1. Добавляем новую Активити – Empty Views Activity – название «ItemsActivity».

2. Заходим на AuthActivity и делаем так, чтобы в момент успешной авторизации мы переходили на другую активити.

В момент когда верно авторизуемся создаем объект на основе класса Intent и указываем что с текущей активити переходим на ItemsActivity:

```
val intent = Intent(this, ItemsActivity::class.java)
startActivity(intent)
```

Получается следующее:

```
if(isAuth){
    Toast.makeText(this, "Пользователь $login авторизован",
Toast.LENGTH_LONG).show()
    userLogin.text.clear()
    userPass.text.clear()

    val intent = Intent(this, ItemsActivity::class.java)
    startActivity(intent)

} else
```

3. Переходим в layout – activity_items

3.1. Выбираем слой LinearLayout (vertical)

Укажем цвет заднего фона такой же как на Активити activity_auth

```
android:background="#e2a600"
```

Копируем с Активити activity_auth TextView с отступом сверху в 40 пикселей и надписью «Товары»

Добавляем в слой новый объект RecyclerView

К нему добавляем id

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

Укажем отступы по бокам 20 пикселей и отступы сверху и снизу по 30 пикселей

```
android:layout_marginHorizontal="20dp"
android:layout_marginVertical="30dp"
```

4. Создаем новый класс на основе которого будем описывать каждый конкретный элемент в нашем проекте – класс «Item». Этот класс будет описывать каждый товар в нашем проекте.

Каждый элемент будет состоять из:

– id;

– картинки image;

– названия title;

– краткого описания desc;

– длинного описания text;

– стоимости price.

```
package com.example.lb1

class Item(val id: Int, val image: String, val title: String, val desc: String, val text: String, val
price : Int) {
}
```

5. Переходим в ItemsActivity и создаем переменную ItemsList основанную на классе RecyclerView, укажем объект из Design itemsList

```
val itemsList: RecyclerView = findViewById(R.id.itemsList)
```

Также создадим здесь список items основанный на arrayListOf и каждый элемент это объект на основе нашего созданного класса Item

```
val items = arrayListOf<Item>()
```

Ниже пишем обращение к items, функции add и указываем что в качестве нового элемента будем добавлять элементы на основе класса Item.

При добавлении элемента мы должны указать:

– id;

– название изображение;

– название товара;

– краткое описание;

– полное описание;

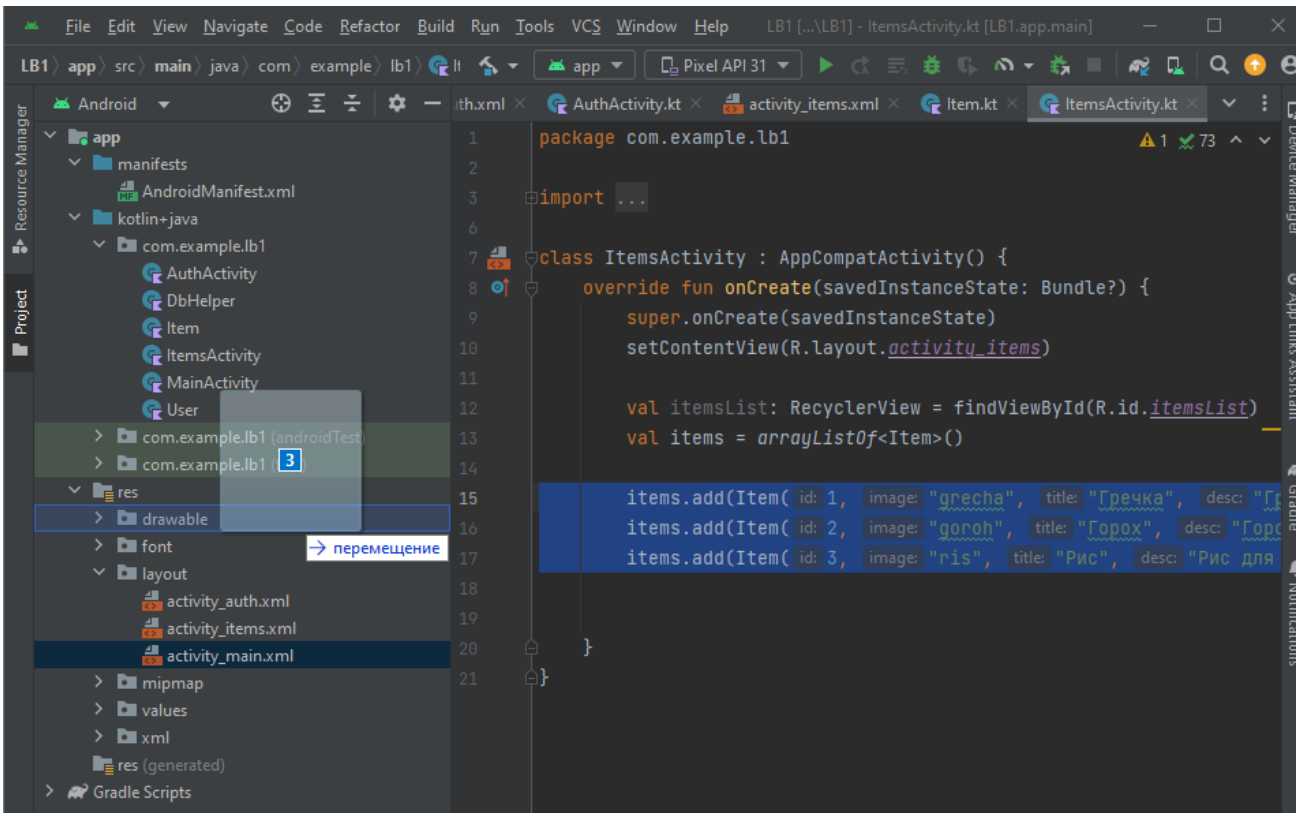
– стоимость.

```
items.add(Item(1, "grecha", "Гречка", "Гречка это XXXX", "Гречка распространена по всему миру, и считается древней культурой. Ее родиной является Индия и Непал, где гречку начали специально выращивать 4 тысячи лет назад.", 100))
items.add(Item(2, "goroh", "Горох", "Горох для XXXX", "Горох шлифованный — это единственный вид крупы, вырабатываемый из семян бобовых. В зависимости от способа обработки крупу из гороха делят на виды: горох целый шлифованный; горох колотый шлифованный", 90))
items.add(Item(3, "ris", "Рис", "Рис для приготовления XXXX", "Рис – один из основных продуктов питания более половины населения земного шара. Эту древнейшую сельскохозяйственную культуру в настоящее время выращивают в 118 странах мира, однако наибольшие площади сосредоточены в странах Юго-Восточной Азии и Дальнего Востока", 50))
```

В итоге:

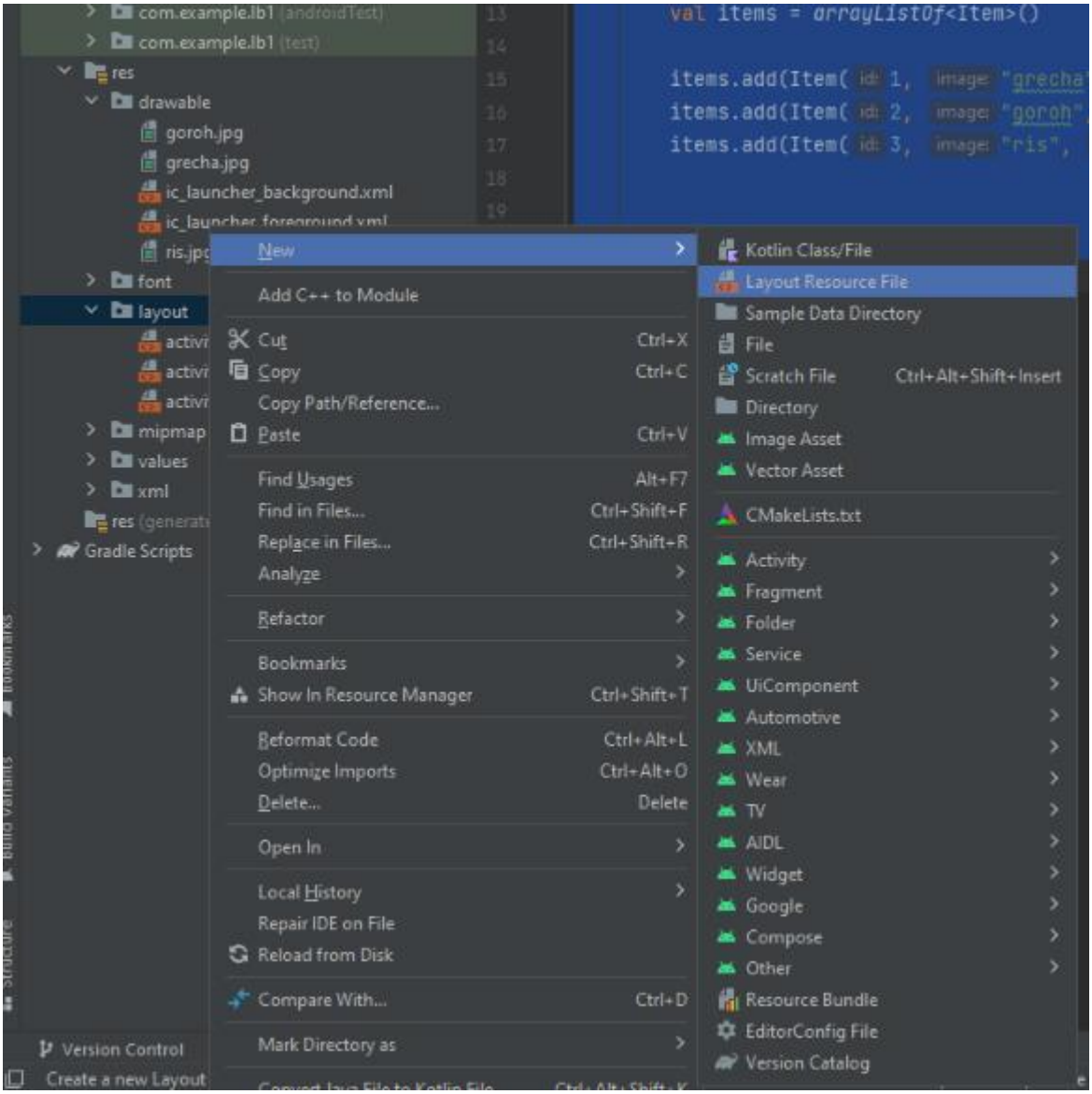
```
package com.example.lb1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.recyclerview.widget.RecyclerView

class ItemsActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_items)

        val itemsList: RecyclerView = findViewById(R.id.itemsList)
        val items = arrayListOf<Item>()

        items.add(Item(1, "grecha", "Гречка", "Гречка это XXXX", "Гречка распространена по всему миру, и считается древней культурой. Ее родиной является Индия и Непал, где гречку начали специально выращивать 4 тысячи лет назад.", 100))
        items.add(Item(2, "goroh", "Горох", "Горох для XXXX", "Горох шлифованный — это единственный вид крупы, вырабатываемый из семян бобовых. В зависимости от способа обработки крупу из гороха делят на виды: горох целый шлифованный; горох колотый шлифованный", 90))
        items.add(Item(3, "ris", "Рис", "Рис для приготовления XXXX", "Рис – один из основных продуктов питания более половины населения земного шара. Эту древнейшую сельскохозяйственную культуру в настоящее время выращивают в 118 странах мира, однако наибольшие площади сосредоточены в странах Юго-Восточной Азии и Дальнего Востока", 50))


    }
}
```

Подготовим три картинки с названиями "grecha", "goroh", "ris" и переносом все три добавляем в папку drawable

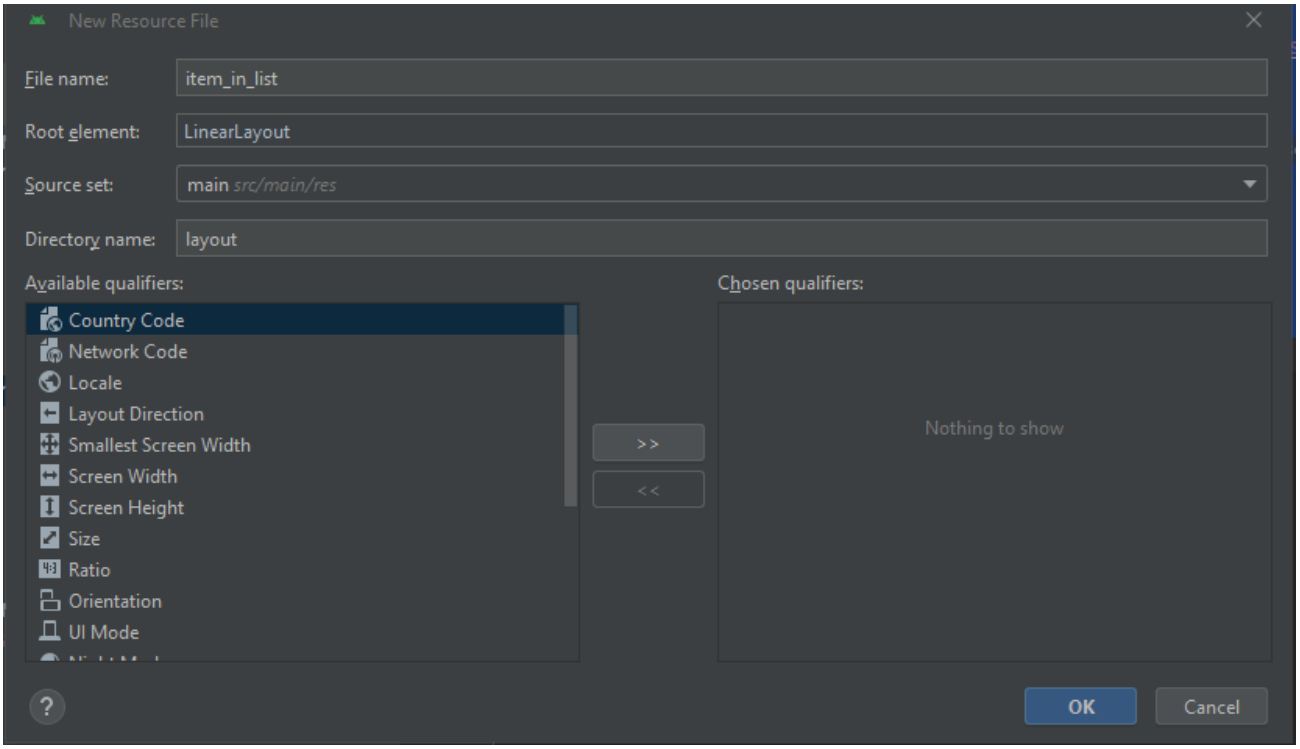


Чуть позже в коде сделаем «привязку» названия картинки к картинке.

6. Добавляем в папку layout новый xml-файл в котором опишем дизайн для каждого элемента в нашем списке. Выбираем Layout Resource File.



Указываем название «item_in_list» и сразу указываем основной слой LinearLayout



Добавляем в слои:

– ImageView (**не стандартную**);

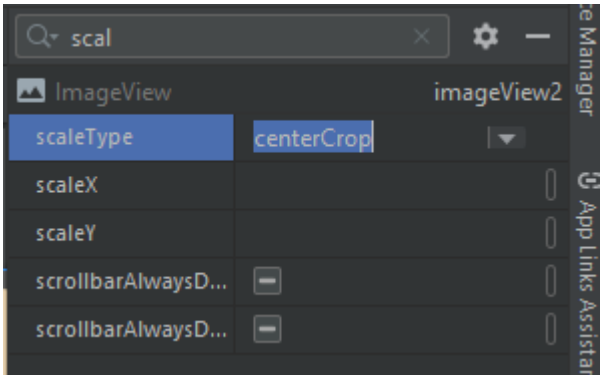
– TextView (для названия);

– TextView (для описания);

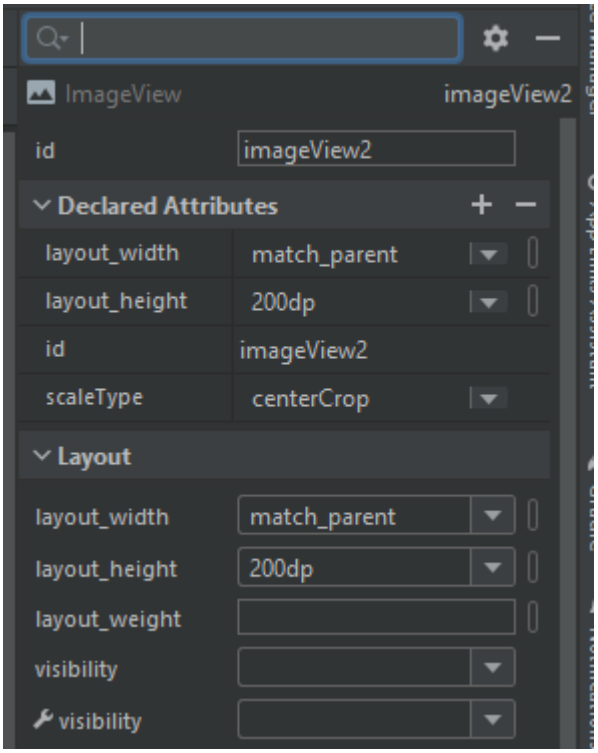
– TextView (для стоимости);

– Button (для перехода на определенный товар).

Для ImageView поставим через поиск в Design свойство centerCrop



Это необходимо чтобы указать высоту картинки, например в 200 пикселей



Перейдем в Split укажем задний фон (чуть темнее предыдущей активити):

```
android:background="#815F02"
```

Укажем в layout_height «wrap_content», чтобы фон занимал столько сколько необходимо, а не весь экран

```
android:layout_height="wrap_content">
```

Укажем отступ снизу в 20 пикселей

```
android:paddingBottom="20dp"
```

7. Поработаем с кодом

7.1 Укажем новый id для TextView

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

7.2 Укажем для новый id ImageView

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

7.3 Укажем новый id для TextView (второго)

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

7.4 Укажем новый id для TextView (третьего)

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

7.5 Укажем новый id для Button

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

7.6 Укажем для каждой текстовой надписи отступы по 20 пикселей:

```
android:paddingHorizontal="20dp"
```

И вертикальные отступы:

```
android:paddingVertical="10dp"
```

7.7 Укажем ширину кнопки 200 пикселей и ее расположение по центру

```
android:layout_width="200dp"
android:layout_gravity="center"
```

На кнопке напишем «Посмотреть»:

```
android:text="Посмотреть"
```

Укажем задний фон у кнопки:

```
android:backgroundTint="#CA9811"
```

7.8 В последнем TextView укажем выравнивание цены по правую сторону:

```
android:gravity="end"
```

7.9 В первом TextView укажем текст «жирный»

```
android:textStyle="bold"
```

и высоту текста 30 пикселей:

```
android:textSize="30sp"
```

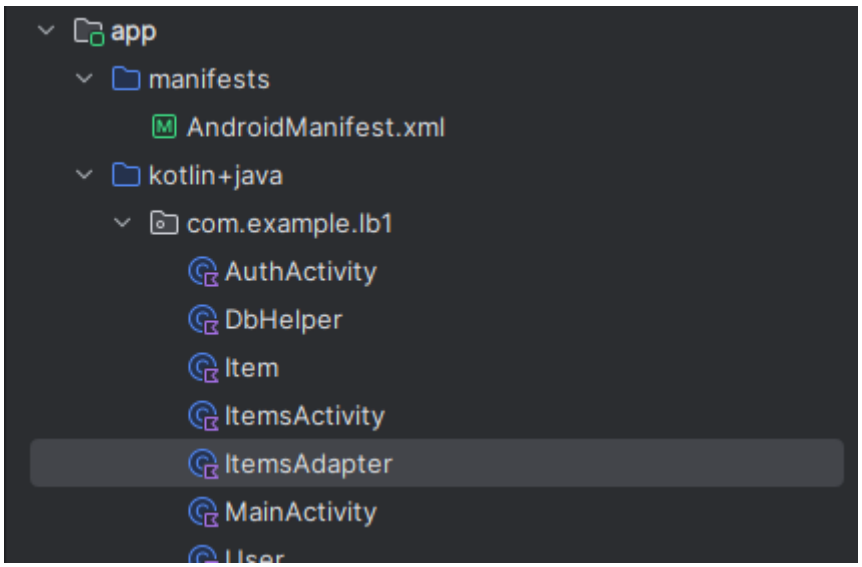
7.10 Во втором TextView размер текста 17 пикселей

```
android:textSize="17sp"
```

7.11 В третьем TextView (цена) сделаем текст «жирный» и размер 20 пикселей

```
android:textStyle="bold"
android:textSize="20sp"
```

8. Создаем класс ItemsAdapter



Класс будет принимать несколько пареметров:

– items основанный на списке, где каждый объект будет на основе класса Item, т.е. будем принимать весь массив который мы создали до этого (гречка, рис, горох);

– контекст в котором мы работаем, т.е. переменную context на основе класса Context;

```
class ItemsAdapter (var items: List<Item>, var context: Context){
```

Внутри класса создаем вложенный класс MyViewHolder в котором принимаем один параметр на основе класса View, также будем наследовать от встроенного класса RecyclerView. ViewHolder (и передаем view, с которым работаем).

```
class MyViewHolder(view: View): RecyclerView.ViewHolder(view){
```

Это вложенный класс за счет которого мы сейчас сможем найти те элементы с которыми мы хотим работать внутри нашего дизайна.

Перед этим укажем, что ItemsAdapter будет наследовать от RecyclerView.Adapter. В качестве типа данных для класса Adapter укажем ItemsAdapter из которого берем MyViewHolder:

```
class ItemsAdapter (var items: List<Item>, var context: Context) :
RecyclerView.Adapter<ItemsAdapter.MyViewHolder>(){
```

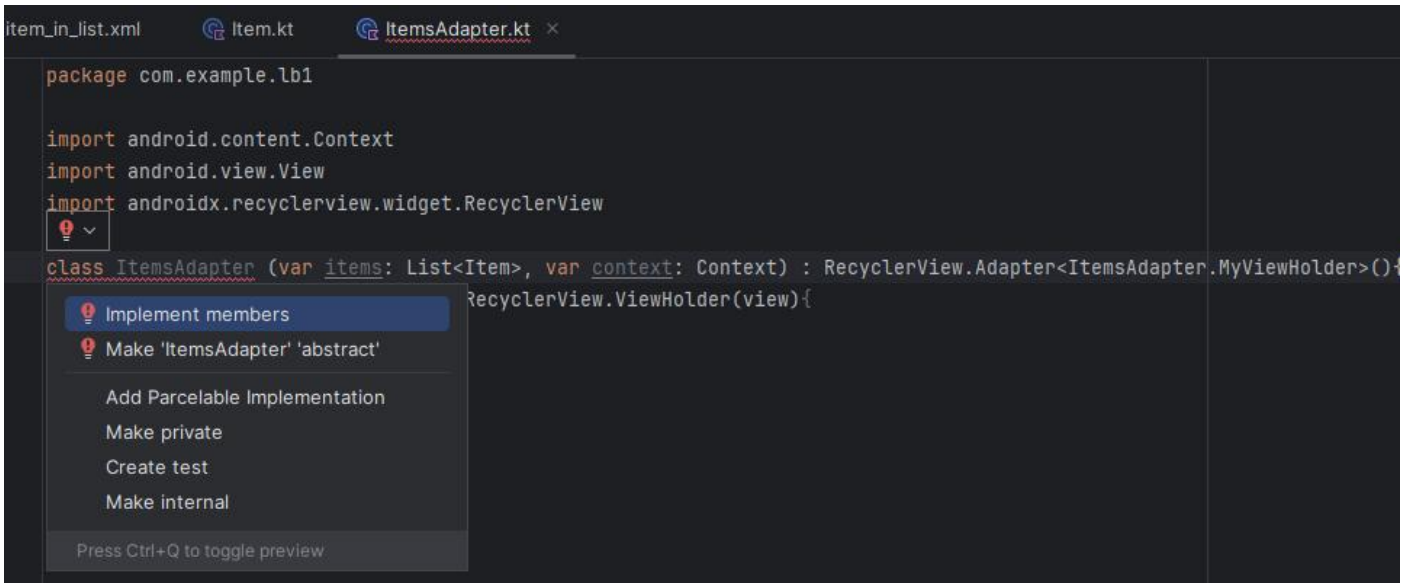
В итоге:

```
package com.example.lb1

import android.content.Context
import android.view.View
import androidx.recyclerview.widget.RecyclerView

class ItemsAdapter (var items: List<Item>, var context: Context) :
RecyclerView.Adapter<ItemsAdapter.MyViewHolder>(){
    class MyViewHolder(view: View): RecyclerView.ViewHolder(view){
    }
}
```

Сейчас создали свой собственный класс Adapter. В этом классе будем принимать массив всех наших элементов, а также контекст в котором мы будем работать – передавая ключевое слово this.

Здесь для того, чтобы все корректно работало мы должны всегда наследовать все от главного класса Adapter. И также в этот главный класс Adapter мы передаем свой собственный класс MyViewHolder, в котором будем находить все элементы из дизайна.

Уберем ошибку через Implement members:



Будут добавлены методы:

```
package com.example.lb1
```

```
import android.content.Context
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView

class ItemsAdapter (var items: List<Item>, var context: Context) :
RecyclerView.Adapter<ItemsAdapter.MyViewHolder>(){
    class MyViewHolder(view: View): RecyclerView.ViewHolder(view){
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
        TODO("Not yet implemented")
    }

    override fun getItemCount(): Int {
        TODO("Not yet implemented")
    }

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        TODO("Not yet implemented")
    }
}
```

Создадим в классе MyViewHolder переменные с какими объектами из Дизайна мы будем работать:

Создаем переменную image, указываем что она будет идти на основе класса ImageView. Указываем что обращаемся к дизайну view.findViewByID. Из этого дизайна находим определенный объект у которого id будет item_list_image:

```
val image: ImageView = view.findViewById(R.id.item_list_image)
```

Скопируем эту строку и создадим еще для 3-ёх переменных, чтобы находить:

– title;

– desc;

– price.

Скорректируем код:

```
class ItemsAdapter (var items: List<Item>, var context: Context) :
RecyclerView.Adapter<ItemsAdapter.MyViewHolder>(){
    class MyViewHolder(view: View): RecyclerView.ViewHolder(view){
        val image: ImageView = view.findViewById(R.id.item_list_image)
        val title: TextView = view.findViewById(R.id.item_list_title)
```

```
        val desc: TextView = view.findViewById(R.id.item_list_desc)
        val price: TextView = view.findViewById(R.id.item_list_price)

    }
```

В методе onCreateViewHolder укажем какой конкретно дизайн будем обрабатывать

Создаем здесь переменную view, далее обращаемся к LayoutInflater, указываем функцию from и сообщаем что работаем с тем контекстом, который сюда передается. Вызываем функцию inflate и указываем с каким дизайном мы работаем. Также передаем parent и false.

Указываем что вызываем вложенный класс MyViewHolder и в него передаем один параметр – дизайн с которым будем работать:

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
    val view = LayoutInflater.from(parent.context).inflate(R.layout.item_in_list, parent, false)
    return MyViewHolder(view)
}
```

Мы сейчас прописали встроенный метод onCreateViewHolder в нем мы находим нужный дизайн и дальше вызываем свой собственный класс в который передаем дизайн. И дальше из этого дизайна мы уже находим определенные объекты и далее с этими объектами будем взаимодействовать.

Реализуем метод getItemCount, который должен возвращать нам подсчет количества элементов, то есть будем обращаться к items, к тому массиву который сюда передается и будем возвращать количество элементов в этом массиве:

```
override fun getItemCount(): Int {
    return items.count()
}
```

Метод onBindViewHolder.

За счет него мы объединяем все наши усилия и укажем какие данные мы будем подставлять в каждое из полей которое здесь были у нас найдены.

Чтобы обратиться к тем данным которые у нас уже были найдены мы обращаемся к параметру holder, он идет на основе класса MyViewHolder, т.е. через этот класс мы находим все элементы и дальше через «точку» обращаемся к любому полю и пишем в качестве текстовой надписи то что находится в массиве items:

```
holder.title.text = items[position].title
```

Аналогично пишем для desc и price.

У price формат Int, поэтому значение переводим к формату строки String

```
override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
    holder.title.text = items[position].title
    holder.desc.text = items[position].desc
    holder.price.text = items[position].price.toString()
}
```

9. Переходим в ItemsActivity

Также указываем layoutManager для указания в каком формате у нас все элементы распологаются

```
itemsList.layoutManager = LinearLayoutManager(this)
```

Обращаемся к itemsList и в качестве свойства обращаемся к adapter и дальше в качестве adaptor подставляем собственный adaptor

```
itemsList.adapter = ItemsAdapter(items, this)
```

В итоге:

```
package com.example.lb1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class ItemsActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_items)

        val itemsList: RecyclerView = findViewById(R.id.itemsList)
        val items = arrayListOf<Item>()

        items.add(Item(1, "grecha", "Гречка", "Гречка это XXXX", "Гречка распространена по всему миру, и считается древней культурой. Ее родиной является Индия и Непал, где гречку начали специально выращивать 4 тысячи лет назад.", 100))
        items.add(Item(2, "goroh", "Горох", "Горох для XXXX", "Горох шлифованный —
```

```
это единственный вид крупы, вырабатываемый из семян бобовых. В зависимости от способа обработки крупу из гороха делят на виды: горох целый шлифованный; горох колотый шлифованный", 90))
        items.add(Item(3, "ris", "Рис", "Рис для приготовления XXXX", "Рис – один из основных продуктов питания более половины населения земного шара. Эту древнейшую сельскохозяйственную культуру в настоящее время выращивают в 118 странах мира, однако наибольшие площади сосредоточены в странах Юго-Восточной Азии и Дальнего Востока", 50))

        itemsList.layoutManager = LinearLayoutManager(this)
        itemsList.adapter = ItemsAdapter(items, this)
    }
}
```

10. Переходим в item_in_list

10.1 Создаем отступы:

```
android:layout_marginBottom="20dp"
```

Создаем в этот отступ в слое <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android":

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:background="#815F02"
    android:paddingBottom="20dp"
    android:layout_marginBottom="20dp"
    android:layout_height="wrap_content">
```

11. Переходим в ItemsAdapter и в методе onBindViewHolder обращаемся к holder/ J,hfoftvcz r aeyrwbb setImageResource и в качестве значения в эту функцию подставляем id нужной картинки. Но id картинки мы не знаем, поэтому до этой функции

```
holder.image.setImageResource()
```

пишем код за счет которго будем узнавать id изображения по ее названию из папке «drawable»

```
val imageId = context.resources.getIdentifier(
    items[position].image,
    "drawable",
    context.packageName


)
```

В итоге код ItemsAdapter:

```
package com.example.lb1

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

class ItemsAdapter (var items: List<Item>, var context: Context) :
RecyclerView.Adapter<ItemsAdapter.MyViewHolder>(){
    class MyViewHolder(view: View): RecyclerView.ViewHolder(view){
        val image: ImageView = view.findViewById(R.id.item_list_image)
        val title: TextView = view.findViewById(R.id.item_list_title)
        val desc: TextView = view.findViewById(R.id.item_list_desc)
        val price: TextView = view.findViewById(R.id.item_list_price)

    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_in_list, parent,
false)
        return MyViewHolder(view)
    }

    override fun getItemCount(): Int {
        return items.count()
    }

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        holder.title.text = items[position].title
        holder.desc.text = items[position].desc
        holder.price.text = items[position].price.toString() + "$"


        val imageId = context.resources.getIdentifier(
```

```
            items[position].image,
            "drawable",
            context.packageName


        )

        holder.image.setImageResource()

    }
}
```

Возможны ошибки при запсуке эмулятора из-за картинок. Целесообразно скачать и заменить в случае ошибок.