3.a1 (OPTIONAL). RecyclerView

Mario Mata
mario.mata@gcu.ac.uk
M215a

GCU
Glasgow Caledonian University

University for the Common Good

6. `RecyclerView` class

- `RecyclerView` is similar to `ListView`, but it is intended to handle a large number of items
- It *recycles* the individual items displayed
  - When an item scrolls off the screen, RecyclerView doesn't destroy its view (like ListView does)
  - Instead, RecyclerView reuses the view for new items that have scrolled into the screen (just updating the contents)
- `ListView` also gets the Views to display from a custom `Adapter` to create each view from the data
  - This custom Adapter must now implement a custom ViewHolder object (extending `RecyclerView.ViewHolder`) which binds objects to each item view's widgets
  - And update an item's view contents with `onBindViewHolder()`

RecyclerView_test.zip – example (I)

- As usual, we define the view for each item (just a TextView in this example, in ***text_row_item.xml***):

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/..."
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height=" wrap_content "
    ……>
</TextView>
```

- Create a new class for our custom Adapter, with a *nested* custom ViewHolder class:

```
public class RecViewCustomAdapter extends
          RecyclerView.Adapter<RecViewCustomAdapter.ViewHolder> {
 private String[] mDataSet; //RecyclerView's data container(s)
 public RecViewCustomAdapter(String[] dataSet) {
      mDataSet = dataSet; //load data to display
 }
 public static class ViewHolder extends RecyclerView.ViewHolder {
      ……
```

RecyclerView_test.zip – example (II)

- The nested, custom ViewHolder class binds objects to the widgets in any item's view (and handles onClick):

```
public static class ViewHolder extends RecyclerView.ViewHolder {
  private final TextView textView;
  public ViewHolder(View v) {
    super(v);
    v.setOnClickListener(new View.OnClickListener() {
    @Override public void onClick(View v) {
       Log.d(TAG, "Element " + getAdapterPosition() + " clicked.");
      }
    });
    textView = (TextView) v.findViewById(R.id.textView);
  }
  public TextView getTextView() {
    return textView;
  }
}
```

RecyclerView_test.zip – example (III)

- Our Adapter must implement `onCreateViewHolder()` to inflate the custom layout into each new item's view
  - This is invoked by a *LayoutManager* that we choose for the RecyclerView, allowing to configure how to arrange the items

```
public class RecViewCustomAdapter extends
            RecyclerView.Adapter<RecViewCustomAdapter.ViewHolder> {
 ……
 //Create each new item view (invoked by the layout manager)
 @Override
 public ViewHolder onCreateViewHolder(ViewGroup viewGroup,
                                      int viewType) {
   // Create a new view inflating our custom item layout
   View v = LayoutInflater.from(viewGroup.getContext())
                .inflate(R.layout.text_row_item, viewGroup, false);
   return new ViewHolder(v);
 }
 ……
```

RecyclerView_test.zip – example (IV)

- Our Adapter must also implement `onBindViewHolder()` to just *update the contents* of an item's view, rather than ceating a new one –the *recycling* process:

```
public class RecViewCustomAdapter extends
            RecyclerView.Adapter<RecViewCustomAdapter.ViewHolder> {
 ……
 // Replace contents of an item's view (invoked by layout manager)
    @Override
    public void onBindViewHolder(ViewHolder viewHolder,
                                 final int position) {
        Log.d(TAG, "Element " + position + " set.");
        // Get element from your dataset at this position
        // and replace the contents of the view with that element
        viewHolder.getTextView().setText(mDataSet[position]);
    }
}
```

RecyclerView_test.zip – example (V)

- We finally setup the RecyclerView in our Activity (picking a LayoutManager for it), and use the custom adapter to "inject" the data into the RecyclerView :

```
……
rv = findViewById(R.id.recView);
……
RecViewCustomAdapter myAdapter = new RecViewCustomAdapter(data);
//Adapt into the recyclerView
LinearLayoutManager llm = new LinearLayoutManager(this);
llm.setOrientation(LinearLayoutManager.VERTICAL);
rv.setLayoutManager(llm);
rv.setAdapter(myAdapter);
```