**Homework Week Two Day Three**

**Research**

1. **What is the difference in recyclerView and listView?**

Both RecyclerView and ListView are used to display information from Collection in a uniformed fashion. There are many differences between the two and they are:

|  |  |
| --- | --- |
| ListView | Recycler View |
| Native (already in the code) | Library must be implemented (3rd party lib) |
| Eager Loading | Lazy Loading |
| Uses an array adapter (ArrayAdapter) | Has its own adapter (RecyclerView.Adapter) |
| ItemClick Listener is built in | ItemClick Listener must be implemented |
| Vertical | Horizontal/Vertical/Grid |
| Does not have a layout manager | Has a layout manager |
| Older | Newer |
| Does not enforce ViewHolder pattern by default | Must use ViewHolder pattern |
| Less efficient | More efficient (lazy) |
| Rendered all at once | Rendered on-demand |
| Slower | Faster |

1. **Define lazy loading**.

Lazy loading is the opposite of ListView’s eager loading. In Recycler View, lazy loading is a popular technique for loading process in which the web page content is loaded as the user scrolls down or up the page. Advantages of lazy loading are that it reduces time consumption and memory usage, therefore, optimizing content delivery of data. In other words, instead of downloading the entire webpage in one go like in the Listview, RecyclerView allows devices to download a series of placeholders that correspond to the actual information on the page.

The browser caches the website’s resources without actually loading all of them at once because it simply knows where to process them. If the user scrolls through the page, the browser will begin to upload and show all the thing in that page right before you get to them. An example of this would be the Google’s image search’s lazy loading with the thumbnail replacing placeholders as the user scrolls down the page.

1. **What is an item decorator in RecyclerViews?**

ItemDecoration is a tool used to decorate the children of a RecyclerView. Each of its children is represented by an item in a list. With ItemDecoration, one can easily modify the appearance of the RecyclerView’s child views. ItemDecoration has 3 methods to override, but only 2 are required to be implemented, which are getItemOffsets() and onDraw(). The method getItemOffsets is needed between list items, so we’re not drawing dividers on top of our child views. This method must be called for each of the RecyclerView’s children. In the second method onDraw(), the bounds of our divider must be determined and drawn onto the RecyclerView.onDraw call just once.

1. **What is the View Holder Pattern?**

The ViewHolder pattern is a design pattern that enables you to access each list item view without the need for the look up, which in turn saving valuable processor cycles. It avoids frequent call of the findViewById() during the ListView scrolling and it will make it smooth. With the ViewHolder pattern, the following are the steps how it works during ListView scrolling:

* The first time it was loaded, convertView is null. We will have to inflate our list item layout, instantiate the ViewHolder, find the TextView via findViewById() and assign it to the ViewHolder and set it as a tag of convertView.
* The second time it was loaded, convertView is not null. We do not have to inflate again and we do not have to call findViewById() again since we can now access the TextView via its ViewHolder.
* The third time it was loaded, convertView is not null, the findViewById() is never called again and this by itself makes our smooth ListView scrolling.

1. **How do you implement an item touch helper for the RecyclerView?**

ItemTouchHelper is a utility class to add swipe to dismiss and drag and drop support to RecyclerView. It works well with a RecyclerView and Callback class, which configures which interactions are enabled and to also receives events when the user performs any of these actions. Depending on which functionality you support, you should implement it by overriding onMove() and/or onSwiped() methods. This class should work with any LayoutManager as it can be optimized for your custom LayoutManager by extending methods in the ItemTouchHelper.Callback class or by implementing ItemTouchHelper.ViewDropHandler interface in your LayoutManager. Most of the time you only need to override onChildDraw to customize ItemTouchHelper repositioning properties behaviors.