Introduction to Recycler View

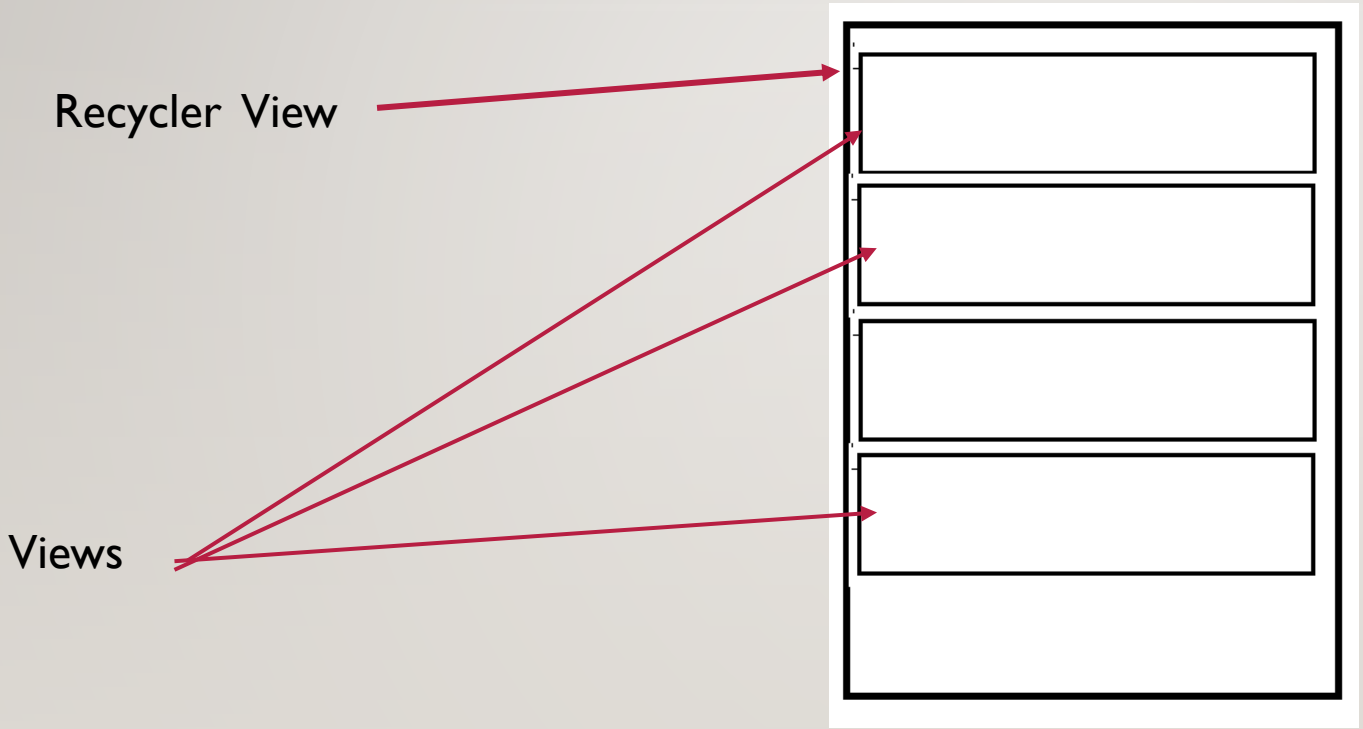
OUTLINE

- Introduction
- Recycler View
- Adapter class
- View Holder class
- Examples

RECYCLER VIEW

- Allow us to display list of data sets to the user in the form of a scrollable list.
- RecyclerView is the ViewGroup that contains the views corresponding to your data.

RECYCLER VIEW

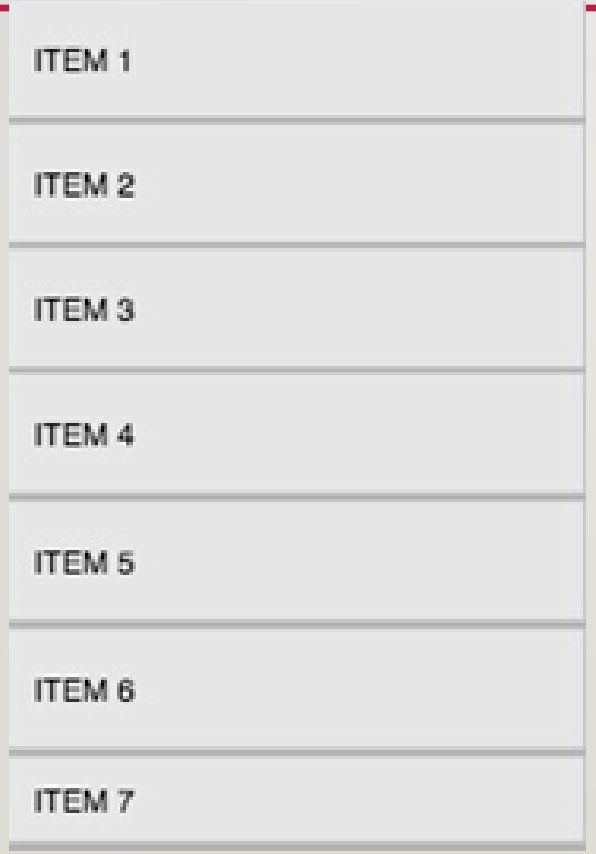


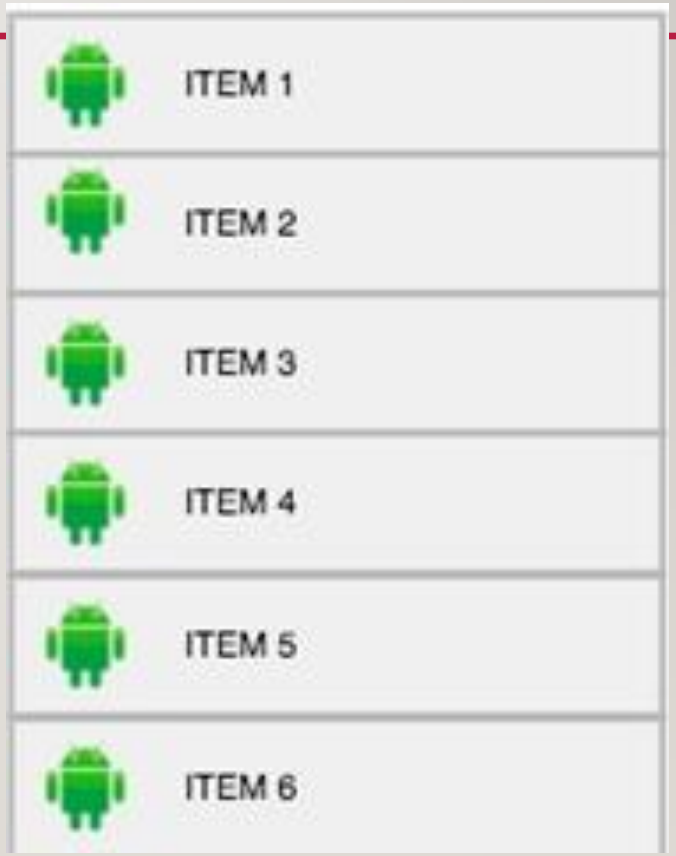
RECYCLER VIEW VS LIST VIEW

- Recycler view has the following advantages over List View
  - Efficient in the way it manages the views (recycling)
  - Performance and reduces the resources used
  - Layout managers to control list presentation style e.g. LinearLayoutManager: is used to present horizontal or vertical scrolling list.

Src: https://developer.android.com/codelabs/kotlin-android-training-recyclerview-fundamentals#2

RECYCLER VIEW EXAMPLES



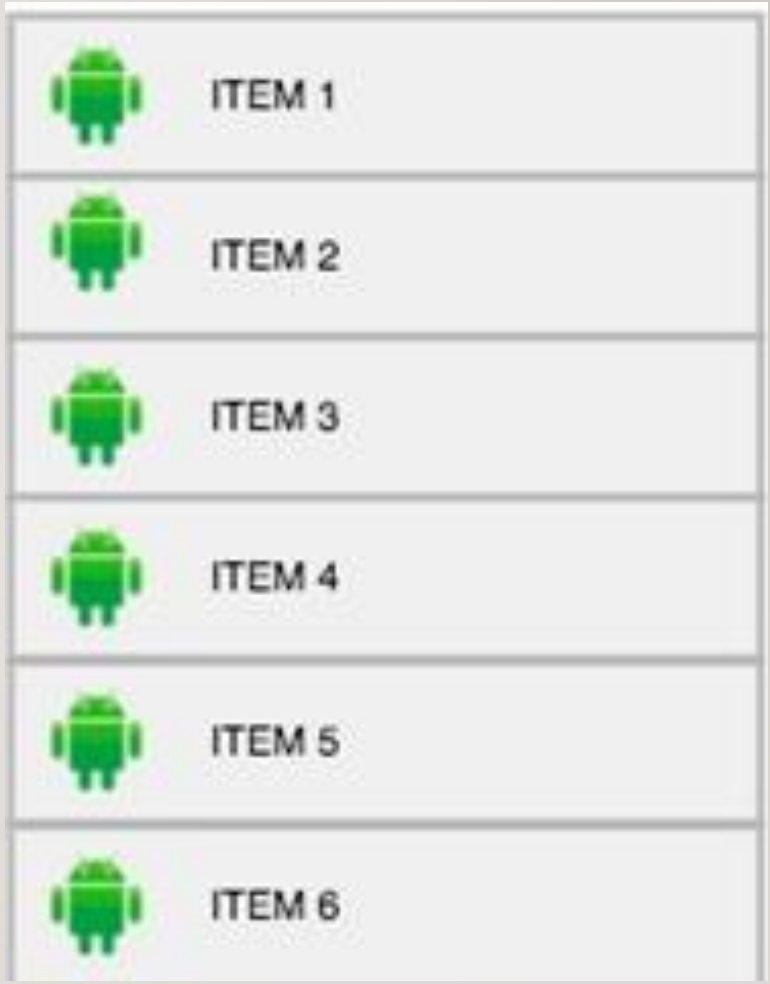


RECYCLER VIEW (EXAMPLE)

Each row(view) represents a childView in the Recycler View

Here each *row* includes

- Text View for username
- Image View for the image



RECYCLER VIEW (EXAMPLE)

Here each *row* of the ListView is comprised of

TextView for province.

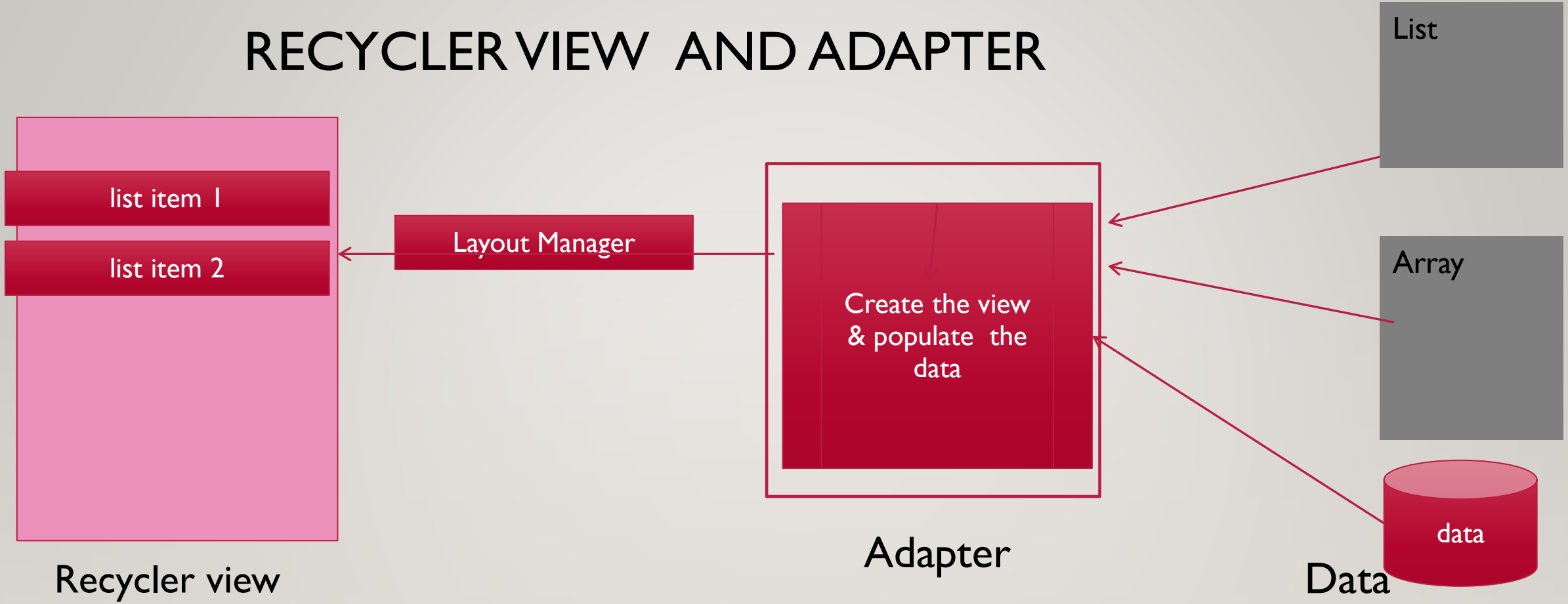
TextView for the capital.

Alberta	**Edmonton**
British Columbia	**Victoria**
Manitoba	**Winnipeg**
New Brunswick	**Fredericton**
Newfoundland and Labrador	**St. John's**
Nova Scotia	**Halifax**
Ontario	**Toronto**
Prince Edward Island	**Charlottetown**
Quebec	**Quebec City**
Saskatchewan	**Regina**

ADAPTER

- An Adapter object acts as a bridge between the Recycler View and the underlying data.
- manage the data and also Responsible for making a View for each item in the data set.

RECYCLER VIEW AND ADAPTER



- Data could come from different locations (array , database etc…)

CUSTOM ADAPTER

- created as a subclass of the RecyclerView.Adapter class and must, at a minimum, implement the following methods,
  - **getItemCount()** – return a count of the number of items the list.
  - **onCreateViewHolder()** – This method creates and returns a ViewHolder object
  - **onBindViewHolder() –** populate the views in the layout with the text and graphics corresponding to the specified item

and

VIEWHOLDER CLASS

- RecyclerView.Adapter implementations should subclass ViewHolder
- The ViewHolder instance contains
  - the information to be displayed and the view layout used to display the item.
  - item view and metadata about its place within the RecyclerView.
  - makes binding view contents easier.

RECYCLER VIEW USING CUSTOM ADAPTER

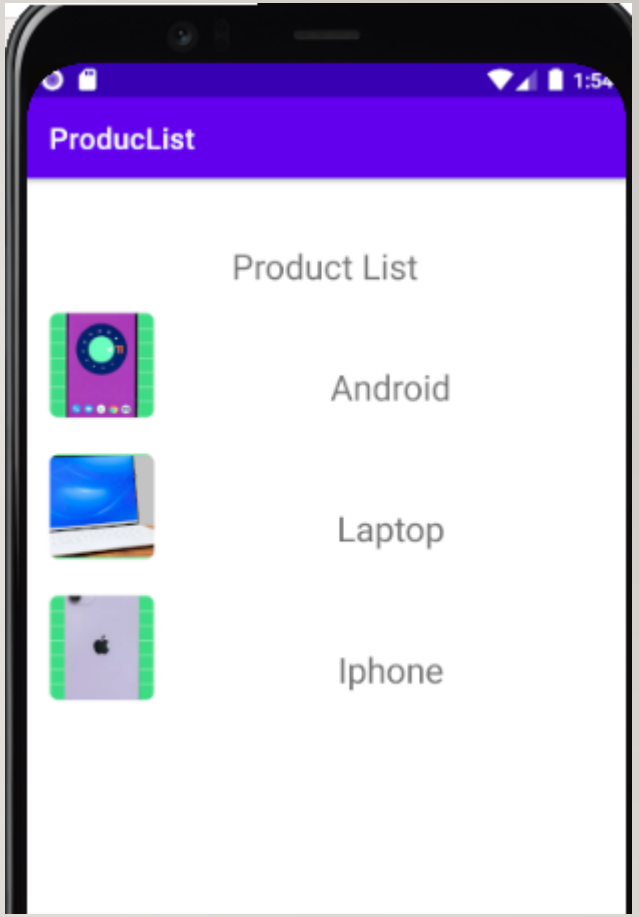
- Create a Model class to represent your data (e.g. Product , Book , User).
- Create an xml-layout file that represent the view that you want each row within the Recycler View to display
- Create adapter class which extends RecyclerViewAdapter and implements the following methods
  - getItemCount()
  - onCreateViewHolder()
  - onBindViewHolder() –
  - Create a ViewHolder class that extends RecyclerView.ViewHolder

EXAMPLE RECYCLER VIEW USING CUSTOM ADAPTER

- Create an instance of Adapter passing in the required information
- Set Layout Manager for Recycler view
- Assign the Adapter to the RecyclerView

EXAMPLE

Recycler view that display list of product as shown in the figure



EXAMPLE

1- get resources from BB week 4 (images)

2- create android project

3- add images to drawable folder

4- add Recycler View to main activity

5- create Product class (Model)

6-Create a custom item_ layout XML file to visualize the item

7-Create a RecyclerView.Adapter and ViewHolder to render the item

8- Bind the adapter to the data source to populate the RecyclerView

ITEM_LAYOUT

```
<?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="wrap_content">

    <EditText
        android:id="@+id/nameEt"
        android:layout_width="222dp"
        android:layout_height="49dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Product Name"
        android:textAlignment="center"
        android:textSize="24sp"
         />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:srcCompat="@tools:sample/avatars" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MODEL CLASS

```
public class Product {
    // fields
    String name;
    int image;
    // constructor
    Product( String n , int im){
    this.name = n;
    this.image = im; }
    // add setters and getters
    //
}
```

GET THE DATA

```
ArrayList<Product> getData() {
    ArrayList<Product> list = new ArrayList<>();
    Product p1 = new Product(" Android ", R.mipmap.androiod);
    Product p2 = new Product(" Laptop ", R.mipmap.laptop1);
    Product p3 = new Product(" iPhone ", R.mipmap.phone);
    list.add(p1);
    list.add(p2);
    list.add(p3);
    return list;
}
```

CREATE CUSTOM ADAPTER

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

    List<Product> list ;

    MyAdapter( List<Product> list){...}
     @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {...}

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {...}

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

    public static class ViewHolder extends RecyclerView.ViewHolder {...}
}
```

CUSTOM ADAPTER CONT.

```
// constructor
MyAdapter( List<Product> list){

    this.list = list;
}
```

On Create View Holder

```
 @NonNull
@Override
public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

     LayoutInflater layoutInflater = LayoutInflater.from(parent.getContext());
     View item = layoutInflater.inflate(R.layout.lrow_layout , parent ,false);
     ViewHolder viewHolder = new ViewHolder(item);
 return viewHolder;
}
```

On Bind View Holder

```
@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position)
{



    holder.im.setImageResource(list.get(position).image);
    holder.name.setText(list.get(position).name);
}
```

Get item count method

```
@Override
public int getItemCount() {
        return list.size()
}
```

View holder

```
public static class ViewHolder extends RecyclerView.ViewHolder {

    ImageView im;
    TextView name;
    public ViewHolder(@NonNull View itemView) {
        super(itemView);

        name = itemView.findViewById(R.id.nameEt);
        im = itemView.findViewById(R.id.imageView);
    }
}
```

REFERENCES

- https://developer.android.com/guide/topics/ui/layout/recyclerview
- https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView
- https://developer.android.com/codelabs/android-training-create-recycler-view#0
- https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.ViewHolder