МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение высшего образования  
 «Кемеровский государственный университет»**

**Институт Цифры**

**ЛАБОРАТОРНАЯ РАБОТА №4**

**ПО ДИСЦИПЛИНЕ**

**“Разработка мобильных приложений”**

студентки 3 курса

**Бердниковой Елизаветы Вадимовны**

Направление 02.03.02 – Фундаментальная информатика и информационные технологии

Преподаватель:

Киселев К.Е.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работа защищена:

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025г.

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_

Кемерово 2025

Цель работы

Изучить возможности создания прокручиваемых списков.

Ход работы:

### 1 Создание прокручиваемых списков.

Скачайте стартовый код по [ссылке](https://github.com/google-developer-training/basic-android-kotlin-compose-training-affirmations/archive/refs/heads/starter.zip) [10]. В данной части практической работы вам необходимо создать приложение «Affirmations», которое отображает список утверждений в сочетании с красивыми изображениями (рис. 5.9)

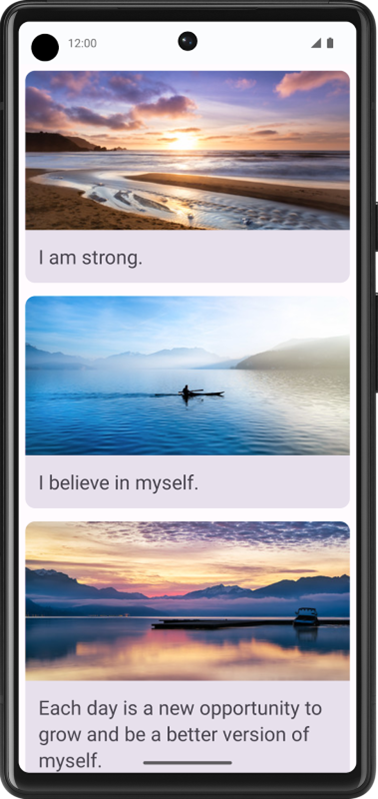
****

Рис. 5.9. Пример готового приложения [Источник: https://developer.android.com/]

В приложениях Android списки состоят из объектов какого-либо класса. Для отдельных фрагментов данных это может быть что-то простое, например строка или целое число. Для элементов списка, которые имеют несколько фрагментов данных, например изображение и текст, вам понадобится класс, содержащий все эти свойства. Data Class – это тип классов, которые содержат только свойства, они могут предоставлять некоторые служебные методы для работы с этими свойствами.

Создайте новый пакет, нажмите правой кнопкой мыши на com.example.affirmations и выберите New > package. Назовите новый пакет “model”. Пакет будет содержать модель данных, которая будет представлена ​​классом данных. Класс данных будет состоять из свойств, которые представляют информацию, относящуюся к тому, что будет являться «Affirmation», которое будет состоять из строкового ресурса и ресурса изображения. Пакеты – это каталоги, которые содержат классы и даже другие каталоги.

Создайте новый класс в пакете com.example.affirmations.model. Назовите новый класс Affirmation и сделайте его Data Class’ом. Каждый Affirmation состоит из одного изображения и одной строки. Создайте два свойства val в классе данных Affirmation. Одно должно называться stringResourceId, а другое imageResourceId. Они оба должны быть целыми числами (Int).

Добавьте полю stringResourceId аннотацию @StringRes. Полю imageResourceId добавьте аннотацию @DrawableRes. Поле stringResourceId Представляет идентификатор для текста, сохраненного в строковом ресурсе. Поле imageResourceId представляет идентификатор для изображения, сохраненного в ресурсе drawable (лист. 5.7).

Листинг 5.7 – Определение data-класса с аннотациями ресурсов строки и изображения

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

В пакете com.example.affirmations.data откройте файл Datasource.kt и раскомментируйте два оператора импорта и содержимое класса Datasource (лист. 5.8).

Листинг 5.8 – Загрузка списка утверждений с привязкой к строковым и графическим ресурсам

|  |
| --- |
| 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))  }  } |

Каждый элемент Affirmation состоит из изображения и строки. Данные для каждого из этих элементов поставляются с начальным кодом, и необходимо создать элемент пользовательского интерфейса для отображения такого элемента.

Элемент будет состоять из Card, содержащего Image и Text. В Compose, a Card – это поверхность, которая отображает содержимое и действия в одном контейнере.

Откройте файл MainActivity.kt. Создайте новый метод под методом AffirmationsApp() с именем AffirmationCard() и добавьте аннотацию @Composable. Добавьте объект Affirmation в качестве параметра. Добавьте параметр modifier и установите для параметра значение по умолчанию Modifier (лист. 5.9).

Листинг 5.9 – Заготовка компонента карточки для отображения утверждения

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

***Примечание:*** *рекомендуется передавать модификатор каждому composable элементу и устанавливать Modifier в качестве значения по умолчанию.*

Внутри метода AffirmationCard вызовите функцию Card. Передайте параметр modifier. Добавьте элемент Column внутрь Card. Элементы внутри Column располагаются вертикально в пользовательском интерфейсе. Это позволяет размещать изображение над связанным текстом. И наоборот, Row элемент располагает содержащиеся в нем элементы горизонтально.

Добавьте Image внутрь тела лямбда выражения Column. Напомним, что Image всегда требует ресурс для отображения и contentDescription. Ресурс изображения Affirmation должен быть передан внутрь метода painterResource, который должен быть передан в параметр painter. Метод painterResource загрузит либо векторные изображения, либо растровые, такие как PNG. Также передайте параметру contentDescription строковый ресурс из объекта Affirmation.

В дополнение к параметрам painter и contentDescription передайте a modifier и contentScale. Параметр contentScale определяет, как изображение должно масштабироваться и отображаться. Объект modifier должен иметь установленный атрибут fillMaxWidth и высоту 194.dp. В качестве параметра для contentScale передайте ContentScale.Crop (лист. 5.10).

Листинг 5.10 – Компонент AffirmationCard с изображением и описанием внутри Card

|  |
| --- |
| 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          )      }  }  } |

Внутри Column создайте компонент Text после компонента Image. Передайте в параметр text строковый ресурс affirmation.stringResourceId, передайте объект Modifier с атрибутом padding, установленным на 16.dp, и задайте тему текста, передав в параметр style материал (лист. 5.11)

Листинг 5.11 – Карточка с изображением и текстом, стилизованным через MaterialTheme.typography.headlineSmall

|  |
| --- |
| MaterialTheme.typography.headlineSmall.  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          )      }  }  } |

Для того, чтобы посмотреть как выглядит карточка, необходимо создать предварительный просмотр

Создайте private метод с именем AffirmationCardPreview(). Добавьте аннотации @Preview и @Composable (лист. 5.12).

Листинг 5.12 – Заготовка функции превью для компонента AffirmationCard

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

Внутри метода вызовите AffirmationCard и передайте ему объект Affirmation со строковым ресурсом R.string.affirmation1 и drawable ресурсом R.drawable.image1 (лист. 5.13).

Листинг 5.13 – Превью компонента AffirmationCard с примером данных

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

В правом углу верхнем углу найдите и откройте вкладку «Разделение» , и вы увидите предварительный просмотр AffirmationCard. При необходимости нажмите «Build & Refresh», чтобы отобразить предварительный просмотр (рис. 5.10).

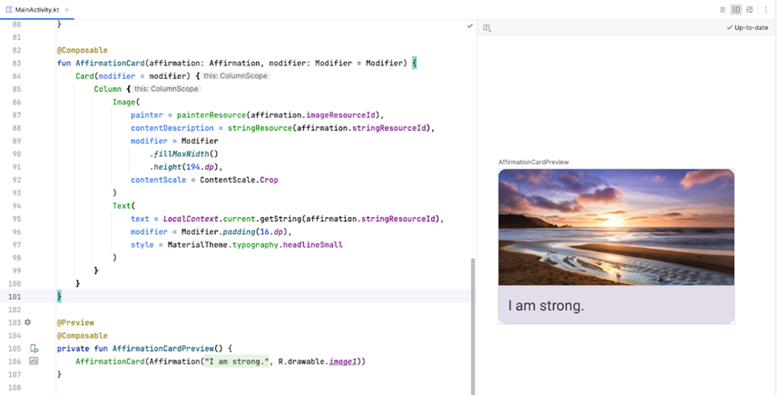
****

Рис. 5.10. Предпросмотр карточки  [Источник: https://developer.android.com/]

Теперь создадим прокручиваемый список. Для этого создайте функцию с именем AffirmationList(), добавьте аннотацию @Composable и объявите набор объектов List<Affirmation> в качестве параметра метода.

Объявите объект modifier как параметр метода со значением по умолчанию Modifier.

Добавьте в метод компонент LazyColumn и передайте в параметр modifier. В теле лямбда-выражения LazyColumn вызовите метод items() и передайте в него affirmationList.

Откройте лямбда выражение в методе items(), именуйте параметр лямбда-выражения и вызовите метод AffirmationCard. Внутрь AffirmationCard передайте в  модификатор Modifier.padding(8.dp), и в параметр affirmation именованый параметр лямбда-выражения

Листинг 5.14 – Отображение списка Affirmation с помощью LazyColumn и карточек AffirmationCard

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

В AffirmationsApp получите текущие направления макета и сохраните их в переменной. Они будут использоваться для настройки отступов позже (лист. 5.15).

Листинг 5.15 – Заготовка компонента приложения с доступом к текущему направлению компоновки

|  |
| --- |
| import com.example.affirmations.data.Datasource   @Composable  fun AffirmationsApp() {  val layoutDirection = LocalLayoutDirection.current  } |

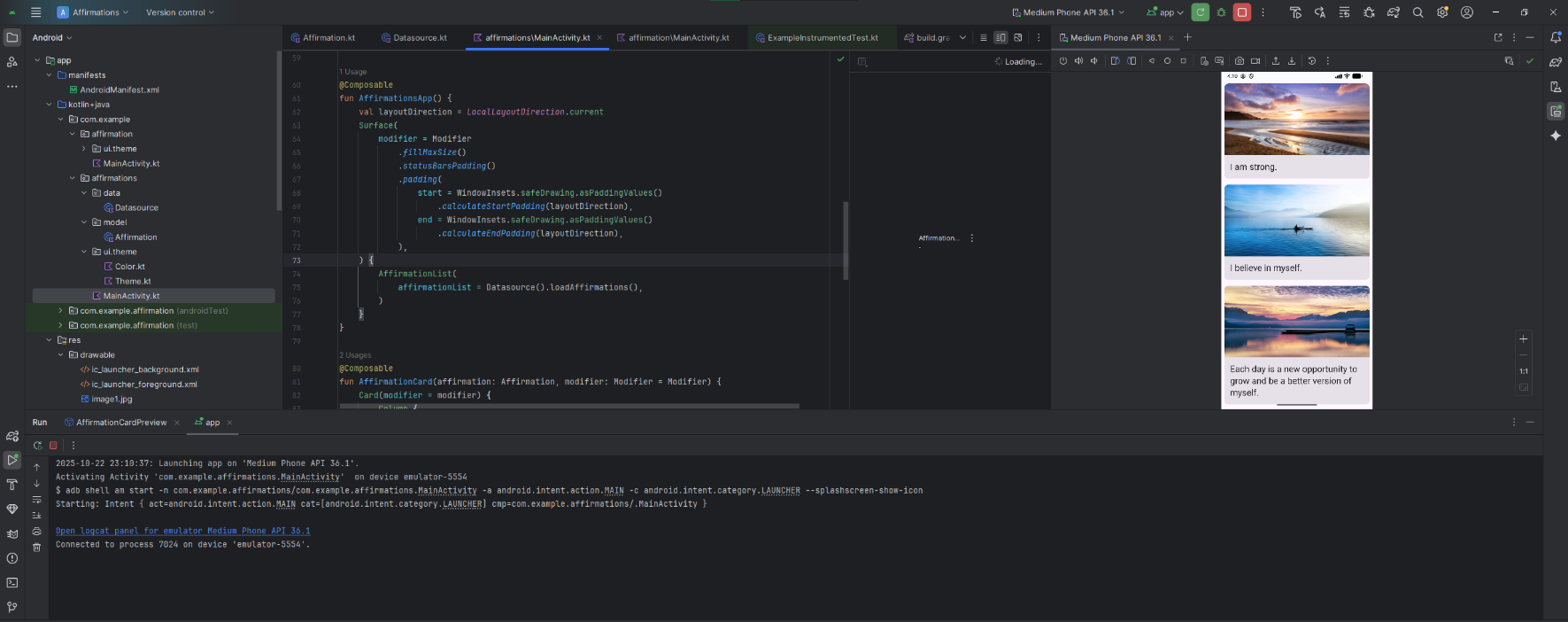
Теперь создайте компонент Surface. Этот элемент установит отступ для AffirmationsList.

Передайте Modifier в Surface, который заполняет максимальную ширину и высоту своего родителя, устанавливает отступ и устанавливает начальный и конечный отступы для layoutDirection. Вот пример того, как преобразовать объект LayoutDirection в отступ: WindowInsets.safeDrawing.asPaddingValues().calculateStartPadding(layoutDirection).

В лямбде для Surface вызовите AffirmationList и передайте параметру affirmationList значение DataSource().loadAffirmations() (лист. 5.16).

Листинг 5.16 – Основной компонент приложения с безопасными отступами и отображением списка утверждений

|  |
| --- |
| import com.example.affirmations.data.Datasource   @Composable  fun AffirmationsApp() {  val layoutDirection = LocalLayoutDirection.current  Surface(      Modifier = Modifier      .fillMaxSize()      .statusBarsPadding()      .padding(          start = WindowInsets.safeDrawing.asPaddingValues()                      .calculateStartPadding(layoutDirection),          end = WindowInsets.safeDrawing.asPaddingValues()                      .calculateEndPadding(layoutDirection),      ),  ) {      AffirmationsList(          affirmationList = Datasource().loadAffirmations(),          )  }  } |

****Запустите приложение Affirmations на устройстве или эмуляторе и посмотрите на конечный результат.

**2 Контрольное задание для самопроверки.**

1)              Создайте новый проект с шаблоном Empty Activity и минимальным SDK 24.

Создайте класс для хранения данных и карточку для отображения топиков с различными темами (рис. 5.14).

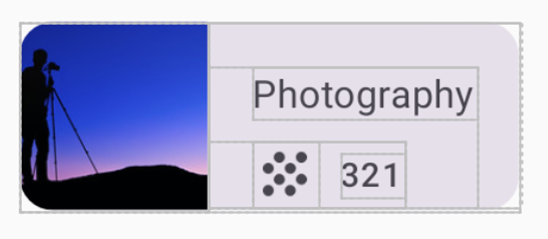
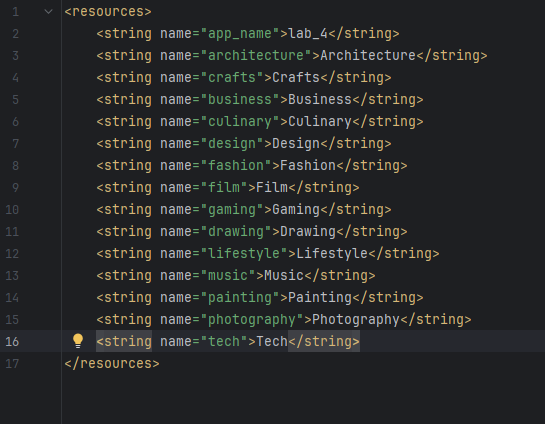
****

Рис. 5.14. Ожидаемый результат карточки темы  [Источник: https://developer.android.com/]

 Каждая тема содержит три части информации, создайте класс для хранения этих данных. Скопируйте следующие элементы в app/src/main/res/values/strings.xml (лист. 5.33)

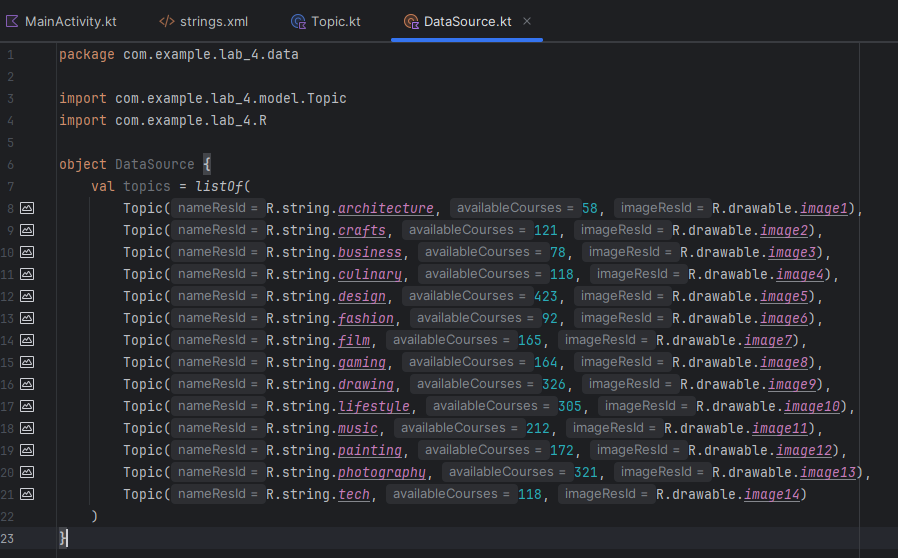
Листинг 5.33 – Ресурсы строк с названиями категорий творчества и хобби

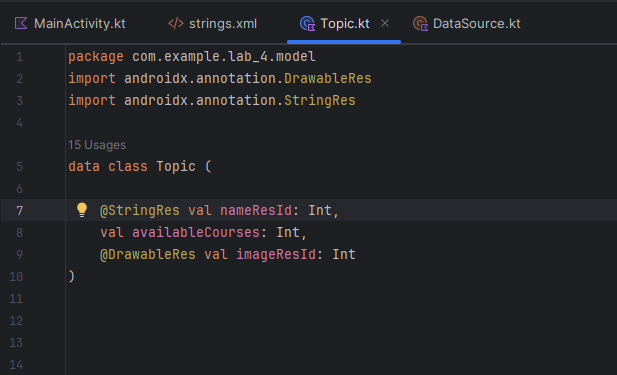
|  |
| --- |
| <string name="architecture">Architecture</string>  <string name="crafts">Crafts</string>  <string name="business">Business</string>  <string name="culinary">Culinary</string>  <string name="design">Design</string>  <string name="fashion">Fashion</string>  <string name="film">Film</string>  <string name="gaming">Gaming</string>  <string name="drawing">Drawing</string>  <string name="lifestyle">Lifestyle</string>  <string name="music">Music</string>  <string name="painting">Painting</string>  <string name="photography">Photography</string>  <string name="tech">Tech</string> |

  
 Самостоятельно найдите картинки для каждой темы. Создайте пустой файл с именем DataSource.kt. Скопируйте в файл следующий код (лист. 5.34):

Листинг 5.34 – Объект DataSource с данными для списка тем (Topic)

|  |
| --- |
| object DataSource {  val topics = listOf(      Topic(R.string.architecture, 58, R.drawable.architecture),      Topic(R.string.crafts, 121, R.drawable.crafts),      Topic(R.string.business, 78, R.drawable.business),      Topic(R.string.culinary, 118, R.drawable.culinary),      Topic(R.string.design, 423, R.drawable.design),      Topic(R.string.fashion, 92, R.drawable.fashion),      Topic(R.string.film, 165, R.drawable.film),      Topic(R.string.gaming, 164, R.drawable.gaming),      Topic(R.string.drawing, 326, R.drawable.drawing),      Topic(R.string.lifestyle, 305, R.drawable.lifestyle),      Topic(R.string.music, 212, R.drawable.music),      Topic(R.string.painting, 172, R.drawable.painting),      Topic(R.string.photography, 321, R.drawable.photography),      Topic(R.string.tech, 118, R.drawable.tech)  )  } |





Используйте сетку отступов как показано на рисунке 5.15.

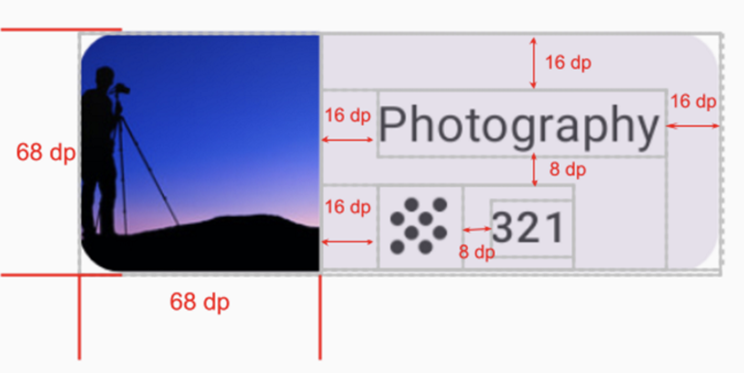
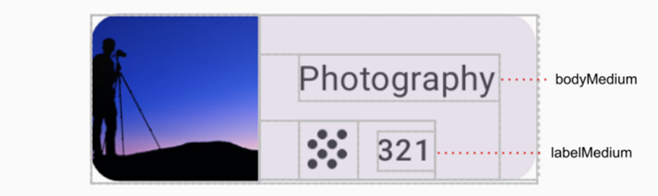
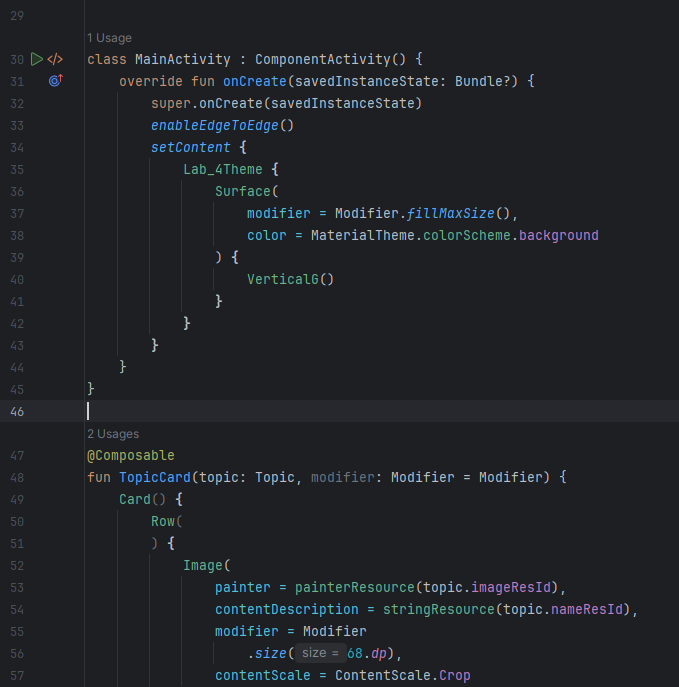
****

Рис. 5.15. Отступы для карточки темы  [Источник: https://developer.android.com/]

Используйте размеры шрифтов как указано на рисунке 5.16

****



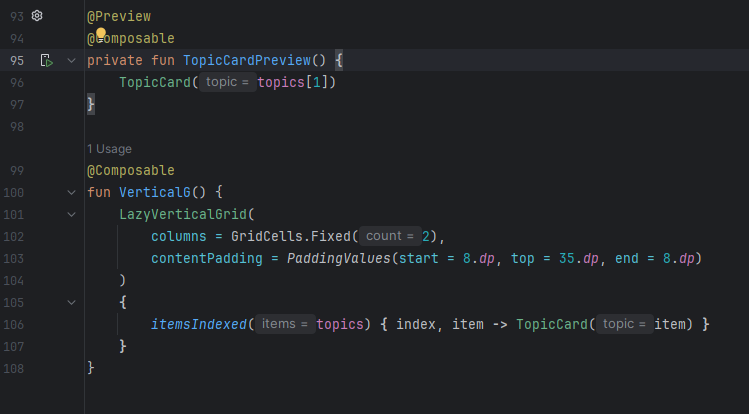
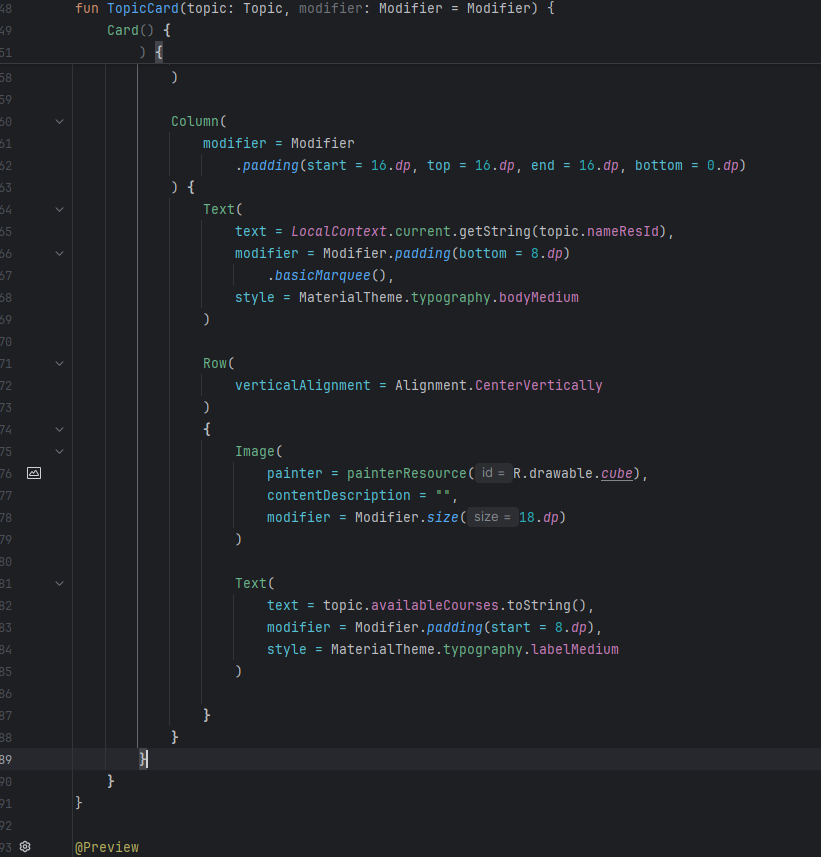


Рис. 5.16. Размеры шрифтов для карточки темы  [Источник: https://developer.android.com/]

Отобразите все карточки в виде сетки с двумя столбцами. Установите отступы между краями экрана и объектами равными 8dp

Ожидаемый результат (рис. 5.17):

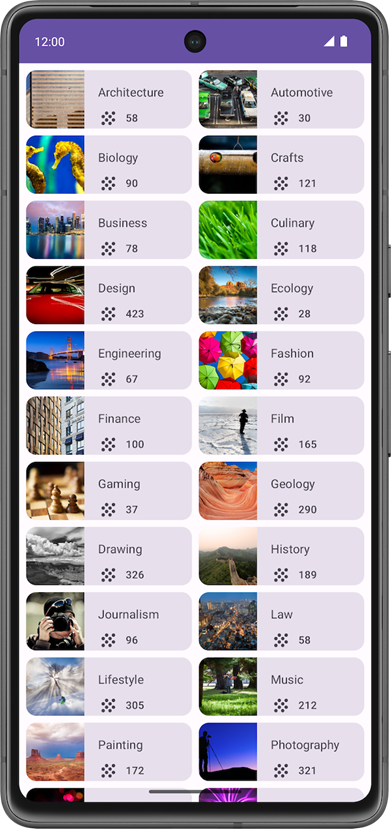
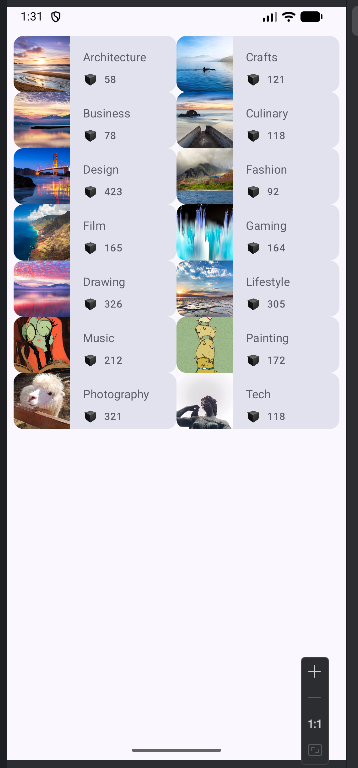
****

Рис. 5.17. Ожидаемый результат задания 1  [Источник: <https://developer.android.com/>]

В итоге:



ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы я познакомилась:  
c созданием прокручиваемых списков.