ПРАКТИЧЕСКАЯ РАБОТА №4

Одним из наиболее частых представлений интерфейса является список. Он используется в ленте новостей, списке сообщений и т.д.

В Android существует несколько возможностей реализации списка: ListView и RecyclerView. ListView был с самого появления Android и имел множество недостатков, например, ListView позволял создать только вертикальный список и держал все элементы списка в памяти.

В свою же очередь RecyclerView позволяет создавать списки различной ориентации. Кроме того, RecyclerView не создает View под каждый элемент списка, а создает лишь ограниченное количество View, для видимых элементов со все элементы списка, а при прокрутке списка верхний элемент уходит за пределы экрана и очищается, а после новый элемент помещается вниз экрана и заполняется новыми данными.

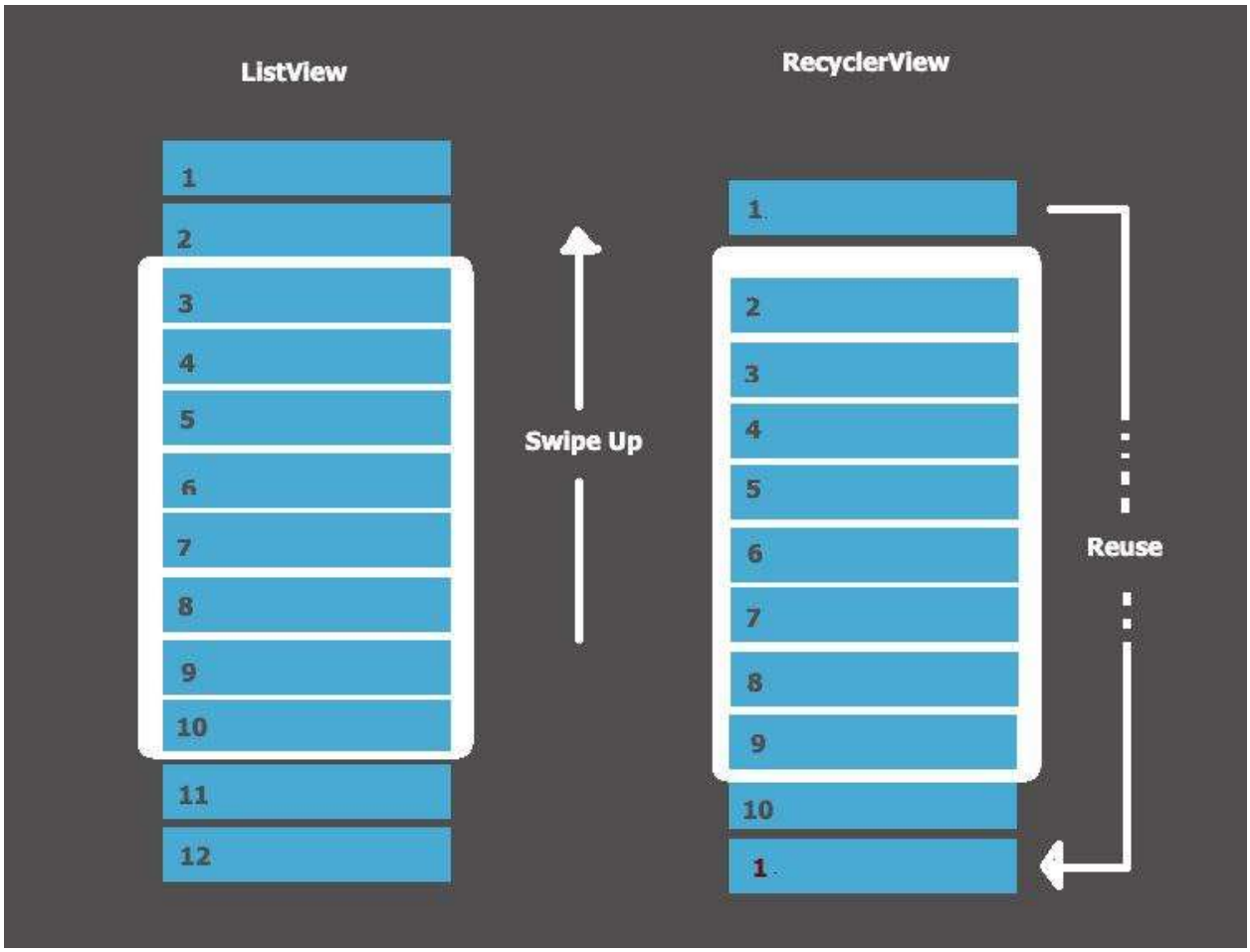

Рисунок 4.1. – Различия между ListView и RecyclerView

Основным компонентом для работы с каждой из реализаций списков является Adapter. С его помощью в ListView и RecyclerView загружаются

данные и определяется разметка элемента списка. Для каждой из реализации создается свой адаптер.

Адаптеры – это такие классы, обеспечивающие базовый функционал для управления содержимым списков. Для ListView базовым является адаптер ArrayAdapter. По умолчанию ArrayAdapter использует метод toString() из объекта массива, чтобы наполнять данными элемент TextView, размещённый внутри указанной разметки. Для создания своего адаптера необходимо создать класс-наследник от ArrayAdapter. Далее, для изменения разметки и наполнения элемента списка, необходимо будет изменять метод getView.

Адаптер для ListView

```
public class YourCustomListAdapter extends ArrayAdapter<Item> {
     private LayoutInflater inflater;
     private int layout;
     private List<Item> items;
     public YourCustomListAdapter(Context context, int resource,
List<Item> items) {
        super(context, resource, items);
        this.items = items;
        this.layout = resource;
        this.inflater = LayoutInflater.from(context);
    }

    public View getView(int position, View convertView,
ViewGroup parent) {
         View view=inflater.inflate(this.layout, parent, false);
         TextView textView = view.findViewById(R.id.textView);
        Item item = items.get(position);
         textView.setText(item.getText());
        return view;
    }
}
```

Пример использования ListView во Fragment.

```
Данный код размещается в классе Fragment, описывающем верстку
вашего экрана.
```

```
// Первым делом необходимо найти список на верстке экрана
        itemsList = findViewById(R.id.itemsList);
        // Далее, создать адаптер и передать в него Context,
ресурс разметки элемента списка, массив элементов
        adapter = new YourCustomListAdapter(this,
R.layout.list_item, items);
        // устанавливаем для списка адаптер
        itemsList.setAdapter(adapter);
```

Для обработки нажатий используется слушатель OnItemClickListener.

```
        itemsList.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
            @Override
            public void onItemClick(AdapterView<?> parent, View
v, int position, long id)
            {
//По позиции элемента возможно получить сам элемент и далее
выполнить необходимые действия
            }
        });
```

Работа с RecyclerView

Для работы с RecyclerView используется ViewHolder. Он помогает сделать прокрутку списка плавной. Он сохраняет ссылки на элементы списка, чтобы затем адаптер мог их переиспользовать. Благодаря этому создание View элемента (inflating) и вызов дорогого метода findViewById() происходит не для каждого элемента списка, а лишь для нескольких из списка.

Для заполнения RecyclerView и определения его внешнего

Адаптер RecyclerView в обязательном порядке требует реализации методов - onCreateViewHolder() и onBindViewHolder().

Адаптер для RecyclerView

. Для RecyclerView базовым является адаптер RecyclerView.Adapter, от которого наследуются все адаптеры для данного элемента списка.

```
public class YourCustomRecyclerViewAdapter  extends
RecyclerView.Adapter< YourCustomRecyclerViewAdapter.ViewHolder>{

    private final LayoutInflater inflater;
    private final List<Item> items;
```

```
    YourCustomRecyclerViewAdapter(Context context, List<State>
items) {
        this.items = items;
        this.inflater = LayoutInflater.from(context);
    }
    @Override
    public YourCustomRecyclerViewAdapter.ViewHolder
onCreateViewHolder(ViewGroup parent, int viewType) {

        View view = inflater.inflate(R.layout.list_item, parent,
false);
        return new ViewHolder(view);
    }

    @Override
    public void
onBindViewHolder(YourCustomRecyclerViewAdapter.ViewHolder
holder, int position) {
        Item item = items.get(position);
        holder.textView.setText(item.getText());
    }

    @Override
    public int getItemCount() {
        return items.size();
    }

    public static class ViewHolder extends
RecyclerView.ViewHolder {
        final TextView textView;
        ViewHolder(View view){
            super(view);
            textView = view.findViewById(R.id.textView);
        }
    }
}
```

Так как RecyclerView поддерживает различные варианты компоновки элементов, то необходимо указывать тип менеджера компоновки, который определит расположения элементов. По умолчанию библиотека RecyclerView предоставляет три реализации данного менеджера компоновки элементов в RecyclerView:

1. LinearLayoutManager: упорядочивает элементы в виде списка с одной колонкой.
2. GridLayoutManager: упорядочивает элементы в виде грида со столлбцами и строками. Грид может упорядочивать элементы по горизонтали (горизонтальный грид) или по вертикали (вертикальный грид).
3. StaggeredGridLayoutManager: аналогичен GridLayoutManager, однако не требует установки для каждого элемента в строке имели одну и ту же высоту (для вертикального грида) и одну и ту же ширну (для горизонтального грида).

Указание менеджера компоновки возможно как с помощью XML, так и динамически в коде.

Пример:

```
app:layoutManager="androidx.recyclerview.widget.GridLayoutManage
r"
LinearLayoutManager layoutManager = new
LinearLayoutManager(getApplicationContext());
recyclerView.setLayoutManager(layoutManager);
```

Пример использования RecyclerView во Fragment.

Данный код размещается в классе Fragment, описывающем верстку экрана.

```
// Первым делом необходимо найти список на верстке экрана
        itemsList = findViewById(R.id.itemsList);
        // Далее, создать адаптер и передать в него Context и
массив элементов
        adapter = new YourCustomListAdapter(this, items);
        LinearLayoutManager layoutManager = new
LinearLayoutManager(getApplicationContext());
        itemsList.setLayoutManager(layoutManager);
        // устанавливаем для списка адаптер
        itemsList.setAdapter(adapter);
```

Для обработки нажатий на элемент списка, необходимо в адаптере RecyclerView создавать метод, который будет назначать слушателя кликов на элемент списка, так как по умолчанию адаптер RecyclerView не содержит методов обработки нажатия на элемент списка.

Важный момент! При изменении массива элементов, чтобы перерисовался список необходимо использовать метод notifyDataSetChanged().

Задание

1. Создать мобильное приложение, состоящее из 3 экранов. Экраны должны быть реализованы с помощью фрагментов.
2. Первый фрагмент содержит 2 кнопки, которые переводят на второй и третий фрагменты.
3. На втором фрагменте, согласно предметной области необходимо разместить список, реализованный с помощью ListView.
4. На третьем фрагменте, согласно предметной области необходимо разместить список, реализованный с помощью RecyclerView.
5. Каждый из элементов списка должен содержать как минимум один ImageView и один TextView
6. Список должен содержать не менее 200 элементов.
7. На каждом из экранов реализовать отображение Toast и сообщений в Log при нажатии на элемент списка.