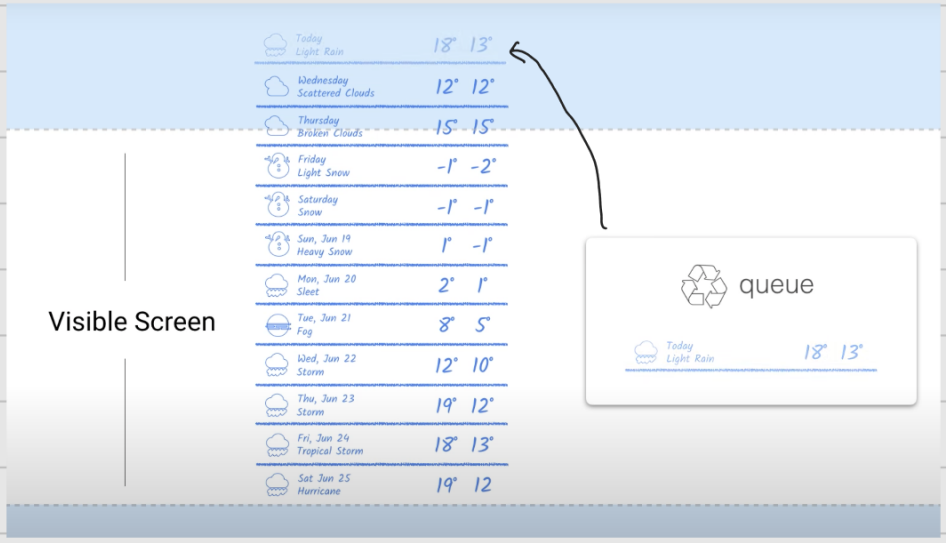
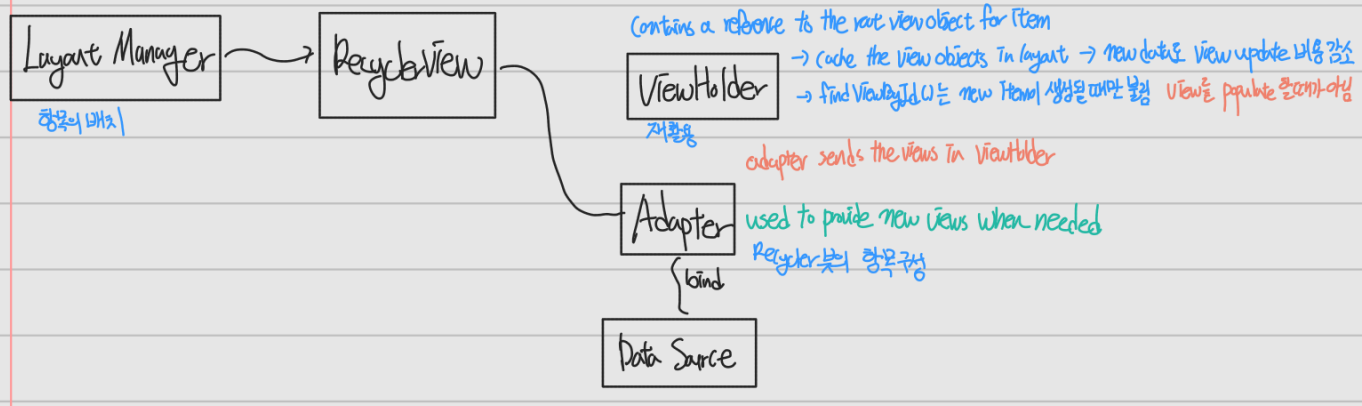
How works?



- queue에 보여질 View 저장 및 재사용을 위해 시야에서 사라진 View 저장

tells how layout all those views. Horizontal? Vertical?



Why ViewHolder?

- To cache the view objects that you're going to be populating with data or images
- When the recycler view is first being populated, you call findViewById() for each view that will be showing data from the adapter (Can be expensive)

=> Best to do it once and cache those views in a view holder

View Holder
Cache 캐쉬

QUIZ QUESTION

Let's say that each item in your RecyclerView list contains four individual data views, and you don't cache these views in a ViewHolder. If eight items fit on screen, approximately how many extra findViewById() calls will be made if you scroll through 30 items?

In addition to the eight items that fit on screen, assume that two extra items are needed for smooth scrolling.

With only eight items on screen, we would need at least eight item views, but the question says that we need two extras for smooth scrolling, which means 10 items total. 10 items times 4 individual data views per item means 40 calls to findViewById. And we could cache these views in a ViewHolder to fill our RecyclerView and then access them later when we scroll and recycle views. If we didn't use a ViewHolder, we would have to call findViewById 4 times for each of the 30 items we scroll through, that's 30x4 or 120 calls to findViewById.

So the difference between using a ViewHolder and not using one is 120-40 calls to findViewById. 80 extra calls for not using a ViewHolder! Android phones today are so fast, that you probably wouldn't notice this optimization, but it will give you slightly better battery usage, which could be noticeable if you are scrolling through very large lists. So it's best to use a ViewHolder.

Creating RecyclerView Layout

'Implementation' ~ 28.0.0'으로 교체

```
// TODO (1) Add RecyclerView dependency
compile 'com.android.support:recyclerview-v7:24.1.1'
```

표준라이브러리의 부가 아님

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!--TODO (2) Replace the TextView with a RecyclerView with an id of "@+id/rv_numbers"-->
    <!--TODO (3) Make the width and height of the RecyclerView match_parent-->
    <android.support.v7.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/rv_numbers">

    </android.support.v7.widget.RecyclerView>
</FrameLayout>
```

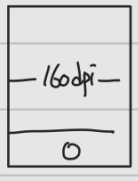
Items & ViewHolders

Not actual pixel size

SP

DP

- scaled to be the same approximate physical size, regardless of the density of the pixels on the screen.



1dpi = 1pixel

px = dp * (dpi/160). Modern device has 480 dpi 수준

- SP는 DP와 유사하지만 also scaled based upon user preferences.

typically text size preferences.

res/layout/number_list_item.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:padding="16dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:textSize="42sp"
        android:fontFamily="monospace"
        android:layout_gravity="center_vertical|start"
        android:id="@+id/tv_item_number"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

</FrameLayout>
```

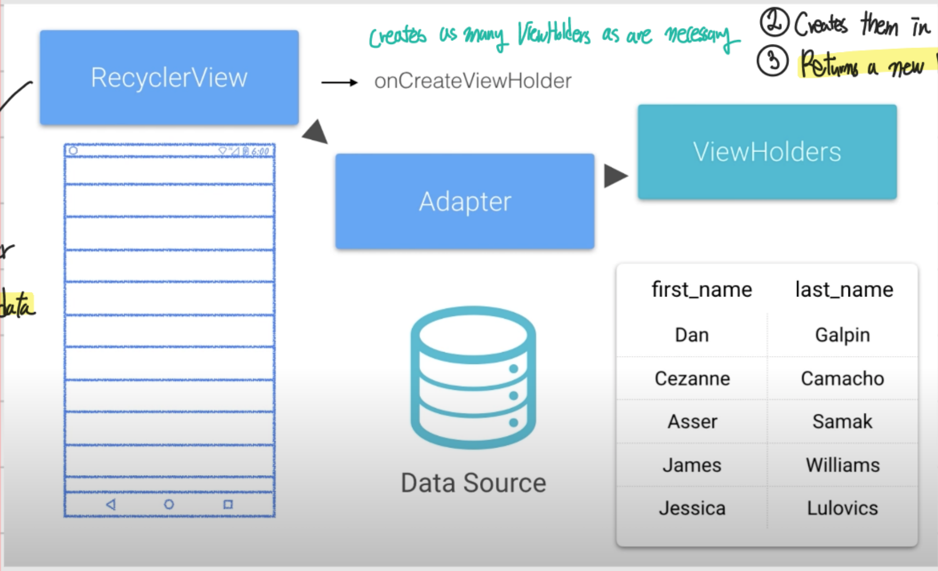
GreenAdapter 안의 Inner class

```
public class NumberViewHolder extends RecyclerView.ViewHolder {

    TextView listItemNumberView;

    public NumberViewHolder(View itemView) {
        super(itemView);
        listItemNumberView = (TextView) itemView.findViewById(R.id.tv_item_number);
    }

    void bind(int listIndex) {
        listItemNumberView.setText(String.valueOf(listIndex));
    }
}
```

RecyclerView and Adapters

- The adapter is called by the RecyclerView to create new items in the form of ViewHolders.
- Populates or binds these items with data.
- Returns information about the data ex) How many items there are in a given data source.
- From ArrayList, JSON result of a network request …

① Inflates the item views from XML
② Creates them in code
③ Returns a new ViewHolder object



Creating an Adapter

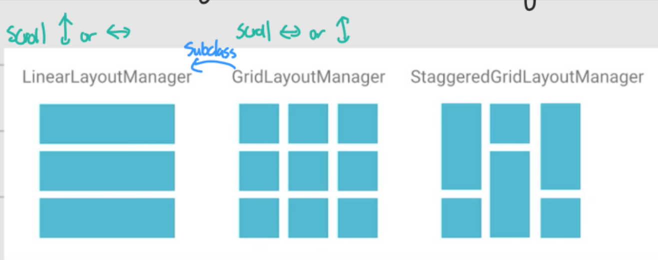
ViewHolder를 내부적으로 이용

```java
public class GreenAdapter extends RecyclerView.Adapter<GreenAdapter.NumberViewHolder> {

    private int mNumberItems;

    GreenAdapter(int numberOfItems) {
        this.mNumberItems = numberOfItems;
    }

    @Override
    public NumberViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        Context context = parent.getContext();
        int layoutIdForListItem = R.layout.number_list_item;
        LayoutInflater inflater = LayoutInflater.from(context);
        boolean shouldAttachToParentImmediately = false;

        View view = inflater.inflate(layoutIdForListItem, parent, shouldAttachToParentImmediately);
        NumberViewHolder viewHolder = new NumberViewHolder(view);

        return viewHolder;
    }

    @Override
    public void onBindViewHolder(NumberViewHolder holder, int position) {
        holder.bind(position);
    }

    @Override
    public int getItemCount() {
        return mNumberItems;
    }
```

- onCreateViewHolder: 레이아웃 XML파일의 inflate 담당
- layoutIdForListItem: XML id. 여러 뷰들의 모음일 수도!
- shouldAttachToParentImmediately: general setting.
- view: inflate된 뷰 계층구조의 root 객체를 ViewHolder의 생성자로 넘김 → 각 뷰 객체를 findViewById()
- viewHolder: Adapter에서 내부적으로 메모리에 저장시킴 → onBindViewHolder() 호출시 매개변수로 전달

RecyclerView Layout Manager

- ViewHolder: Determines how an individual entry is displayed.
- Layout Manager: Determines how the collection of items is displayed.
- Determines when to recycle items views that are no longer visible



Wiring up RecyclerView

```
public class MainActivity extends AppCompatActivity {

    private static final int NUM_LIST_ITEMS = 100;

    private GreenAdapter mAdapter;
    private RecyclerView mNumberList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mNumberList = (RecyclerView) findViewById(R.id.rv_numbers);
        LinearLayoutManager layoutManager = new LinearLayoutManager( context: this);
        mNumberList.setLayoutManager(layoutManager);
        //to designate that the contents of the RecyclerView won't change an item's size
        mNumberList.setHasFixedSize(true);
        mAdapter = new GreenAdapter(NUM_LIST_ITEMS);
        mNumberList.setAdapter(mAdapter);
    }
}
```

Responding to Item Clicks

GreenAdapter의 Inner Interface로 생성

```
// COMPLETED (1) Add an interface called ListItemClickListener
public interface ListItemClickListener {
    // COMPLETED (2) Within that interface, define a void method called onListItemClick that takes an int as a parameter
    void onListItemClick(int index);
}
```

```
public class GreenAdapter extends RecyclerView.Adapter<GreenAdapter.NumberViewHolder> {

    private static final String TAG = GreenAdapter.class.getSimpleName();
      멤버변수로 들고있음
    // COMPLETED (3) Create a final private ListItemClickListener called mOnClickListener
    private final ListItemClickListener mOnClickListener;
```

```
// COMPLETED (4) Add a ListItemClickListener as a parameter to the constructor and store it in mOnClickListener
public GreenAdapter(int numberOfItems, ListItemClickListener listItemClickListener) {
    mNumberItems = numberOfItems;
    mOnClickListener = listItemClickListener;
    viewHolderCount = 0;
}
```

생성자주입. MainActivity가 onCreate()에서 자기자신주입

GreenAdapter 안 Innerclass에 OnClickListener Implements.

```
// (5) Implement OnClickListener in the NumberViewHolder class
class NumberViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
```

NumberViewHolder는 GreenAdapter의 Innerclass 이므로 바깥 클래스 접근가능

```
// (6) Override onClick, passing the clicked item's position (getAdapterPosition()) to mOnClickListener via its onListItemClick method
@Override
public void onClick(View view) {
    int position = getAdapterPosition();
    mOnClickListener.onListItemClick(position);
}
```

```
public NumberViewHolder(View itemView) {
    super(itemView);

    listItemNumberView = (TextView) itemView.findViewById(R.id.tv_item_number);
    viewHolderIndex = (TextView) itemView.findViewById(R.id.tv_view_holder_instance);
    // (7) Call setOnClickListener on the View passed into the constructor (use 'this' as the OnClickListener)
    itemView.setOnClickListener(this);
}
```

itemView가 눌리면 NumberViewHolder의 onClick()이 불리는듯

```
// (8) Implement GreenAdapter.ListItemClickListener from the MainActivity
public class MainActivity extends AppCompatActivity implements GreenAdapter.ListItemClickListener {
```

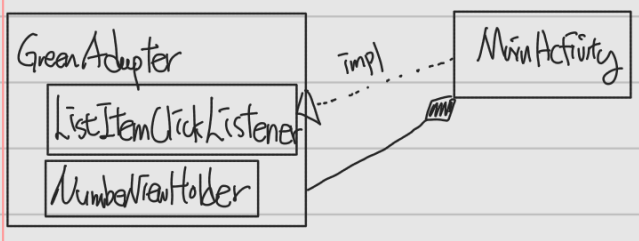
```
// (10) Override ListItemClickListener's onListItemClick method
@Override
public void onListItemClick(int index) {
    // (11) In the beginning of the method, cancel the Toast if it isn't null
    if (mToast != null) {
        mToast.cancel();
    }
    // (12) Show a Toast when an item is clicked, displaying that item number that was clicked
    mToast.makeText( context: MainActivity.this, String.valueOf(index), Toast.LENGTH_SHORT).show();
}
```

MainActivity의 Implement. 실제 동작구현

MainActivity의 onCreate()

```
// (13) Pass in this as the ListItemClickListener to the GreenAdapter constructor
/*
 * The GreenAdapter is responsible for displaying each item in the list.
 */
mAdapter = new GreenAdapter(NUM_LIST_ITEMS,  listItemClickListener: this);
mNumbersList.setAdapter(mAdapter);
```

Adapter를 작동하여 점점 RecycleView 완성



ViewHolder

- 항목을 구성하기 위한 뷰들을 findViewById 해주는 역할

ListView에서는 개별적의 선택

장점: ViewHolder 객체를 Adapter 내부에서 메모리에 유지 → ViewHolder에 의해 최초에 한번만 findViewById