Android представляет широкую палитру элементов,которые представляют списки. Все они является наследниками класса **android.widget.AdapterView**. Это такие виджеты как ListView, GridView, Spinner. Они могут выступать контейнерами для других элементов управления

При работе со списками мы имеем дело с тремя компонентами. Во-первых, это визуальный элемент или виджет, который на экране представляет список (ListView, GridView) и который отображает данные. Во-вторых, это источник данных - массив, объект ArrayList, база данных и т.д., в котором находятся сами отображаемые данные. И в-третьих, это адаптер - специальный компонент, который связывает источник данных с виджетом списка.

Одним из самых простых и распространенных элементов списка является виджет **ListView**. Рассмотрим связь элемента ListView с источником данных с помощью одного из таких адаптеров - класса **ArrayAdapter**.

Класс **ArrayAdapter** представляет собой простейший адаптер, который связывает массив данных с набором элементов `TextView`, из которых, к примеру, может состоять `ListView`. То есть в данном случае источником данных выступает массив объектов. ArrayAdapter вызывает у каждого объекта
метод `toString()` для приведения к строковому виду и полученную строку устанавливает в элемент TextView.
Но теперь его время ушло, недаром на панели инструментов студии он находится в разделе **Legacy** (устаревший код). Сейчас актуален RecyclerView.

Компонент **RecyclerView** появился в Android 5.0 Lollipop и находится в разделе **Containers**. Для простоты буду называть его списком, хотя на самом деле это универсальный элемент управления с большими возможностями.

Раньше для отображения прокручиваемого списка использовался **ListView**. Со временем у него обнаружилось множество недостатков, которые было трудно исправить. Тогда решили создать новый элемент с нуля.

Вначале это был сырой продукт, потом его доработали. На данном этапе можно считать, что он стал полноценной заменой устаревшего **ListView**.

Схематично работу **RecyclerView** можно представить следующим образом. На экране отображаются видимые элементы списка. Когда при прокрутке списка

верхний элемент уходит за пределы экрана и становится невидимым, его содержимое очищается. При этом сам "чистый" элемент помещается вниз экрана и заполняется новыми данными, иными словами переиспользуется, отсюда название **Recycle**.

Компонент **RecyclerView** не является родственником **ListView** и относится к семейству **ViewGroup**. Он часто используется как замена **ListView**, но его возможности шире.

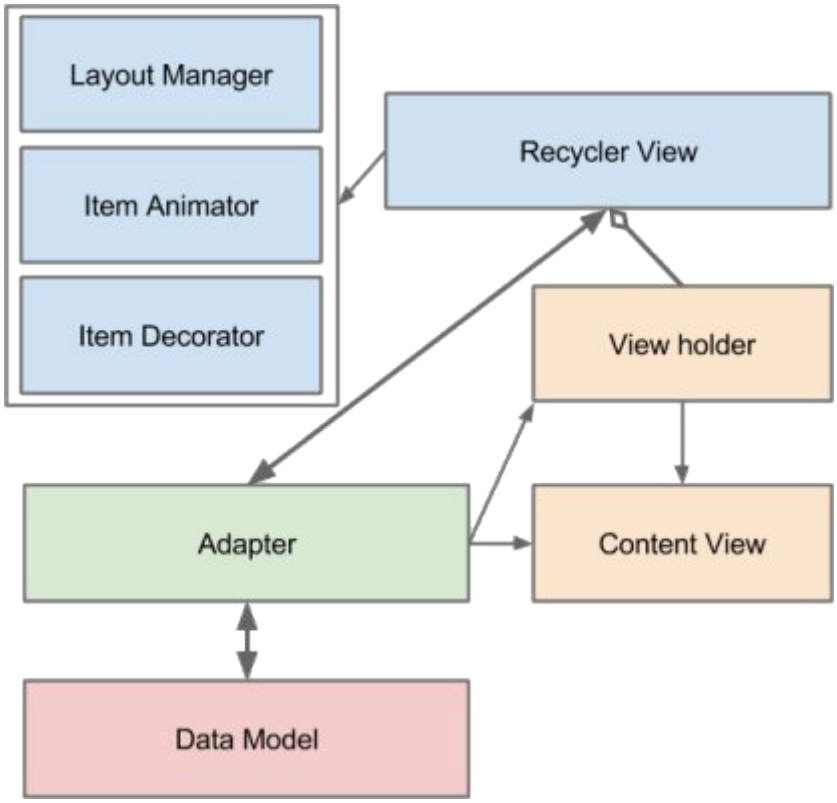
Следует сказать, что при работе с ним приходится писать много кода, который пугает новичков. Если с **RecyclerView** работать не постоянно, то порой забываются детали и сложно вспомнить необходимые шаги. Многие просто сохраняют отдельные шаблоны и копируют в разные проекты.

Внешний вид можно представить не только в виде обычного списка, но и в виде сетки. При необходимости можно быстро переключиться между разными типами отображения.

Для размещения своих дочерних элементов используется специальный менеджер макетов **LayoutManager**. Он может быть трёх видов.

- **LinearLayoutManager** - дочерние элементы размещаются вертикально (как в **ListView**) или горизонтально
- **GridLayoutManager** - дочерние элементы размещаются в сетке, как в **GridView**
- **StaggeredGridLayoutManager** - неравномерная сетка

Общая модель работы компонента.



Задание 1.

Создайте по следующему гайду новую активность в своём проекте.

Для начала создадим макет для отдельного элемента списка. Варианты могут быть самыми разными - можно использовать один **TextView** для отображения строк (имена котов), можно использовать связку **ImageView** и **TextView** (имена котов и их наглые морды). Мы возьмём для примера две текстовые метки. Создадим новый файл **res/layout/recyclerview_item.xml**.(Название можете использовать любое).

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:background="?colorPrimaryContainer"
    android:layout_height="100dp">
    <TextView
        android:id="@+id/productNameTextView"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:textColor="?colorPrimaryInverse"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHeight_percent="0.3"
        app:layout_constraintHorizontal_bias="0.069"
        app:layout_constraintStart_toEndOf="@+id/iconImageView"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.485" />
    <ImageView
        android:id="@+id/iconImageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.044"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.492" />
    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="0dp"
```

```
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHeight_percent="0.5"
        app:layout_constraintWidth_percent="0.1"
        app:layout_constraintHorizontal_bias="0.951"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.407" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Можете подобавлять какие-нибудь картинки из интернета.

Добавим компонент в разметку экрана активности.

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout

    xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">
```

```
    <androidx.recyclerview.widget.RecyclerView
    android:id="@+id/catalogRecyclerView"
    android:layout_width="match_parent"
    app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
    android:layout_height="0dp"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/catalogToolbar"
    android:visibility="gone"
    app:layout_constraintVertical_bias="0.0">
```

</androidx.constraintlayout.widget.ConstraintLayout>

Далее, создадим отдельный НОВЫЙ класс для адаптера нашей recyclerView.

```
public class SitesRecyclerAdapter extends
RecyclerView.Adapter<SitesRecyclerAdapter.SitesViewHolder>{

    public SitesRecyclerAdapter() {

    }

    @NonNull
    @Override
    public SitesRecyclerAdapter.SitesViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        View view;
        view = inflater.inflate(R.layout.servers_item,parent,false);
        return new SitesRecyclerAdapter.SitesViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull SitesRecyclerAdapter.SitesViewHolder
holder, int position) {
        SitesRecyclerAdapter.SitesViewHolder sitesViewHolder =
(SitesRecyclerAdapter.SitesViewHolder)holder;
sitesViewHolder.siteNameTextView.setText(siteList.get(position).getUrl());

    }

    @Override
    public int getItemCount() {
        return 1;
    }
    public static class SitesViewHolder extends RecyclerView.ViewHolder {
        public TextView siteNameTextView;
        AnimatedStatusView animatedStatusView;
        SitesViewHolder(View view){
            super(view);
            siteNameTextView = view.findViewById(R.id.siteNameTextView);
            animatedStatusView = view.findViewById(R.id.statusView);
        }
    }
}
```

SitesViewHolder Должен называться подобно основному классу по смыслу, а также включать в себя элементы вашего макета для элемента. Holder – ваш элемент.

В onCreateViewHolder inflater.inflate(R.layout.*servers_item*,parent,false); должен иметь ссылку на ваш макет.

В onBindViewHolder вы обозначаете значения для ваших элементов.

getItemCount() – метод, обозначающий, сколько элементов у вас будет выводится.

Обычно привязывается к размеру передаваемого списка.

Отлично, мы всё сделали, теперь перейдём к коду вашей Activity.

Там мы должны создать RecyclerView вне ваших методов жизненного цикла.

После, вы должны найти свой recyclerView на разметке.
Пример:
recyclerView = findViewById(R.id.*sitesRecyclerView*);

После этого создать объект своего класса адаптера:
SitesRecyclerAdapter adapter = new SitesRecyclerAdapter();

И после этого обозначить адаптер для вашей recyclerView:
recyclerView.setAdapter(adapter);

После этого запускаете приложение и видите 1 элемент, который мы сделали.

Полезные методы:

```
sitesViewHolder.itemView.setOnClickListener(v ->{});
```

Позволит вам обозначить, что будет происходить по нажатию на ваш элемент.

Задание 2.

Сделайте свой класс сущности, список которых вы будете выводить в recyclerView. Сделайте вывод списка.

Задание 3.

Сделайте, чтобы по нажатию на элемент вашего списка вы переходили на новую Activity. Сделать это можно с помощью создания объекта вашего класса, в котором будет находиться RecyclerView.