**import** android.os.Bundle;  
**import** android.support.v4.app.Fragment;  
**import** android.support.v4.app.FragmentManager;  
**import** android.support.v7.app.AppCompatActivity;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
*// Add FragmentManager for Fragment with RecycleView* FragmentManager fm = getSupportFragmentManager();  
 Fragment fragment = fm.findFragmentById(R.id.***fragmentContainer***);  
 **if** (fragment == **null**) {  
 fragment = **new** ListFragment();  
 fm.beginTransaction()  
 .add(R.id.***fragmentContainer***, fragment)  
 .commit();  
 }  
 }  
}

PopMovie **Main**

*<?***xml version="1.0" encoding="utf-8"***?>*<**FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id = "@+id/fragmentContainer"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"** />

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:baselineAligned="false"  
 android:orientation="horizontal"  
 android:weightSum="3"**>  
  
 <**fragment  
 android:id="@+id/fragmentContainer"  
 android:name="com.ex.popmovie.ListFragment"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"**/>  
  
 <**FrameLayout  
 android:id="@+id/fragment\_container"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="2"**/>  
</**LinearLayout**>

<**android.support.v7.widget.RecyclerView  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/rv\_list"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"** />

*<?***xml version="1.0" encoding="utf-8"***?>*<**FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"**>  
  
 <**ImageView  
 android:id="@+id/iv\_small\_poster"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="@dimen/iv\_small\_height"  
 android:adjustViewBounds="true"  
 android:contentDescription="@string/image\_description"  
 android:scaleType="centerCrop"** />  
</**FrameLayout**>

**package** com.ex.popmovie.data;  
  
**import** android.content.ContentProvider;  
**import** android.content.ContentValues;  
**import** android.content.UriMatcher;  
**import** android.database.Cursor;  
**import** android.database.SQLException;  
**import** android.database.sqlite.SQLiteQueryBuilder;   
**import** android.net.Uri;  
  
**import** com.ex.popmovie.R;  
**import** com.ex.popmovie.data.MovieContract.MovieTable;  
  
**public class** DbContentProvider **extends** ContentProvider {  
 **private** MovieDbHelper **dbHelper**;  
 **private static final** UriMatcher ***uriMatcher*** = **new** UriMatcher(UriMatcher.***NO\_MATCH***);  
 **private static final int *ONE\_MOVIE*** = 1;  
 **private static final int *MOVIES*** = 2;  
  
 **static** {  
 ***uriMatcher***.addURI(MovieContract.***AUTHORITY***, MovieTable.***TABLE\_NAME*** + **"/#"**, ***ONE\_MOVIE***);  
 ***uriMatcher***.addURI(MovieContract.***AUTHORITY***, MovieTable.***TABLE\_NAME***, ***MOVIES***);  
 }  
 **public** DbContentProvider() {  
 }  
 @Override  
 **public boolean** onCreate() {  
 *// Implement this to initialize your content provider on startup.* **dbHelper** = **new** MovieDbHelper(getContext());  
 **return true**;  
 }  
 @Override  
 **public** Cursor query(Uri uri, String[] projection, String selection,  
 String[] selectionArgs, String sortOrder) {  
 *// Implement this to handle query requests from clients,  
 // create SQLiteQueryBuilder for querying movies table* SQLiteQueryBuilder queryBuilder = **new** SQLiteQueryBuilder();  
 queryBuilder.setTables(MovieTable.***TABLE\_NAME***);  
  
 **switch** (***uriMatcher***.match(uri)) {  
 **case *ONE\_MOVIE***: *// movie with specified id will be selected* queryBuilder.appendWhere(  
 MovieTable.***\_ID*** + **"="** + uri.getLastPathSegment());  
 **break**;  
 **case *MOVIES***: *// all movies will be selected* **break**;  
 **default**:  
 **throw new** UnsupportedOperationException(  
 getContext().getString(R.string.***invalid\_query\_uri***) + uri);  
 }  
 *// execute the query to select one or all movies* Cursor cursor = queryBuilder.query(**dbHelper**.getWritableDatabase(),  
 projection, selection, selectionArgs, **null**, **null**, sortOrder);  
 *// configure to watch for content changes* cursor.setNotificationUri(getContext().getContentResolver(), uri);  
 **return** cursor;  
 }  
 @Override  
 **public** Uri insert(Uri uri, ContentValues values) {  
 *// Implement this to handle requests to insert a new row.* Uri newMovieUri = **null**;  
  
 **switch** (***uriMatcher***.match(uri)) {  
 **case *MOVIES***:  
 *// insert the new movie--success yields new movie's row id* **long** rowId = **dbHelper**.getWritableDatabase().insert(  
 MovieTable.***TABLE\_NAME***, **null**, values);  
 *// if the movie was inserted, create an appropriate Uri;  
 // otherwise, throw an exception* **if** (rowId > 0) { *// SQLite row IDs start at 1* newMovieUri = MovieTable.*buildMovieUri*(rowId);  
 *// notify observers that the database changed* getContext().getContentResolver().notifyChange(uri, **null**);  
 }  
 **else  
 throw new** SQLException(  
 getContext().getString(R.string.***insert\_failed***) + uri);  
 **break**;  
 **default**:  
 **throw new** UnsupportedOperationException(  
 getContext().getString(R.string.***invalid\_insert\_uri***) + uri);  
 }  
 **return** newMovieUri;  
 }  
 @Override  
 **public int** update(Uri uri, ContentValues values, String selection,  
 String[] selectionArgs) {  
 *// Implement this to handle requests to update one or more rows.* **int** numberOfRowsUpdated; *// 1 if update successful; 0 otherwise* **switch** (***uriMatcher***.match(uri)) {  
 **case *ONE\_MOVIE***:  
 *// get from the uri the id of movie to update* String id = uri.getLastPathSegment();  
 *// update the row* numberOfRowsUpdated = **dbHelper**.getWritableDatabase().update(  
 MovieTable.***TABLE\_NAME***, values, MovieTable.***\_ID*** + **"="** + id,  
 selectionArgs);  
 **break**;  
 **default**:  
 **throw new** UnsupportedOperationException(  
 getContext().getString(R.string.***invalid\_update\_uri***) + uri);  
 }  
 *// if changes were made, notify observers that the database changed* **if** (numberOfRowsUpdated != 0) {  
 getContext().getContentResolver().notifyChange(uri, **null**);  
 }  
 **return** numberOfRowsUpdated;  
 }  
 @Override  
 **public int** delete(Uri uri, String selection, String[] selectionArgs) {  
 *// Implement this to handle requests to delete one or more rows.* **int** numberOfRowsDeleted;  
  
 **switch** (***uriMatcher***.match(uri)) {  
 **case *ONE\_MOVIE***:  
 *// get from the uri the id of contact to update* String id = uri.getLastPathSegment();  
 *// delete the contact* numberOfRowsDeleted = **dbHelper**.getWritableDatabase().delete(  
 MovieTable.***TABLE\_NAME***, MovieTable.***\_ID*** + **"="** + id, selectionArgs);  
 **break**;  
 **default**:  
 **throw new** UnsupportedOperationException(  
 getContext().getString(R.string.***invalid\_delete\_uri***) + uri);  
 }  
 *// notify observers that the database changed* **if** (numberOfRowsDeleted != 0) {  
 getContext().getContentResolver().notifyChange(uri, **null**);  
 }  
 **return** numberOfRowsDeleted;  
 }  
 @Override  
 **public** String getType(Uri uri) {  
 *// Implement this to handle requests for the MIME type of the data at the given URI.  
// throw new UnsupportedOperationException("Not yet implemented");* **return null**;  
 }  
}

PopMovie **01**

**import** android.content.ContentUris;  
**import** android.net.Uri;  
**import** android.provider.BaseColumns;  
  
*// Class MovieContract define table name and hir column names***public class** MovieContract {  
 **public static final** String ***AUTHORITY*** = **"com.ex.popmovie.data"**;  
 **private static final** Uri ***BASE\_URI*** = Uri.*parse*(**"content://"** + ***AUTHORITY***);  
 *// Create class MovieTable:* **public static final class** MovieTable **implements** BaseColumns {  
 *// Create table name and each of the db columns* **public final static** String ***TABLE\_NAME*** = **"movies"**;  
 **public final static** String ***COLUMN\_ID\_MOVIE*** = **"idMovie"**;  
 **public final static** String ***COLUMN\_IS\_FAV*** = **"isFav"  
 public final static** String ***COLUMN\_TITLE*** = **"title"**;  
 **public final static** String ***COLUMN\_POSTER\_PATH*** = **"posterPath"**;  
 **public final static** String ***COLUMN\_OVERVIEW*** = **"overview"**;  
 **public final static** String ***COLUMN\_VOTE*** = **"vote"**;  
 **public final static** String ***COLUMN\_POP*** = **"pop"**;  
 **public final static** String ***COLUMN\_RELEASE*** = **"releaseDate"**;  
 *// Create Uri for table:* **public static final** Uri ***CONTENT\_URI*** = ***BASE\_URI***.buildUpon().appendPath(***TABLE\_NAME***).build();  
 *// Create Uri* **public static** Uri buildMovieUri(**long** id) {  
 **return** ContentUris.*withAppendedId*(***CONTENT\_URI***, id);  
 }  
 }  
}

**import** android.content.Context;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteOpenHelper;  
  
**import** com.ex.popmovie.data.MovieContract.\*;  
  
**public class** MovieDbHelper **extends** SQLiteOpenHelper {  
 *// Create datebase file name:* **private static final** String ***DB\_NAME*** = **"FavMovie.db"**;  
 *// If you change the database schema, you must increment the database version* **private static final int *DB\_VERSION*** = 1;  
 **public** MovieDbHelper(Context context) {  
 **super**(context, ***DB\_NAME***, **null**, ***DB\_VERSION***);  
 }  
 @Override  
 **public void** onCreate(SQLiteDatabase database) {  
 **final** String SQL\_CREATE\_DB\_TABLE = **"CREATE TABLE "** + MovieTable.***TABLE\_NAME*** + **" ("** +  
 MovieTable.***\_ID*** + **" INTEGER PRIMARY KEY AUTOINCREMENT, "** +  
 MovieTable.***COLUMN\_ID\_MOVIE*** + **" TEXT NOT NULL, "** +  
 MovieTable.***COLUMN\_IS\_FAV*** + **" ,"** +  
 MovieTable.***COLUMN\_TITLE*** + **" TEXT NOT NULL, "** +  
 MovieTable.***COLUMN\_POSTER\_PATH*** + **" TEXT NOT NULL, "** +  
 MovieTable.***COLUMN\_OVERVIEW*** + **" TEXT, "** +  
 MovieTable.***COLUMN\_VOTE*** + **" TEXT, "** +  
 MovieTable.***COLUMN\_POP*** + **" TEXT, "** +  
 MovieTable.***COLUMN\_RELEASE*** + **" TEXT"** +  
 **");"**;  
 database.execSQL(SQL\_CREATE\_DB\_TABLE);  
 }  
 *// Override the onUpgrade method* @Override  
 **public void** onUpgrade(SQLiteDatabase db, **int** i, **int** ii) {  
 *// For now simply drop the table and create a new one. This means if you change the  
 // DB\_VERSION the table will be dropped.  
 // In a production app, this method might be modified to ALTER the table  
 // instead of dropping it, so that existing data is not deleted.  
 // Inside, execute a drop table query, and then call onCreate to re-create it* db.execSQL(**"DROP TABLE IF EXISTS "** + MovieTable.***TABLE\_NAME***);  
 onCreate(db);  
 }  
}

**package** com.ex.popmovie.utilities;  
  
**import** android.content.Context;  
**import** android.support.annotation.NonNull;  
**import** android.support.v7.widget.RecyclerView;   
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.View.OnClickListener;  
**import** android.view.ViewGroup;  
**import** android.widget.ImageView;  
  
**import** com.ex.popmovie.R;  
**import** com.ex.popmovie.data.Movie;  
**import** com.squareup.picasso.Picasso;  
  
*// Creates a RecyclerView Adapter:***public class** RecyclerViewAdapter **extends** RecyclerView.Adapter<RecyclerViewAdapter.ViewHolder>{  
 **private static final** String ***MOVIE\_URL*** = **"http://image.tmdb.org/t/p/w342"**;  
 **public** Movie[] **movieList**;  
 *// An on-click handler that we've defined to make it easy for an Activity to interface with our RecyclerView* **private final** RecyclerViewAdapterOnClickHandler **mClickHandler**;  
 *// The interface that receives onClick messages.* **public interface** RecyclerViewAdapterOnClickHandler {  
 **void** onClick(**int** position);  
 }  
 */\* Creates a RecyclerViewAdapter.  
 \* @param clickHandler The on-click handler for this adapter. This single handler is called when an item is clicked.  
 \*/* **public** RecyclerViewAdapter(RecyclerViewAdapterOnClickHandler clickHandler) {  
 **mClickHandler** = clickHandler;  
 }  
 */\*\*  
 \* Cache of the children views for a list item.  
 \*/* **public class** ViewHolder **extends** RecyclerView.ViewHolder **implements** OnClickListener {  
 **private final** ImageView **ivPoster**;  
*// Create a constructor for this class that accepts a View as a parameter  
// Call super(itemView)  
// Using itemView.findViewById, get a reference to these layout's TextViews and save it.* ViewHolder(View itemView) {  
 **super**(itemView);  
 **ivPoster** = itemView.findViewById(R.id.***iv\_small\_poster***);  
 itemView.setOnClickListener(**this**);  
 }  
 *// This gets called by the child views during a click. @param v The View that was clicked* @Override  
 **public void** onClick(View v) {  
 **int** adapterPosition = getAdapterPosition();  
 **mClickHandler**.onClick(adapterPosition);  
 }  
 }  
 *// Override onCreateViewHolder  
 // Within onCreateViewHolder, inflate the list item xml into a view  
 // Within onCreateViewHolder, return a new ViewHolder with the above view passed in as a parameter  
 /\*\*  
 \* This gets called when each new ViewHolder is created. This happens when the RecyclerView  
 \* is laid out. Enough ViewHolders will be created to fill the screen and allow for scrolling.  
 \** ***@param parent*** *The ViewGroup that these ViewHolders are contained within.  
 \** ***@param viewType*** *If your RecyclerView has more than one type of item (which ours doesn't) you  
 \* can use this viewType integer to provide a different layout. See  
 \* {****@link*** *android.support.v7.widget.RecyclerView.Adapter#getItemViewType(int)}  
 \* for more details.  
 \** ***@return*** *A new ViewHolder that holds the View for each list item  
 \*/* @NonNull  
 @Override  
 **public** ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, **int** viewType) {  
 Context context = parent.getContext();  
 **int** layoutIdForListItem = R.layout.***item\_list***;  
 LayoutInflater inflater = LayoutInflater.*from*(context);  
*// boolean shouldAttachToParentImmediately = false; // don't attach to parent immediately* View view = inflater.inflate(layoutIdForListItem, parent, **false**);  
 **return new** ViewHolder(view);  
 }  
 *// Override onBindViewHolder  
 // Set the text of the TextView to data for this list item's position  
 /\*\*  
 \* OnBindViewHolder is called by the RecyclerView to display the data at the specified  
 \* position. In this method, we update the contents of the ViewHolder to display data  
 \* details for this particular position, using the "position" argument that is conveniently  
 \* passed into us.  
 \** ***@param viewHolder*** *The ViewHolder which should be updated to represent the  
 \* contents of the item at the given position in the data set.  
 \** ***@param position*** *The position of the item within the adapter's data set.  
 \*/* @Override  
 **public void** onBindViewHolder(@NonNull ViewHolder viewHolder, **int** position) {  
 Movie dataMovie = **movieList**[position];  
 String posterUrl = ***MOVIE\_URL*** + dataMovie.getPosterPath();  
 Picasso.*with*(viewHolder.**itemView**.getContext())  
 .load(posterUrl)  
 .into(viewHolder.**ivPoster**);  
 }  
 *// Override getItemCount  
 // Return 0 if mList is null, or the size of mList if it is not null  
 /\*\*  
 \* This method simply returns the number of items to display.  
 \* It is used behind the scenes to help layout our Views and for animations.  
 \** ***@return*** *The number of items available in our data  
 \*/* @Override  
 **public int** getItemCount() {  
 **if** (**movieList** == **null**) **return** 0;  
 **return movieList**.**length**;  
 }  
 */\*\*  
 \* This method is used to set the response data on a DataAdapter if we've already created one.  
 \* This is handy when we get new data from the web but don't want to create a new RecyclerViewAdapter to display it.  
 \* The new data to be displayed.  
 \*/  
 // Create a setList method that saves the movies to mList* **public void** setList(Movie[] movies) {  
 **movieList** = movies;  
 notifyDataSetChanged(); *// After you save movieList, call notifyDataSetChanged* }  
}

RecycleView

PopMovie

**import** android.support.v4.app.Fragment;   
  
**public class** DetailActivity **extends** FragmentActivity {  
 @Override  
 **protected** Fragment createFragment() {  
 **return new** DetailFragment();  
 }  
}

PopMovie

DetailActivity

**import** com.ex.popmovie.data.Movie;  
**import** com.ex.popmovie.data.MovieContract;  
**import** com.squareup.picasso.Picasso;  
*/\*\*  
 \* A simple {****@link*** *Fragment} subclass.  
 \*/***public class** DetailFragment **extends** Fragment {  
*// implements LoaderManager.LoaderCallbacks<Cursor> ???* **private static final** String ***MOVIE\_URL*** = **"http://image.tmdb.org/t/p/w185"**;  
 **public static final** String ***EXTRA\_OBJECT*** = **"mark\_movie"**;  
 *// constant used to identify the Loader* **private static final int *MOVIE\_LOADER*** = 0;  
  
 **boolean isFav** = **false**;  
 Movie **movieDetail**;  
  
 **public** DetailFragment() {  
 *// Required empty public constructor* }  
 @Nullable  
 @Override  
 **public** View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 *// Inflate the layout for this fragment* View v = inflater.inflate(R.layout.***fragment\_detail***, container, **false**);  
  
 ImageView ivPoster = v.findViewById(R.id.***iv\_poster***);  
 TextView tvTitle = v.findViewById(R.id.***tv\_detail\_title***);  
 TextView tvOverview = v.findViewById(R.id.***tv\_detail\_overview***);  
 TextView tvVote = v.findViewById(R.id.***tv\_detail\_vote***);  
 TextView tvPop = v.findViewById(R.id.***tv\_detail\_pop***);  
 TextView tvReleaseDate = v.findViewById(R.id.***tv\_detail\_release***);  
 TextView tvIdMovie = v.findViewById(R.id.***tv\_detail\_id***);  
 *// Intent intent = getActivity().getIntent();  
 // (2) Display the data that was passed from MainActivity  
// if (intent == null) { closeOnError(); return; }  
// intent.getIntExtra(EXTRA\_POSITION, position);* **movieDetail** = getActivity().getIntent().getParcelableExtra(***EXTRA\_OBJECT***);  
  
 String mPosterPath = **movieDetail**.getPosterPath();  
 String mTitle = **movieDetail**.getTitle();  
 String mOverview = **movieDetail**.getOverview();  
 String mVote = **movieDetail**.getVote();  
 String mPop = **movieDetail**.getPop();  
 String mReleaseDate = **movieDetail**.getReleaseDate();  
 **final** String mIdMovie = **movieDetail**.getIdMovie();  
  
 String posterUrl = ***MOVIE\_URL*** + mPosterPath;  
 Picasso.*with*(getContext())  
 .load(posterUrl)  
 .into(ivPoster);  
  
 tvTitle.setText(mTitle);  
 tvOverview.setText(mOverview);  
 tvVote.setText(mVote);  
 tvPop.setText(mPop);  
 tvReleaseDate.setText(mReleaseDate);  
 tvIdMovie.setText(mIdMovie);  
*// Add Button for Favorite movie:* ImageButton imgButtonFav = v.findViewById(R.id.***imgButton\_fav***);  
*// check in Favorite movie* **if** (checkIsFav(mIdMovie)) {  
 imgButtonFav.setImageResource(R.drawable.***ic\_favorite\_red\_24dp***);  
 } **else** {  
 imgButtonFav.setImageResource(R.drawable.***ic\_favorite\_border\_24dp***);  
 };  
  
 imgButtonFav.setOnClickListener(  
 **new** View.OnClickListener() {  
 *// displays the AddEditFragment when FAB is touched* @Override  
 **public void** onClick(View view) {  
 updateFavMovie(mIdMovie); *// save Favorite movie to the database* }  
 }  
 );  
 **return** v;  
 }  
 **private boolean** checkIsFav(String id) {  
*// check in movie with id is favorite* Cursor cursor = getActivity().getContentResolver().query(  
 MovieContract.MovieTable.***CONTENT\_URI***,  
 **null**,  
 MovieContract.MovieTable.***COLUMN\_ID\_MOVIE*** + **"="** + id,  
 **null**,  
 **null**);  
 */\* return true if the cursor is not empty \*/* **isFav** = (cursor.getCount() > 0);  
 cursor.close();  
 **return isFav**;  
 }  
*// update* **private boolean** updateFavMovie(String id) {  
 **if**(**isFav**) {  
 **isFav** = **false**;  
 deleteFavMovie(id);  
 } **else** {  
 **isFav** = **true**;  
 saveFavMovie();  
 };  
 **return isFav**;  
 }  
*// saves Favorite movie information to the database* **private void** saveFavMovie() {  
 *// create ContentValues object containing movie's key-value pairs* ContentValues contentValues = **new** ContentValues();  
 contentValues.put(MovieContract.MovieTable.***COLUMN\_TITLE***, **movieDetail**.getTitle().toString());  
 contentValues.put(MovieContract.MovieTable.***COLUMN\_ID\_MOVIE***, **movieDetail**.getIdMovie().toString());  
 contentValues.put(MovieContract.MovieTable.***COLUMN\_POSTER\_PATH***, **movieDetail**.getPosterPath().toString());  
 contentValues.put(MovieContract.MovieTable.***COLUMN\_OVERVIEW***, **movieDetail**.getOverview().toString());  
 contentValues.put(MovieContract.MovieTable.***COLUMN\_VOTE***, **movieDetail**.getVote().toString());  
 contentValues.put(MovieContract.MovieTable.***COLUMN\_POP***, **movieDetail**.getPop().toString());  
 contentValues.put(MovieContract.MovieTable.***COLUMN\_RELEASE***, **movieDetail**.getReleaseDate().toString());  
  
 Uri newMovieUri = getActivity().getContentResolver().insert(  
 MovieContract.MovieTable.***CONTENT\_URI***, contentValues);  
 }  
*// saves Favorite movie information to the database* **private void** deleteFavMovie(String id) {  
 *// delete movie with id from Favorite movie's database* **int** rowsDelete = getActivity().getContentResolver().delete(MovieContract.MovieTable.***CONTENT\_URI***,  
 MovieContract.MovieTable.***COLUMN\_ID\_MOVIE*** + **" = ?"** ,**new** String[] {id});  
 }  
}

**import** com.ex.popmovie.data.Movie;  
**import** com.ex.popmovie.utilities.JsonUtils;  
**import** com.ex.popmovie.utilities.NetworkUtils;  
**import** com.ex.popmovie.utilities.RecyclerViewAdapter;  
  
**import static** com.ex.popmovie.DetailFragment.***EXTRA\_OBJECT***;  
  
*/\*\*  
 \* A simple {****@link*** *Fragment} subclass.  
 \*/***public class** ListFragment **extends** Fragment **implements** RecyclerViewAdapter.RecyclerViewAdapterOnClickHandler {  
 **private static final** String ***POPULAR*** = **"/popular?"**;  
 **private static final** String ***TOP\_RATED*** = **"/top\_rated?"**;  
 **private static final int *DEFAULT\_SIZE*** = 180;  
 **private** RecyclerView **recyclerView**;  
 **private** RecyclerViewAdapter **recyclerViewAdapter**;  
  
 **public** ListFragment() {  
 *// Required empty public constructor* }  
  
 @Override  
 **public void** onCreate(@Nullable Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setHasOptionsMenu(**true**);  
 }  
  
 @Override  
 **public** View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 *// Inflate the layout for this fragment* View v = inflater.inflate(R.layout.***fragment\_list***, container, **false**);  
  
 **recyclerView** = v.findViewById(R.id.***rv\_list***);  
 Context context = getContext();  
 **if**(context == **null**) closeOnError();  
 **int** numberOfColumns = *calculateColumns*(getContext());  
 *// Set the gridLayoutManager on recyclerView* **recyclerView**.setLayoutManager(**new** GridLayoutManager(getContext(), numberOfColumns));  
 **recyclerView**.setHasFixedSize(**true**);  
 **recyclerViewAdapter** = **new** RecyclerViewAdapter(**this**);  
 **recyclerView**.setAdapter(**recyclerViewAdapter**);  
  
 *// Call loadData to perform the network request to get data* loadData(***TOP\_RATED***);  
  
 **return** v;  
 }  
  
 **private void** loadData(String queryType) {  
 String apiKey = getResources().getString(R.string.***key\_api***);  
 **new** QueryAsyncTask().execute(apiKey, queryType);  
 }  
  
 *// Create a class that extends AsyncTask to perform network requests* @SuppressLint(**"StaticFieldLeak"**)  
 **class** QueryAsyncTask **extends** AsyncTask<String, Void, Movie[]> {  
 *// Override the doInBackground method to perform your network requests* @Override  
 **protected** Movie[] doInBackground(String... params) {  
 **if** (params.**length** == 0) {  
 **return null**;  
 }  
 String apiKey = params[0];  
 String queryType = params[1];  
 URL requestUrl = NetworkUtils.*buildUrl*(apiKey, queryType);  
 **try** {  
 String jsonResponse = NetworkUtils.*getResponseFromHttpUrl*(requestUrl);  
 **return** JsonUtils.*parseJson*(jsonResponse);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return null**;  
 }  
  
 *// Override the onPostExecute method to display the results of the network request* @Override  
 **protected void** onPostExecute(Movie[] movieList) {  
 **if** (movieList != **null**) {  
 **recyclerView**.setVisibility(View.***VISIBLE***);  
 *// Instead of iterating through every string, use recyclerViewAdapter.setList and pass in data* **recyclerViewAdapter**.setList(movieList);  
 } **else** {  
 Toast.*makeText*(getActivity(), **"ErrorQuery. Check out correct api\_key"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 }  
  
 *// Override method in order to handle RecyclerView item clicks.* **public void** onClick(**int** position) {  
 Intent intent = **new** Intent(getActivity(), DetailActivity.**class**);  
 *// Pass the data to the DetailActivity* Movie markMovie = **this**.**recyclerViewAdapter**.**movieList**[position];  
 intent.putExtra(***EXTRA\_OBJECT***, markMovie);  
 startActivity(intent);  
 }  
  
 */\* https://stackoverflow.com/questions/33575731/gridlayoutmanager-how-to-auto-fit-columns  
 \* calculate number of columns in GridLayoutManager  
 \*/* **private static int** calculateColumns(Context context) {  
 DisplayMetrics displayMetrics = context.getResources().getDisplayMetrics();  
 **float** dpWidth = displayMetrics.**widthPixels** / displayMetrics.**density**;  
 **return** (**int**) (dpWidth / ***DEFAULT\_SIZE***);  
 }  
  
 **private void** closeOnError() {  
 getActivity().finish();  
 Toast.*makeText*(getContext(), R.string.***detail\_error\_context***, Toast.***LENGTH\_SHORT***).show();  
 }  
  
 *// Creating Menu: Popular TopRating Favorite and HomeButton* @Override  
 **public void** onCreateOptionsMenu(Menu menu, MenuInflater inflater) {  
 inflater.inflate(R.menu.***main\_menu***, menu);  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 **switch** (item.getItemId()) {  
 **case** R.id.***menu\_pop***:  
 loadData(***POPULAR***);  
 **return true**;  
 **case** R.id.***menu\_rate***:  
 loadData(***TOP\_RATED***);  
 **return true**;  
 **default**:  
 **return super**.onOptionsItemSelected(item);  
 }  
 }  
}