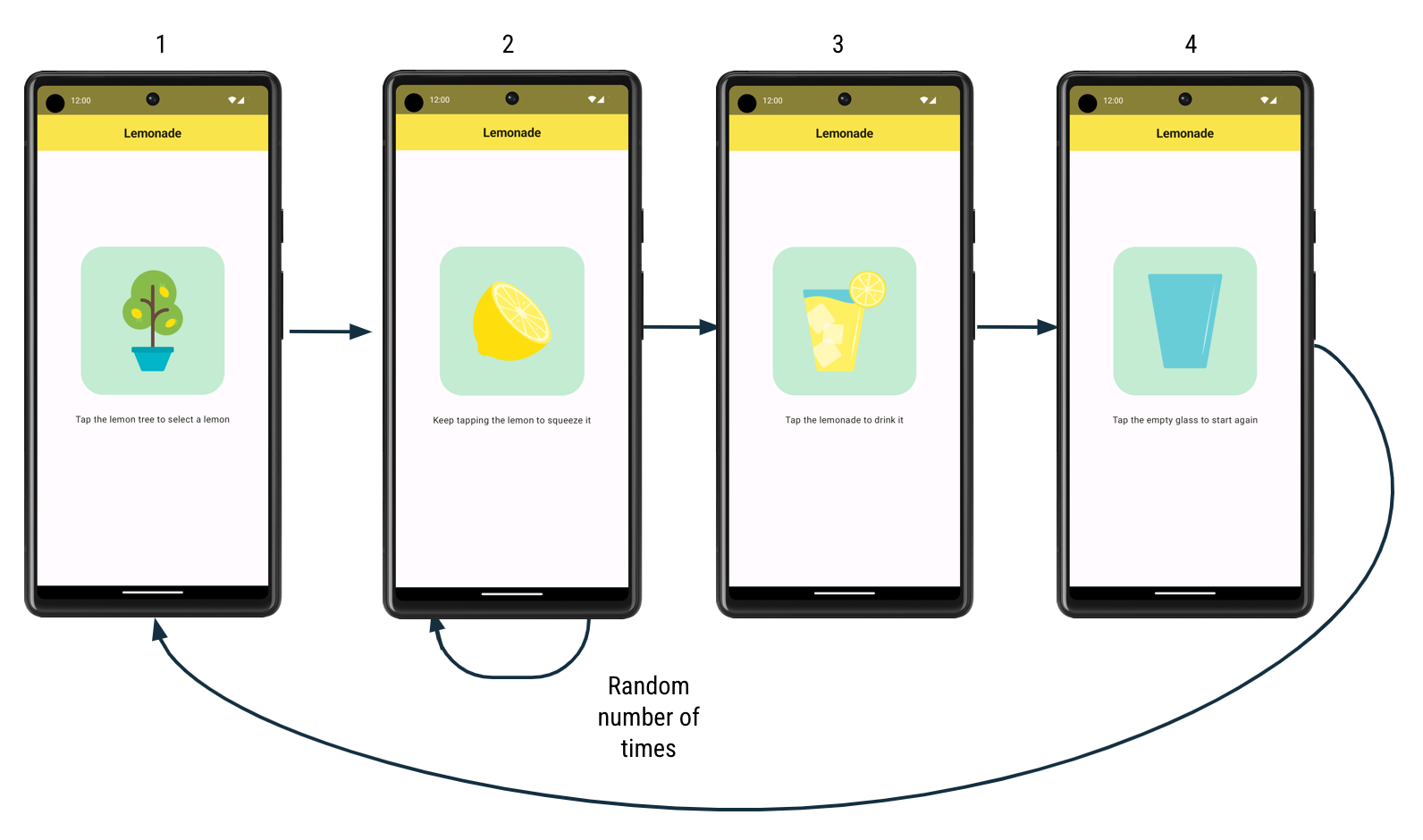
Лабораторная работа: Приложение Lemonade (клики)

## [**2. Обзор приложения**](https://developer.android.com/codelabs/basic-android-kotlin-compose-button-click-practice-problem?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-2-pathway-2%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-button-click-practice-problem#1)

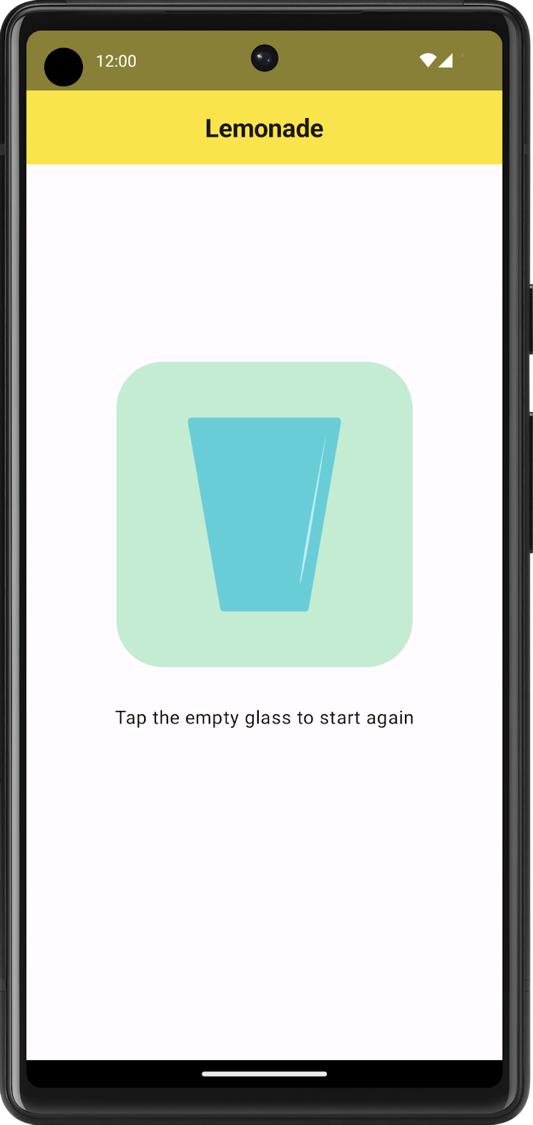
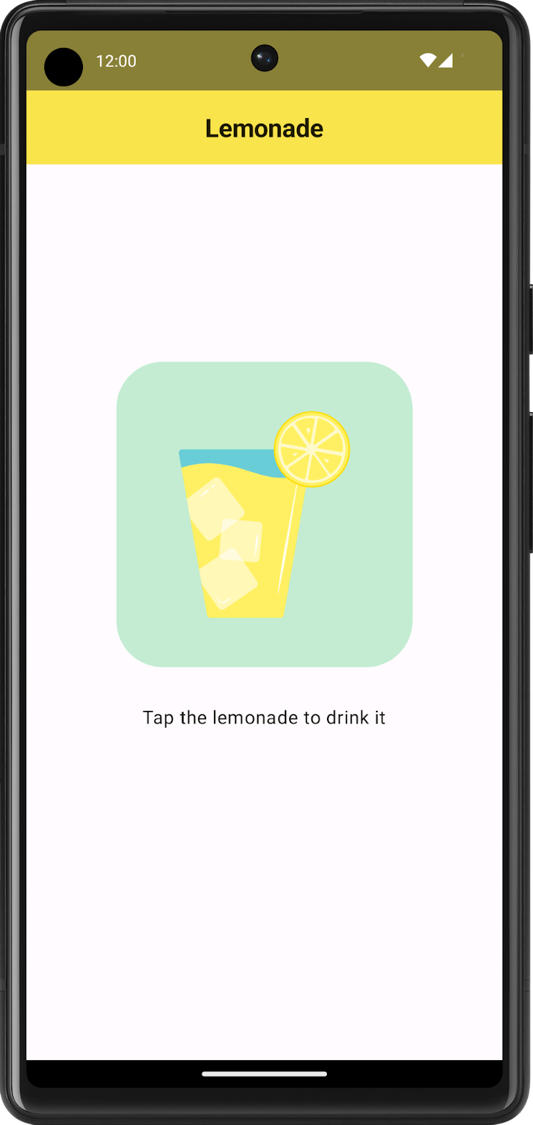
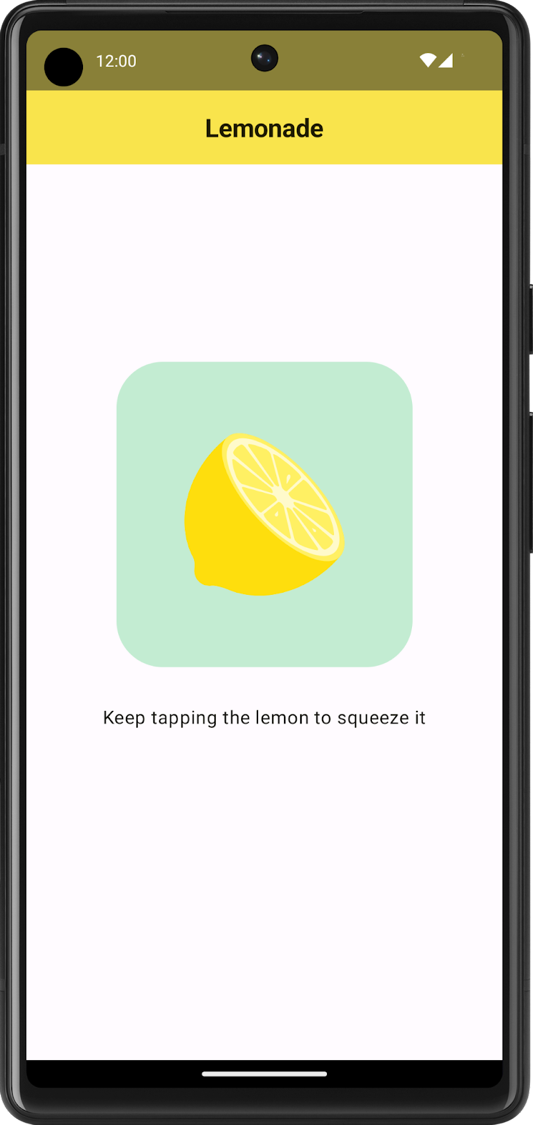
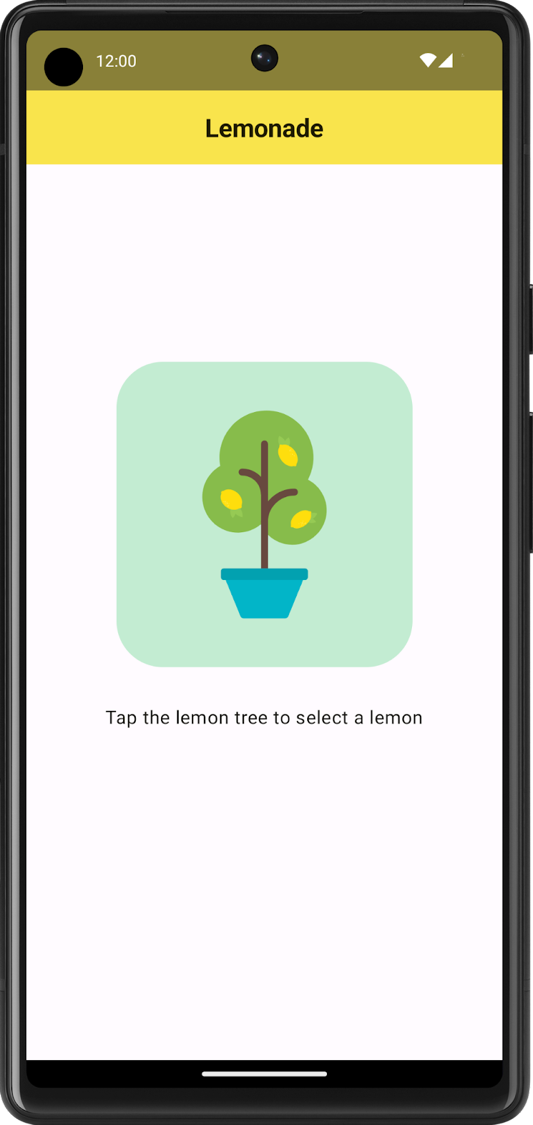
Вы поможете нам воплотить в жизнь нашу идею создания цифрового лимонада! Цель — создать простое интерактивное приложение, которое позволит вам выжимать сок из лимона, нажимая на изображение на экране, пока вы не выпьете стакан лимонада. Считайте это метафорой или просто забавным способом скоротать время!



Вот как работает приложение:

1. Когда пользователь впервые запускает приложение, он видит лимонное дерево. Есть метка, которая предлагает им нажать на изображение лимонного дерева, чтобы «выбрать» лимон из дерева.
2. После того, как пользователь коснется лимонного дерева, он увидит лимон. Им предлагается постучать по лимону, чтобы «выжать» его и приготовить лимонад. Им нужно несколько раз постучать по лимону, чтобы выжать его. Количество нажатий, необходимых для выжимания лимона, каждый раз разное и представляет собой случайно сгенерированное число от 2 до 4 (включительно).
3. После того, как они постучали по лимону необходимое количество раз, они увидели освежающий стакан лимонада! Их просят постучать по стакану, чтобы «выпить» лимонад.
4. После того, как они постучали по стакану с лимонадом, они увидели пустой стакан. Их просят постучать по пустому стакану, чтобы начать заново.
5. После того, как они стукнут по пустому стакану, они увидят лимонное дерево и смогут начать процесс заново. Еще лимонада, пожалуйста!

Вот увеличенные скриншоты того, как выглядит приложение:



Для каждого этапа приготовления лимонада на экране отображается разное изображение и текстовая метка, а также разное поведение приложения на щелчок. Например, когда пользователь нажимает на лимонное дерево, приложение показывает лимон.

Ваша задача — создать макет пользовательского интерфейса приложения и реализовать логику, позволяющую пользователю пройти все этапы приготовления лимонада.

## [**3. Начните работу**](https://developer.android.com/codelabs/basic-android-kotlin-compose-button-click-practice-problem?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-2-pathway-2%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-button-click-practice-problem#2)

## **Создать проект**

В Android Studio создайте новый проект с шаблоном **Empty Activity** со следующими данными:

* Название: Lemonade
* Имя пакета: com.example.lemonade
* Минимальный SDK: 24

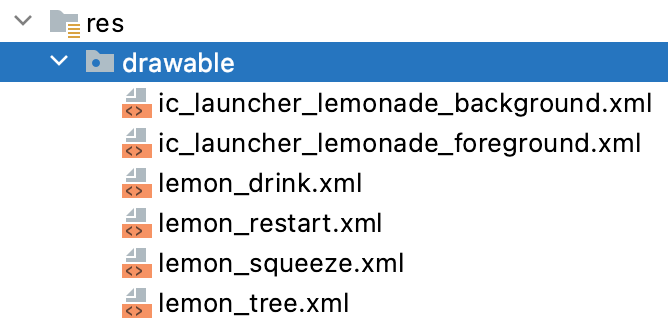
## **Добавить изображения**

Вам предоставляются четыре векторных файла с возможностью рисования, которые вы используете в приложении Lemonade.

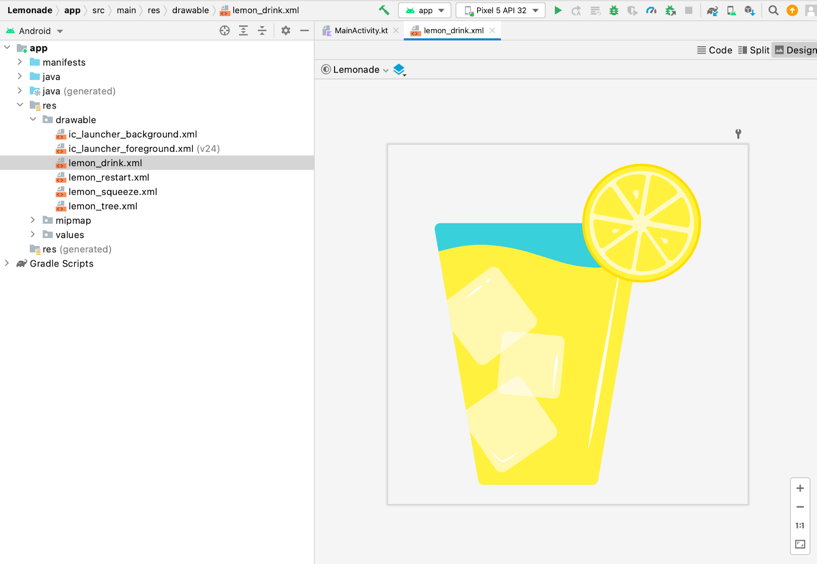
Получите файлы:

1. Загрузите [zip-файл изображений](https://github.com/google-developer-training/basic-android-kotlin-compose-training-lemonade/raw/main/lemonade_images.zip" \t "_blank) для приложения.
2. Дважды щелкните zip-файл. На этом этапе изображения разархивируются в папку.
3. Добавьте изображения в папку drawable вашего приложения. Если вы не помните, как это сделать, в лабораторной с приложением Dice Roller.

Папка вашего проекта должна выглядеть так, как показано на следующем снимке экрана, на котором ресурсы теперь отображаются в каталоге **res > drawable** : lemon\_drink.xml , lemon\_restart.xml, lemon\_squeeze.xml, lemon\_tree.xml



1. Дважды щелкните векторный файл, который можно рисовать, чтобы просмотреть предварительное изображение.
2. Выберите панель **«Design»** , чтобы просмотреть изображение во всю ширину.



После включения файлов изображений в ваше приложение вы можете ссылаться на них в своем коде. Например, если векторный файл для рисования называется lemon\_tree.xml, то в коде Kotlin вы можете обратиться к рисуемому файлу, используя его идентификатор ресурса в формате R.drawable.lemon\_tree.

## **Добавить строковые ресурсы**

Добавьте следующие строки в свой проект в файле **res > Values > strings.xml** :

* Tap the lemon tree to select a lemon
* Keep tapping the lemon to squeeze it
* Tap the lemonade to drink it
* Tap the empty glass to start again

Следующие строки также необходимы в вашем проекте. Они не отображаются на экране в пользовательском интерфейсе, но используются для описания содержимого изображений в вашем приложении, чтобы описать, что это за изображения. Добавьте эти дополнительные строки в файл вашего приложения strings.xml:

* Lemon tree
* Lemon
* Glass of lemonade
* Empty glass

Присвойте каждому строковому ресурсу подходящее имя-идентификатор, описывающее содержащееся в нем значение. Например, для строки "Lemon" вы можете объявить ее в файле strings.xml с идентификатором name «lemon\_content\_description», а затем ссылаться на нее в своем коде, используя идентификатор ресурса R.string.lemon\_content\_description.

## **Этапы приготовления лимонада**

Теперь у вас есть строковые ресурсы и ресурсы изображений, необходимые для реализации приложения. Вот краткое описание каждого шага приложения и того, что отображается на экране:

|  |  |
| --- | --- |
| Шаг 1:   * Текст: Tap the lemon tree to select a lemon * Изображение: Лимонное дерево ( lemon\_tree.xml) |  |
| Шаг 2:   * Текст:Keep tapping the lemon to squeeze it * Изображение: Лимон ( lemon\_squeeze.xml) |  |
| Шаг 3:   * Текст:Tap the lemonade to drink it * Изображение: Полный стакан лимонада ( lemon\_drink.xml) |  |
| Шаг 4:   * Текст:Tap the empty glass to start again * Изображение: Пустой стакан ( lemon\_restart.xml) |  |

## **Добавьте визуальный блеск**

Чтобы ваша версия приложения выглядела так, как показано на последних скриншотах, в приложение нужно внести еще пару визуальных изменений:

* Увеличьте размер шрифта текста, чтобы он был больше размера шрифта по умолчанию (например, 18sp).
* Добавьте дополнительное пространство между текстовой меткой и изображением под ней, чтобы они не располагались слишком близко друг к другу (например, 16dp).
* Придайте кнопке акцентный цвет и слегка закруглите углы, чтобы пользователи знали, что они могут нажать на изображение.

Если вы хотите бросить вызов себе, создайте остальную часть приложения на основе описания того, как оно должно работать. Если вам нужны дополнительные рекомендации, перейдите к следующему разделу.

**Предупреждение:** не продолжайте читать остальную часть инструкций, если вы не хотите, чтобы были раскрыты подсказки. Рекомендуется попробовать решить проблему самостоятельно и обращаться к остальным инструкциям только в том случае, если вы застряли.

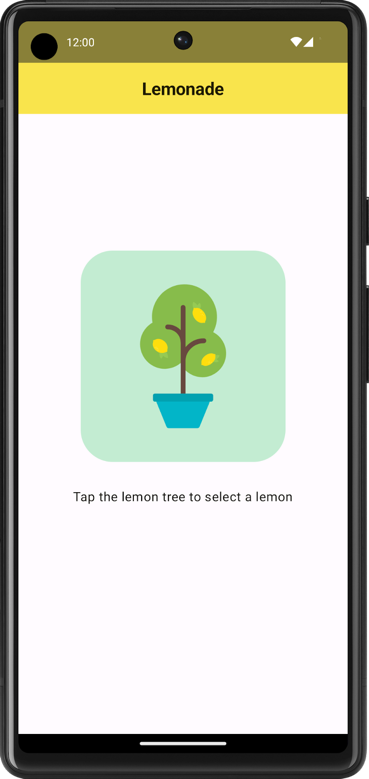
## [**4. Спланируйте, как создать приложение.**](https://developer.android.com/codelabs/basic-android-kotlin-compose-button-click-practice-problem?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-2-pathway-2%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-button-click-practice-problem#3)

При создании приложения рекомендуется сначала создать минимальную рабочую версию приложения. Затем постепенно добавляйте больше функций, пока не завершите все необходимые функции. Определите небольшую часть комплексной функциональности, которую вы можете создать в первую очередь.

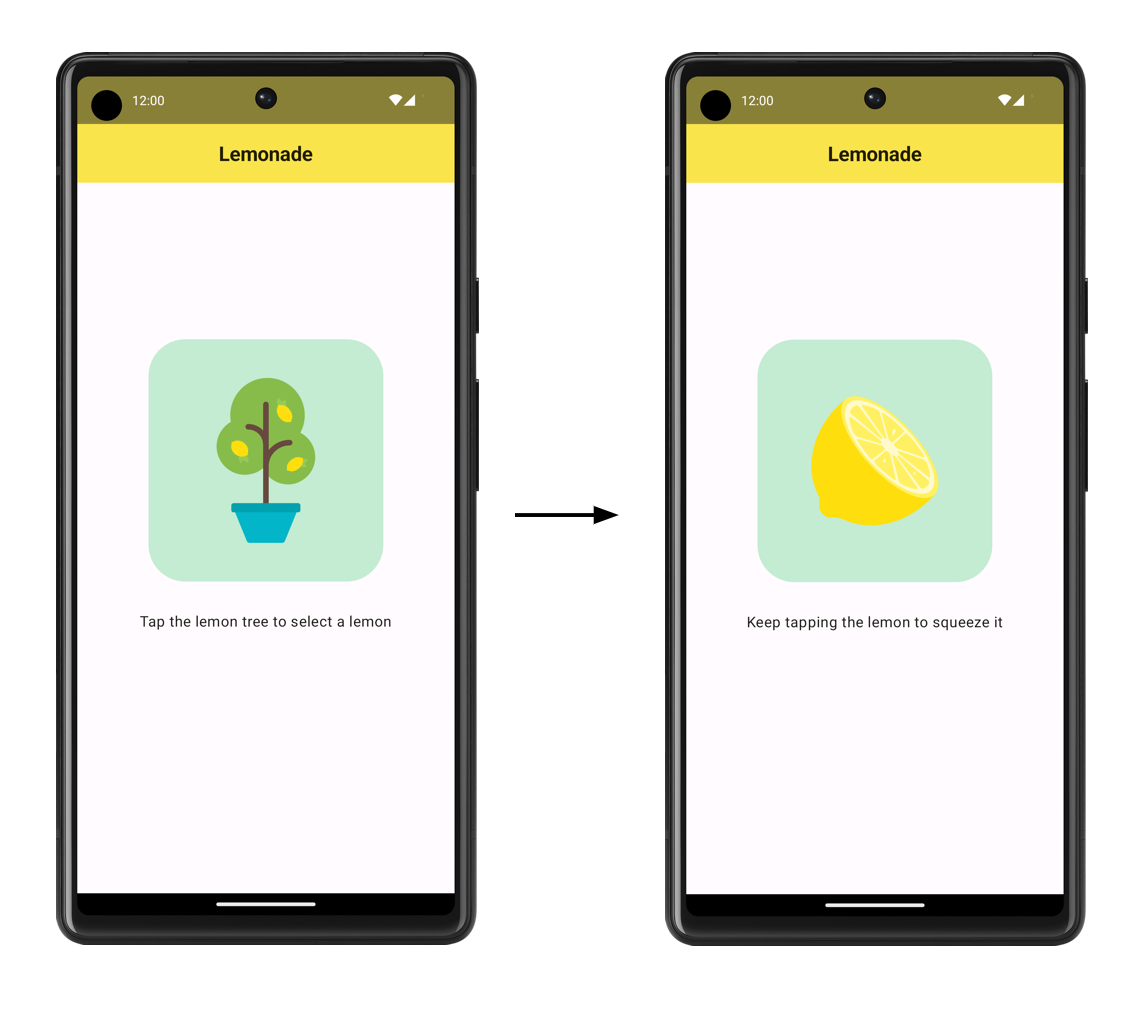
Обратите внимание, что в приложении Lemonade ключевая часть приложения переходит от одного шага к другому, каждый раз показывая другое изображение и текстовую метку. Первоначально вы можете игнорировать особое поведение состояния сжатия, поскольку вы можете добавить эту функциональность позже, после того, как создадите основу приложения.

Ниже приведен общий обзор шагов, которые вы можете предпринять для создания приложения:

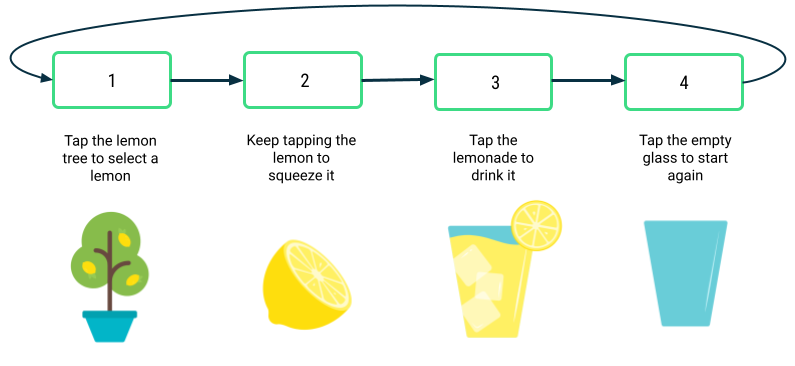
1. Создайте макет пользовательского интерфейса для первого этапа приготовления лимонада, который предлагает пользователю выбрать лимон из дерева. На данный момент вы можете пропустить рамку вокруг изображения, потому что это визуальная деталь, которую вы сможете добавить позже.



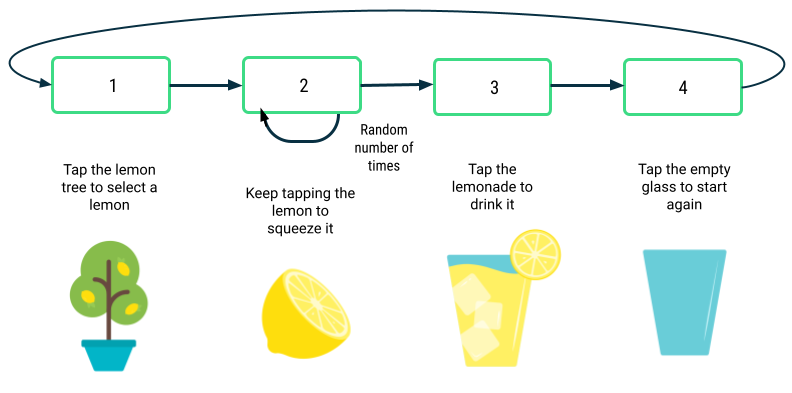
1. Реализуйте поведение в приложении так, чтобы, когда пользователь касается лимонного дерева, приложение отображало изображение лимона и соответствующую ему текстовую метку. Здесь описаны первые два этапа приготовления лимонада.



1. Добавьте код, чтобы приложение отображало остальные шаги по приготовлению лимонада при каждом нажатии на изображение. На этом этапе одно нажатие на лимон может перейти к отображению стакана лимонада.



1. Добавьте пользовательское поведение для этапа выжимания лимона, чтобы пользователю нужно было «сжать» или постучать по лимону определенное количество раз, которое генерируется случайным образом от 2 до 4.



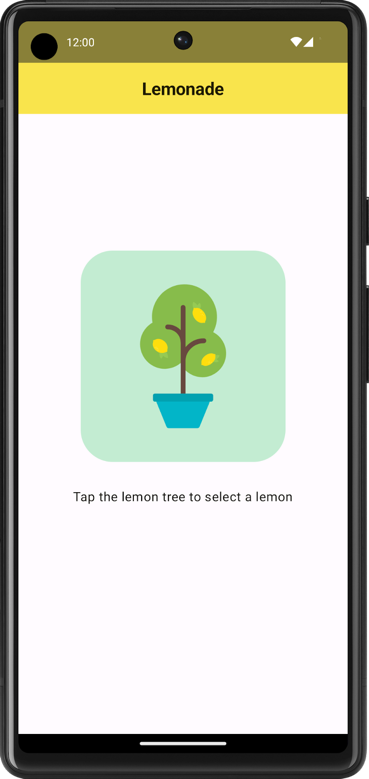
1. Завершите приложение, указав любые другие необходимые детали визуальной доработки. Например, измените размер шрифта и добавьте рамку вокруг изображения, чтобы приложение выглядело более изысканно. Убедитесь, что приложение соответствует хорошим практикам кодирования, таким как соблюдение [рекомендаций по стилю кодирования Kotlin](https://developer.android.com/kotlin/style-guide) и добавление комментариев в ваш код.

Если вы можете использовать эти общие шаги для реализации приложения Lemonade, продолжайте и создайте приложение самостоятельно. Если вы обнаружите, что вам нужны дополнительные рекомендации по каждому из этих пяти шагов, переходите к следующему разделу.

## [**5. Создание приложения**](https://developer.android.com/codelabs/basic-android-kotlin-compose-button-click-practice-problem?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-2-pathway-2%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-button-click-practice-problem#4)

## **Создайте макет пользовательского интерфейса**

Сначала измените приложение так, чтобы в центре экрана отображалось изображение лимонного дерева и соответствующая ему текстовая метка с надписью Tap the lemon tree to select a lemon.  Между текстом и изображением под ним также должно быть пространство 16dp.



Если это поможет, вы можете использовать в файле следующий стартовый код MainActivity.kt:

package com.example.lemonade  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.material3.MaterialTheme  
import androidx.compose.material3.Surface  
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.lemonade.ui.theme.LemonadeTheme  
  
class MainActivity : ComponentActivity() {  
   override fun onCreate(savedInstanceState: Bundle?) {  
       super.onCreate(savedInstanceState)  
       setContent {  
           LemonadeTheme {  
               LemonApp()  
           }  
       }  
   }  
}  
  
@Composable  
fun LemonApp() {  
   // A surface container using the 'background' color from the theme  
   Surface(  
       modifier = Modifier.fillMaxSize(),  
       color = MaterialTheme.colorScheme.background  
   ) {  
       Text(text = "Hello there!")  
   }  
}  
  
@Preview(showBackground = true)  
@Composable  
fun DefaultPreview() {  
   LemonadeTheme {  
       LemonApp()  
   }  
}

Этот код похож на код, автоматически созданный Android Studio. Однако вместо  составного объекта Greeting() определен составной объект LemonApp(), который не ожидает параметра. Составной элемент DefaultPreview()также обновлен для использования  составного элемента LemonApp(), поэтому вы можете легко просмотреть свой код.

После ввода этого кода в Android Studio измените компонуемый объект LemonApp(), который должен содержать содержимое приложения. Вот несколько вопросов, которые помогут вам в мыслительном процессе:

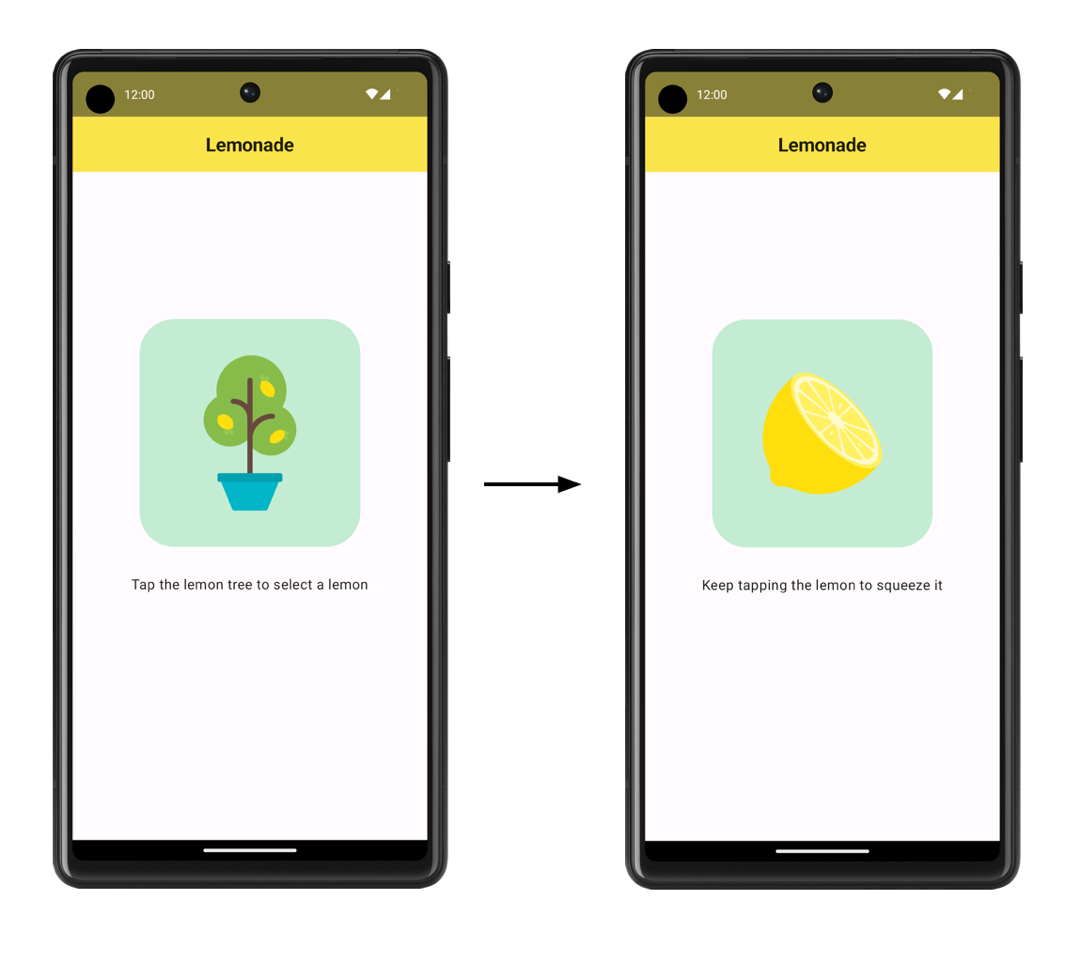
* Какие составные части вы будете использовать?
* Существует ли стандартный [компонент компоновки Compose](https://developer.android.com/jetpack/compose/layouts/basics#standard-layouts) , который поможет вам расположить составные элементы в нужных позициях?

Выполните этот шаг, чтобы лимонное дерево и текстовая метка отображались в вашем приложении при запуске приложения. [Предварительно просмотрите](https://developer.android.com/jetpack/compose/tooling#preview) компоновку в Android Studio, чтобы увидеть, как будет выглядеть пользовательский интерфейс при изменении кода. Запустите приложение и убедитесь, что оно выглядит так, как показано на снимке экрана, который вы видели ранее в этом разделе.

Когда закончите, вернитесь к этим инструкциям, если вам нужны дополнительные инструкции о том, как добавить поведение при нажатии на изображение.

## **Добавить поведение клика**

Далее вы добавите код, чтобы, когда пользователь нажимает на изображение лимонного дерева, изображение лимона появлялось вместе с текстовой меткой Keep tapping the lemon to squeeze it. Другими словами, когда вы нажимаете на лимонное дерево, текст и изображение меняются.



Ранее вы узнали, как сделать кнопку нажимаемой. В случае с приложением Lemonade – тут нет кнопок. Однако вы можете сделать любой составной элемент, а не только кнопки, кликабельными, если указать  на нем модификатор clickable. Пример см. на [интерактивной](https://developer.android.com/reference/kotlin/androidx/compose/foundation/package-summary#(androidx.compose.ui.Modifier).clickable(kotlin.Boolean,kotlin.String,androidx.compose.ui.semantics.Role,kotlin.Function0)) странице документации.

Что должно произойти при нажатии на изображение? Код реализации такого поведения нетривиален, поэтому сделайте шаг назад и снова посетите знакомое приложение.

### **Посмотрите приложение Dice Roller.**

Вернитесь к коду приложения Dice Roller, чтобы увидеть, как приложение отображает различные изображения игральных костей в зависимости от значения броска:

**MainActivity.kt в приложении Dice Roller**

...  
  
@Composable  
fun DiceWithButtonAndImage(modifier: Modifier = Modifier) {  
   var result by remember { mutableStateOf(1) }  
   val imageResource = when(result) {  
       1 -> R.drawable.dice\_1  
       2 -> R.drawable.dice\_2  
       3 -> R.drawable.dice\_3  
       4 -> R.drawable.dice\_4  
       5 -> R.drawable.dice\_5  
       else -> R.drawable.dice\_6  
   }  
   Column(modifier = modifier, horizontalAlignment = Alignment.CenterHorizontally) {  
       Image(painter = painterResource(id = imageResource), contentDescription = result.toString())  
       Button(onClick = { result = (1..6).random() }) {  
          Text(stringResource(id = R.string.roll))  
       }  
   }  
}  
  
...

Ответьте на эти вопросы о коде приложения Dice Roller:

* Значение какой переменной определяет, какое изображение игральной кости будет отображаться?
* Какое действие пользователя вызывает изменение этой переменной?

Составная функция DiceWithButtonAndImage() сохраняет последний бросок кубика в переменной result, которая была определена с помощью составной функции remember и mutableStateOf() в этой строке кода:

var result by remember { mutableStateOf(1) }

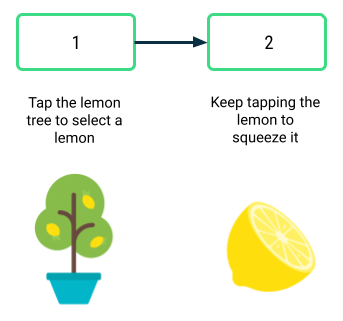
Когда переменная result обновляется до нового значения, Compose запускает рекомпозицию составного объекта DiceWithButtonAndImage(), что означает, что составной объект будет выполнен снова. Значение result запоминается при рекомпозиции, поэтому при повторном запуске компонуемого объекта DiceWithButtonAndImage() используется самое последнее значение result.  Используя оператор when значения переменной result, составной объект определяет новый идентификатор рисуемого ресурса, который нужно отобразить, и  составной объект Image отображает его.

### **Примените полученные знания в приложении Lemonade.**

Теперь ответьте на аналогичные вопросы о приложении Lemonade:

* Есть ли переменная, которую вы можете использовать, чтобы определить, какой текст и изображение должны отображаться на экране? Определите эту переменную в своем коде.
* Можете ли вы использовать условные выражения в Kotlin, чтобы приложение выполняло различное поведение в зависимости от значения этой переменной? Если да, напишите это условное выражение в своем коде.
* Какое действие пользователя вызывает изменение этой переменной? Найдите подходящее место в вашем коде, где это происходит. Добавьте туда код для обновления переменной.

**Примечание.** Возможно, вы захотите обозначить каждый этап приготовления лимонада цифрой. Например, на шаге 1 приложения есть изображение лимонного дерева, и щелчок по изображению позволяет перейти к шагу 2 приложения. Это может помочь вам организовать, какая текстовая строка соответствует тому или иному изображению.



Этот раздел может быть довольно сложным для реализации, и для правильной работы потребуется внести изменения в несколько мест вашего кода. Не расстраивайтесь, если приложение сразу не заработает так, как вы ожидаете. Помните, что существует несколько правильных способов реализовать такое поведение.

Когда вы закончите, запустите приложение и убедитесь, что оно работает. Когда вы запускаете приложение, оно должно отображать изображение лимонного дерева и соответствующую ему текстовую метку. Одно касание изображения лимонного дерева должно обновить текстовую метку и отобразить изображение лимона. Нажатие на изображение лимона пока ничего не даст.

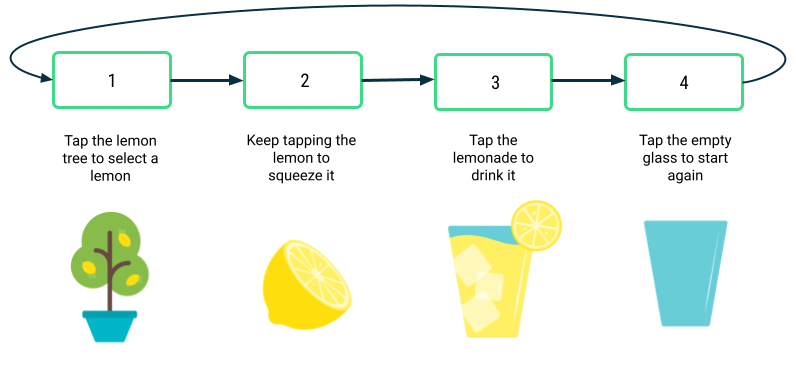
## **Добавить оставшиеся шаги**

Теперь ваше приложение может отображать два шага приготовления лимонада! На этом этапе ваш составной объект LemonApp() может выглядеть примерно так, как показано в следующем фрагменте кода. Ничего страшного, если ваш код выглядит немного иначе, но поведение в приложении такое же.

**MainActivity.kt**

...  
@Composable  
fun LemonApp() {  
*// Текущий шаг, который отображает приложение.*   **var currentStep by remember { mutableStateOf(1) }**  
*// Контейнер, использующий цвет темы*   Surface(  
       modifier = Modifier.fillMaxSize(),  
       color = MaterialTheme.colorScheme.background  
   ) {  
       when (currentStep) {  
           1 -> {  
               Column (  
                   horizontalAlignment = Alignment.CenterHorizontally,  
                   verticalArrangement = Arrangement.Center,  
                   modifier = Modifier.fillMaxSize()  
               ){  
                   Text(text = stringResource(R.string.lemon\_select))  
                   Spacer(modifier = Modifier.height(32.dp))  
                   Image(  
                       painter = painterResource(R.drawable.lemon\_tree),  
                       contentDescription = stringResource(R.string.lemon\_tree\_content\_description),  
                       modifier = Modifier  
                           .wrapContentSize()  
                           .clickable {  
                               currentStep = 2  
                           }  
                   )  
               }  
           }  
           2 -> {  
               Column (  
                   horizontalAlignment = Alignment.CenterHorizontally,  
                   verticalArrangement = Arrangement.Center,  
                   modifier = Modifier.fillMaxSize()  
               ){  
                   Text(text = stringResource(R.string.lemon\_squeeze))  
                   Spacer(modifier = Modifier.height(32  
                       .dp))  
                   Image(  
                       painter = painterResource(R.drawable.lemon\_squeeze),  
                       contentDescription = stringResource(R.string.lemon\_content\_description),  
                       modifier = Modifier.wrapContentSize()  
                   )  
               }  
           }  
       }  
   }  
}  
...

Далее вы добавите остальные шаги для приготовления лимонада. Одно нажатие на изображение должно перевести пользователя на следующий этап приготовления лимонада, где обновятся и текст, и изображение. Вам нужно будет изменить свой код, чтобы сделать его более гибким для обработки всех этапов работы приложения, а не только первых двух шагов.



Чтобы при каждом щелчке по изображению поведение было разным, вам необходимо настроить поведение, позволяющее щелкнуть мышью. Точнее, лямбда, которая выполняется при нажатии на изображение, должна знать, к какому шагу мы переходим.

Вы можете заметить, что в вашем приложении повторяется код для каждого этапа приготовления лимонада. Для  оператора when из предыдущего фрагмента кода код случая 1очень похож на код случая 2с небольшими отличиями. Если это полезно, создайте новую составную функцию, например LemonTextAndImage(), которая отображает текст над изображением в пользовательском интерфейсе. Создав новую компонуемую функцию, которая принимает некоторые входные параметры, вы получаете функцию многократного использования, которая будет полезна во многих сценариях, если вы измените входные данные, которые вы передаете. Ваша задача — выяснить, какими должны быть входные параметры. После создания этой составной функции обновите существующий код, чтобы вызывать эту новую функцию в соответствующих местах.

Еще одним преимуществом наличия отдельного компонуемого объекта LemonTextAndImage() является то, что ваш код становится более организованным и надежным. Когда вы вызываете LemonTextAndImage(), вы можете быть уверены, что и текст, и изображение будут обновлены до новых значений. В противном случае можно случайно пропустить один случай, когда обновленная текстовая метка отображается с неверным изображением.

Вот еще один совет: вы даже можете передать лямбда-функцию в компонуемый объект. Обязательно используйте обозначение типа функции, чтобы указать, какой тип функции следует передать. В следующем примере WelcomeScreen() определен составной объект, который принимает два входных параметра:  строку name и функцию onStartClicked() типа () -> Unit. Это означает, что функция не принимает никаких входных данных (пустые круглые скобки перед стрелкой) и не имеет возвращаемого значения (после стрелки Unit). Любая функция, соответствующая этому типу функции () -> Unit, может использоваться для установки обработчика onClick этого объекта Button. При нажатии кнопки вызывается функция onStartClicked().

@Composable  
fun WelcomeScreen(name: String, onStartClicked: () -> Unit) {  
    Column {  
        Text(text = "Welcome $name!")  
        Button(  
            onClick = onStartClicked  
        ) {  
            Text("Start")  
        }  
    }  
}

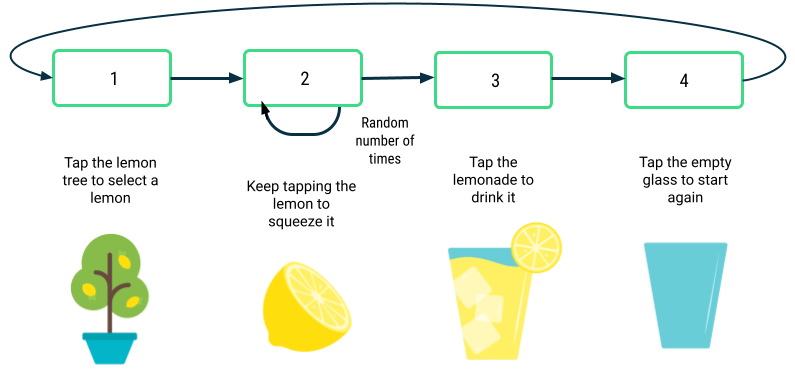
Передача лямбды в составной объект — полезный шаблон, поскольку тогда составной объект WelcomeScreen() можно будет повторно использовать в разных сценариях. Имя пользователя и  поведение кнопки onClick могут каждый раз различаться, поскольку они передаются в качестве аргументов.

Обладая этими дополнительными знаниями, вернитесь к своему коду и добавьте в свое приложение оставшиеся этапы приготовления лимонада.

Вернитесь к этим инструкциям, если вам нужны дополнительные инструкции о том, как добавить собственную логику для выжимания лимона случайное количество раз.

## **Добавить логику сжатия**

Отличная работа! Теперь у вас есть основа приложения. Нажатие на изображение должно переходить от одного шага к другому. Пришло время добавить поведение, при котором необходимо несколько раз выжать лимон, чтобы приготовить лимонад. Количество раз, которое пользователю необходимо сжать или постучать по лимону, должно быть случайным числом от 2 до 4 (включительно). Это случайное число меняется каждый раз, когда пользователь выбирает новый лимон с дерева.



Вот несколько вопросов, которые помогут вам в мыслительном процессе:

* Как вы генерируете случайные числа в Котлине?
* В каком месте вашего кода следует генерировать случайное число?
* Как гарантировать, что пользователь постучал по лимону необходимое количество раз, прежде чем перейти к следующему шагу?
* Нужны ли вам какие-либо переменные, хранящиеся в  компонуемом объекте remember, чтобы данные не сбрасывались каждый раз при перерисовке экрана?

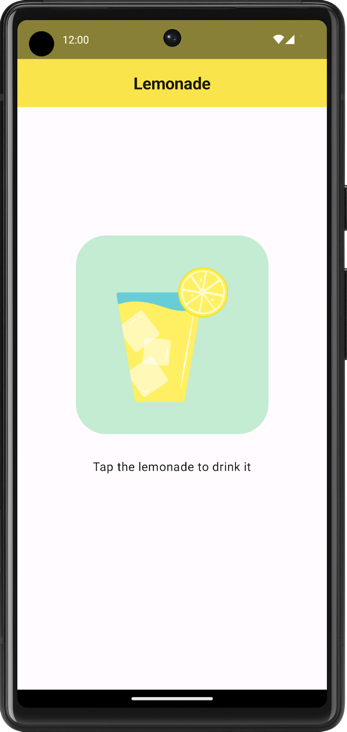
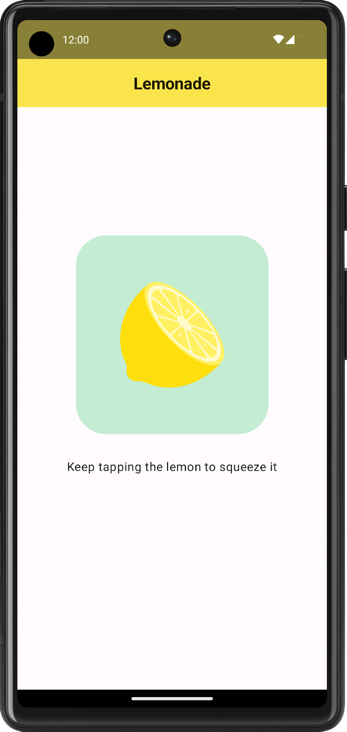
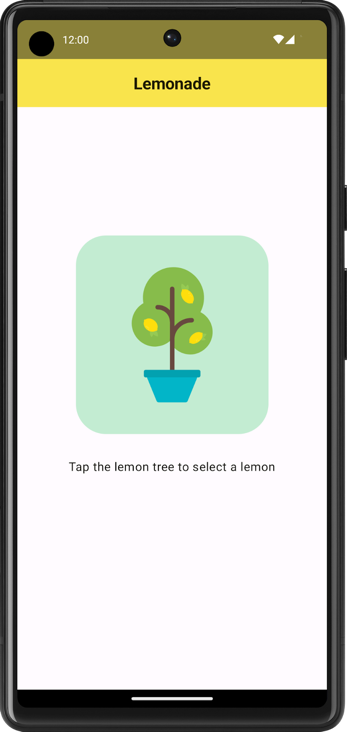
Когда вы закончите реализацию этого изменения, запустите приложение. Убедитесь, что для перехода к следующему шагу требуется несколько нажатий на изображение лимона и что количество нажатий, необходимое каждый раз, представляет собой случайное число от 2 до 4. Если при одном касании изображения лимона отображается стакан с лимонадом, вернитесь к своему коду, чтобы выяснить, чего не хватает, и повторите попытку.

Вернитесь к этим инструкциям, если вам нужны дополнительные инструкции о том, как завершить работу над приложением.

## **Завершите работу над приложением**

Вы почти закончили! Добавьте последние детали, чтобы усовершенствовать приложение.

Напоминаем, вот финальные скриншоты того, как выглядит приложение:



* Центрируйте текст и изображения на экране по вертикали и горизонтали.
* Установите размер шрифта текста 18sp.
* Добавьте пространство 16dp между текстом и изображением.
* Добавьте тонкую рамку 2dp вокруг изображений со слегка закругленными углами 4dp. Граница имеет значение цвета [RGB](https://en.wikipedia.org/wiki/RGB_color_model) : 105 красный, 205 зеленый и 216 синий. Примеры того, как добавить рамку, можно найти в Google. Или вы можете обратиться к документации по [Border](https://developer.android.com/jetpack/compose/modifiers-list" \l "Border" \t "_blank) .

После внесения этих изменений запустите приложение и сравните его с окончательными снимками экрана, чтобы убедиться, что они совпадают.

Отличная работа! Вы проделали потрясающую работу по внедрению приложения Lemonade! Это было сложное приложение, в котором нужно было разобраться со многими деталями.