Лабораторная работа № 1
Установка Android Studio и первый проект

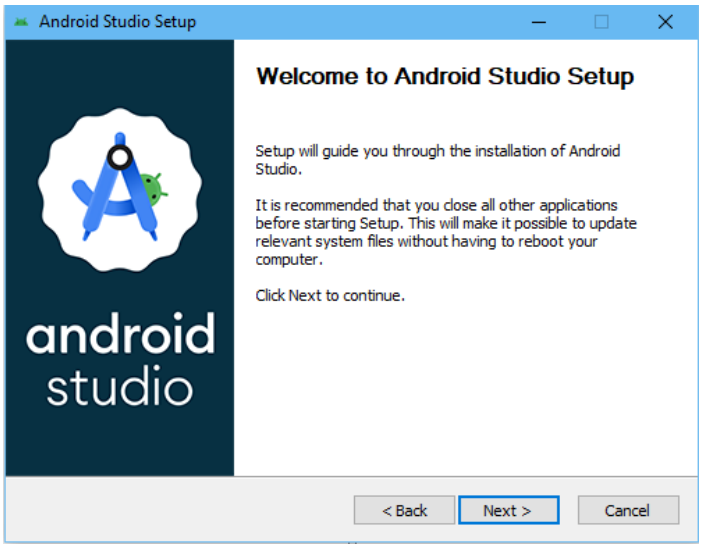
Задание 1: скачать и установить последнюю версию Android Studio, установить необходимые компоненты SDK, сконфигурировать эмулятор, создать новый проект, внести правки и запустить проект на эмуляторе.

Установка Android Studio

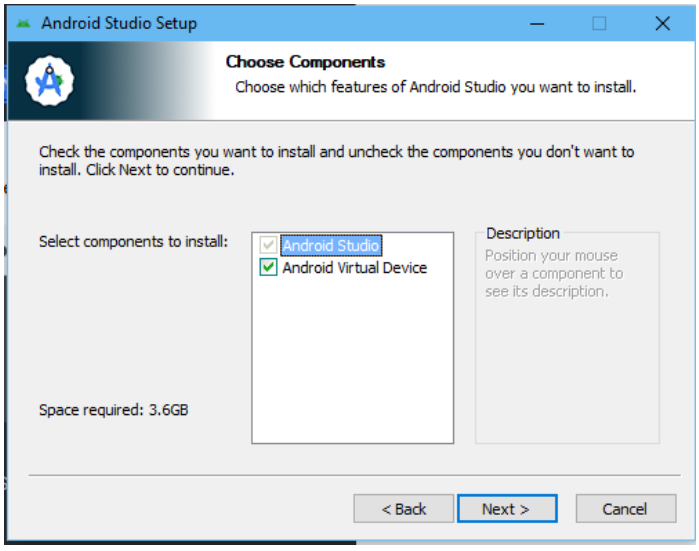
Для разработки под Android с помощью Jetpack Compose необходимо установить Android Studio, вместе с которой также будут установлены все необходимые компоненты.

Загрузить Android Studio можно с официального сайта с адреса https://developer.android.com/studio

Запускаем загруженное приложение – отобразится начальный экран программы установки:

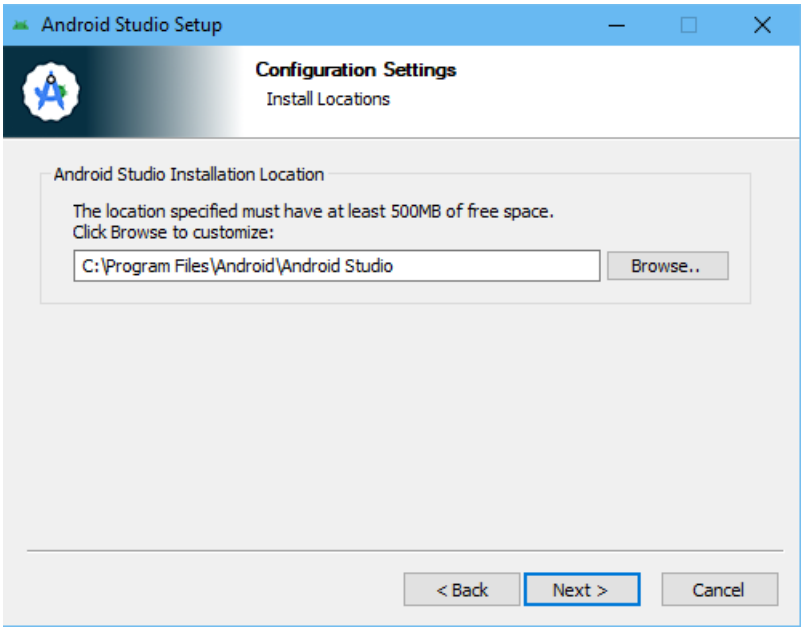


После нажатия Next откроется окно с предложением выбора компонентов для установки



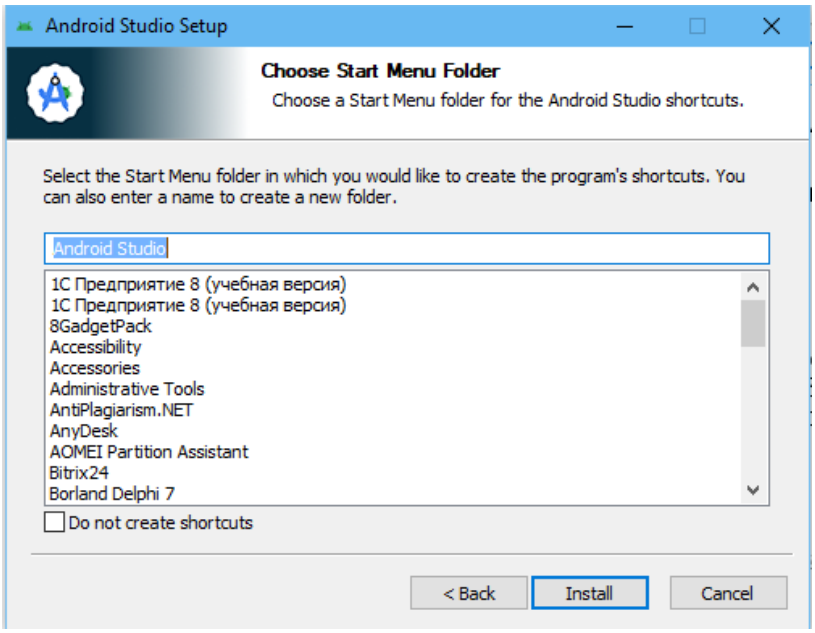
Здесь надо указать, надо ли устанавливать компонент Android Virtual Device, то есть эмулятор. Если для тестирования будут использованы реальные устройства – смартфоны и планшеты, то в принципе эмулятор устанавливать необязательно.

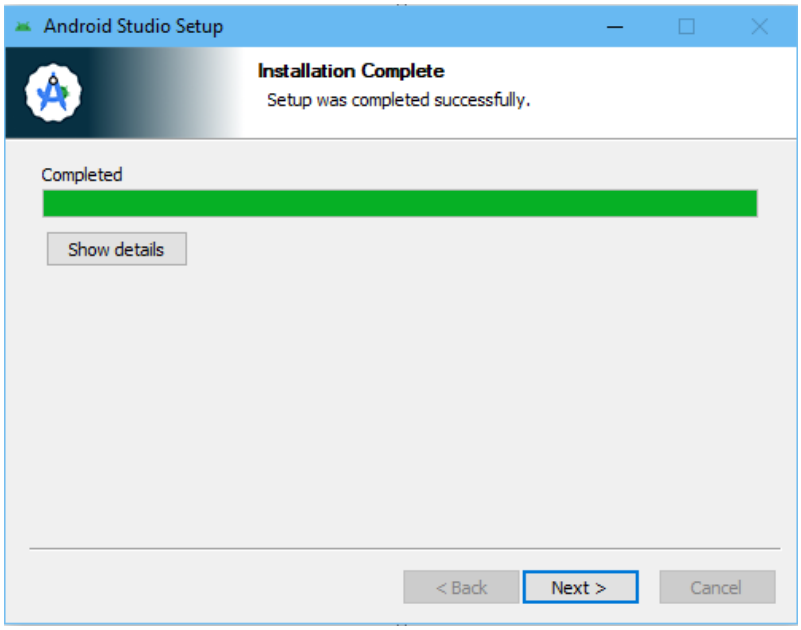
На следующем шаге надо указать папку для установки Android Studio:



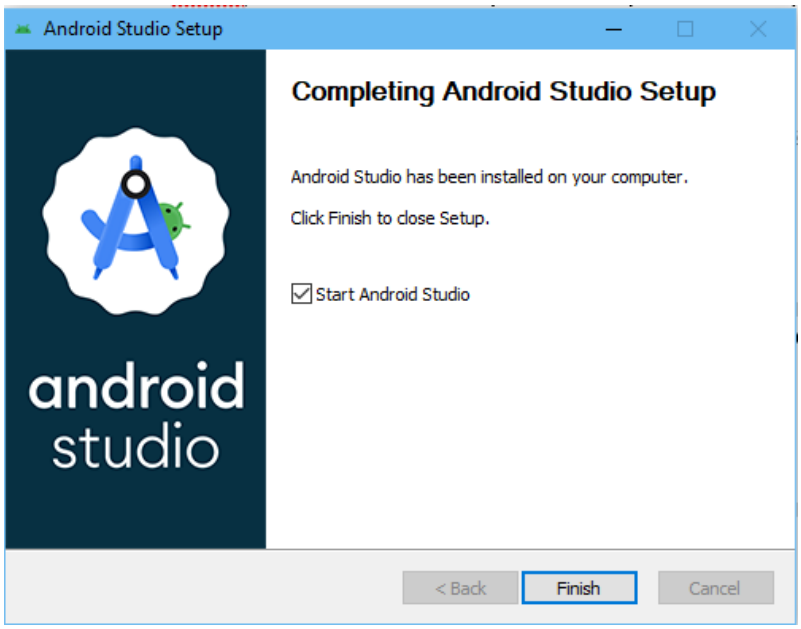
По умолчанию это папка "C:\Program Files\Android\Android Studio".

Далее надо указать, создавать ли иконку Android Studio на рабочем столе, после чего собственно и начнется установка.





Когда программа установится, появится следующее окно.



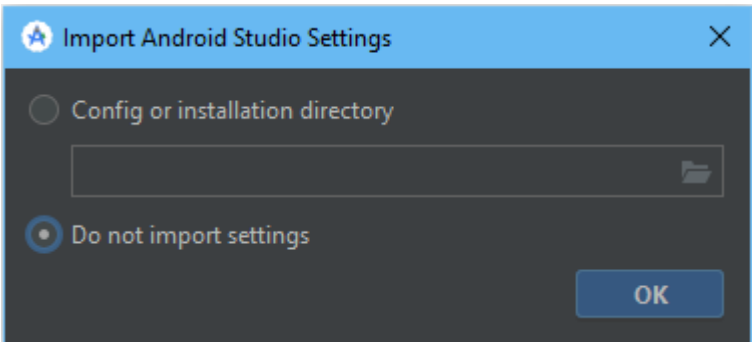
Это сообщение о завершении установки. Нажимаем Finish, предварительно убедившись, что установлен флажок **Start Android Studio**.

При первом запуске Android Studio первым делом надо установить ряд дополнительных компонентов, в частности, Android SDK.

Установка Android SDK

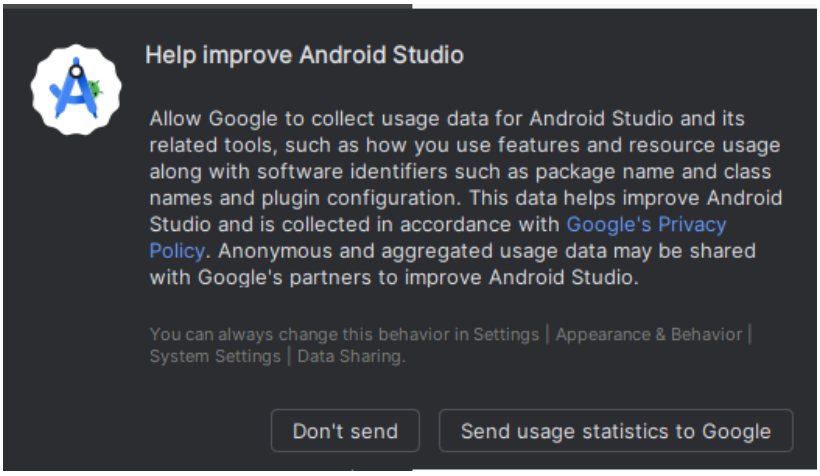
Итак, запускаем Android Studio.

При запуске будет предложено импортировать предыдущие настройки.

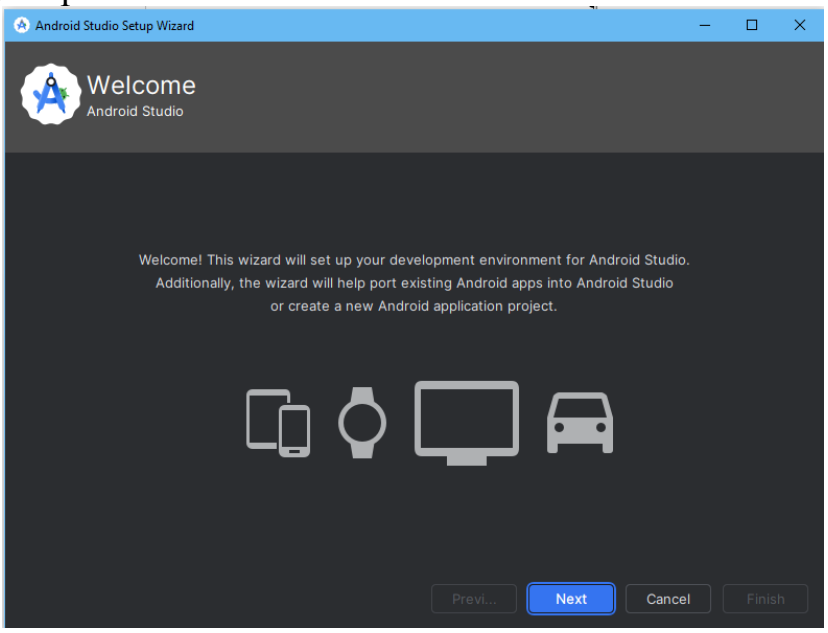


Если никаких предыдущих настроек нет или нет желания импортировать их, следует выбрать второй вариант.

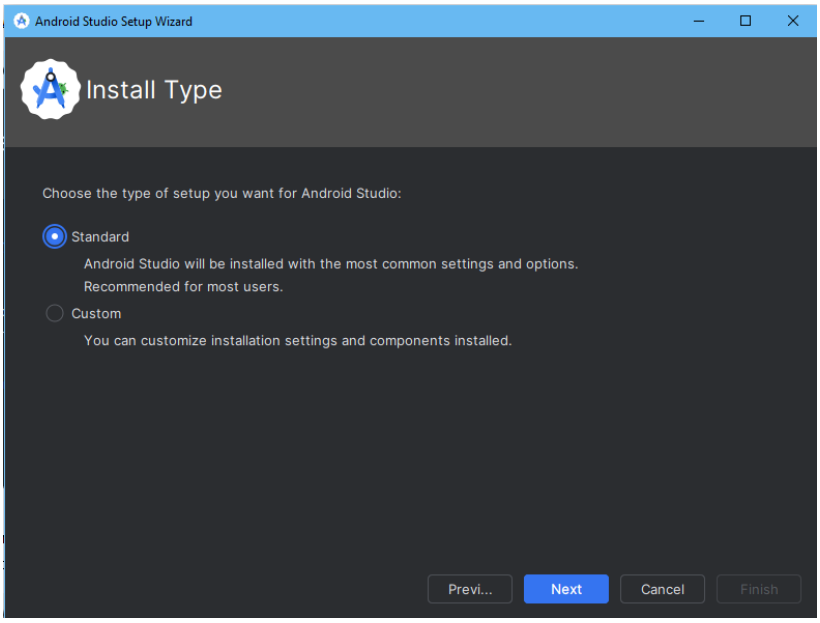
Далее запрос на сбор статистики для улучшения Android studio



И наконец, стартовое окно

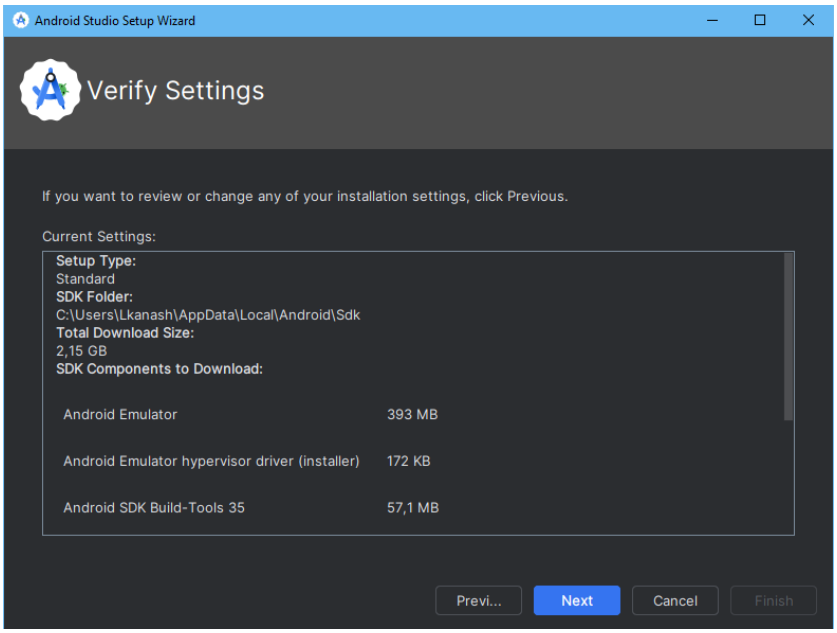


Нажимаем на Next. Далее будет предложено выбрать тип установки Android Studio:

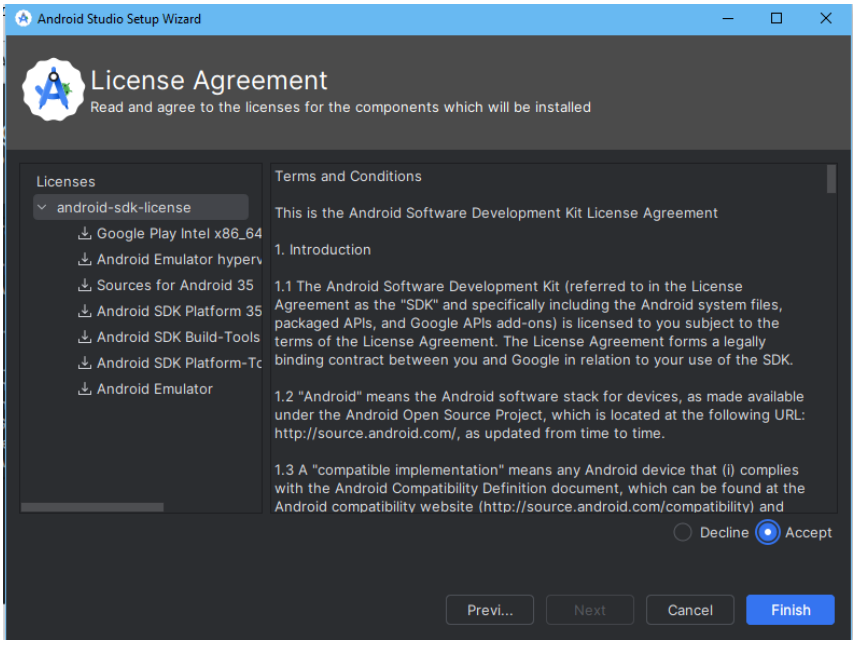


Оставим отмеченный по умолчанию тип "Standard" и нажимаем на кнопку Next.

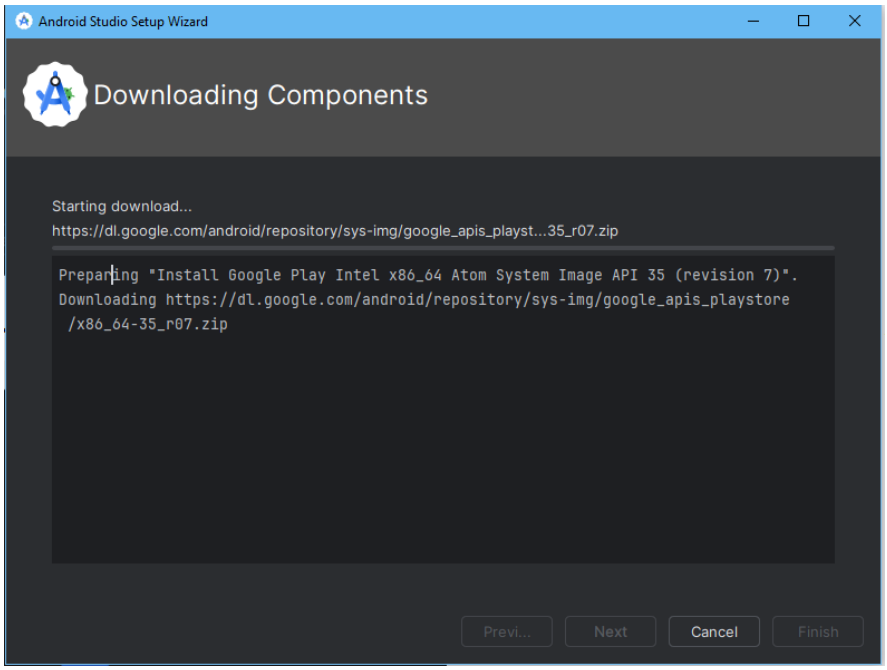
В следующем окне отобразится путь к устанавливаемому Android SDK и список компонентов, которые будут установлены:



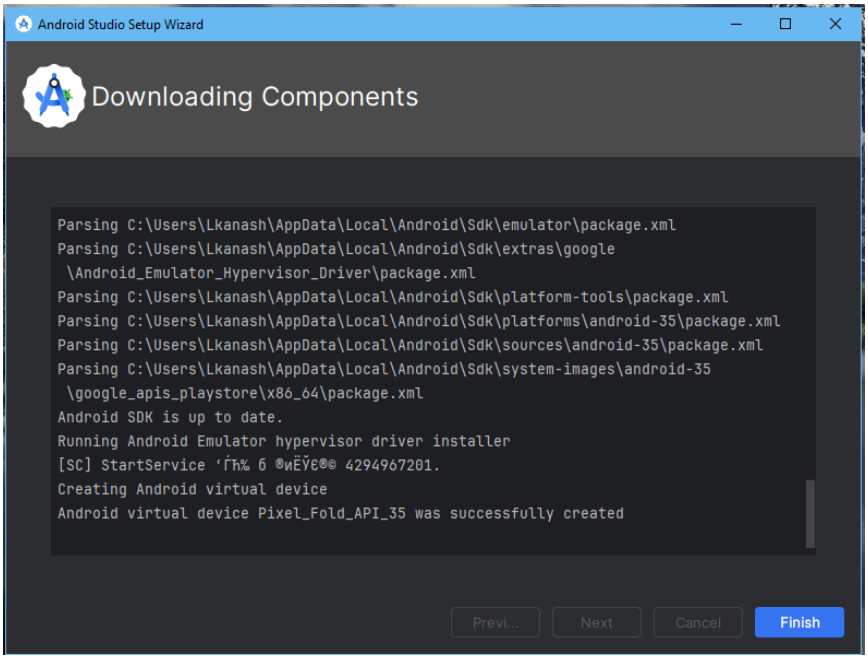
Жмем Next и принимаем лицензионные соглашения к устанавливаемым компонентам:



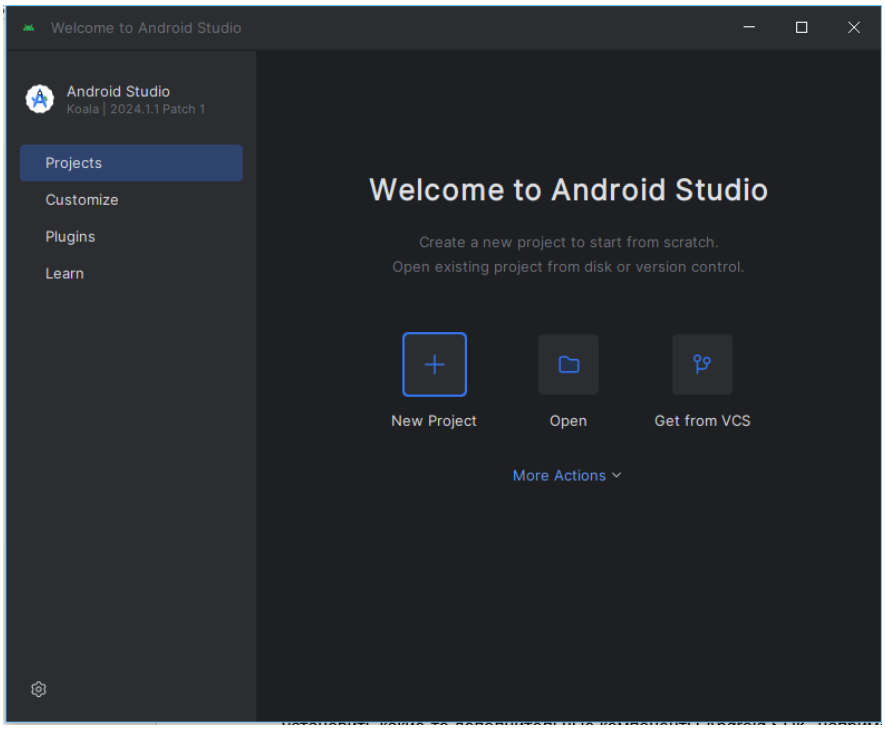
Жмем Finish – начнется скачка дополнительных компонентов, установка Android SDK и эмулятора:



Оповещение о том, что установка завершена

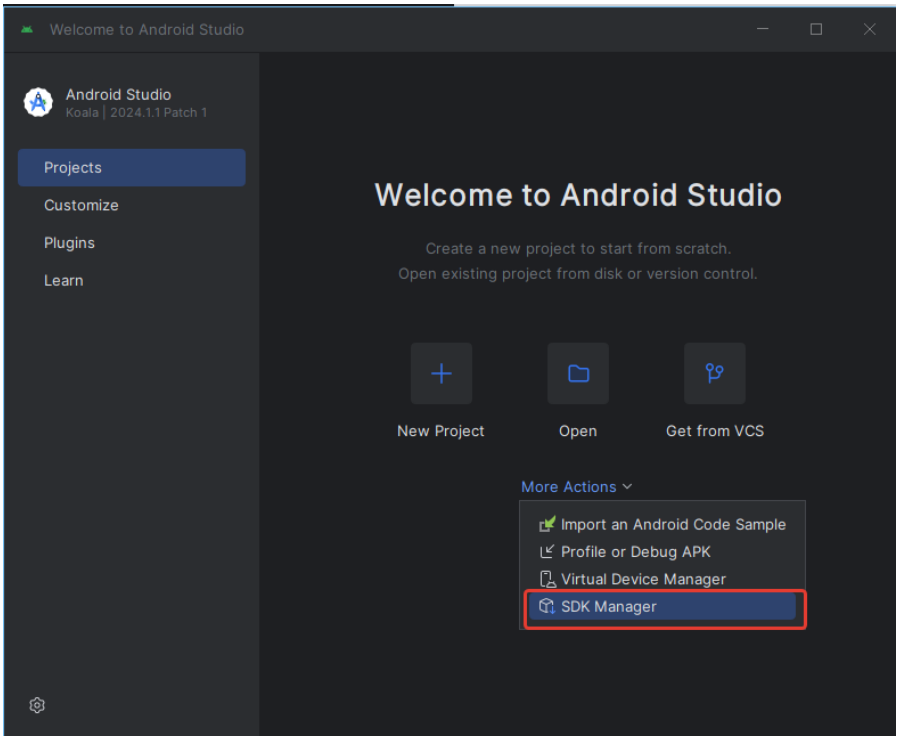


После завершения установки откроется стартовый экран Android Studio, и можно начать создавать приложения под Android:

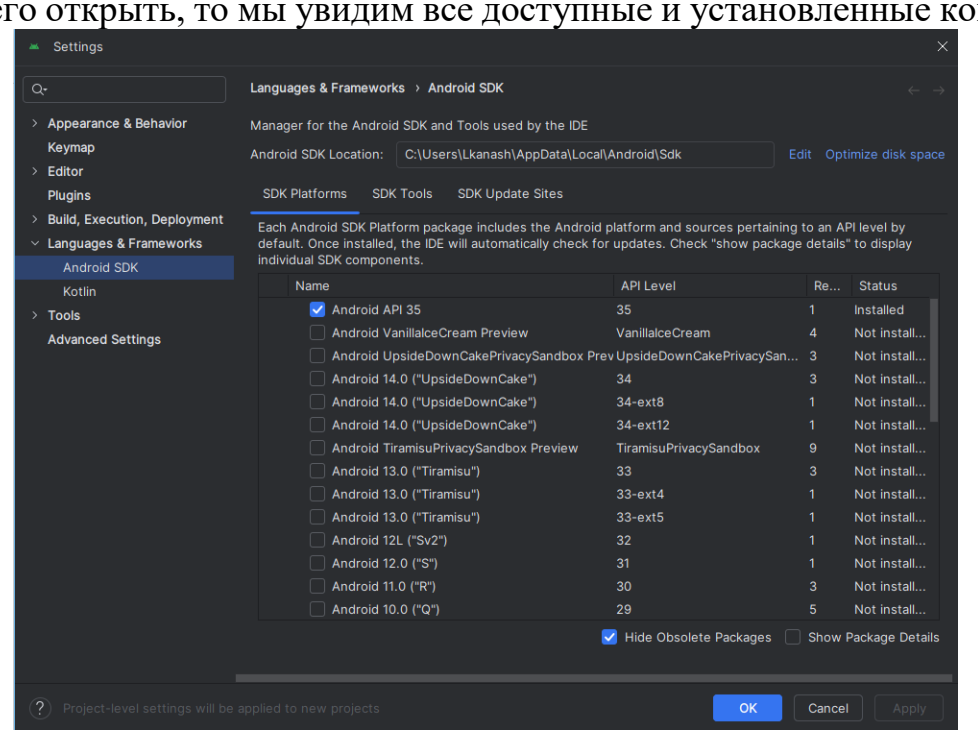


Управление Android SDK

Возможно, что в какой-то момент времени работы с Android Studio потребуется установить какие-то дополнительные компоненты Android SDK, например, новые платформы, или удалить устаревшие компоненты. В этом случае Android Studio предоставляет инструменты для управления Android SDK. В частности, при запуске Android Studio на главном экране можно выбрать пункт SDK Manager



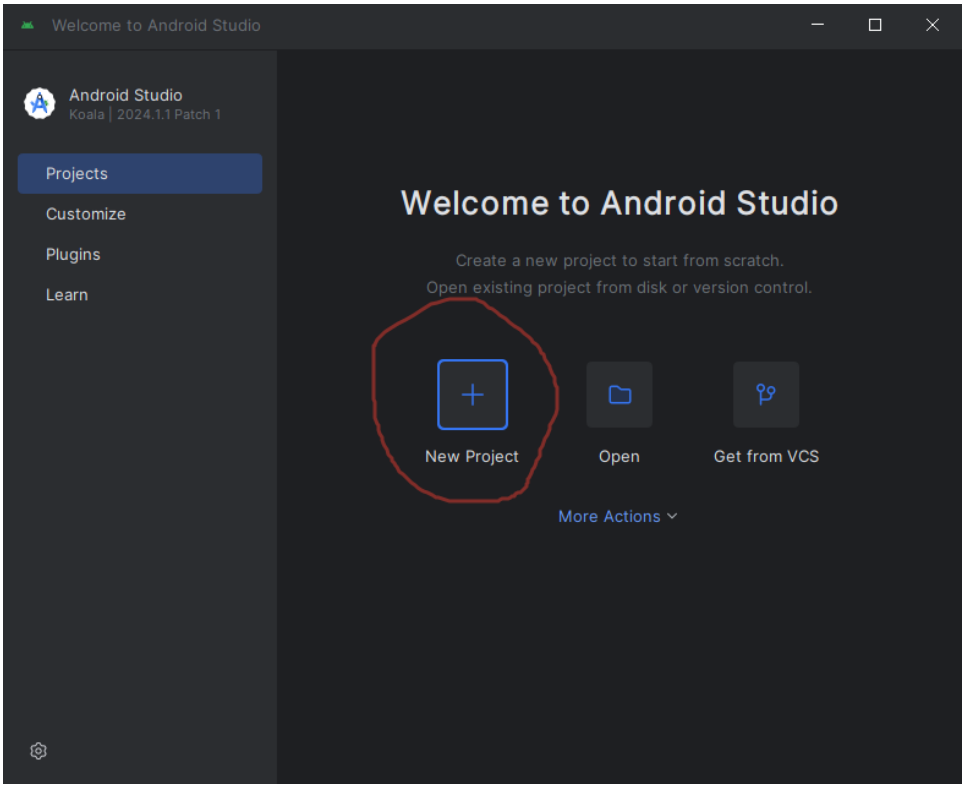
Если его открыть, то мы увидим все доступные и установленные компоненты:



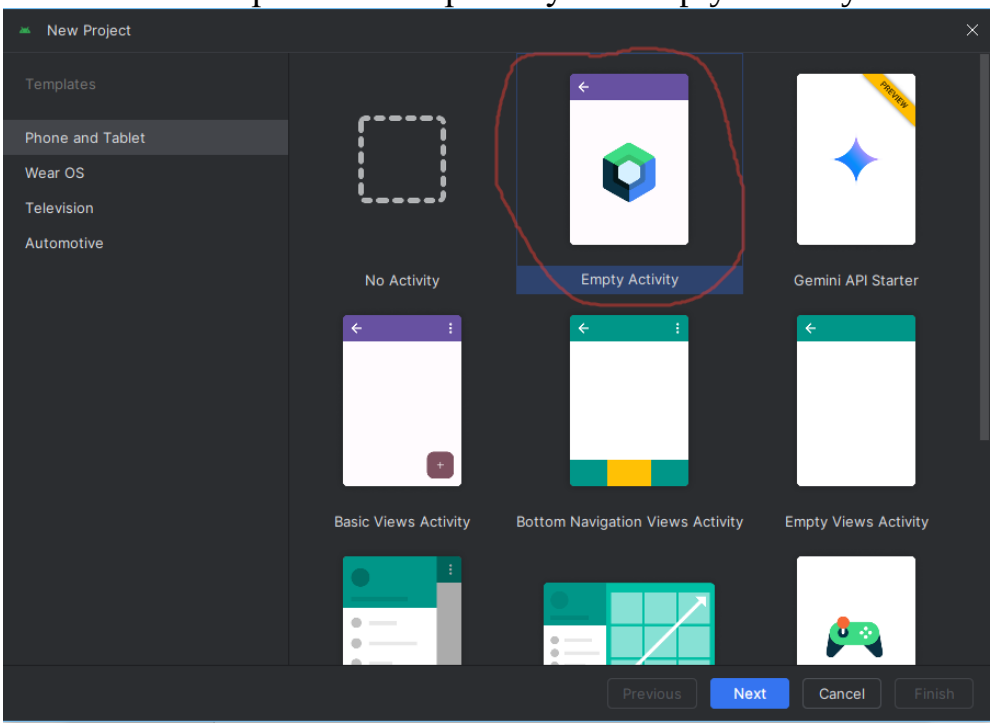
При необходимости можно выделить дополнительные компоненты и установить или снять отметку с ранее установленных компонентов и удалить их.

Первый проект на Jetpack Compose

1. Запустите среду Android Studio.
2. Нажмите New Project в окне Welcome to Android Studio
или File → New → New Project, если уже открыт какой-то проект



3. В окне шаблонов проекта выберите пункт Empty Activity → Next.



4. В окне настройки проекта установите начальные настройки:

В поле **Name** вводится название приложения. Укажите в качестве имени название HelloApp

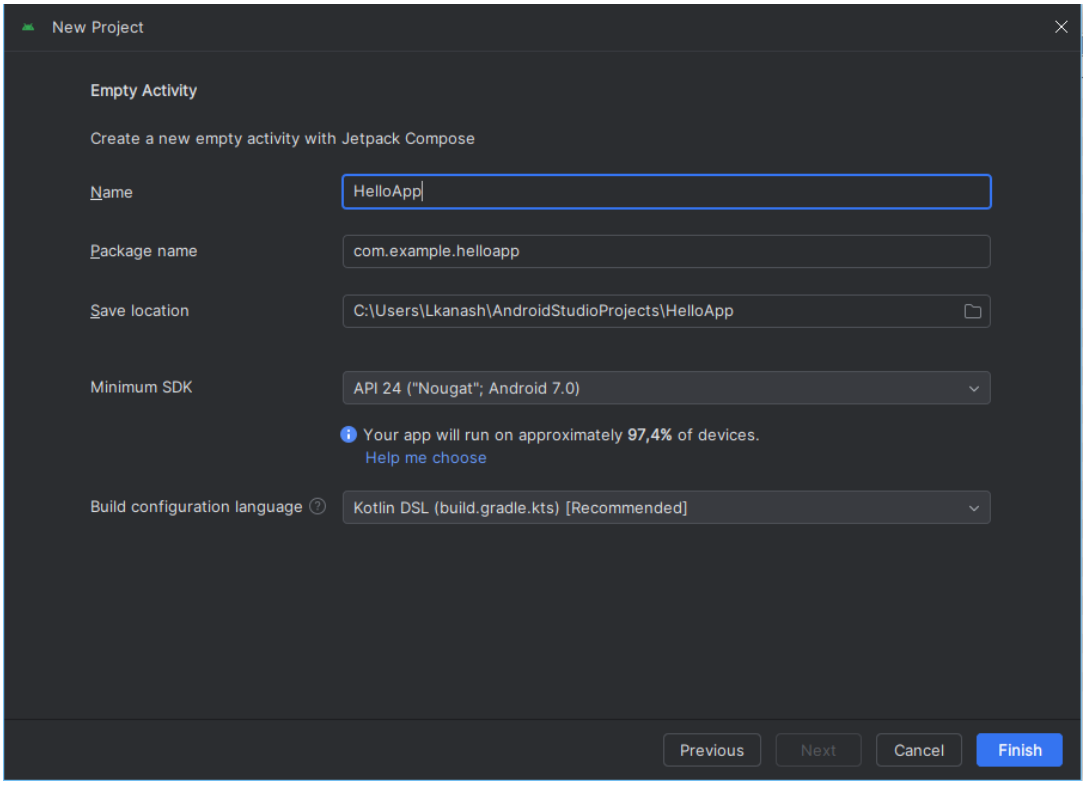
В поле **Package Name** указывается имя пакета, где будет размещаться главный класс приложения. В данном случае для тестовых проектов это не имеет большого значения, поэтому установите com.example.helloapp.

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

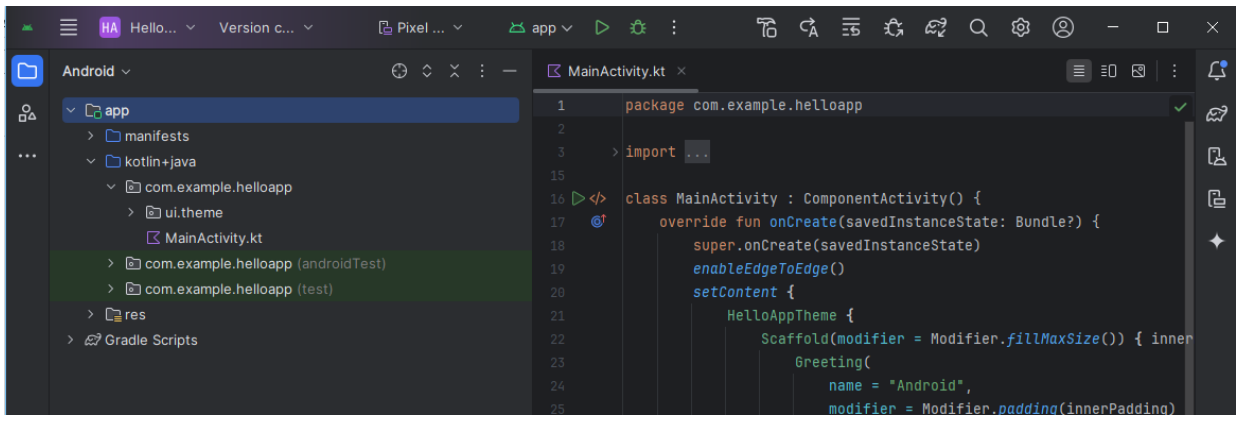
В поле **Minimum SDK** указывается самая минимальная поддерживаемая версия SDK. Оставьте значение по умолчанию. Например, API 24 ("Nougat"; Android 7.0), которая означает, что приложение можно будет запустить начиная с Android 7.0, а это 97,4% устройств). На более старых устройствах запустить будет нельзя.

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

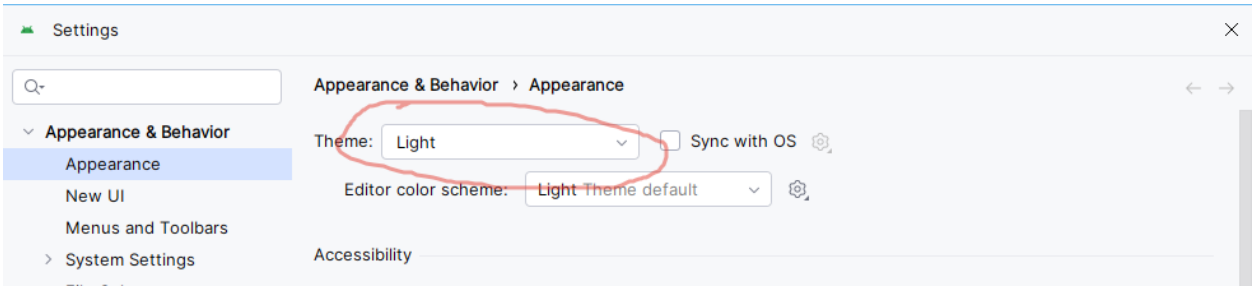
В поле **Build configuration language** указывается язык, который применяется для определения конфигурации построения проекта. Оставьте здесь значение по умолчанию.



4. Нажмите Finish, и Android Studio создаст новый проект:

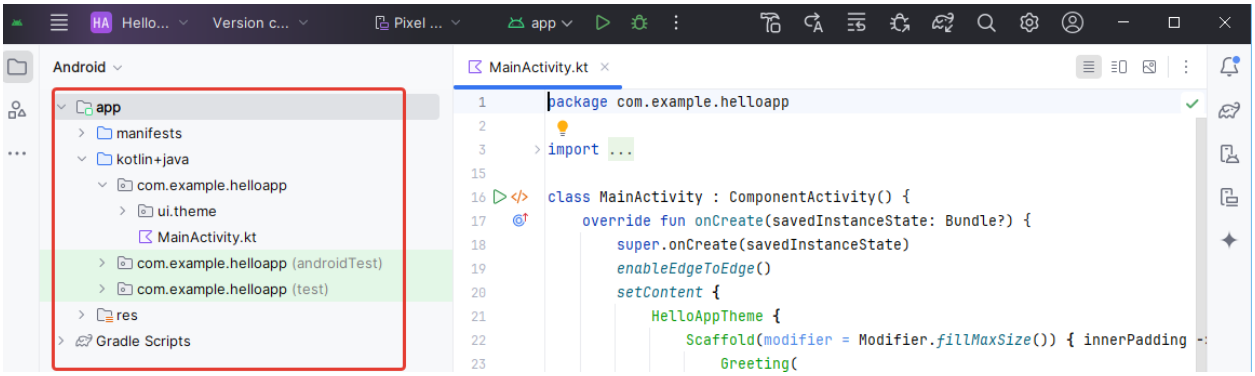


Изменить темную тему на светлую или наоборот можно в настройках 



Структура проекта по умолчанию

Проект Android может состоять из различных модулей. По умолчанию, при создании проекта, создается один модуль → app. Модуль по умолчанию включает три каталога:



manifests: хранит файл манифеста AndroidManifest.xml, который описывает конфигурацию приложения и определяет каждый из компонентов данного приложения.

kotlin+java: хранит файлы кода на языке Kotlin, которые структурированы по отдельным пакетам. Так, в папке com.example.helloapp (название которого было указано на этапе создания проекта) имеется по умолчанию файл MainActivity.kt с кодом на языке Kotlin, который представляет класс MainActivity, запускаемый по умолчанию при старте приложения

Также в этой папке определен подкаталог ***ui.theme,*** который содержит ряд файлов на языке Kotlin: Color.kt, Theme.kt и Type.kt, которые определяют ряд вспомогательных типов, функций, переменных, применяемых для создания графического интерфейса.

res: содержит используемые в приложении ресурсы. Все ресурсы разбиты на подпапки.

папка d***rawable*** предназначена для хранения изображений, используемых в приложении

папки ***mipmap*** содержат файлы изображений, которые предназначены для создания иконки приложения при различных разрешениях экрана.

папка ***values*** хранит различные xml-файлы, содержащие коллекции ресурсов – различных данных, которые применяются в приложении. По умолчанию здесь есть два файла и одна папка:

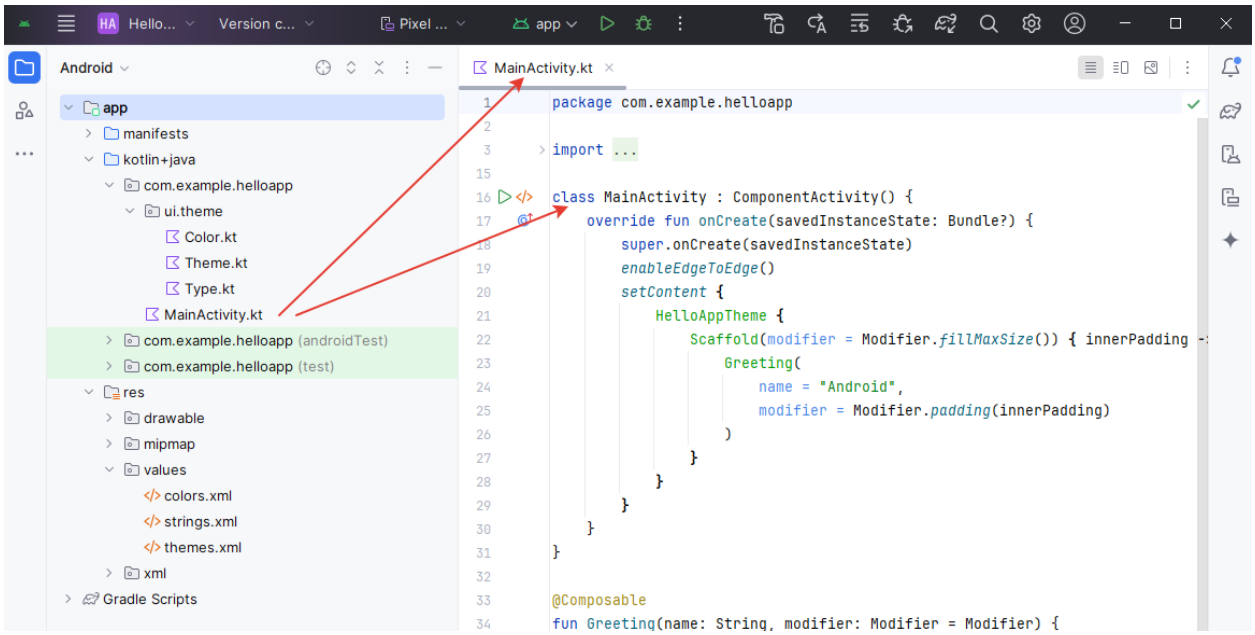
файл ***colors.xml*** хранит описание цветов, используемых в приложении

файл ***strings.xml*** содержит строковые ресурсы, используемые в приложении

папка ***themes*** хранит две темы приложения − светлую (дневную) и темную (ночную)

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

Во всей этой структуре следует выделить файл ***MainActivity.kt***, который открыт в Android Studio и который содержит логику приложения и собственно с него начинается выполнение приложения.



Запуск проекта

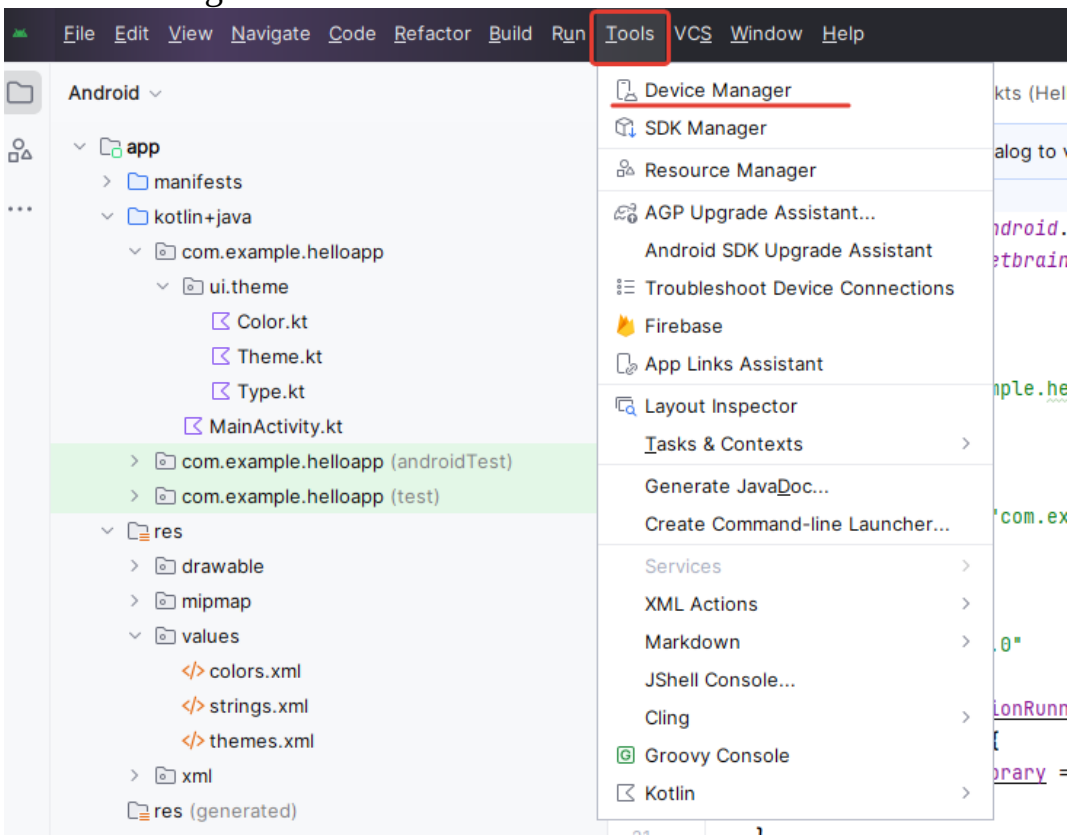
Для запуска и тестирования проекта можно использовать как различные эмуляторы, так и реальные устройства. Для тестирования на реальном устройстве на нем должен быть включен режим разработчика.

Для запуска проекта нажмите на зеленую стрелочку на панели инструментов.

При первом запуске в режиме эмулятора может потребоваться установить Android Emulator hypervisor driver (для компьютеров с AMD процессорами) (в BIOS должна быть включена виртуализация).

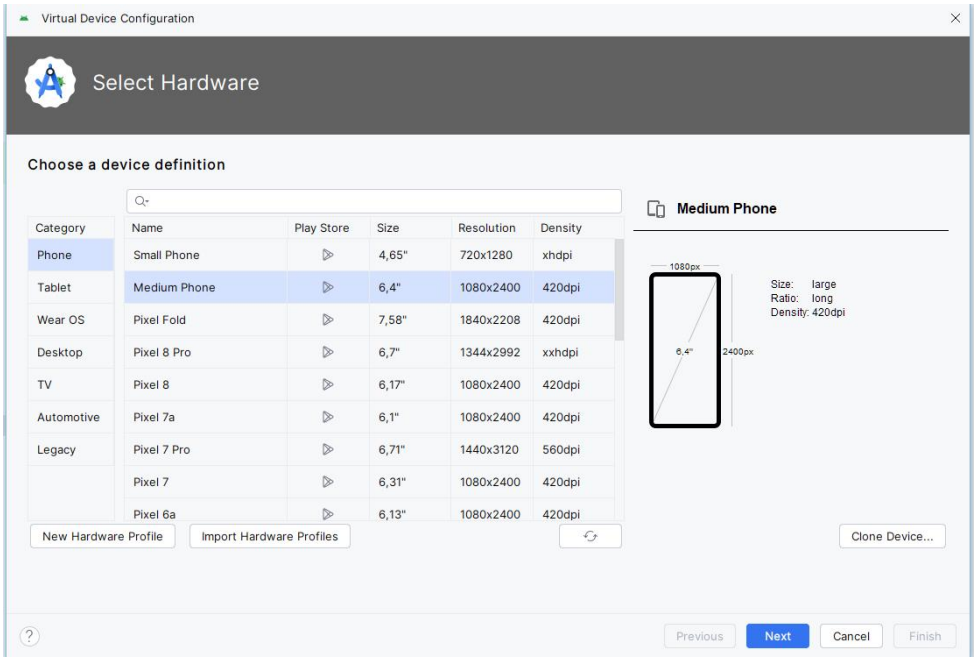
После успешного запуска на устройстве/эмуляторе можно насладиться надписью "Hello Android":

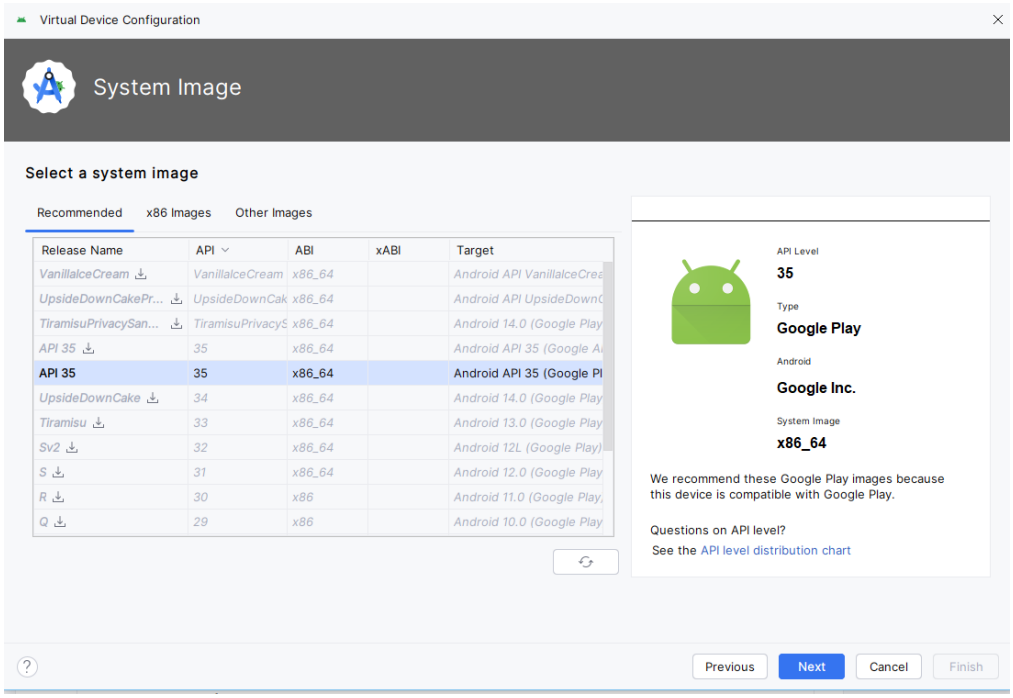
Чтобы сменить виртуальное устройство, откройте Main menu → Tools → Device Manager

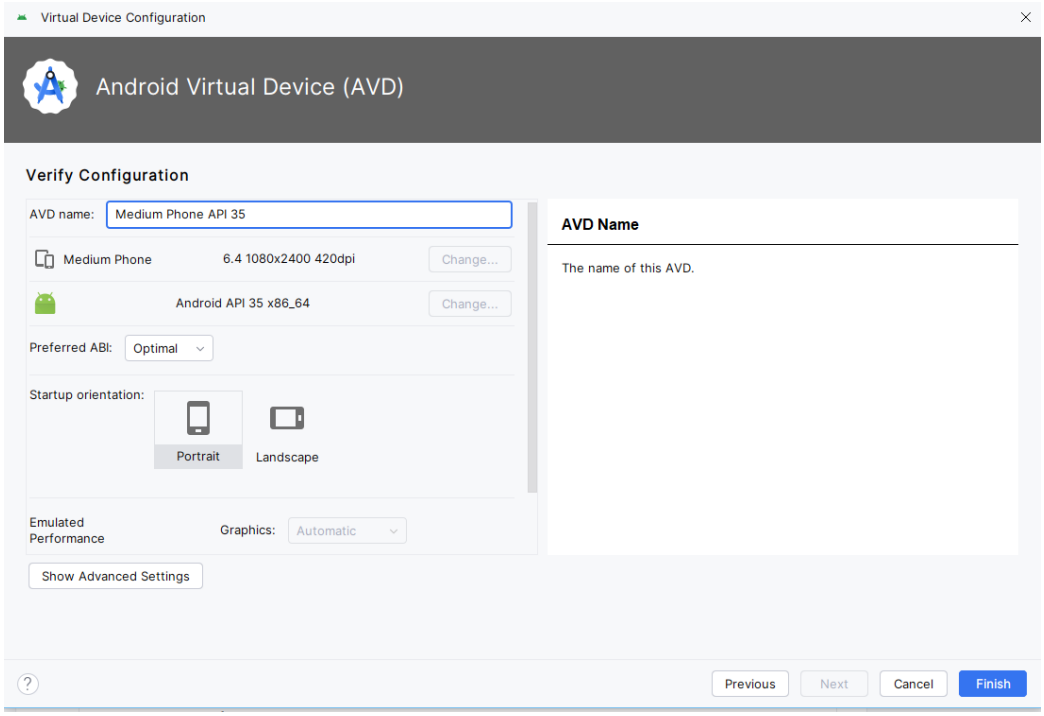


В окне Device Manager нажмите на + и выберите Create Virtual Device

В окне Select Hardware выберите устройство и в последующих окнах выберите настройки

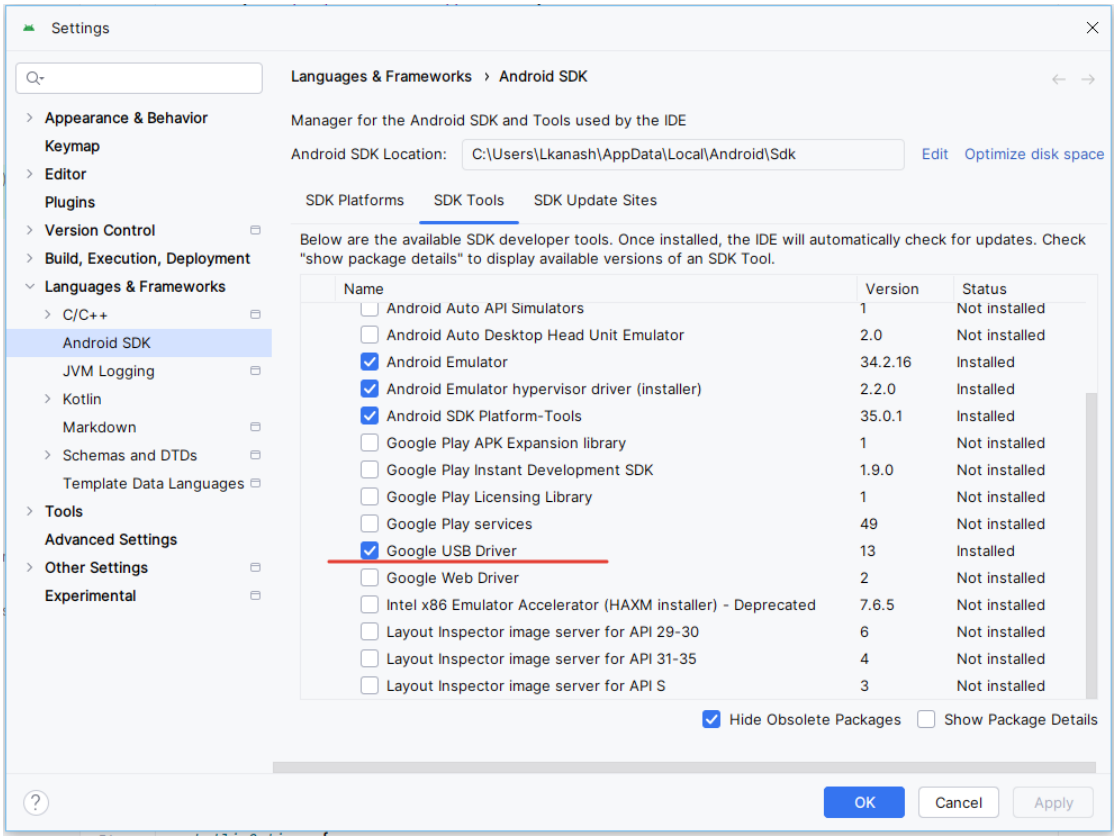






Подключение телефона для тестирования приложений андроид студио.

1. в Android Studio перейти Main Menu → Tools → SDK Manager →SDK Tools и отметить галочкой Google USB Driver (если он не отмечен).



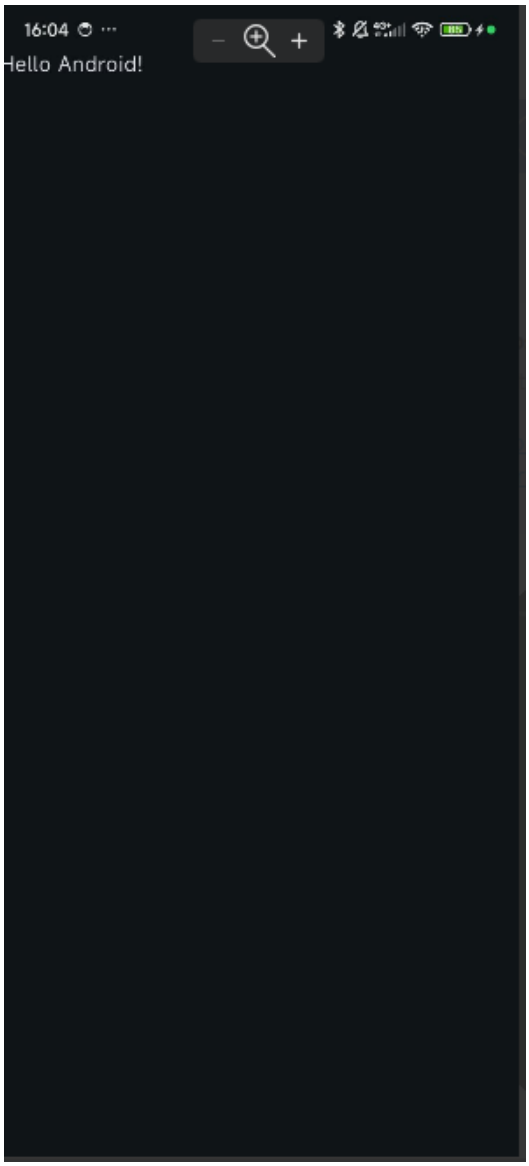
После нажатия ОК драйвер установится (если не установлен).

2. На телефоне включить режим разработчика. Порядок включения примерно такой: в настройках на номере сборки тапнуть несколько раз (5-10), появится всплывающее сообщение о том, что режим разработчика включен. Далее в настройках ищете пункт меню Режим разработчика, переходите туда и активируете «Не выключать экран…» и «Отладка по USB…», «Установка через USB»

3. Подключите смартфон к компьютеру через USB

4. Если необходимо, то в настройках смартфона укажите тип взаимодействия с компьютером − «Передача файлов»

5. Если сделано все верно, то имя подключенного устройства появится в списке Running Device в Android Studio.



Изменение проекта

При создании проекта, который использует Jetpack Compose, в Android Studio, в проект по умолчанию добавляется файл MainActivity.kt, который содержит одноименный класс MainActivity. Этот класс определяет интерфейс, который можно увидеть на устройстве/эмуляторе при запуске проекта

Основные моменты кода, который добавляется в файл MainActivity.kt и который определяет подобный интерфейс:

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.helloapp.ui.theme.HelloAppTheme

class MainActivity : ComponentActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    enableEdgeToEdge()
    setContent {
      HelloAppTheme {
        Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
          Greeting(
            name = "Android",
            modifier = Modifier.padding(innerPadding)
          )
        }
      }
    }
  }
}

@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
  Text(
    text = "Hello $name!",
    modifier = modifier
  )
}

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
  HelloAppTheme {
    Greeting("Android")
  }
}
```

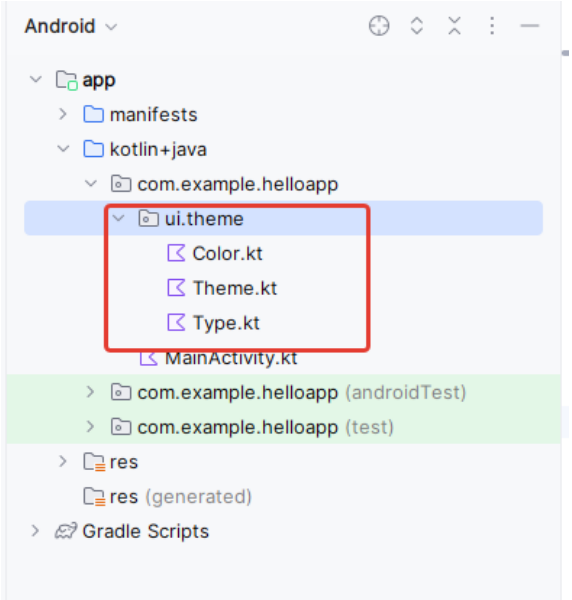
Сначала идет определение пакета, к которому принадлежит класс MainActivity. Здесь это пакет com.example.helloapp

```
package com.example.helloapp
```

Далее идут подключаемые пакеты, функционал который использует класс MainActivity:

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.helloapp.ui.theme.HelloAppTheme
```

Обратите внимание на последний подключаемый пакет – он представляет функционал из каталога ui.theme, который также добавляется в проект по умолчанию и который содержит ряд файлов на языке Kotlin: Color.kt, Theme.kt и Type.kt.



Далее идет определение класса MainActivity

```
class MainActivity : ComponentActivity() {
```

Приложение на Android определяется как набор объектов, которые называются activity. Объект activity фактически представляет отдельное окно графического приложения. И класс MainActivity как раз представляет окно, которое отображается при запуске приложения.

MainActivity наследуется от встроенного класса ComponentActivity. ComponentActivity обеспечивает построение интерфейса из визуальных компонентов и для этого предоставляет минимальный функционал. В частности, ComponentActivity предоставляет метод onCreate().

```
    override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    enableEdgeToEdge()
    setContent {......
```

onCreate() вызывается, когда приложение создаёт и отображает разметку активности. Метод помечен ключевым словом override (переопределён из базового класса).

Строка super.onCreate(savedInstanceState) – это конструктор родительского класса, выполняющий необходимые операции для работы активности. Эту строчку не надо трогать, оставляйте ее без изменений.

enableEdgeToEdge() позволяет включить отображение приложения "от края до края".

Функция setContent() подключает другие функции, которые отвечают за внешний вид приложения. В частности, функция HelloAppTheme() подключает определённую тему.

```
    setContent {
     HelloAppTheme {
       Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
         Greeting(
           name = "Android",
           modifier = Modifier.padding(innerPadding)
         )
       }
     }
   }
```

Её имя создаётся на основе имени проекта, а определена она в файле Theme.kt:

```
    ...
@Composable
fun HelloAppTheme(
  darkTheme: Boolean = isSystemInDarkTheme(),
  // Dynamic color is available on Android 12+
  dynamicColor: Boolean = true,
  content: @Composable () -> Unit
) {
  val colorScheme = when {
    dynamicColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
      val context = LocalContext.current
      if (darkTheme) dynamicDarkColorScheme(context) else
dynamicLightColorScheme(context)
    }

    darkTheme -> DarkColorScheme
    else -> LightColorScheme
  }

  MaterialTheme(
    colorScheme = colorScheme,
    typography = Typography,
    content = content
  )
}
```

Scaffold − это набор виджетов, которые визуально представляют собой пользовательский интерфейс для Activity. Как правило, Activity используется для отображения одного экрана, который состоит из многих View-компонентов, таких как тулбар, меню, боковое меню, снэк-бар, FAB и т.д. В scaffold всё это представлено в виде виджетов.

Таким образом Scaffold представляет собой предварительно созданный составной файл, который обеспечивает базовую структуру макета для создания экранов пользовательского интерфейса.

В более ранних версиях Android Studio использовалась функция Surface, которая отвечала за фон приложения.

У функции Greeting() есть параметр name, который в нашем случае равен «Android». Понятно, что если заменить это слово на другое, то это уже будет приложение с изменённым текстом.

Функцию с аннотацией ***@Composable*** можно вызывать только в составе других composable-функций или в функции setContent() (как в примере). Подобные функции принято начинать с заглавной буквы.

В примере у функции два параметра − name и modifier.

Строковый параметр name добавляет имя к слову Hello. Второй параметр имеет тип по умолчанию Modifier, поэтому при вызове функции можно его опустить.

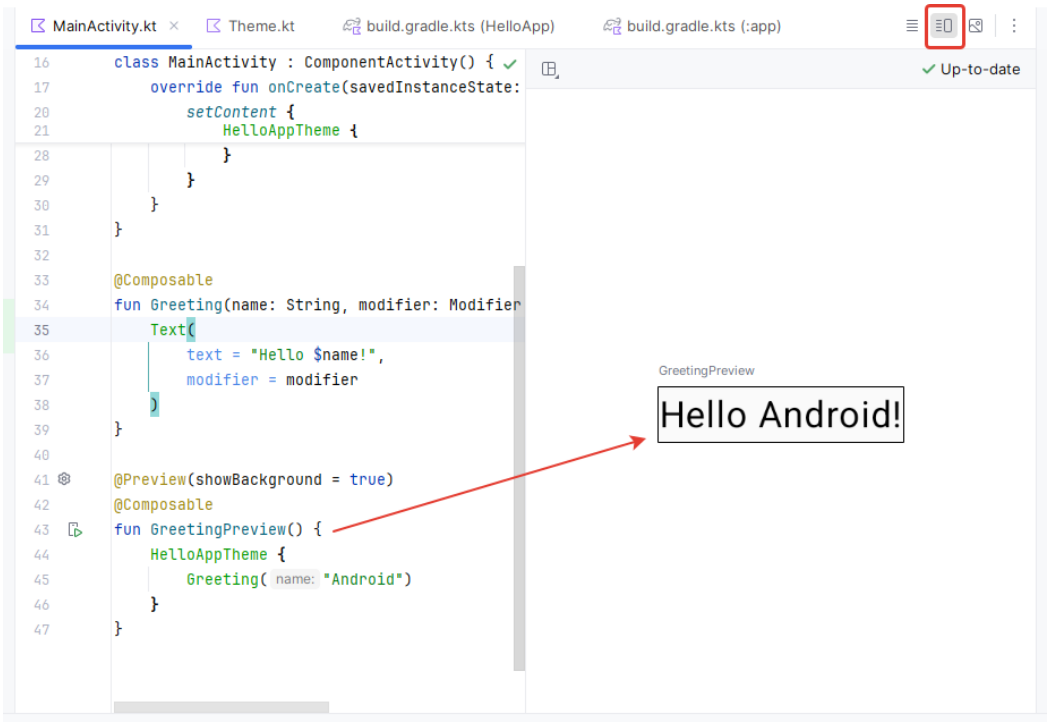
Предварительный вид компонентов

Для предварительного просмотра внешнего вида приложения используется аннотация @Preview.

```
@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
  HelloAppTheme {
    Greeting("Android")
  }
}
```

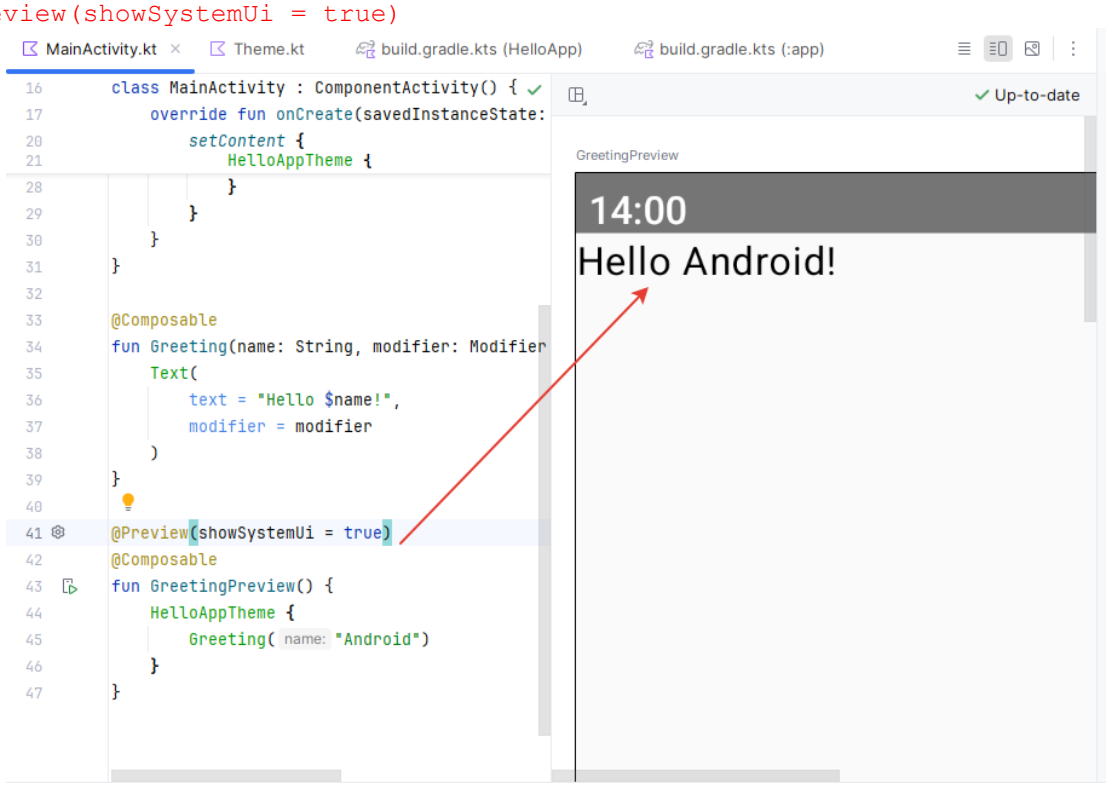
Здесь GreetingPreview представляет некоторую часть визуального интерфейса. В реальности он скрывает компонент HelloAppTheme − то есть фактически весь тот же самый интерфейс, что и определен в классе MainActivity. Но важная часть определения этого компонента − аннотация @Preview − указывает, что данный компонент будет применяться для предварительного просмотра. То есть, чтобы узнать, как будет выглядеть созданный интерфейс, необязательно запускать приложение на устройстве/эмуляторе. В случае небольших, но частых изменений это может быть довольно утомительно. А с помощью этого компонента можно увидеть, что это будет за интерфейс.

Весь интерфейс в компоненте GreetingPreview можно увидеть в Android Studio при просмотре в режиме Design или Split:



С помощью параметров аннотации @Preview можно настраивать предварительный просмотр. Например, если надо отобразить остальную часть экрана, то можно установить опцию showSystemUi = true:

```
@Preview(showSystemUi = true)
```



Стоит отметить, что окно предварительного просмотра автоматически подхватывает все изменения в интерфейсе приложения.

Теперь попробуйте заменить «Android» на «Student»

```
...
setContent {
 HelloAppTheme {
   Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
     Greeting(
       name = "Student",
       modifier = Modifier.padding(innerPadding)
     )
   }
 }
}


...
@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
  HelloAppTheme {
    Greeting("Student")
  }
}
```

Поменяйте фон у экрана активности.

Сейчас в функции Greeting находится всего один компонент Text. В реальности, у вас будет несколько компонентов, которые нужно размещать на экране определённым образом внутри какого-нибудь контейнера. Поместите курсор внутри строки кода со словом Text и нажмите комбинацию клавиш Alt+Enter.

В контекстном меню выберите последовательно Show Context Action → Surround with widget → Surround with Column. Это самый популярный контейнер

```
Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
  Column {
    Text(
      text = "Hello $name!",
      modifier = modifier
    )
  }
}
```

Здесь Column не использует аргументы. Однако, нам нужно поменять цвет у контейнера, поэтому добавляем к функции круглые скобки и внутри прописываем нужные аргументы.

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
  Column (modifier = modifier.fillMaxSize().background(Color(0xFF2196F3))
  ){
    Text(
      text = "Hello $name!",
      modifier = modifier
    )
  }
}
```

Если вы видите, что, например, background и color отображаются красным цветом, это означает что нужно подключить соответствующие пакеты. Пользуйтесь подсказками студии. Здесь это пакеты:

```
import androidx.compose.foundation.background
import androidx.compose.ui.graphics.Color
```

В студии слева от строчки кода, где задан цвет, появится небольшой значок. Нажав на него, вы можете открыть специальное окно для выбора цвета и задать свой цвет.

```
@Composable
fun Greeting(modifier: Modifier = Modifier) {

    Column (modifier = modifier
        .fillMaxSize()
        .background(Color( color: 0xFF2196F3)),
        horizontalAlignment = Alignment.CenterHorizontally
    ){
        var catName by remember{mutableStateOf( value: "Student")}
        var textValue by remember{mutableStateOf( value: "")}
        Text(
            text = "Hello $catName!",
```

Результат после внесения изменений

Здесь текст прижался к левому верхнему углу, поэтому
измените поведение – пусть все компоненты центрируются по горизонтальной оси.

```
Column (modifier = modifier.fillMaxSize().background(Color(0xFFF3C2D1)),
      horizontalAlignment = Alignment.CenterHorizontally
)
```

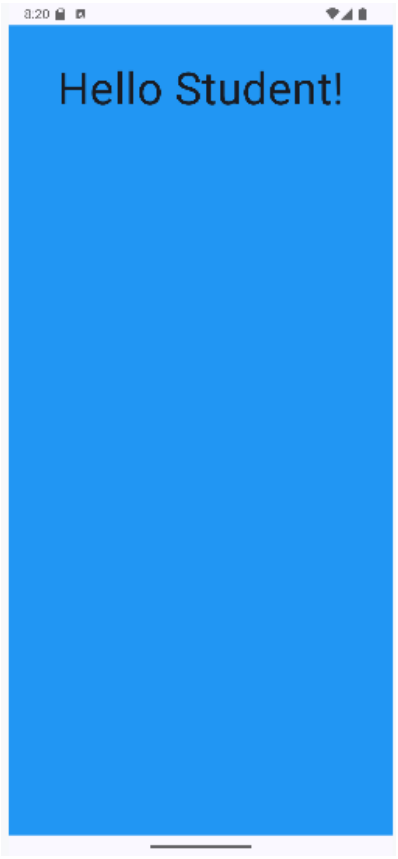
И увеличьте шрифт текста, например, до 48.sp

```
Text(
  text = "Hello $name!",
  modifier = modifier,
  fontSize = 48.sp
)
```

Добавьте к тексту отступы (можно задать индивидуально для каждой стороны или сразу для всех сторон).

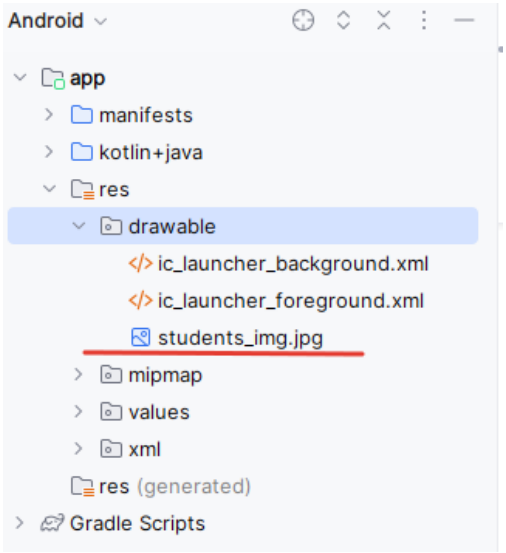
```
Text(
  text = "Hello $name!",
```

```
    modifier = modifier.padding(16.dp),
    fontSize = 48.sp
)
```



Добавьте картинку на экран активности. Для этого найдите подходящее изображение и добавьте его в папку res/drawable.

Можно просто перетащить картинку из проводника мышкой в папку drawable, при этом картинка перенесётся. Если вы хотите перенести копию картинки, оставив оригинал в своей папке, то попробуйте следующий вариант – скопируйте вашу картинку в буфер, затем щёлкните правой кнопкой мыши на папке drawable в студии, выберите команду Paste.



Сразу после функции Text() добавьте новую функцию Image.

```
Text(
  text = "Hello $name!",
  modifier = modifier.padding(16.dp),
  fontSize = 48.sp
)
Image(
    painter = painterResource(id = R.drawable.students_img),
    contentDescription = "Hello Student Image"
)
```

Полностью код для приложения.

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
```

```
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.helloapp.ui.theme.HelloAppTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            HelloAppTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    Greeting(
                        name = "Student",
                        modifier = Modifier.padding(innerPadding)
                    )
                }
            }
        }
    }
}

@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Column (modifier = modifier.fillMaxSize().background(Color(0xFF2196F3)),
        horizontalAlignment = Alignment.CenterHorizontally) {
        Text(
            text = "Hello $name!",
            modifier = modifier.padding(16.dp),
            fontSize = 48.sp
        )
        Image(
            painter = painterResource(id = R.drawable.students_img),
            contentDescription = "Hello Student Image"
        )
    }
}

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    HelloAppTheme {
        Greeting("Student")
    }
}
```

Создайте две переменные. Первая будет содержать имя студента, а вторая – текст из текстового поля.

```
var MyName by remember{mutableStateOf("Student")}
var textValue by remember{mutableStateOf("")}
```

Добавьте два новых компонента: текстовое поле (TextFild)

```
TextField(
  value = textValue,
  onValueChange = {textValue = it},
  label = { Text(text = "Введите имя")},
  placeholder = { Text(text = "Student") },
  modifier = modifier.padding(4.dp)
)
```

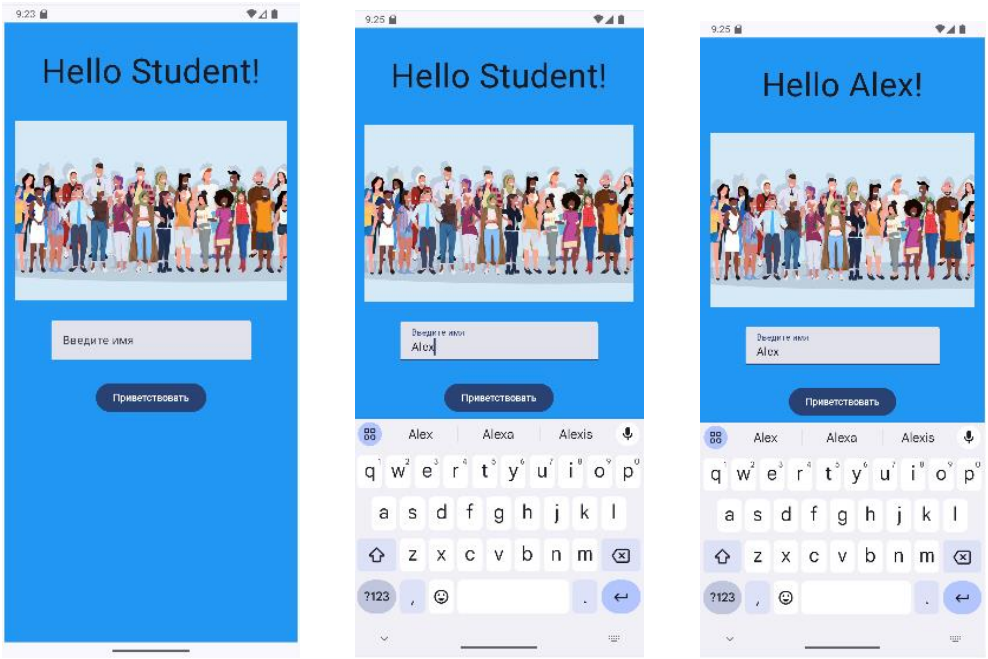
Через modifier.padding задаются отступы. Необязательный параметр placeholder выводит текст при пустом поле. Другой необязательный параметр label выводит подсказку над полем. Параметр value содержит текст из текстового поля, который нам нужен. В нём содержится имя, которое ввёл пользователь. Параметр onValueChange отслеживает в режиме реального времени изменение текста.

и кнопку (Button)

```
Button(
  onClick =
  {
    MyName = textValue
  }
) {
  Text(text = "Приветствовать")
}
```

Здесь при помощи Text задаётся текст на кнопке, а в параметре onClick код для нажатия. В данном случае переменной для имени кота присваивается значение из текстового поля.

У функции Greeting можно убрать первый параметр name, так как теперь имя будем узнавать программно.

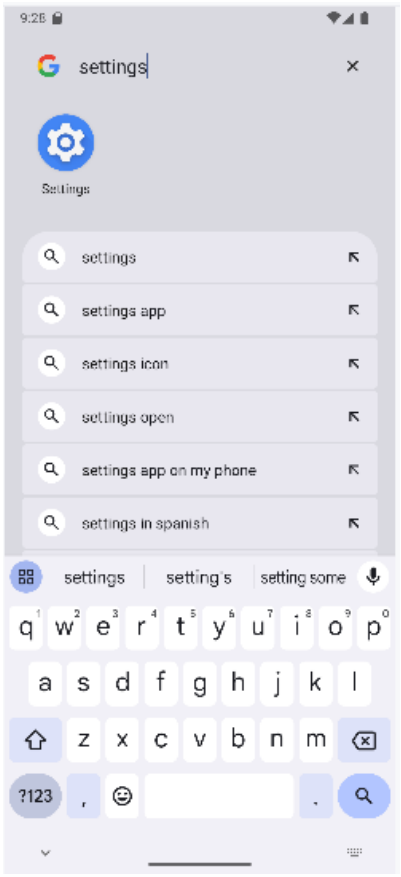


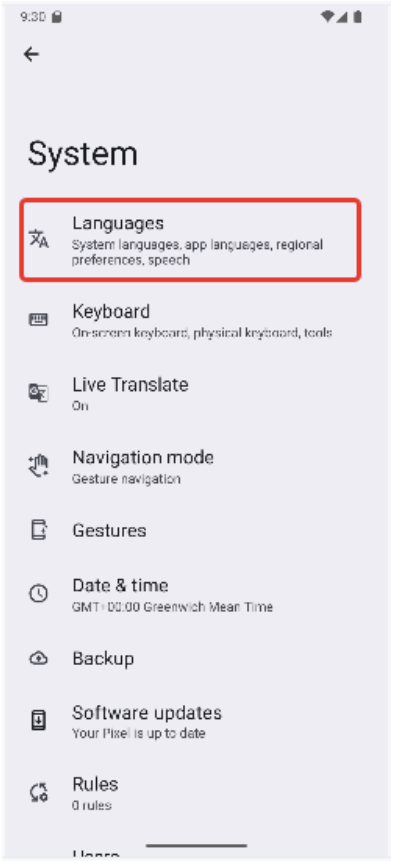
Код после правок:

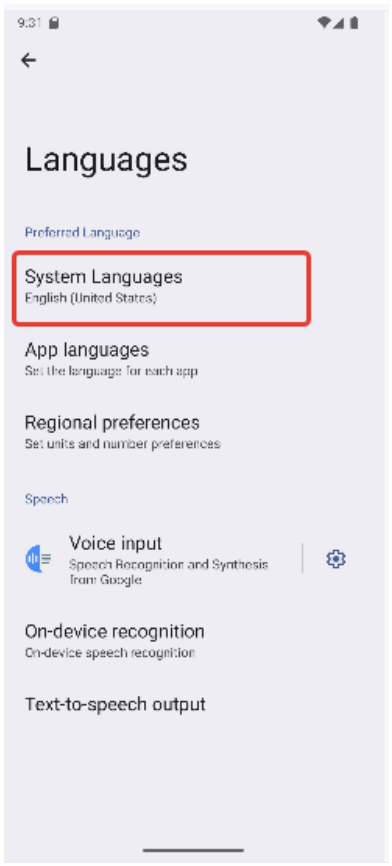
```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Button
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.runtime.Composable
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.helloapp.ui.theme.HelloAppTheme
import androidx.compose.runtime.getValue
import androidx.compose.runtime.setValue

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            HelloAppTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    Greeting(
                        modifier = Modifier.padding(innerPadding)
                    )
```

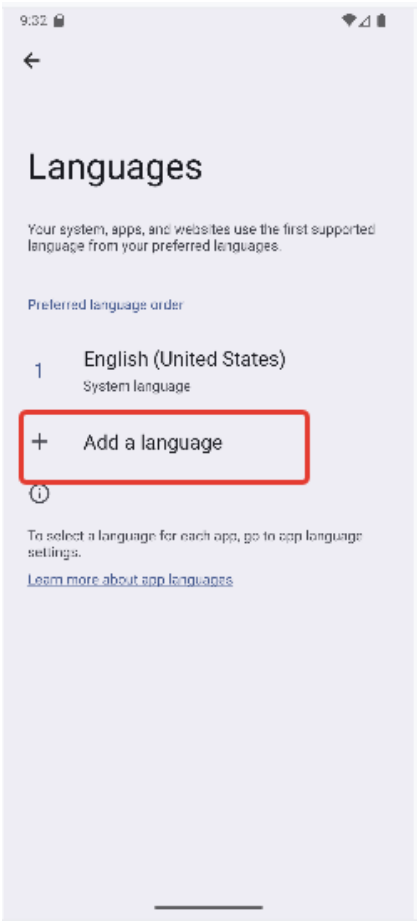
```
                }
            }
        }
    }
}

@Composable
fun Greeting(modifier: Modifier = Modifier) {
    Column (modifier = modifier.fillMaxSize().background(Color(0xFF2196F3)),
        horizontalAlignment = Alignment.CenterHorizontally) {
        var MyName by remember{mutableStateOf("Student")}
        var textValue by remember{mutableStateOf("")}
        Text(
            text = "Hello $MyName!",
            modifier = modifier.padding(16.dp),
            fontSize = 48.sp
        )
        Image(
            painter = painterResource(id = R.drawable.students_img),
            contentDescription = "Hello Student Image"
        )
        TextField(
            value = textValue,
            onValueChange = {textValue = it},
            label = { Text(text = "Введите имя")},
            placeholder = { Text(text = "Student") },
            modifier = modifier.padding(4.dp)
        )
        Button(
            onClick =
            {
                MyName = textValue
            }
        ) {
            Text(text = "Приветствовать")
        }
    }
}

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    HelloAppTheme {
        Greeting()
    }
}
```

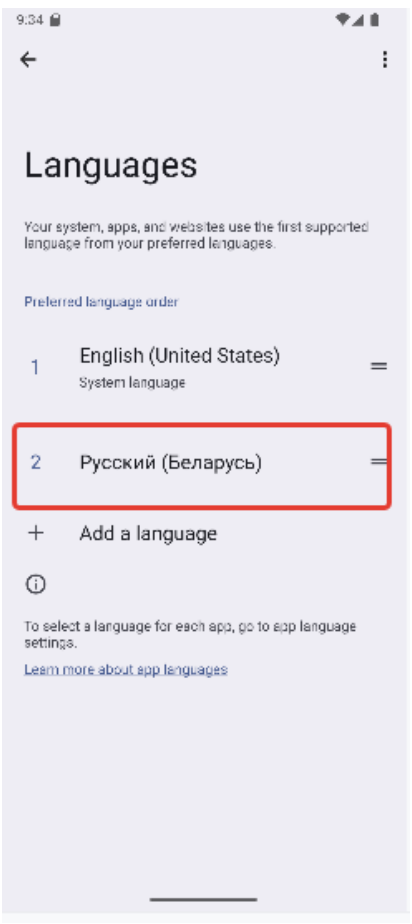
Подключить русскую клавиатуру можно добавив русский язык в настройках виртуального устройства

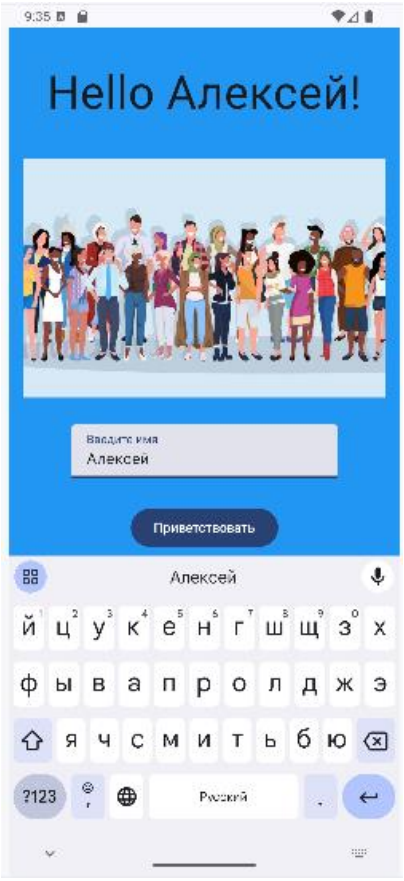

settings
Settings
settings
settings app
settings icon
settings open
settings app on my phone
settings in spanish


System
Languages
System languages, app languages, regional preferences, speech
Keyboard
Live Translate
On
Navigation mode
Gesture navigation
Gestures
Date & time
Backup
Software updates
Your Pixel is up to date
Rules
0 rules

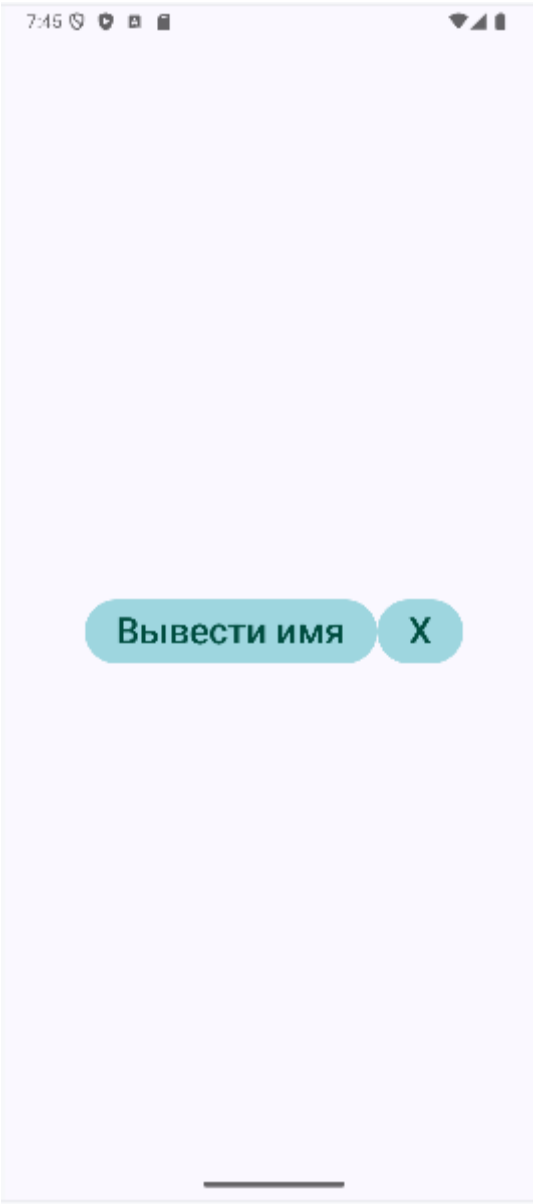

Languages
Preferred Language
System Languages
English (United States)
App languages
Regional preferences
Set units and number preferences
Speech
Voice input
On-device recognition
Text-to-speech output

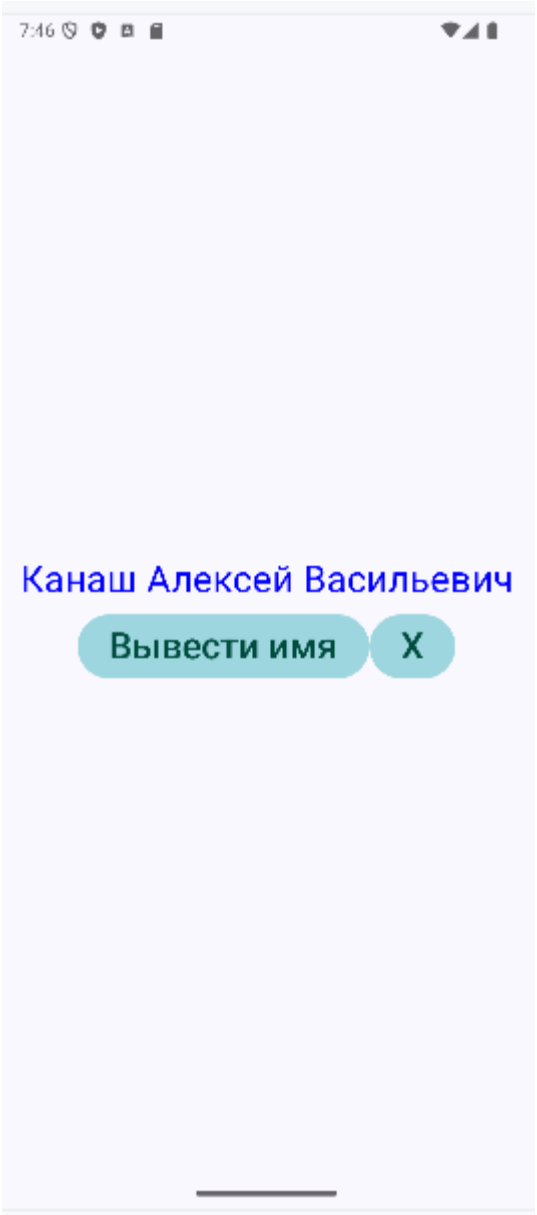

Languages
Your system, apps, and websites use the first supported language from your preferred languages.
Preferred language order
1
English (United States)
System language
Add a language
To select a language for each app, go to app language settings.
Learn more about app languages


Languages
Your system, apps, and websites use the first supported language from your preferred languages.
Preferred language order
1
English (United States)
System language
2
Русский (Беларусь)
Add a language
To select a language for each app, go to app language settings.
Learn more about app languages

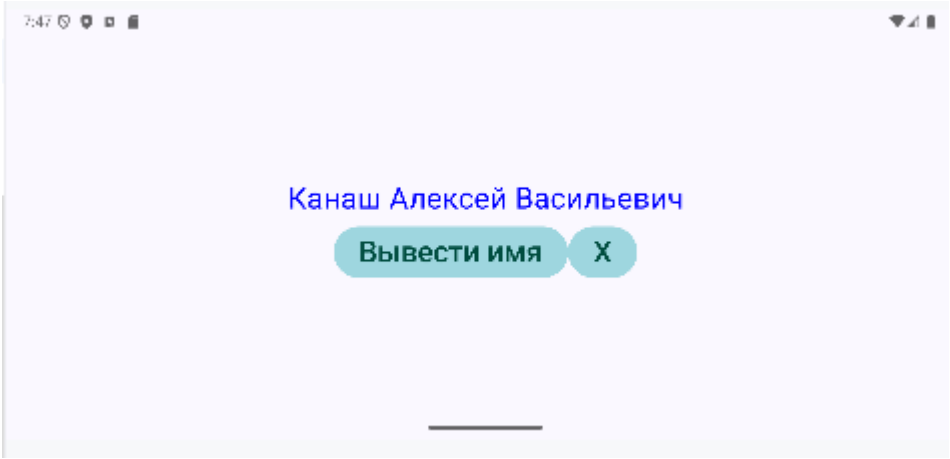

9:35
Hello Алексей!
Введите имя
Алексей
Приветствовать
Алексей
?123
Русский

Задание 2: Создать приложение с одним текстовым полем и двумя кнопками. При нажатии на кнопку 1 выводить в текстовое поле свое ФИО. ФИО использовать как строковый ресурс. При нажатии на кнопку 2 очищать информацию в текстовом поле.





n



Если меняется состояние (здесь по клику меняется текст), то при изменении ориентации мобильного устройства (например, при переходе от портретного режима к альбомному) измененное состояние пропадет.

То же самое касается и других конфигурационных изменений, например, изменения системных настроек. Подобные изменения вызывают не просто рекомпозицию отдельных компонентов, а уничтожение и пересоздание всего класса MainActivity. В итоге вновь созданный объект MainActivity не помнит никаких изменений состояния.

Для сохранения состояния между поворотами экрана и другими подобными системными изменениями конфигурации применяется обертка над функцией remember − функция rememberSaveable, которая сохраняет данные в объект Bundle. Правда, не все объекты можно сохранить, а только данные примитивных типов, либо классов, которые реализуют интерфейс Parcelable.

```
import androidx.compose.runtime.saveable.rememberSaveable
```

Пример использования:

```
var value by rememberSaveable{mutableStateOf("Hello BSUIR!")}
```