```java
package ca.camosun.androidmoviedatabase;

import android.content.Context;
import android.content.Intent;
import android.content.res.Resources;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.content.res.ResourcesCompat;
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.ImageView;
import android.widget.RatingBar;
import android.widget.TextView;

import java.util.List;

import ca.camosun.androidmoviedatabase.movie.Movie;
import ca.camosun.androidmoviedatabase.movie.MovieContent;

/**
 * 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;
        }
```

```
 59
 60          View recyclerView = findViewById(R.id.item_list);
 61          assert recyclerView != null;
 62          setupRecyclerView((RecyclerView) recyclerView);
 63      }
 64
 65      // setup the recycler view
 66      private void setupRecyclerView(@NonNull RecyclerView recyclerView
     ) {
 67          recyclerView.setAdapter(new SimpleItemRecyclerViewAdapter(
     this, MovieContent.ITEMS, mTwoPane));
 68      }
 69
 70      // A class containing a RecyclerView for displaying items
 71      public static class SimpleItemRecyclerViewAdapter
 72              extends RecyclerView.Adapter<
     SimpleItemRecyclerViewAdapter.ViewHolder> {
 73
 74          private final ItemListActivity mParentActivity;
 75          private final List<Movie> mValues;
 76          private final boolean mTwoPane;
 77          private final View.OnClickListener mOnClickListener = new
     View.OnClickListener() {
 78              // When item is selected, pass the item's id to the
     detail screen so it can load the
 79              // appropriate item details
 80              @Override
 81              public void onClick(View view) {
 82                  Movie item = (Movie) view.getTag();
 83                  if (mTwoPane) {
 84                      Bundle arguments = new Bundle();
 85                      arguments.putString(ItemDetailFragment.
     ARG_ITEM_ID, Integer.toString(item.getId()));
 86                      ItemDetailFragment fragment = new
     ItemDetailFragment();
 87                      fragment.setArguments(arguments);
 88                      mParentActivity.getSupportFragmentManager().
     beginTransaction()
 89                              .replace(R.id.item_detail_container,
     fragment)
 90                              .commit();
 91                  } else {
 92                      Context context = view.getContext();
 93                      Intent intent = new Intent(context,
     ItemDetailActivity.class);
 94                      intent.putExtra(ItemDetailFragment.ARG_ITEM_ID,
     Integer.toString(item.getId()));
 95
 96                      context.startActivity(intent);
 97                  }
 98              }
 99          };
100
101          // Constructor for a SimpleItemRecyclerViewAdapter
102          SimpleItemRecyclerViewAdapter(ItemListActivity parent,
103                                        List<Movie> items,
104                                        boolean twoPane) {
105              mValues = items;
106              mParentActivity = parent;
107              mTwoPane = twoPane;
108          }
```

```
109
110         // Creates the view for each
111         @Override
112         public ViewHolder onCreateViewHolder(ViewGroup parent, int
    viewType) {
113             View view = LayoutInflater.from(parent.getContext())
114                     .inflate(R.layout.item_list_content, parent,
    false);
115             return new ViewHolder(view);
116         }
117
118         // Sets the appropriate list item's id, name, and image
119         @Override
120         public void onBindViewHolder(final ViewHolder holder, int
    position) {
121             holder.mContentView.setText(mValues.get(position).getName
    ());
122             holder.mRatingBar.setRating(mValues.get(position).
    getRating());
123             Resources res = mParentActivity.getResources();
124             holder.mImageView.setImageDrawable(ResourcesCompat.
    getDrawable(res, mValues.get(position).getImageID(), null));
125             holder.itemView.setTag(mValues.get(position));
126             holder.itemView.setOnClickListener(mOnClickListener);
127         }
128
129         // The count of Movie items in the list
130         @Override
131         public int getItemCount() {
132             return mValues.size();
133         }
134
135         // The view for each item in the list
136         class ViewHolder extends RecyclerView.ViewHolder {
137             final TextView mContentView;
138             final RatingBar mRatingBar;
139             final ImageView mImageView;
140
141             ViewHolder(View view) {
142                 super(view);
143                 mContentView = (TextView) view.findViewById(R.id.
    content);
144                 mRatingBar = (RatingBar) view.findViewById(R.id.
    ratingBar);
145                 // disable buttons for this rating bar, display only
146                 mRatingBar.setEnabled(false);
147                 mImageView = (ImageView) view.findViewById(R.id.image
    );
148             }
149         }
150     }
151 }
152
```

```java
package ca.camosun.androidmoviedatabase;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
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) {
```

```
58          int id = item.getItemId();
59          if (id == android.R.id.home) {
60              // This ID represents the Home or Up button. In the case
    of this
61              // activity, the Up button is shown. For
62              // more details, see the Navigation pattern on Android
    Design:
63              //
64              // http://developer.android.com/design/patterns/
    navigation.html#up-vs-back
65              //
66              navigateUpTo(new Intent(this, ItemListActivity.class));
67              return true;
68          }
69          return super.onOptionsItemSelected(item);
70      }
71  }
72
```

```java
package ca.camosun.androidmoviedatabase;

import android.app.Activity;
import android.content.res.Resources;
import android.os.Bundle;
import android.support.design.widget.CollapsingToolbarLayout;
import android.support.v4.app.Fragment;
import android.support.v4.content.res.ResourcesCompat;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.RatingBar;
import android.widget.TextView;
import java.util.List;

import ca.camosun.androidmoviedatabase.movie.Genres;
import ca.camosun.androidmoviedatabase.movie.Movie;
import ca.camosun.androidmoviedatabase.movie.MovieContent;

/**
 * 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 Movie content this fragment is presenting.
     */
    private Movie mItem;

    /**
     * The Button for toggling if the current Movie is a favourite or not
     */
    private Button favButton;

    /**
     * The RatingBar for updating the user Movie rating for the current Movie
     */
    private RatingBar ratingBar;


    /**
     * For feedback when the user alters the current Movies rating or favourite status
     */
    private TextView notification;

    /**
```

```
58      * Mandatory empty constructor for the fragment manager to instantiate the
59      * fragment (e.g. upon screen orientation changes).
60      */
61     public ItemDetailFragment() {
62     }
63
64     // Setup the current activity from a savedInstanceState
65     @Override
66     public void onCreate(Bundle savedInstanceState) {
67         super.onCreate(savedInstanceState);
68
69         if (getArguments().containsKey(ARG_ITEM_ID)) {
70             // Load the Movie content specified by the fragment arguments.
71             mItem = MovieContent.ITEM_MAP.get(getArguments().getString(ARG_ITEM_ID));
72
73             Activity activity = this.getActivity();
74             CollapsingToolbarLayout appBarLayout = (CollapsingToolbarLayout) activity.findViewById(R.id.toolbar_layout);
75             if (appBarLayout != null) {
76                 appBarLayout.setTitle(mItem.getName());
77             }
78         }
79     }
80
81     // Setup the activity as a new view.  This handles first time navigation to this page.
82     @Override
83     public View onCreateView(LayoutInflater inflater, ViewGroup container,
84                              Bundle savedInstanceState) {
85         View rootView = inflater.inflate(R.layout.item_detail, container, false);
86
87         // Show the Movie content as text in a TextView.
88         if (mItem == null) {
89             //selected movie was not set correctly
90             Log.e("onCreateView", "movie was not set properly");
91         }
92
93         // setup the favourite button
94         favButton = ((Button) rootView.findViewById(R.id.favouriteButton));
95         favButton.setText(updateFavouriteButtonText());
96         favButton.setOnClickListener(new View.OnClickListener() {
97             @Override
98             public void onClick(View view) {
99                 favouriteButtonClicked(view);
100            }
101        });
102
103        // setup the RatingBar
104        ratingBar = (RatingBar) rootView.findViewById(R.id.ratingBarDetail);
105        ratingBar.setRating(mItem.getRating());
106        ratingBar.setOnRatingBarChangeListener(new RatingBar.OnRatingBarChangeListener(){
107            @Override
108            public void onRatingChanged(RatingBar ratingBar, float
```

```
108 val, boolean fromUser){
109                 updateRating(val);
110             }
111         });
112 
113         // setup genres
114         TextView genresLabel = (TextView) rootView.findViewById(R.id.
    genresLabel);
115         genresLabel.setText("Genres:");
116         TextView genresContent = (TextView) rootView.findViewById(R.
    id.genresContent);
117 
118         String genresString = "";
119         List<Genres> movieGenresList = mItem.getGenres();
120         for(int i = 0; i < movieGenresList.size(); i++){
121             genresString += movieGenresList.get(i);
122             if(i < movieGenresList.size() - 1){
123                 genresString += ", ";
124             }
125         }
126         genresContent.setText(genresString);
127 
128         //setup actors
129         TextView actorsLabel = (TextView) rootView.findViewById(R.id.
    actorsLabel);
130         actorsLabel.setText("Actors:");
131         TextView actorsContent = (TextView) rootView.findViewById(R.
    id.actorsContent);
132 
133         String actorsString = "";
134         List<String> actorsList = mItem.getActors();
135         for(int i = 0; i < actorsList.size(); i++){
136             actorsString += actorsList.get(i);
137             if(i < actorsList.size() - 1){
138                 actorsString += ", ";
139             }
140         }
141         actorsContent.setText(actorsString);
142 
143         //setup comments
144         TextView commentsLabel = (TextView) rootView.findViewById(R.
    id.commentsLabel);
145         commentsLabel.setText("Comments: ");
146         TextView commentsContent = (TextView) rootView.findViewById(R
    .id.commentsContent);
147         commentsContent.setText(mItem.getComments());
148 
149         //setup detail label
150         TextView detailLabel = (TextView) rootView.findViewById(R.id.
    detailsLabel);
151         detailLabel.setText("Details:");
152 
153         // grab a reference to the notification area and init to
    empty
154         notification = (TextView) rootView.findViewById(R.id.
    notificationArea);
155         notification.setText("");
156 
157         //setup image
158         ImageView movieImage = (ImageView) rootView.findViewById(R.id
    .movieImage);
```

```
159         Resources res = getResources();
160         movieImage.setImageDrawable(ResourcesCompat.getDrawable(res,
    mItem.getImageID(), null));
161 
162         return rootView;
163     }
164 
165     /**
166      * Called when the favourite button was clicked.  Adds or removes
     the current movie from the
167      * user's favourite list
168      * @param view View the current View context
169      */
170     public void favouriteButtonClicked(View view){
171         if(mItem == null){
172             // selected movie was never set properly
173             // log and abort
174             Log.e("favouriteButtonClicked", "movie was not set
    properly");
175             return;
176         }
177 
178         // Update movie
179         if(mItem.isFavourite){
180             mItem.isFavourite = false;
181             notification.setText("Movie Removed from Favourites!");
182         } else{
183             mItem.isFavourite = true;
184             notification.setText("Movie Added to Favourites!");
185         }
186 
187         // update button text
188         favButton.setText(updateFavouriteButtonText());
189     }
190 
191     /**
192      * Updates the text for the favourite button depending on the
    favourite context
193      */
194     public String updateFavouriteButtonText(){
195         if(mItem.isFavourite){
196             return "Remove from Favourites";
197         } else {
198             return "Add to Favourites";
199         }
200     }
201 
202     /**
203      * Updates the rating of the current Movie with the passed rating
204      * @param newRating float the new rating to set
205      */
206     public void updateRating(float newRating){
207         if(mItem == null){
208             // selected movie was never set properly
209             // log and abort
210             Log.e("updateRating", "movie was not set properly");
211             return;
212         }
213 
214         // update the rating
215         try {
```

```
216             mItem.setRating(newRating);
217
218             // update the notification area
219             notification.setText("Rating Updated!");
220         } catch (IllegalArgumentException e){
221             // rating was out of acceptable range, relay to user
222             Log.e("updateRating", e.getMessage());
223             notification.setText("Error updating rating");
224         }
225     }
226 }
227
```

```java
package ca.camosun.androidmoviedatabase.movie;

import java.util.ArrayList;
import java.util.List;

import ca.camosun.androidmoviedatabase.R;

/**
 * A class for storing details about a Movie with a unique id and additional details.
 */
public class Movie {
    // the unique id of the conversion
    private int id;
    // the name to display for the conversion
    private String name;
    // the genres for the movie
    private List<Genres> genres;
    // the actors in the movie
    private List<String> actors;
    // the user rating for the movie
    private float rating;
    // if the movie is a user favourite
    public boolean isFavourite;
    // additional comments for the movie
    private String comments;
    // the resourceID for the movie's image in the android package
    private int imageID;

    // tracks the next unique id for each new conversion
    // call generateID() to increment consistently
    private static int IDCOUNTER = 0;

    // returns the next unique id for each new conversion and increments static counter
    private int generateID(){
        return IDCOUNTER++;
    }

    // Initializes a Movie with all of its variables

    /**
     * Initializes a Movie instance with the passed parameters.  This constructor initializes all
     * class attributes for the Movie instance.
     * @param name String the name of the Movie
     * @param genres [Genres] all the genres the Movie belongs to
     * @param actors [String] all the actors that are in the Movie
     * @param rating int the rating for the Movie
     * @param isFavourite boolean if the Movie is a favourite or not
     * @param comments String additional comments about the Movie
     * @param imageResourceID int the Android identifier for the Drawable the Movie should use for
     *                        the Movie image
     * @throws IllegalArgumentException if rating is < 0 or > 5.0
     */
    public Movie(String name, List<Genres> genres, List<String> actors, int rating, boolean isFavourite, String comments, int imageResourceID) {
        this.id = generateID();
        this.name = name;
```

```
 56         this.genres = genres;
 57         this.actors = actors;
 58
 59         // use setRating to verify value is in range
 60         setRating(rating);
 61
 62         this.isFavourite = isFavourite;
 63         this.comments = comments;
 64         this.imageID = imageResourceID;
 65     }
 66
 67     /**
 68      * Initializes a Movie instance with the passed parameters.  This
     constructor uses the default
 69      * movie image.  Use the other Movie constructor if you wish to
    pass an image. The id for the
 70      * movie is automatically generated based on the next available
    id in the sequence.
 71      * @param name String the name of the Movie
 72      * @param genres [Genres] all the genres the Movie belongs to
 73      * @param actors [String] all the actors that are in the Movie
 74      * @param rating int the rating for the Movie
 75      * @param isFavourite boolean if the Movie is a favourite or not
 76      * @param comments String additional comments about the Movie
 77      * @throws IllegalArgumentException if rating is < 0 or > 5.0
 78      */
 79     public Movie(String name, List<Genres> genres, List<String>
    actors, int rating, boolean isFavourite, String comments){
 80         this(name, genres, actors, rating, isFavourite, comments, R.
    drawable.moviedefault);
 81     }
 82
 83     // public accessor for id
 84     public int getId(){
 85         return this.id;
 86     }
 87
 88     // public accessor for name
 89     public String getName(){
 90         return this.name;
 91     }
 92
 93     // public accessor for genres
 94     public List<Genres> getGenres(){ return this.genres; }
 95
 96     // public accessor for actors
 97     public List<String> getActors(){ return this.actors; }
 98
 99     //public accessor for comments
100     public String getComments(){ return this.comments; }
101
102     // public accessor for imageID
103     public int getImageID(){
104         return this.imageID;
105     }
106
107     // public accessor for rating
108     public float getRating() { return this.rating; }
109
110     /**
111      * Updates the Movie with the newRating after checking if the
```

```
111 value is in range
112      * @param newRating float the new rating to set.  newRating must
    be > 0 and <= 5.0
113      * @throws IllegalArgumentException if newRating is < 0 or > 5.0
114      */
115     public void setRating(float newRating){
116         if(newRating < 0 || newRating > 5.0){
117             throw new IllegalArgumentException("Unable to set rating
    , value was out of range. " +
118                     " newRating must be > 0 and <= 5.0");
119         }
120         this.rating = newRating;
121     }
122
123
124     /**
125      * This method generates a number of predefined Movies.
126      * @return ArrayList<Movie> the generated Movie objects
127      */
128     public static ArrayList<Movie> generateMovies(){
129
130         // Initialize all the movies
131         // Store all conversions together
132         ArrayList<Movie> movies = new ArrayList<>();
133
134         ArrayList<Genres> currentGenres = new ArrayList<>();
135         currentGenres.add(Genres.Action);
136         currentGenres.add(Genres.Comedy);
137
138         ArrayList<String> currentActors = new ArrayList<>();
139         currentActors.add("Jennifer");
140         currentActors.add("Bob");
141
142         movies.add(new Movie("testTitle", currentGenres,
    currentActors, 5, true, "This is a great movie!!!"));
143
144         currentGenres = new ArrayList<>();
145         currentGenres.add(Genres.Romance);
146
147         currentActors = new ArrayList<>();
148         currentActors.add("Leo");
149
150         movies.add(new Movie("Titanic", currentGenres, currentActors,
     3, false, "Very sad", R.drawable.titanic));
151
152         currentGenres = new ArrayList<>();
153         currentGenres.add(Genres.Action);
154
155         currentActors = new ArrayList<>();
156         currentActors.add("Tom Hanks");
157
158         movies.add(new Movie("Saving Private Ryan", currentGenres,
    currentActors, 5, true, "A classic", R.drawable.savingprivateryan));
159
160         currentGenres = new ArrayList<>();
161         currentGenres.add(Genres.SciFi);
162
163         currentActors = new ArrayList<>();
164         currentActors.add("Mark");
165
166         movies.add(new Movie("Star Wars Episode I", currentGenres,
```

```
166 currentActors, 4, false, "First prequel movie", R.drawable.starwarsi)
    );
167         movies.add(new Movie("Star Wars Episode II", currentGenres,
    currentActors, 4, false, "Second prequel movie", R.drawable.
    starwarsii));
168
169         currentGenres = new ArrayList<>();
170         currentGenres.add(Genres.SciFi);
171         currentGenres.add(Genres.Action);
172
173         currentActors = new ArrayList<>();
174         currentActors.add("Jane");
175         currentActors.add("Jon");
176         currentActors.add("Mary");
177
178         movies.add(new Movie("Avatar", currentGenres, currentActors,
    3, false, "This is a really long test comment.  This is to test that
    multi-line comments work.", R.drawable.avatar));
179
180         currentGenres = new ArrayList<>();
181         currentGenres.add(Genres.Comedy);
182
183         currentActors = new ArrayList<>();
184         currentActors.add("Geoff");
185         currentActors.add("Bob");
186
187         movies.add(new Movie("testTitle2", currentGenres,
    currentActors, 4, true, "This is a good movie"));
188
189         currentGenres = new ArrayList<>();
190         currentGenres.add(Genres.Documentary);
191
192         currentActors = new ArrayList<>();
193         currentActors.add("Jack");
194         currentActors.add("Bob");
195
196         movies.add(new Movie("testTitle3", currentGenres,
    currentActors, 2, true, "This is an ok movie"));
197
198         // Return the populated movie collection
199         return movies;
200     }
201 }
```

```java
package ca.camosun.androidmoviedatabase.movie;

/**
 * Enumeration for Movie Genres
 */
public enum Genres {
    Action, Comedy, Romance, Documentary, SciFi
}

```

```java
package ca.camosun.androidmoviedatabase.movie;

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

/**
 *  Stores all the Movie objects for the app
 */
public class MovieContent {

    /**
     * An array of all Movie items.
     */
    public static final List<Movie> ITEMS = Movie.generateMovies();

    /**
     * A map of all Movie items, by ID.
     */
    public static final Map<String, Movie> ITEM_MAP = new HashMap<String, Movie>();

    /**
     * The number of Movie items
     */
    private static final int COUNT = ITEMS.size();

    static {
        // Add all Movie ITEMS to the Map with their ID
        for (int i = 0; i < COUNT; i++) {
            ITEM_MAP.put(Integer.toString(ITEMS.get(i).getId()), ITEMS.get(i));
        }
    }
}

```