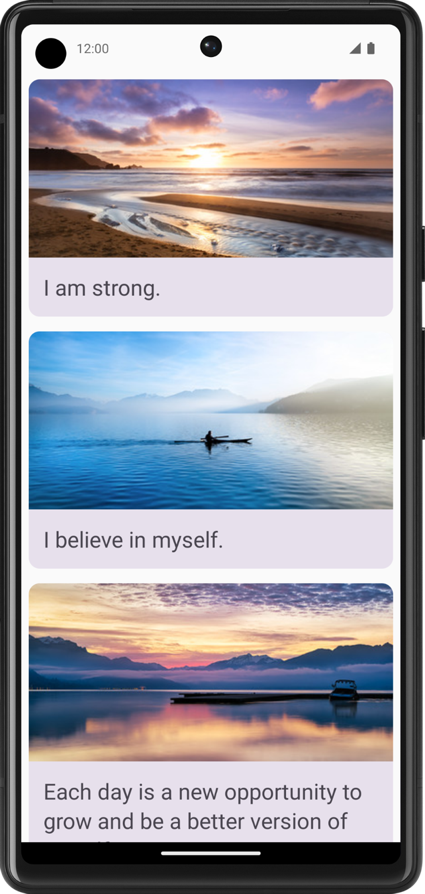
**19 Практическая работа Прокручиваемый список**

Вы будете работать с приложением **«Аффирмации»** , которое отображает список аффирмаций в сочетании с красивыми изображениями, чтобы привнести позитив в ваш день!

Данные уже есть, все, что вам нужно сделать, это взять эти данные и отобразить их в пользовательском интерфейсе.

Готовое изделие будет выглядеть так:



Загрузите стартовый код

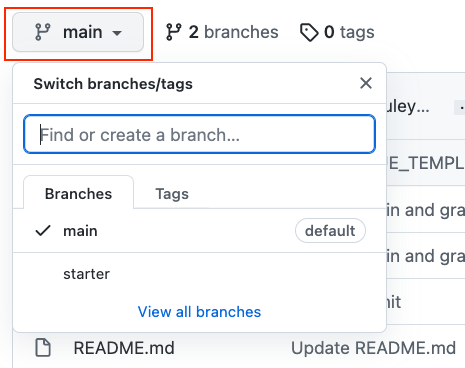
В Android Studio откройте папку basic-android-kotlin-compose-training-affirmations.

**URL стартового кода:**

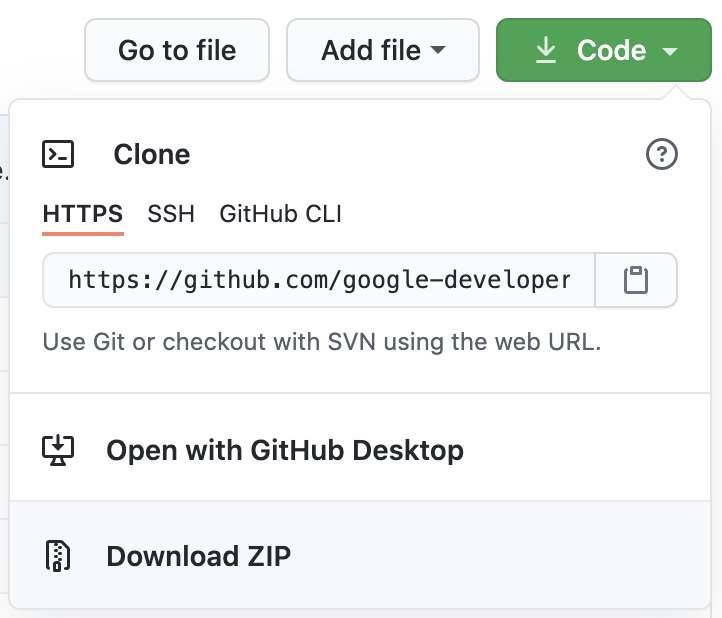
[https://github.com/google-developer-training/basic-android-kotlin-compose-training-affirmations](https://github.com/google-developer-training/basic-android-kotlin-compose-training-affirmations/tree/starter)

**Название ветки со стартовым кодом:** starter

1. Перейдите на предоставленную страницу репозитория GitHub для проекта.
2. Убедитесь, что имя ветки соответствует имени ветки, указанному в кодовой лаборатории. Например, на следующем снимке экрана имя ветки — **main** .



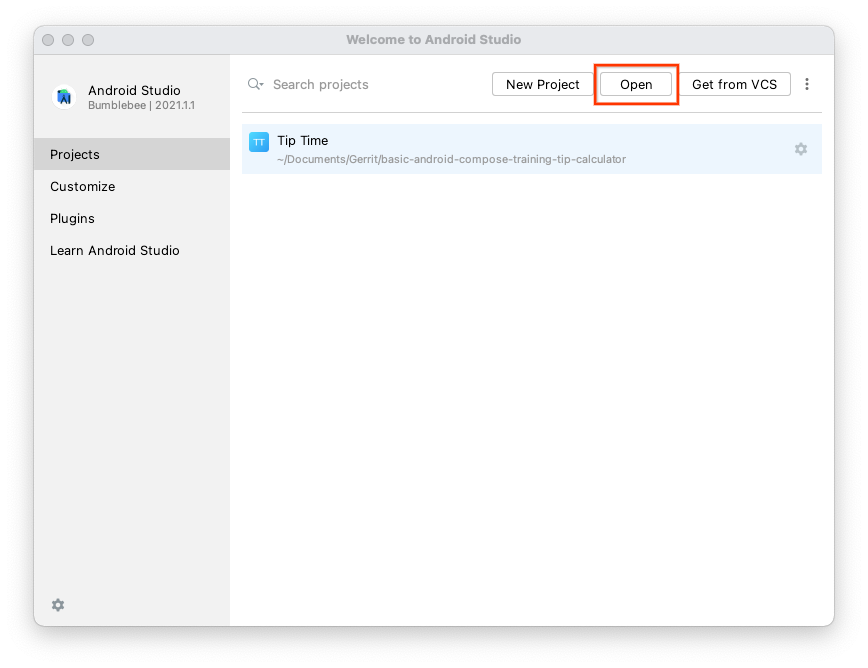
1. На странице проекта GitHub нажмите кнопку **«Код»** , после чего появится всплывающее окно.



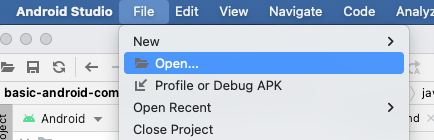
1. Во всплывающем окне нажмите кнопку **«Download ZIP»** , чтобы сохранить проект на свой компьютер. Подождите завершения загрузки.
2. Найдите файл на своем компьютере (вероятно, в папке **«Загрузки»** ).
3. Дважды щелкните ZIP-файл, чтобы распаковать его. При этом создается новая папка, содержащая файлы проекта.

Откройте проект в Android Studio.

1. Запустите Android-студию.
2. В окне **«Добро пожаловать в Android Studio»** нажмите « **Открыть»** .

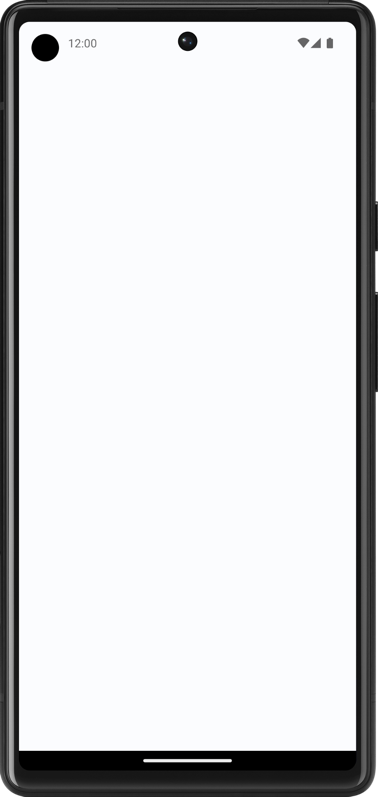


Примечание. Если Android Studio уже открыта, выберите пункт меню **«Файл»** > **«Открыть»** .



1. В браузере файлов перейдите туда, где находится распакованная папка проекта (вероятно, в папке **«Загрузки»** ).
2. Дважды щелкните папку этого проекта.
3. Подождите, пока Android Studio откроет проект.
4. Нажмите кнопку **«Run»** , чтобы создать и запустить приложение. Убедитесь, что он строится так, как ожидалось.

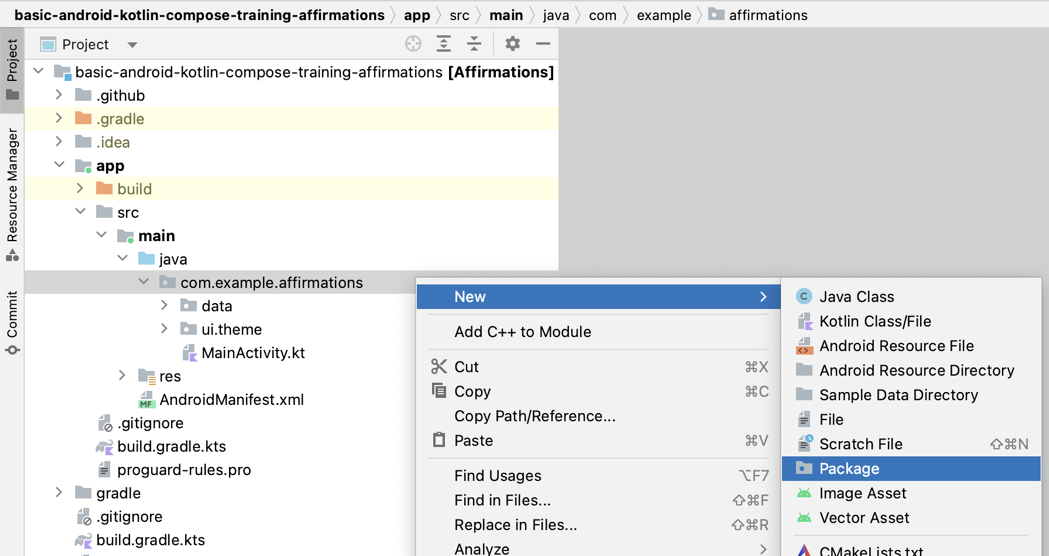
Ожидается, что приложение будет отображать пустой экран, если оно создано на основе  кода ветки starter.



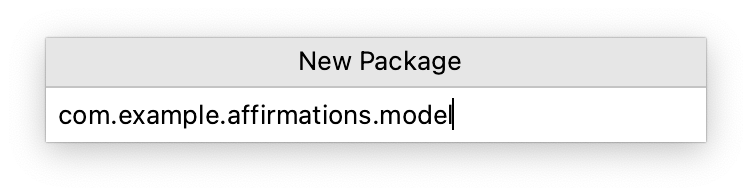
[2. Создайте класс данных элемента списка.](https://developer.android.com/codelabs/basic-android-kotlin-compose-training-add-scrollable-list?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-2%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-training-add-scrollable-list#1)

Создайте класс данных для подтверждения

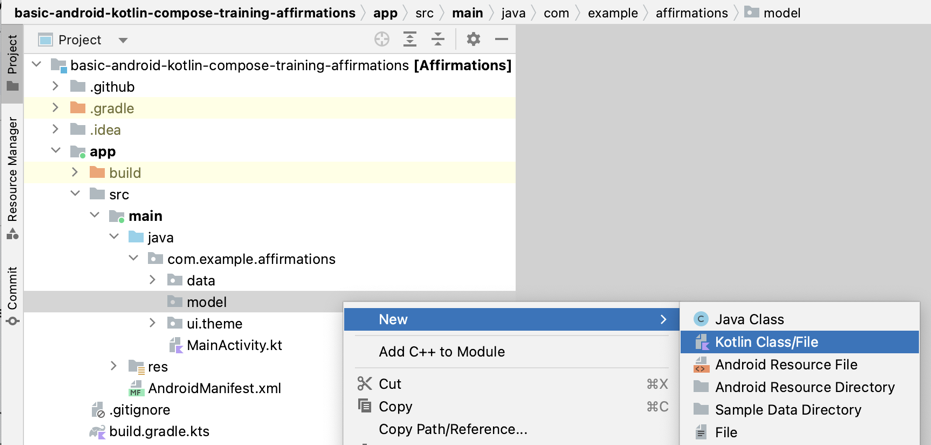
В приложениях Android списки состоят из элементов списков. Для отдельных фрагментов данных это может быть что-то простое, например строка или целое число. Для элементов списка, содержащих несколько фрагментов данных, таких как изображение и текст, вам понадобится класс, содержащий все эти свойства. Классы данных — это тип класса, который содержит только свойства. Они могут предоставлять некоторые служебные методы для работы с этими свойствами.

1. Создайте новый пакет в **com.example.affirmations** .

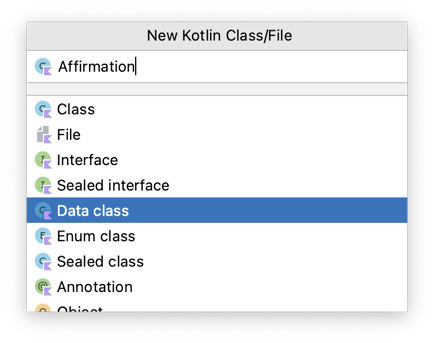
Назовите новую **модель пакета.** Пакет модели будет содержать модель данных, которая будет представлена классом данных. Класс данных будет состоять из свойств, которые представляют информацию, относящуюся к тому, что будет «Подтверждением», которое будет состоять из строкового ресурса и ресурса изображения. Пакеты — это каталоги, содержащие классы и даже другие каталоги.



1. Создайте новый класс в пакете **com.example.affirmations.model** .



Назовите новый класс **Affirmation** и сделайте его **классом Data** .



1. Каждый Affirmation состоит из одного изображения и одной строки. Создайте два val свойства в классе данных Affirmation. Назовите stringResourceId и imageResourceId. Оба они должны быть целыми числами.

Affirmation.kt

data class Affirmation(  
    val stringResourceId: Int,  
    val imageResourceId: Int  
)

1. Аннотируйте свойство stringResourceId с помощью @StringRes аннотации и аннотируйте свойство imageResourceId с помощью @DrawableRes аннотации. stringResourceId - идентификатор текста подтверждения, хранящегося в строковом ресурсе. imageResourceId - идентификатор изображения подтверждения, хранящегося в доступном для рисования ресурсе.

Affirmation.kt

import androidx.annotation.DrawableRes  
import androidx.annotation.StringRes  
  
data class Affirmation(  
    @StringRes val stringResourceId: Int,  
    @DrawableRes val imageResourceId: Int  
)

1. В пакете **com.example.affirmations.data** откройте файл **Datasource.kt** и раскомментируйте два оператора импорта и содержимое класса Datasource.

Datasource.kt

import com.example.affirmations.R  
import com.example.affirmations.model.Affirmation  
  
class Datasource() {  
    fun loadAffirmations(): List<Affirmation> {  
        return listOf<Affirmation>(  
            Affirmation(R.string.affirmation1, R.drawable.image1),  
            Affirmation(R.string.affirmation2, R.drawable.image2),  
            Affirmation(R.string.affirmation3, R.drawable.image3),  
            Affirmation(R.string.affirmation4, R.drawable.image4),  
            Affirmation(R.string.affirmation5, R.drawable.image5),  
            Affirmation(R.string.affirmation6, R.drawable.image6),  
            Affirmation(R.string.affirmation7, R.drawable.image7),  
            Affirmation(R.string.affirmation8, R.drawable.image8),  
            Affirmation(R.string.affirmation9, R.drawable.image9),  
            Affirmation(R.string.affirmation10, R.drawable.image10))  
    }

}

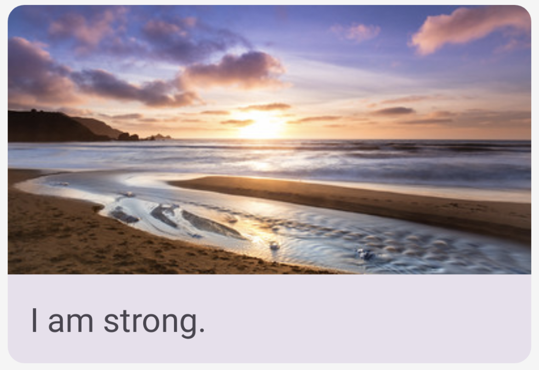
**Примечание.** Метод loadAffirmations()собирает все данные, представленные в стартовом коде, и возвращает их в виде списка. Вы будете использовать это позже для создания прокручиваемого списка.

[3. Добавьте список в свое приложение](https://developer.android.com/codelabs/basic-android-kotlin-compose-training-add-scrollable-list?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-2%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-training-add-scrollable-list#2)

Создайте карточку элемента списка

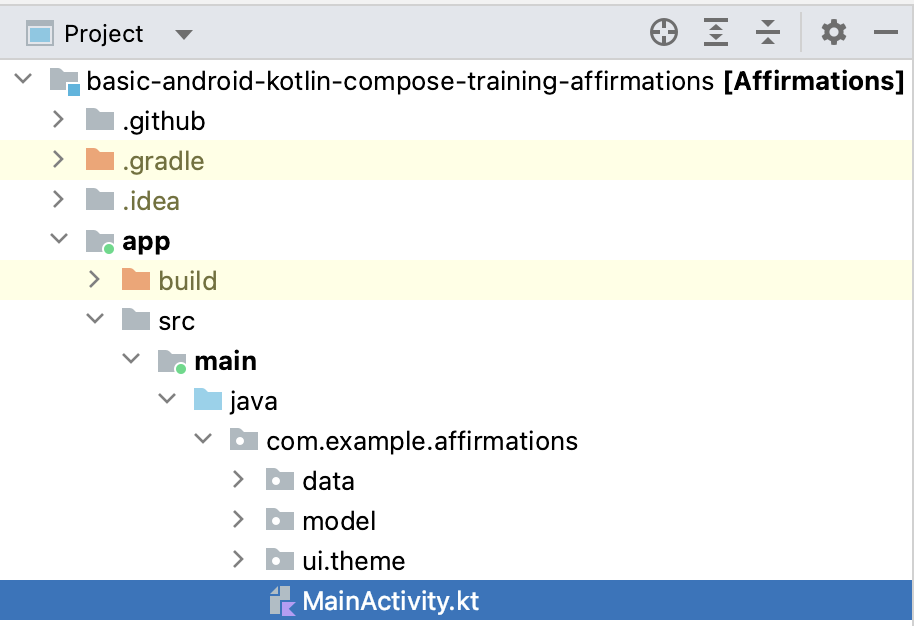
Это приложение предназначено для отображения списка аффирмаций. Первым шагом в настройке пользовательского интерфейса для отображения списка является создание элемента списка. Каждый элемент подтверждения состоит из изображения и строки. Данные для каждого из этих элементов поставляются со стартовым кодом, и вы создадите компонент пользовательского интерфейса для отображения такого элемента.

Элемент будет состоять из составного объекта Card, содержащего Image и Text. В Compose Card— это поверхность, которая отображает содержимое и действия в одном контейнере. Карточка подтверждения будет выглядеть в предварительном просмотре следующим образом:



На карточке показано изображение с текстом под ним. Такого вертикального макета можно добиться, используя Column, завернутый в составной элемент Card. Вы можете попробовать это самостоятельно или выполнить следующие действия, чтобы добиться этого.

1. Откройте файл **MainActivity.kt** .



1. Создайте новый метод под методом AffirmationsApp() с именем AffirmationCard() и добавьте к нему аннотацию @Composable.

MainActivity.kt

@Composable  
fun AffirmationsApp() {  
}  
  
@Composable  
fun AffirmationCard() {  
  
}

1. Отредактируйте сигнатуру метода, чтобы использовать объект Affirmation в качестве параметра. Объект Affirmation исходит из пакета model.

MainActivity.kt

import com.example.affirmations.model.Affirmation  
  
@Composable  
fun AffirmationCard(affirmation: Affirmation) {  
  
}

1. Добавьте параметр modifier в подпись. Установите значение по умолчанию для параметра Modifier.

MainActivity.kt

@Composable  
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier = Modifier) {  
  
}

**Примечание.** Рекомендуется передавать модификатор каждому составному элементу и устанавливать для него значение по умолчанию.

1. Внутри метода AffirmationCard вызовите составной объект Card. Передайте параметр modifier.

MainActivity.kt

import androidx.compose.material3.Card  
  
@Composable  
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier = Modifier) {  
    Card(modifier = modifier) {  
  
    }  
}

1. Добавьте Column внутри Card. Элементы в Column располагаются в пользовательском интерфейсе вертикально. Это позволяет разместить изображение над соответствующим текстом. И наоборот, Row располагает содержащиеся в нем элементы горизонтально.

MainActivity.kt

import androidx.compose.foundation.layout.Column  
  
@Composable  
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier = Modifier) {  
    Card(modifier = modifier) {  
        Column {  
  
        }  
    }  
  
}

1. Добавьте Image внутри лямбда-тела Column. Напомним, что для компонуемого объекта Image всегда требуется ресурс, а также файл contentDescription. Ресурс painterResource должен быть передан в параметр painter. Этот метод загружает либо векторные рисунки, либо растровые форматы ресурсов, такие как PNG. Кроме того, передайте stringResource в параметр contentDescription.

MainActivity.kt

import androidx.compose.foundation.Image  
import androidx.compose.ui.res.painterResource  
import androidx.compose.ui.res.stringResource  
  
@Composable  
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier = Modifier) {  
    Card(modifier = modifier) {  
        Column {  
            Image(  
                painter = painterResource(affirmation.imageResourceId),  
                contentDescription = stringResource(affirmation.stringResourceId),  
            )  
        }  
    }  
}

1. В дополнение к параметрам painter and contentDescription передайте modifier и contentScale. contentScale определяет, как изображение должно масштабироваться и отображаться. Объект Modifier должен иметь набор атрибутов fillMaxWidth и высоту 194.dp. Должно contentScale быть ContentScale.Crop.

MainActivity.kt

import androidx.compose.foundation.layout.fillMaxWidth  
import androidx.compose.foundation.layout.height  
import androidx.compose.ui.unit.dp  
import androidx.compose.ui.layout.ContentScale  
  
@Composable  
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier = Modifier) {  
    Card(modifier = modifier) {  
        Column {  
            Image(  
                painter = painterResource(affirmation.imageResourceId),  
                contentDescription = stringResource(affirmation.stringResourceId),  
**modifier = Modifier  
                    .fillMaxWidth()  
                    .height(194.dp),  
                contentScale = ContentScale.Crop**            )  
        }  
    }  
}

1. Внутри Column создайте Text после Image. Передайте параметру stringResource объект с атрибутом, установленным в значение affirmation.stringResourceId, и установите текстовую тему, передав параметр .textModifierpadding16.dpMaterialTheme.typography.headlineSmallstyle

MainActivity.kt

import androidx.compose.material3.Text  
import androidx.compose.foundation.layout.padding  
import androidx.compose.ui.platform.LocalContext  
  
@Composable

fun AffirmationCard(affirmation: Affirmation, modifier: Modifier = Modifier) {  
    Card(modifier = modifier) {  
        Column {  
            Image(  
                painter = painterResource(affirmation.imageResourceId),  
                contentDescription = stringResource(affirmation.stringResourceId),  
                modifier = Modifier  
                    .fillMaxWidth()  
                    .height(194.dp),  
                contentScale = ContentScale.Crop  
            )  
            Text(  
                text = LocalContext.current.getString(affirmation.stringResourceId),  
                modifier = Modifier.padding(16.dp),  
                style = MaterialTheme.typography.headlineSmall  
            )  
        }  
    }  
}

Предварительный просмотр составной карты AffirmationCard

Карта — это основа пользовательского интерфейса приложения **«Аффирмации»** , и вы усердно работали над ее созданием! Чтобы убедиться, что карточка выглядит правильно, вы можете создать составной объект, который можно просмотреть, не запуская все приложение.

1. Создайте private метод с именем AffirmationCardPreview(). Аннотируйте метод с помощью @Preview и @Composable.

MainActivity.kt

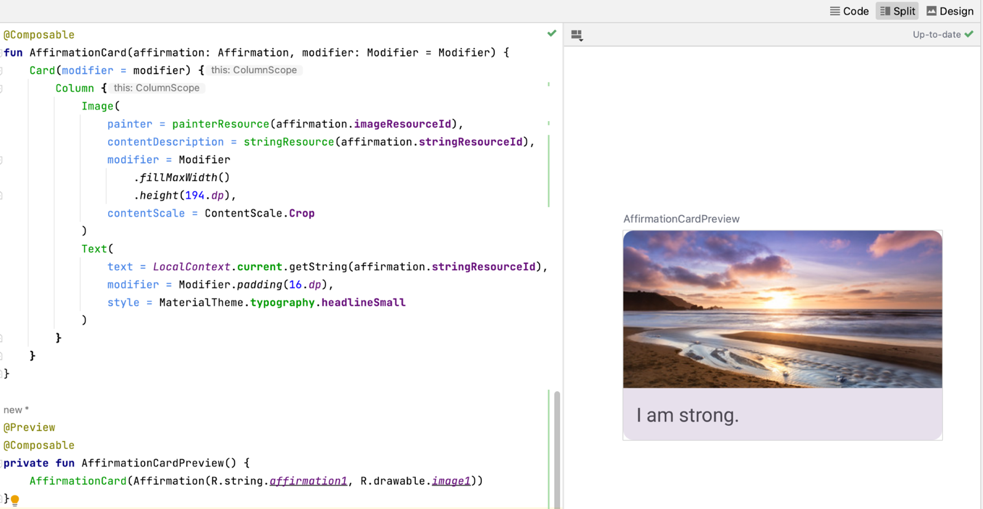
import androidx.compose.ui.tooling.preview.Preview  
  
@Preview  
@Composable  
private fun AffirmationCardPreview() {  
  
}

1. Внутри метода вызовите составной объект AffirmationCard и передайте ему новый объект Affirmation со R.string.affirmation1 строковым ресурсом и R.drawable.image1 доступным для рисования ресурсом, переданным в его конструктор.

MainActivity.kt

@Preview  
@Composable  
private fun AffirmationCardPreview() {  
    AffirmationCard(Affirmation(R.string.affirmation1, R.drawable.image1))  
}

1. Откройте вкладку **«Split»** , и вы увидите предварительный просмотр файла AffirmationCard. При необходимости нажмите  **«Build & Refresh»** на панели **«Design»** , чтобы отобразить предварительный просмотр.



Создать список

Компонент элемента списка является строительным блоком списка. После создания элемента списка вы можете использовать его для создания самого компонента списка.

1. Создайте функцию с именем AffirmationList(), добавьте к ней аннотацию @Composable и объявите объект Affirmation в качестве параметра List в сигнатуре метода.

MainActivity.kt

@Composable  
fun AffirmationList(affirmationList: List<Affirmation>) {  
      
}

1. Объявите объект modifier в качестве параметра в сигнатуре метода со значением по умолчанию Modifier.

MainActivity.kt

@Composable  
fun AffirmationList(affirmationList: List<Affirmation>, modifier: Modifier = Modifier) {  
  
}

1. В Jetpack Compose прокручиваемый список можно создать с помощью LazyColumn. Разница между LazyColumn и Column заключается в том, что Column следует использовать, когда у вас небольшое количество элементов для отображения, поскольку Compose загружает их все одновременно. Column может содержать только заранее определенное или фиксированное количество составных элементов. LazyColumn может добавлять контент по требованию, что удобно для длинных списков, особенно когда длина списка неизвестна. LazyColumn также обеспечивает прокрутку по умолчанию без дополнительного кода. Объявите составную часть LazyColumn функции AffirmationList(). Передайте объект modifier в качестве аргумента методу LazyColumn.

MainActivity.kt

import androidx.compose.foundation.lazy.LazyColumn  
  
@Composable  
fun AffirmationList(affirmationList: List<Affirmation>, modifier: Modifier = Modifier) {  
    LazyColumn(modifier = modifier) {  
  
    }  
}

1. В теле лямбда-выражения LazyColumn вызовите метод items()и передайте ему affirmationList. Этот метод items() заключается в том, как вы добавляете элементы в файл LazyColumn. Этот метод в некоторой степени уникален для этого составного объекта и в остальном не является обычной практикой для большинства составных объектов.

MainActivity.kt

import androidx.compose.foundation.lazy.items  
  
@Composable  
fun AffirmationList(affirmationList: List<Affirmation>, modifier: Modifier = Modifier) {  
    LazyColumn(modifier = modifier) {  
        items(affirmationList) {  
  
        }  
    }  
}

1. Для вызова метода items()требуется лямбда-функция. В этой функции укажите параметр affirmation,  который представляет один элемент подтверждения из файла affirmationList.

MainActivity.kt

@Composable  
fun AffirmationList(affirmationList: List<Affirmation>, modifier: Modifier = Modifier) {  
    LazyColumn(modifier = modifier) {  
        items(affirmationList) **{ affirmation ->**  
  
        }  
    }  
}

1. Для каждого утверждения в списке укажем AffirmationCard(). Передайте ему affirmation и объект Modifier с  атрибутом padding = 8.dp.

MainActivity.kt

@Composable  
fun AffirmationList(affirmationList: List<Affirmation>, modifier: Modifier = Modifier) {  
    LazyColumn(modifier = modifier) {  
        items(affirmationList) { affirmation ->  
            **AffirmationCard(  
                affirmation = affirmation,  
                modifier = Modifier.padding(8.dp)  
            )**        }  
    }  
}

Показать список

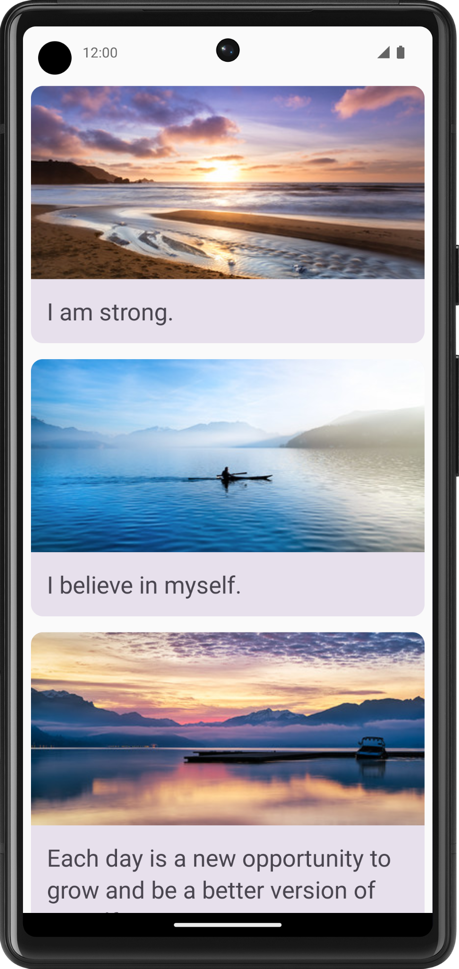
1. В лямбде вызовите компонуемый объект AffirmationList и перейдите DataSource().loadAffirmations() к параметру affirmationList.

**Примечание.** Класс DataSource находится в пакете data.

MainActivity.kt

import com.example.affirmations.data.Datasource  
  
@Composable  
fun AffirmationsApp() {  
    AffirmationList(  
        affirmationList = Datasource().loadAffirmations(),  
    )  
}

Запустите приложение  **«Аффирмации»** на устройстве или в эмуляторе и посмотрите готовый продукт!



[5. Вывод](https://developer.android.com/codelabs/basic-android-kotlin-compose-training-add-scrollable-list?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-2%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-training-add-scrollable-list#4)

Теперь вы знаете, как создавать карточки, элементы списков и прокручиваемые списки с помощью Jetpack Compose! Имейте в виду, что это всего лишь основные инструменты для создания списка. Вы можете дать волю своему творчеству и настраивать элементы списка по своему усмотрению!