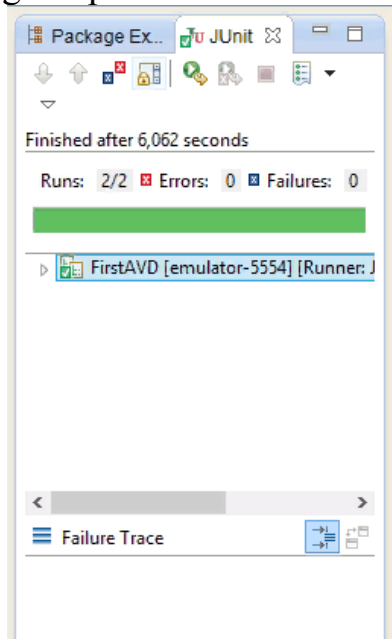
```
import android.widget.TextView;
```

9. Запуск теста и просмотр результатов.

Перед запуском теста необходимо удостовериться, что запущен эмулятор или подключено устройство, на котором будет производиться тестирование.

Для запуска теста в Package Explorer щелкните правой кнопкой на названии проекта тестов (SpinnerActivityTest), а затем выберите Run As→Android JUnit Test

После запуска теста появится новая вкладка JUnit рядом с вкладкой Package Explorer.



Эта вкладка содержит две панели. Верхняя панель содержит информацию о тестах, которые были запущены, нижняя панель содержит запись ошибок.

Информация о запущенных тестах содержит следующие пункты:

- общее время, затраченное на исполнение теста;
- количество запусков (Runs) — количество тестов в классе;
- количество ошибок (Errors) — количество ошибок и исключений, обнаруженных в приложении за время работы теста;
- количество провалов (failures) — количество неудачных испытаний во время работы теста;
- полоса состояния, если тесты прошли успешно, полоса остается зеленой, если тест провален, полоса становится красной;
- под полосой состояния можно увидеть описание каждого класса в тестирующем приложении, можно просмотреть результаты каждого метода тест-класса.

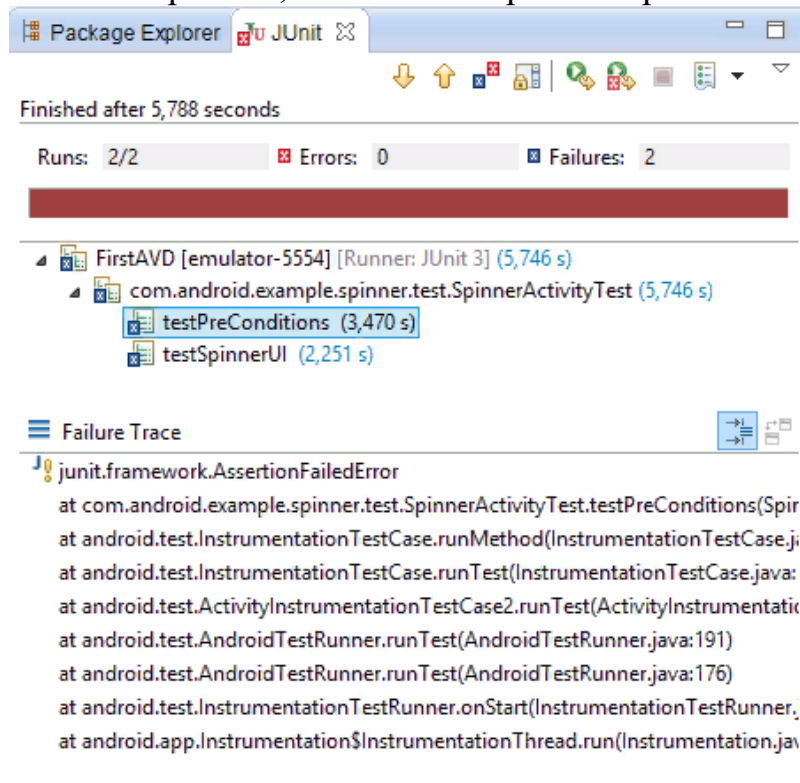
10. Спровоцируем провал некоторых тестов

Посмотрим, что произойдет в Android IDE, когда тест будет провален. Можно быстро увидеть какой тест-класс был провален, найти метод или методы, которые были провалены, используя запись ошибки точно найти проблему.

Испортим в файле SpinnerActivity.java метод onCreate(), в этом файле есть закомментированная строка:

```
// spinner.setOnItemClickListener(null);
```

Снимите комментарий, тем самым проверим работу метода `testPreCondition()` класса `SpinnerActivityTest`, в этом методе в строке: `assertTrue(mSpinner.getItemSelectedListener() != null);` утверждается, что слушатель событий визуального компонента `Spinner` не равен `null`. После удаления символов комментария и запуска тестирующего проекта, это утверждение будет выдавать ошибку, т.е. тест будет провален. Теперь полоса состояния стала красной, количество провалов равно 2.



Чтобы найти место в коде, на котором был провален тест, достаточно щелкнуть мышью по строке `testPreCondition`, выделенной на рисунке в нижней панели `Failure Trace` появится список вызовов, который привел к провалу теста. Первая строка в этой панели сообщает об ошибке, чтобы увидеть место в коде, где произошел провал теста достаточно дважды щелкнуть по строке с ошибкой.

ПРАКТИЧЕСКАЯ РАБОТА № 25

Тема: Тестирование и оптимизация мобильного приложения

Цель работы: научиться тестировать и производить оптимизацию мобильного приложения

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Справочный материал:

Тестирование — это наблюдение за функционированием программного обеспечения в специфических условиях с целью определения степени соответствия *ПО* требованиям к нему.

Тестирование подразделяется на уровни.

- **Модульное тестирование** (автономное или Unit-тестирование). На данном уровне тестируются по отдельности небольшие разработанные компоненты системы, максимально отделенные от других компонентов, но при этом пригодные для тестирования. Обычно проводится сразу после разработки компонента, направлено на оценку соответствия его функциональности спроектированной архитектуре компонентов системы (Component Design).
- **Комплексное тестирование** (сборочное тестирование, Integration testing). На данном уровне тестируются объединенные элементы (компоненты или подсистемы) общей системы, направлено на проверку взаимодействия компонентов в соответствии с архитектурой системы (System Design). Тесты данного уровня обычно проверяют все интерфейсы взаимодействия между компонентами, определенные в системной архитектуре, до тех пор, пока все компоненты не будут разработаны, отлажены и проинтегрированы друг с другом в единую систему.
- **Системное тестирование** (System testing). На данном уровне тестируется разрабатываемая система целиком, проверяется соответствие системным спецификациям (System specification), а также реализация всех функциональных и нефункциональных требований к разрабатываемой системе.
- **Приемочное тестирование** (приемо-сдаточное тестирование, Acceptance testing). На данном уровне производится тестирование завершенной системы заказчиком, конечным пользователем или соответствующим уполномоченным с целью определения соответствия системы требованиям заказчика и ее готовности к внедрению. На этом уровне оформляется процесс передачи продукта от разработчика заказчику.
- **Операционное тестирование** (Release testing). На данном уровне проверяется функционирование системы в среде эксплуатации, оценивается соответствие требованиям пользователя и выполнение роли, определенной в бизнес-модели (Business case или Business model) системы. Необходимо учитывать, что бизнес-модель также может содержать ошибки, поэтому очень важно провести операционное тестирование как финальный шаг валидации.

Содержание работы:

Задание 1. Произведите тестирование каждого уровня всех ранее созданных мобильных приложений. При обнаружении ошибок – оптимизируйте их.

ПРАКТИЧЕСКАЯ РАБОТА № 26

Тема: Тестирование и оптимизация мобильного приложения

Цель работы: применять полученные практические навыки при тестировании и оптимизации мобильных приложений

Оборудование: ПК, программное обеспечение – Android Studio, инструкции по выполнению работы.

Справочный материал:

Основные типы тестирования.

- **Инсталляционное тестирование** (Installation testing). Проверяется корректность инсталляции и деинсталляции программного продукта в среде максимально приближенной к эксплуатационной.
- **Конфигурационное тестирование** (Configuration testing). Проверяется работоспособность при различных конфигурациях, предполагает тестирование работы системы на различных платформах: различных вариантах аппаратной конфигурации, версиях операционной системы и окружения.
- **Тестирование интернационализации** (Internationalization testing). Проверяется возможность быстрой локализации продукта под необходимую локаль потенциальных пользователей системы.
- **Дымное тестирование** (Smoke testing). Первый прогон программы, проверяется готовность программы для проведения более обширного тестирования. Чаще всего тестируется основная бизнес-логика программы.
- **Регрессионное тестирование** (Regression testing). Повторное тестирование после внесения изменений в программный продукт или его окружение. Предполагается выявление потенциальных проблем, которые могли возникнуть в результате изменений, проверка исправления найденных ранее дефектов.
- **Функциональное тестирование** (Functional testing). Проверяется соответствие продукта функциональным требованиям и спецификациям.
- **Тестирование графического интерфейса пользователя** (User interface testing). Предполагается проверка работы интерфейса в целом и его отдельных компонентов, выполняется поиск ошибок функциональности посредством интерфейса.
- **Тестирование удобства использования** (Usability testing). Включает в себя тесты на человеческий фактор, эстетику интерфейса и его непротиворечивость, наличие и качество оперативной и контекстной помощи, руководств пользователя и учебных материалов.
- **Тестирование производительности** (Performance testing). Проверяется скорость работы системы (время отклика, частота транзакций) в имитационной и реальной средах, с целью установить реальную производительность программного продукта.

Содержание работы:

Задание 1. Произведите тестирование каждым типом всех ранее созданных мобильных приложений. При обнаружении ошибок – оптимизируйте их.

Информационное обеспечение обучения

Печатные издания:

Основные учебные издания:

1. Введение в разработку приложений для ОС Android: учебное пособие / Ю. В. Березовская, О. А. Юфрякова, В. Г. Вологодина [и др.]. — 3-е изд. — Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 427 с. — ISBN 978-5-4497-0890-8. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <https://www.iprbookshop.ru/102000.html>

Дополнительные учебные издания:

2. Пирская, Л. В. Разработка мобильных приложений в среде Android Studio: учебное пособие / Л. В. Пирская. — Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2019. — 123 с. — ISBN 978-5-9275-3346-6. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <https://www.iprbookshop.ru/100196.html>
3. Семакова, А. Введение в разработку приложений для смартфонов на ОС Android: учебное пособие для СПО / А. Семакова. — Саратов: Профобразование, 2021. — 102 с. — ISBN 978-5-4488-0994-1. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/102187>

Электронные издания (электронные ресурсы)

4. Учебники по программированию <http://programm.ws/index.php>

Электронно-библиотечная система:

5. ЭБС «IPRbooks», ООО «Ай Пи Ар Медиа»
6. ЭБС «Znanium»
7. ЭБС «PROФобразование»
8. ЭБС «Book.ru»