```java
package ca.camosun.masterdetailconverter;

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.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import java.util.List;

import ca.camosun.masterdetailconverter.conversion.Conversion;
import ca.camosun.masterdetailconverter.conversion.ConversionContent;

/**
 * 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;


    // Setup the activity from a savedInstanceState
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_item_list);

        // Set the title
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        toolbar.setTitle(getTitle());

        if (findViewById(R.id.item_detail_container) != null) {
            // The detail container view will be present only in the
            // large-screen layouts (res/values-w900dp).
            // If this view is present, then the
            // activity should be in two-pane mode.
            mTwoPane = true;
        }

        View recyclerView = findViewById(R.id.item_list);
        assert recyclerView != null;
        setupRecyclerView((RecyclerView) recyclerView);
    }

    // setup the recycler view
    private void setupRecyclerView(@NonNull RecyclerView recyclerView) {
        recyclerView.setAdapter(new SimpleItemRecyclerViewAdapter(this, ConversionContent.ITEMS, mTwoPane));
    }

    // A class containing a RecyclerView for displaying items
    public static class SimpleItemRecyclerViewAdapter
            extends RecyclerView.Adapter<SimpleItemRecyclerViewAdapter.ViewHolder> {

        private final ItemListActivity mParentActivity;
        private final List<Conversion> mValues;
        private final boolean mTwoPane;
        private final View.OnClickListener mOnClickListener = new View.OnClickListener() {
            // When item is selected, pass the item's id to the detail screen so it can load the
            // appropriate item details
            @Override
            public void onClick(View view) {
                Conversion item = (Conversion) view.getTag();
                if (mTwoPane) {
                    Bundle arguments = new Bundle();
                    arguments.putString(ItemDetailFragment.ARG_ITEM_ID, Integer.toString(item.getId()));
                    ItemDetailFragment fragment = new ItemDetailFragment();
                    fragment.setArguments(arguments);
                    mParentActivity.getSupportFragmentManager().beginTransaction()
                            .replace(R.id.item_detail_container, fragment)
                            .commit();
                } else {
                    Context context = view.getContext();
                    Intent intent = new Intent(context, ItemDetailActivity.class);
                    intent.putExtra(ItemDetailFragment.ARG_ITEM_ID, Integer.toString(item.getId()));

                    context.startActivity(intent);
                }
            }
        };

        // Constructor for a SimpleItemRecyclerViewAdapter
        SimpleItemRecyclerViewAdapter(ItemListActivity parent,
                                      List<Conversion> items,
                                      boolean twoPane) {
            mValues = items;
            mParentActivity = parent;
            mTwoPane = twoPane;
        }

        // Creates the view for each
        @Override
        public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
            View view = LayoutInflater.from(parent.getContext())
                    .inflate(R.layout.item_list_content, parent, false);
            return new ViewHolder(view);
        }

        // Sets the appropriate list item's id and name
        @Override
        public void onBindViewHolder(final ViewHolder holder, int position) {
            holder.mIdView.setText(Integer.toString(mValues.get(position).getId()));
            holder.mContentView.setText(mValues.get(position).getName());

            holder.itemView.setTag(mValues.get(position));
            holder.itemView.setOnClickListener(mOnClickListener);
        }

        // The count of Conversion items in the list
        @Override
        public int getItemCount() {
            return mValues.size();
        }

        // The view for each item in the list
        class ViewHolder extends RecyclerView.ViewHolder {
            final TextView mIdView;
            final TextView mContentView;

            ViewHolder(View view) {
                super(view);
                mIdView = (TextView) view.findViewById(R.id.id_text);
                mContentView = (TextView) view.findViewById(R.id.content);
            }
        }
    }
}

```

```
package ca.camosun.masterdetailconverter;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
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 be re-added
        // to its container so we don't need to manually add it.
        // For more information, see the Fragments API guide at:
        //
        // http://developer.android.com/guide/components/fragments.
html
        //
        if (savedInstanceState == null) {
            // Create the detail fragment and add it to the activity
            // using a fragment transaction.
            Bundle arguments = new Bundle();
            arguments.putString(ItemDetailFragment.ARG_ITEM_ID,
                    getIntent().getStringExtra(ItemDetailFragment.
ARG_ITEM_ID));
            ItemDetailFragment fragment = new ItemDetailFragment();
            fragment.setArguments(arguments);
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.item_detail_container, fragment)
                    .commit();
        }
    }

    // This method is called when the user wants to return to the
homepage from the detail page
    // (They click the 'back' button)
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
```

```java
        int id = item.getItemId();
        if (id == android.R.id.home) {
            // This ID represents the Home or Up button. In the case of this
            // activity, the Up button is shown. For
            // more details, see the Navigation pattern on Android Design:
            //
            // http://developer.android.com/design/patterns/navigation.html#up-vs-back
            //
            navigateUpTo(new Intent(this, ItemListActivity.class));
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}

```

```java
package ca.camosun.masterdetailconverter;

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

import ca.camosun.masterdetailconverter.conversion.Conversion;
import ca.camosun.masterdetailconverter.conversion.ConversionContent;
import ca.camosun.masterdetailconverter.conversion.PerformsConversion;

/**
 * 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";

    /**
     * The Conversion content this fragment is presenting.
     */
    private Conversion mItem;

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

    // Setup the current activity from a savedInstanceState
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (getArguments().containsKey(ARG_ITEM_ID)) {
            // Load the Conversion content specified by the fragment arguments.
            mItem = ConversionContent.ITEM_MAP.get(getArguments().getString(ARG_ITEM_ID));

            Activity activity = this.getActivity();
            CollapsingToolbarLayout appBarLayout = (CollapsingToolbarLayout) activity.findViewById(R.id.toolbar_layout);
            if (appBarLayout != null) {
                appBarLayout.setTitle(mItem.getName());
            }
        }
    }

    // Setup the activity as a new view.  This handles first time navigation to this page.
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.item_detail, container, false);

        // Show the Conversion content as text in a TextView.
        if (mItem != null) {
            // setup reference to userInputValue
            EditText userValueField = rootView.findViewById(R.id.userInputValue);

            // setup the conversion buttons with their name, listener and action
            Button leftButton = ((Button) rootView.findViewById(R.id.leftButton));
            leftButton.setText(mItem.getLeftButton().getName());
            leftButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    leftButton(view);
                }
            });

            Button rightButton = ((Button) rootView.findViewById(R.id.rightButton));
            rightButton.setText(mItem.getRightButton().getName());
            rightButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    rightButton(view);
                }
            });
        }

        return rootView;
    }

    // Called when the left button was clicked.  Converts using the currently selected conversion
    // from the conversion spinner.
    public void leftButton(View view){
        if(mItem == null){
            // selected conversion was never set properly
            // log and abort
            Log.e("leftButton", "selectedConversion was not set properly");
            return;
        }

        // Call helper method
        convertValue(mItem.getLeftButton().getAction());
    }

    // Called when the right button was clicked.  Converts using the currently selected conversion
    // from the conversion spinner.
    public void rightButton(View view){
        if(mItem == null){
            // selected conversion was never set properly
            // log and abort
            Log.e("rightButton", "selectedConversion was not set properly");
            return;
        }

        // Call helper method
        convertValue(mItem.getRightButton().getAction());
    }

    // Converts the user value using the passed action.  The action is a lambda expression as per
    // the PerformsConversion interface.
    private void convertValue(PerformsConversion action){
        // Grab the user variable
        EditText converterField = (EditText) this.getActivity().findViewById(R.id.userInputValue);

        try {
            // Throws if Null or no Double found
            double temp = Double.parseDouble(converterField.getText().toString());

            // Convert the user variable and output the result
            double convertedTemp = action.convert(temp);
            converterField.setText(Double.toString(convertedTemp));
        } catch (NullPointerException|NumberFormatException ex){
            // Failed to convert to a double - the value either contained no double or was empty
            converterField.setText("N/A");
        }
    }
}

```