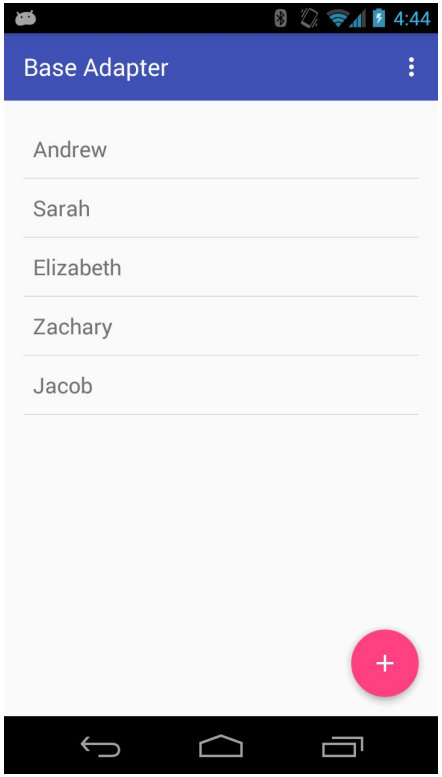
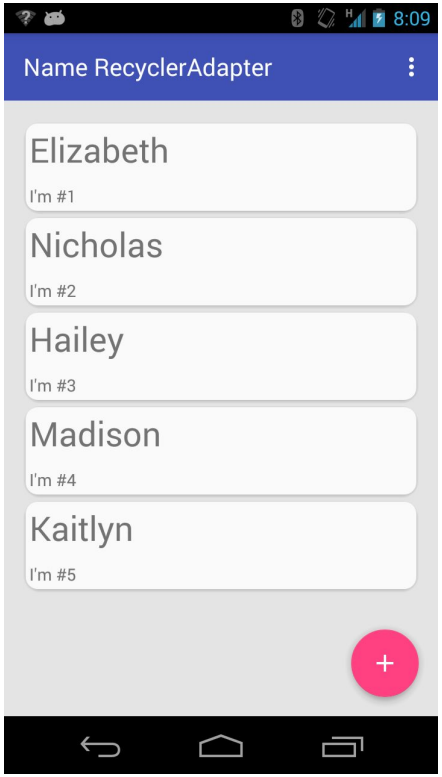
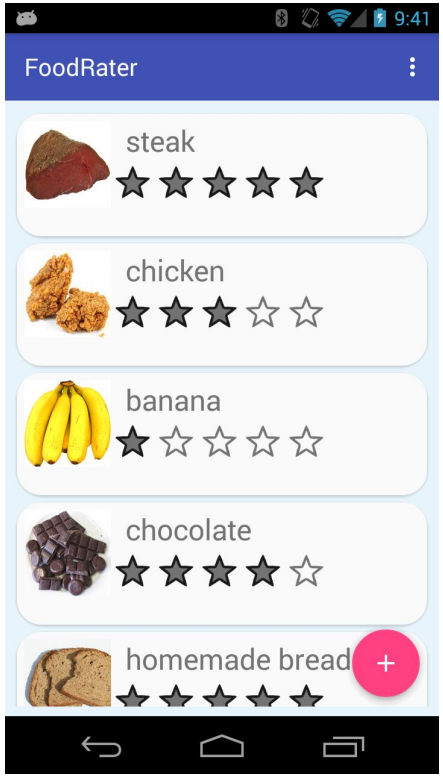
Recycler View and Adapters



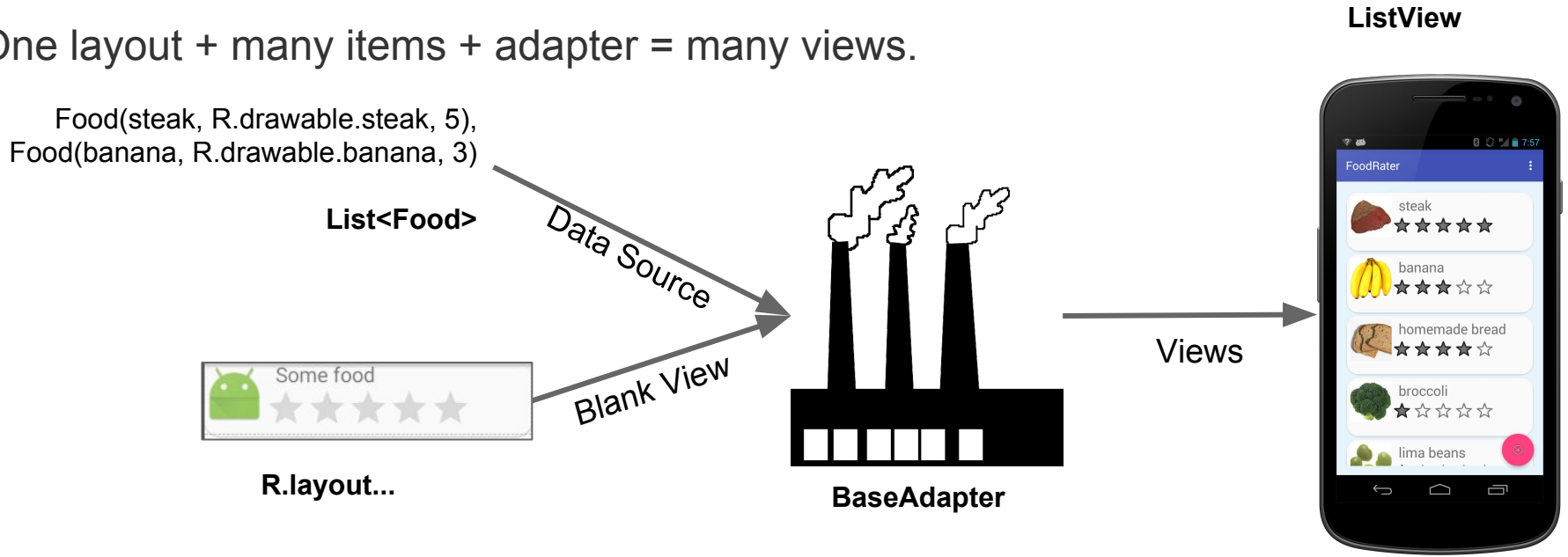




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Adapters tie array data to views by replicating the view for each element

One layout + many items + adapter = many views.

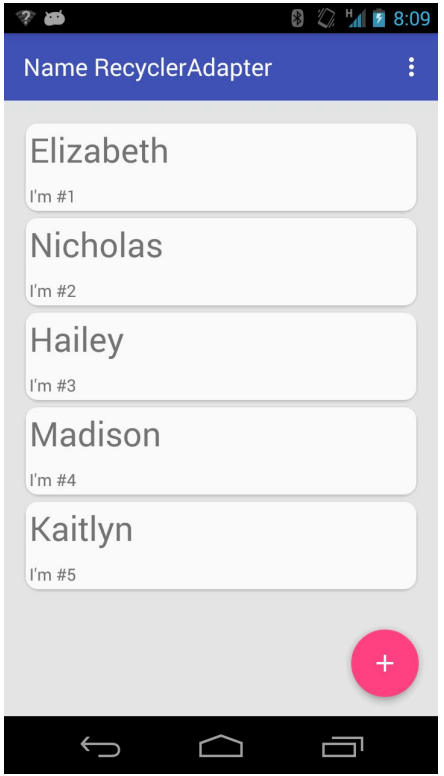


http://developer.android.com/guide/topics/ui/declaring-layout.html#AdapterViews

https://pixabay.com/static/uploads/photo/2013/07/12/15/01/air-pollution-149241_640.png

RecyclerView and Adapter

In this lesson you will learn how to use a RecyclerView and RecyclerAdapter



RecyclerViews have several improvements over ListViews

+ **Flexible**: 3 layout managers

+ **Efficient**: minimizes use of findViewById(): capture once, bind to data often; reported 15% faster

+ **Nice looking**: built-in animations for CRUD on lists, for example, notify**Item**Inserted(pos)

https://developer.android.com/training/improving-layouts/smooth-scrolling.html

Anatomy of a RecyclerAdapter: getView's 3 functions are split up

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

    class ViewHolder extends RecyclerView.ViewHolder {
        private TextView mNameTextView;
        private TextView mPositionTextView;

        public ViewHolder(View itemView) {
            super(itemView);
            mNameTextView = (TextView)itemView.findViewById(R.id.name_view);
            mPositionTextView = (TextView)itemView.findViewById(R.id.position_view);
        }
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext()).inflate(R.layout.name_view, parent, false);
        return new ViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        String name = mNames.get(position);
        holder.mNameTextView.setText(name);
        holder.mPositionTextView.setText(String.format("I'm #%d",(position+1)));
    }
```

2. ViewHolder captures

1. onCreateVH inflates

3. onBindVH pop's with data

Converting a BaseAdapter to a RecyclerAdapter

NameAdapter now extends RecyclerAdapter<NameAdapter.ViewHolder>

- Add the required methods

Create inner ViewHolder class that extends RecyclerAdapter.ViewHolder

Split up getView() into 3 smaller functions:

- createViewHolder() inflates the view
- ViewHolder constructor captures views
- bindViewHolder() assigns values

Changes to layout and gradle file

In content_main.xml, change ListView to a RecyclerView:

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Include recyclerview as a dependency in your module's build.gradle

(The version numbers change rapidly, so you'll need to edit!)

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'
    compile 'com.android.support:recyclerview-v7:23.1.1'
}
```

Changes to MainActivity

Change ListView to RecyclerView.

setLayoutManager() is new

setHasFixedSize() is for further efficiency

```
public class MainActivity extends AppCompatActivity {
    private NameAdapter mAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener((view) → { mAdapter.addName(); });

        RecyclerView recyclerView = (RecyclerView)findViewById(R.id.recycler_view);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setHasFixedSize(true);
        mAdapter = new NameAdapter(this);
        recyclerView.setAdapter(mAdapter);
    }
```

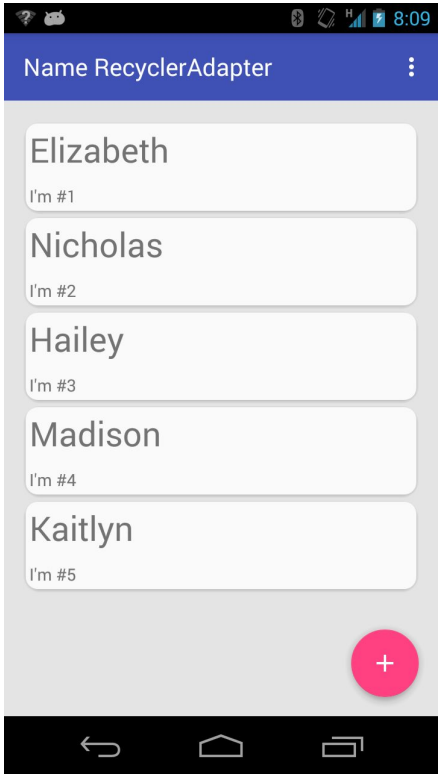
The ViewHolder implements LongClickListener

```
class ViewHolder extends RecyclerView.ViewHolder implements View.OnLongClickListener {
    private TextView mNameTextView;
    private TextView mPositionTextView;

    public ViewHolder(View itemView) {
        super(itemView);
        itemView.setOnLongClickListener(this);
        mNameTextView = (TextView) itemView.findViewById(R.id.name_view);
        mPositionTextView = (TextView) itemView.findViewById(R.id.position_view);
    }

    @Override
    public boolean onLongClick(View v) {
        deleteName(getAdapterPosition());
        return true;
    }
}
```

Clean design, since natural for views to listen for their own clicks

Taking advantage of RecyclerViews

In this lesson, you'll learn about simple animations and GridLayoutManagers and get a preview of CardViews



Add animations to insert/delete

So easy!

Try it out now.

Couple “glitches”:

1. Numbers are messed up.

   To fix, add

   ```
   notifyItemRangeChanged(position, mNames.size());
   ```

   after deleting the item

2. Adding to top once the screen is full doesn’t re-position it at the top. Hint: the recyclerView’s layout manager has a scrollToPosition() method

```
public void addName() {
    mNames.add(0, getRandomName());
    notifyItemInserted(0);
}

public void deleteName(int position) {
    mNames.remove(position);
    notifyItemRemoved(position);
}
```

This calls onBindViewHolder() for the views in this range to update the data. Only call it in your app if needed.

http://stackoverflow.com/questions/26076965/android-recyclerview-addition-removal-of-items

Another hint on scrollTo():

```
public void addName() {
    mNames.add(0, getRandomName());
    notifyItemInserted(0);
    mRecyclerView.getLayoutManager().scrollToPosition(0);
}
```

The activity needs to pass in the recyclerview to the adapter when you construct it.

UI upgrade: CardViews

You can wrap your row view layout in a CardView.

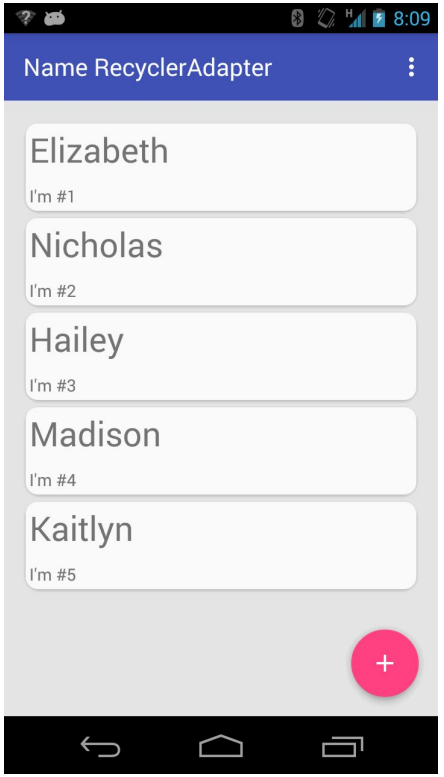
http://developer.android.com/training/material/lists-cards.html#CardView

cardCornerRadius of 10dp is obvious

Pro-tip: if you only get 1 card per screen (super-tall), go make sure that your CardView's height is set to wrap_content. (It wouldn't hurt to have your LinearLayout's height also be wrap_content)

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

More details in lab.



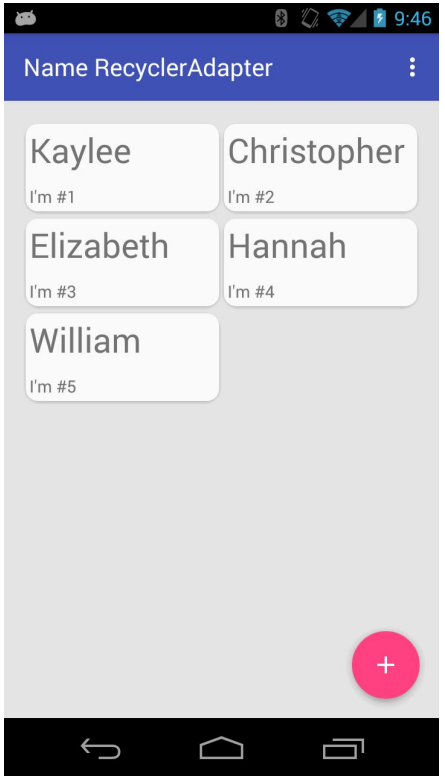
Other Layout Managers

Change LinearLayoutManager(this) to **GridLayoutManager(this, 2)**

What's the 2 for?

StaggeredGridLayoutManager is for multiple columns with rows of varying widths:

http://stackoverflow.com/questions/31667785/recyclerview-with-gridlayoutmanager



BaseAdapter vs RecyclerAdapter summary

BaseAdapter	RecyclerAdapter
getCount() →	getItemCount()
getView() →	ViewHolder, onCreateVH(), onBindVH()
getItem(), getItemId() →	Not needed
onItemLongClick() →	VH's onLongClick()

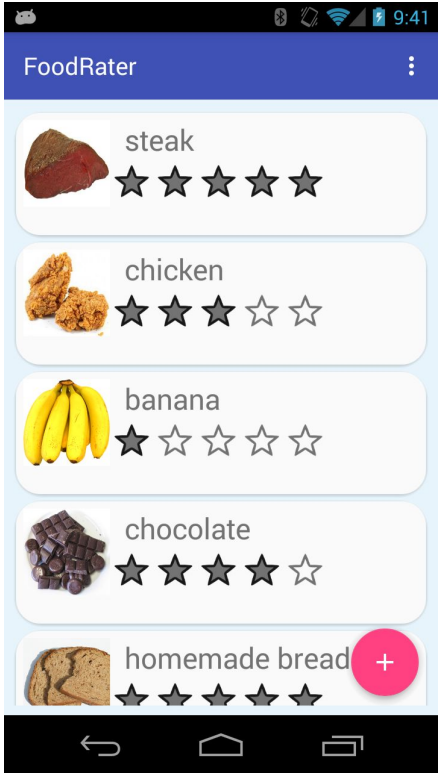
Lab: FoodRater

Your turn!

More realistic model object

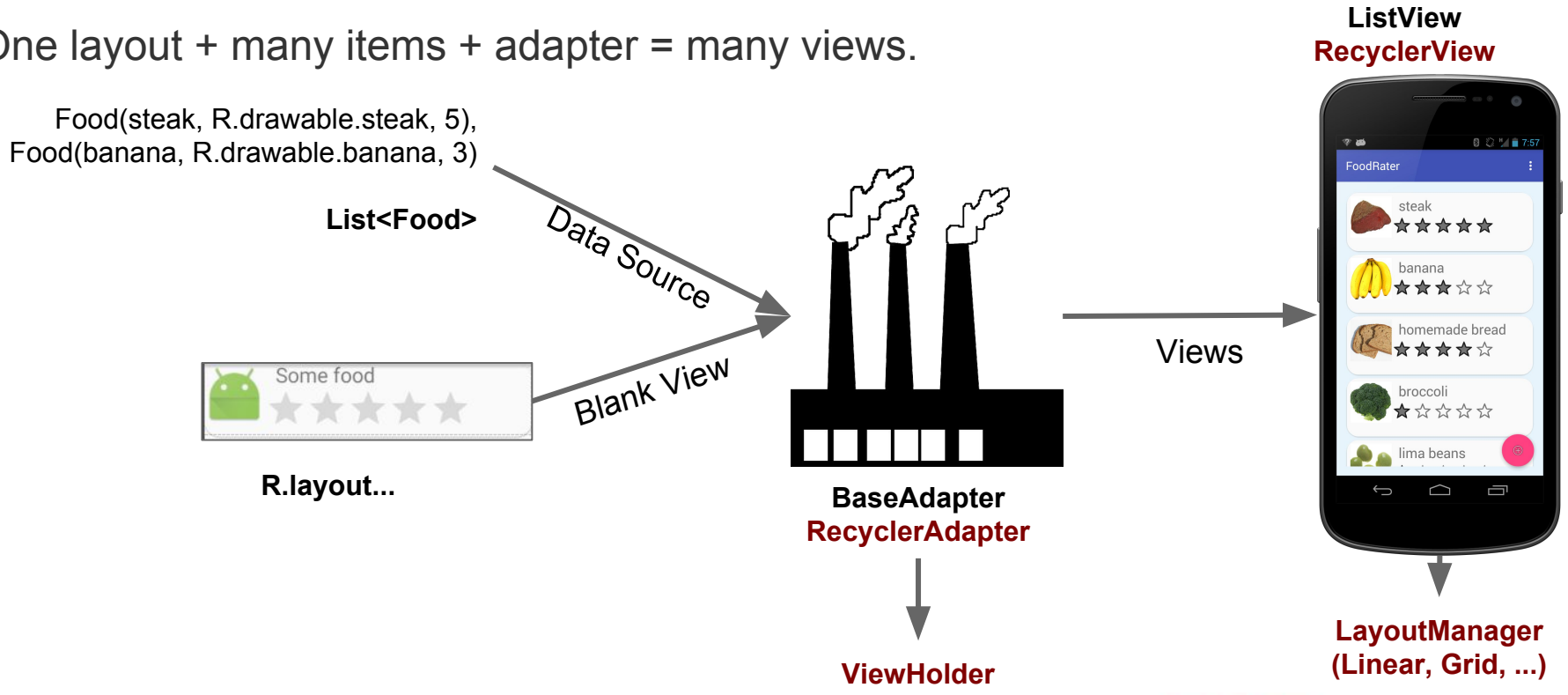
RatingBar and images

CardView



Summary: adapters replicate layouts for a data source

One layout + many items + adapter = many views.



Next steps for further study

1. The lab has CardViews
2. Explore other widgets that use adapters:
   - http://developer.android.com/guide/topics/ui/controls/spinner.html
   - http://developer.android.com/guide/topics/ui/controls/text.html for autocomplete, which works similarly
3. Create an app using adapters with menus and dialogs:
   - Next exam