File - D:\Android apps\MasterDetailFlow\app\src\main\java\mackenzieapps\nathan\masterdetailflow\model\Converters.java

```java
package mackenzieapps.nathan.masterdetailflow.model;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Helper class for providing sample content for user interfaces created by
 * Android template wizards.
 * <p>
 */
public class Converters {

    public interface Lambda {
        double conversion(double value);
    }
    /**
     * An array of conversion object items.
     * labels and lambda expressions
     */
    public static final List<ConverterItem> ITEMS = new ArrayList<>();

    /**
     * A map of conversion items, by name.
     */
    public static final Map<String, ConverterItem> ITEM_MAP = new HashMap<>();

    /**
     * This pupulates the list view
     */
    static {
        addItem(createConverterItem("Area", "A->H", "H->A", (a) -> a * 0.404686, (h) -> h * 2.47105));
        addItem(createConverterItem("Length", "ft > m", "m > ft", (f) -> f * 0.3048, (m) -> m * 3.28084));
        addItem(createConverterItem("Temperature", "F > C", "C > F", (f) -> (f - 32.0) * 5.0 / 9.0, (c) -> c * 9.0 / 5.0 + 32.0));
        addItem(createConverterItem("Weight", "lbs > kg", "kg > lbs", (l) -> l * 0.453592, (k) -> k * 2.20462));
    }
```

```
38
39      private static void addItem(ConverterItem item) {
40          ITEMS.add(new ConverterItem(item.conversionName,
    item.leftConversionLabel, item.rightConversionLabel, item.
    leftConversion, item.rightConversion));
41          ITEM_MAP.put(item.conversionName, item);
42      }
43
44      /**
45       * for each item in the list view this populates the
    details view
46       */
47      private static ConverterItem createConverterItem(
    String conversionName, String leftConversionLabel, String
    rightConversionLabel, Lambda leftConversion, Lambda
    rightConversion) {
48          return new ConverterItem(conversionName,
    leftConversionLabel, rightConversionLabel, leftConversion,
     rightConversion);
49      }
50
51      /**
52       * A converter item representing a piece of content.
53       */
54      public static class ConverterItem {
55          public final String conversionName;
56          public final String leftConversionLabel;
57          public final String rightConversionLabel;
58          public Lambda leftConversion;
59          public Lambda rightConversion;
60
61          public ConverterItem(String conversionName, String
     leftConversionLabel, String rightConversionLabel, Lambda
    leftConversion, Lambda rightConversion) {
62              this.conversionName = conversionName;
63              this.leftConversionLabel = leftConversionLabel
    ;
64              this.rightConversionLabel =
    rightConversionLabel;
65              this.leftConversion = leftConversion;
66              this.rightConversion = rightConversion;
67          }
68
69          @Override
70          public String toString() {
```

```
71                return conversionName;
72            }
73        }
74    }
75
```

```java
package mackenzieapps.nathan.masterdetailflow;

import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.app.ActionBar;
import android.view.MenuItem;

/**
 * An activity representing a single Item detail screen. This
 * activity is only used on narrow width devices. On tablet-size devices,
 * item details are presented side-by-side with a list of items
 * in a {@link ItemListActivity}.
 */
public class ItemDetailActivity extends AppCompatActivity {

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

        // Show the Up button in the action bar.
        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) {
            actionBar.setDisplayHomeAsUpEnabled(true);
        }

        // savedInstanceState is non-null when there is fragment state
        // saved from previous configurations of this activity
        // (e.g. when rotating the screen from portrait to landscape).
        // In this case, the fragment will automatically
```

```
37 be re-added
38         // to its container so we don't need to manually
   add it.
39         // For more information, see the Fragments API
   guide at:
40         //
41         // http://developer.android.com/guide/components/
   fragments.html
42         //
43         if (savedInstanceState == null) {
44             // Create the detail fragment and add it to
   the activity
45             // using a fragment transaction.
46             Bundle arguments = new Bundle();
47             arguments.putString(ItemDetailFragment.
   ARG_ITEM_ID,
48                     getIntent().getStringExtra(
   ItemDetailFragment.ARG_ITEM_ID));
49             ItemDetailFragment fragment = new
   ItemDetailFragment();
50             fragment.setArguments(arguments);
51             getSupportFragmentManager().beginTransaction()
52                     .add(R.id.item_detail_container,
   fragment)
53                     .commit();
54         }
55     }
56
57     @Override
58     public boolean onOptionsItemSelected(MenuItem item) {
59         int id = item.getItemId();
60         if (id == android.R.id.home) {
61             // This ID represents the Home or Up button.
   In the case of this
62             // activity, the Up button is shown. For
63             // more details, see the Navigation pattern on
    Android Design:
64             //
65             // http://developer.android.com/design/
   patterns/navigation.html#up-vs-back
66             //
67             navigateUpTo(new Intent(this, ItemListActivity
   .class));
68             return true;
69         }
```

```
70            return super.onOptionsItemSelected(item);
71        }
72    }
73
```

```java
package mackenzieapps.nathan.masterdetailflow;

import android.app.Activity;
import android.support.design.widget.CollapsingToolbarLayout;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;

import mackenzieapps.nathan.masterdetailflow.model.Converters;

import static mackenzieapps.nathan.masterdetailflow.model.Converters.ITEMS;

/**
 * A fragment representing a single Item detail screen.
 * This fragment is either contained in a {@link ItemListActivity}
 * in two-pane mode (on tablets) or a {@link ItemDetailActivity}
 * on handsets.
 */
public class ItemDetailFragment extends Fragment {
    /**
     * The fragment argument representing the item ID that this fragment
     * represents.
     */
    public static final String ARG_ITEM_ID = "item_id";
    private Converters.ConverterItem mItem;

    /**
     * Mandatory empty constructor for the fragment manager to instantiate the
     * fragment (e.g. upon screen orientation changes).
     */
    public ItemDetailFragment() {
    }

    Button leftButton;
```

```
39      Button rightButton;
40      EditText value;
41      @Override
42      public void onCreate(Bundle savedInstanceState) {
43          super.onCreate(savedInstanceState);
44
45          if (getArguments().containsKey(ARG_ITEM_ID)) {
46              // Load the dummy content specified by the
    fragment
47              // arguments. In a real-world scenario, use a
    Loader
48              // to load content from a content provider.
49              mItem = Converters.ITEM_MAP.get(getArguments()
    .getString(ARG_ITEM_ID));
50
51              Activity activity = this.getActivity();
52              CollapsingToolbarLayout appBarLayout = (
    CollapsingToolbarLayout) activity.findViewById(R.id.
    toolbar_layout);
53              if (appBarLayout != null) {
54                  appBarLayout.setTitle(mItem.conversionName
    );
55              }
56          }
57      }
58
59      @Override
60      public View onCreateView(LayoutInflater inflater,
    ViewGroup container,
61                               Bundle savedInstanceState) {
62
63          View rootView = inflater.inflate(R.layout.
    item_detail, container, false);
64
65          // Show the converter item content as text in a
    TextView.
66          if (mItem != null) {
67              leftButton = rootView.findViewById(R.id.button
    );
68              leftButton.setText(mItem.leftConversionLabel);
69              rightButton = rootView.findViewById(R.id.
    button2);
70              rightButton.setText(mItem.rightConversionLabel
    );
71              value =  rootView.findViewById(R.id.valueField
```

```
71 );
72             leftButton.setOnClickListener(v -> leftButton
   ());
73             rightButton.setOnClickListener(v ->
   rightButton());
74         }
75         return rootView;
76     }
77
78     /**
79      * method for left button interaction will check if a
    value can be read from the input field
80      * if unable to will default text to N/A
81      */
82     public void leftButton(){
83         try {
84             Object o = value.getText();
85             String tempValue = o.toString();
86             value.setText(String.valueOf(mItem.
   leftConversion.conversion(Double.parseDouble(tempValue)))
   );
87         } catch (Exception err) {
88             value.setText("N/A");
89         }
90     }
91
92     /**
93      * method for right button interaction will check if
   a value can be read from the input field
94      * if unable to will default text to N/A
95      */
96     public void rightButton(){
97      try {
98         Object o = value.getText();
99         String tempValue = o.toString();
100        value.setText(String.valueOf(mItem.
   rightConversion.conversion(Double.parseDouble(tempValue))
   ));
101    } catch (Exception err) {
102        value.setText("N/A");
103    }
104    }
105 }
106
```

```java
package mackenzieapps.nathan.masterdetailflow;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;


import java.util.List;

import mackenzieapps.nathan.masterdetailflow.model.Converters;

/**
 * An activity representing a list of Items. This activity
 * has different presentations for handset and tablet-size devices. On
 * handsets, the activity presents a list of items, which when touched,
 * lead to a {@link ItemDetailActivity} representing
 * item details. On tablets, the activity presents the list of items and
 * item details side-by-side using two vertical panes.
 */
public class ItemListActivity extends AppCompatActivity {

    /**
     * Whether or not the activity is in two-pane mode, i.e. running on a tablet
     * device.
     */
    private boolean mTwoPane;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
41          setContentView(R.layout.activity_item_list);
42
43          Toolbar toolbar = (Toolbar) findViewById(R.id.
    toolbar);
44          setSupportActionBar(toolbar);
45          toolbar.setTitle(getTitle());
46
47          if (findViewById(R.id.item_detail_container) !=
    null) {
48              // The detail container view will be present
    only in the
49              // large-screen layouts (res/values-w900dp).
50              // If this view is present, then the
51              // activity should be in two-pane mode.
52              mTwoPane = true;
53          }
54
55          View recyclerView = findViewById(R.id.item_list);
56          assert recyclerView != null;
57          setupRecyclerView((RecyclerView) recyclerView);
58      }
59
60      private void setupRecyclerView(@NonNull RecyclerView
    recyclerView) {
61          recyclerView.setAdapter(new
    SimpleItemRecyclerViewAdapter(this, Converters.ITEMS,
    mTwoPane));
62      }
63
64      public static class SimpleItemRecyclerViewAdapter
65              extends RecyclerView.Adapter<
    SimpleItemRecyclerViewAdapter.ViewHolder> {
66
67          private final ItemListActivity mParentActivity;
68          private final List<Converters.ConverterItem>
    mValues;
69          private final boolean mTwoPane;
70          private final View.OnClickListener
    mOnClickListener = new View.OnClickListener() {
71              @Override
72              public void onClick(View view) {
73                  Converters.ConverterItem item = (
    Converters.ConverterItem) view.getTag();
74                  if (mTwoPane) {
75                      Bundle arguments = new Bundle();
```

```
 76                      arguments.putString(
    ItemDetailFragment.ARG_ITEM_ID, item.conversionName);
 77                      ItemDetailFragment fragment = new
    ItemDetailFragment();
 78                      fragment.setArguments(arguments);
 79                      mParentActivity.
    getSupportFragmentManager().beginTransaction()
 80                              .replace(R.id.
    item_detail_container, fragment)
 81                              .commit();
 82                  } else {
 83                      Context context = view.getContext();
 84                      Intent intent = new Intent(context,
    ItemDetailActivity.class);
 85                      intent.putExtra(ItemDetailFragment.
    ARG_ITEM_ID, item.conversionName);
 86
 87                      context.startActivity(intent);
 88                  }
 89              }
 90          };
 91
 92          SimpleItemRecyclerViewAdapter(ItemListActivity
    parent,
 93                                        List<Converters.
    ConverterItem> items,
 94                                        boolean twoPane) {
 95              mValues = items;
 96              mParentActivity = parent;
 97              mTwoPane = twoPane;
 98          }
 99
100          @Override
101          public ViewHolder onCreateViewHolder(ViewGroup
    parent, int viewType) {
102              View view = LayoutInflater.from(parent.
    getContext())
103                      .inflate(R.layout.item_list_content,
    parent, false);
104              return new ViewHolder(view);
105          }
106
107          @Override
108          public void onBindViewHolder(final ViewHolder
    holder, int position) {
```

```
109             holder.mIdView.setText(mValues.get(position).
    conversionName);
110
111             holder.itemView.setTag(mValues.get(position))
    ;
112             holder.itemView.setOnClickListener(
    mOnClickListener);
113         }
114
115         @Override
116         public int getItemCount() {
117             return mValues.size();
118         }
119
120         class ViewHolder extends RecyclerView.ViewHolder
    {
121             final TextView mIdView;
122             final TextView mContentView;
123
124             ViewHolder(View view) {
125                 super(view);
126                 mIdView = (TextView) view.findViewById(R.
    id.id_text);
127                 mContentView = (TextView) view.
    findViewById(R.id.content);
128             }
129         }
130     }
131 }
132
```