*//* ***MainActivity.java*** *// Hosts the app's fragments and handles communication between them***package** com.ex.and.abook;  
  
**import** android.net.Uri;  
**import** android.os.Bundle;  
**import** android.support.v4.app.FragmentTransaction;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.support.v7.widget.Toolbar;  
  
**public class** MainActivity **extends** AppCompatActivity  
 **implements** ContactsFragment.ContactsFragmentListener,  
 DetailFragment.DetailFragmentListener,  
 AddEditFragment.AddEditFragmentListener {  
  
 *// key for storing a contact's Uri in a Bundle passed to a fragment* **public static final** String ***CONTACT\_URI*** = **"contact\_uri"**;  
  
 **private** ContactsFragment **contactsFragment**; *// displays contact list  
  
 // display ContactsFragment when MainActivity first loads* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 Toolbar toolbar = (Toolbar) findViewById(R.id.***toolbar***);  
 setSupportActionBar(toolbar);  
  
 *// if layout contains fragmentContainer, the phone layout is in use;  
 // create and display a ContactsFragment* **if** (savedInstanceState == **null** &&  
 findViewById(R.id.***fragmentContainer***) != **null**) {  
 *// create ContactsFragment* **contactsFragment** = **new** ContactsFragment();  
  
 *// add the fragment to the FrameLayout* FragmentTransaction transaction =  
 getSupportFragmentManager().beginTransaction();  
 transaction.add(R.id.***fragmentContainer***, **contactsFragment**);  
 transaction.commit(); *// display ContactsFragment* }  
 **else** {  
 **contactsFragment** =  
 (ContactsFragment) getSupportFragmentManager().  
 findFragmentById(R.id.***contactsFragment***);  
 }  
 }  
 *// display DetailFragment for selected contact* @Override  
 **public void** onContactSelected(Uri contactUri) {  
 **if** (findViewById(R.id.***fragmentContainer***) != **null**) *// phone* displayContact(contactUri, R.id.***fragmentContainer***);  
 **else** { *// tablet  
 // removes top of back stack* getSupportFragmentManager().popBackStack();  
  
 displayContact(contactUri, R.id.***rightPaneContainer***);  
 }  
 }  
 *// display AddEditFragment to add a new contact* @Override  
 **public void** onAddContact() {  
 **if** (findViewById(R.id.***fragmentContainer***) != **null**) *// phone* displayAddEditFragment(R.id.***fragmentContainer***, **null**);  
 **else** *// tablet* displayAddEditFragment(R.id.***rightPaneContainer***, **null**);  
 }  
 *// display a contact* **private void** displayContact(Uri contactUri, **int** viewID) {  
 DetailFragment detailFragment = **new** DetailFragment();  
  
 *// specify contact's Uri as an argument to the DetailFragment* Bundle arguments = **new** Bundle();  
 arguments.putParcelable(***CONTACT\_URI***, contactUri);  
 detailFragment.setArguments(arguments);  
  
 *// use a FragmentTransaction to display the DetailFragment* FragmentTransaction transaction =  
 getSupportFragmentManager().beginTransaction();  
 transaction.replace(viewID, detailFragment);  
 transaction.addToBackStack(**null**);  
 transaction.commit(); *// causes DetailFragment to display* }  
  
 *// display fragment for adding a new or editing an existing contact* **private void** displayAddEditFragment(**int** viewID, Uri contactUri) {  
 AddEditFragment addEditFragment = **new** AddEditFragment();  
  
 *// if editing existing contact, provide contactUri as an argument* **if** (contactUri != **null**) {  
 Bundle arguments = **new** Bundle();  
 arguments.putParcelable(***CONTACT\_URI***, contactUri);  
 addEditFragment.setArguments(arguments);  
 }  
  
 *// use a FragmentTransaction to display the AddEditFragment* FragmentTransaction transaction =  
 getSupportFragmentManager().beginTransaction();  
 transaction.replace(viewID, addEditFragment);  
 transaction.addToBackStack(**null**);  
 transaction.commit(); *// causes AddEditFragment to display* }  
  
 *// return to contact list when displayed contact deleted* @Override  
 **public void** onContactDeleted() {  
 *// removes top of back stack* getSupportFragmentManager().popBackStack();  
 **contactsFragment**.updateContactList(); *// refresh contacts* }  
  
 *// display the AddEditFragment to edit an existing contact* @Override  
 **public void** onEditContact(Uri contactUri) {  
 **if** (findViewById(R.id.***fragmentContainer***) != **null**) *// phone* displayAddEditFragment(R.id.***fragmentContainer***, contactUri);  
 **else** *// tablet* displayAddEditFragment(R.id.***rightPaneContainer***, contactUri);  
 }  
  
 *// update GUI after new contact or updated contact saved* @Override  
 **public void** onAddEditCompleted(Uri contactUri) {  
 *// removes top of back stack* getSupportFragmentManager().popBackStack();  
 **contactsFragment**.updateContactList(); *// refresh contacts* **if** (findViewById(R.id.***fragmentContainer***) == **null**) { *// tablet  
 // removes top of back stack* getSupportFragmentManager().popBackStack();  
  
 *// on tablet, display contact that was just added or edited* displayContact(contactUri, R.id.***rightPaneContainer***);  
 }  
 }  
}

*//* ***ContactsFragment.java*** *// Fragment subclass that displays the alphabetical list of contact names***package** com.ex.and.abook;  
  
**import** android.content.Context;  
**import** android.database.Cursor;  
**import** android.net.Uri;  
**import** android.os.Bundle;  
**import** android.support.design.widget.FloatingActionButton;  
**import** android.support.v4.app.Fragment;  
**import** android.support.v4.app.LoaderManager;  
**import** android.support.v4.content.CursorLoader;  
**import** android.support.v4.content.Loader;  
**import** android.support.v7.widget.LinearLayoutManager;  
**import** android.support.v7.widget.RecyclerView;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
  
**import** com.ex.and.abook.data.DatabaseDescription.Contact;  
  
**public class** ContactsFragment **extends** Fragment  
 **implements** LoaderManager.LoaderCallbacks<Cursor> {  
  
 *// callback method implemented by MainActivity* **public interface** ContactsFragmentListener {  
 *// called when contact selected* **void** onContactSelected(Uri contactUri);  
 *// called when add button is pressed* **void** onAddContact();  
 }  
 **private static final int *CONTACTS\_LOADER*** = 0; *// identifies Loader  
 // used to inform the MainActivity when a contact is selected* **private** ContactsFragmentListener **listener**;  
 **private** ContactsAdapter **contactsAdapter**; *// adapter for recyclerView  
 // configures this fragment's GUI* @Override  
 **public** View onCreateView(  
 LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 **super**.onCreateView(inflater, container, savedInstanceState);  
 setHasOptionsMenu(**true**); *// fragment has menu items to display  
 // inflate GUI and get reference to the RecyclerView* View view = inflater.inflate(  
 R.layout.***fragment\_contacts***, container, **false**);  
 RecyclerView recyclerView =  
 (RecyclerView) view.findViewById(R.id.***recyclerView***);  
 *// recyclerView should display items in a vertical list* recyclerView.setLayoutManager(  
 **new** LinearLayoutManager(getActivity().getBaseContext()));  
 *// create recyclerView's adapter and item click listener* **contactsAdapter** = **new** ContactsAdapter(  
 **new** ContactsAdapter.ContactClickListener() {  
 @Override  
 **public void** onClick(Uri contactUri) {  
 **listener**.onContactSelected(contactUri);  
 }  
 }  
 );  
 recyclerView.setAdapter(**contactsAdapter**); *// set the adapter  
  
 // attach a custom ItemDecorator to draw dividers between list items* recyclerView.addItemDecoration(**new** ItemDivider(getContext()));  
  
 *// improves performance if RecyclerView's layout size never changes* recyclerView.setHasFixedSize(**true**);  
  
 *// get the FloatingActionButton and configure its listener* FloatingActionButton addButton =  
 (FloatingActionButton) view.findViewById(R.id.***addButton***);  
 addButton.setOnClickListener(  
 **new** View.OnClickListener() {  
 *// displays the AddEditFragment when FAB is touched* @Override  
 **public void** onClick(View view) {  
 **listener**.onAddContact();  
 }  
 }  
 );  
 **return** view;  
 }  
 *// set ContactsFragmentListener when fragment attached* @Override  
 **public void** onAttach(Context context) {  
 **super**.onAttach(context);  
 **listener** = (ContactsFragmentListener) context;  
 }  
 *// remove ContactsFragmentListener when Fragment detached* @Override  
 **public void** onDetach() {  
 **super**.onDetach();  
 **listener** = **null**;  
 }  
 *// initialize a Loader when this fragment's activity is created* @Override  
 **public void** onActivityCreated(Bundle savedInstanceState) {  
 **super**.onActivityCreated(savedInstanceState);  
 getLoaderManager().initLoader(***CONTACTS\_LOADER***, **null**, **this**);  
 }  
 *// called from MainActivity when other Fragment's update database* **public void** updateContactList() {  
 **contactsAdapter**.notifyDataSetChanged();  
 }  
 *// called by LoaderManager to create a Loader* @Override  
 **public** Loader<Cursor> onCreateLoader(**int** id, Bundle args) {  
 *// create an appropriate CursorLoader based on the id argument;  
 // only one Loader in this fragment, so the switch is unnecessary* **switch** (id) {  
 **case *CONTACTS\_LOADER***:  
 **return new** CursorLoader(getActivity(),  
 Contact.***CONTENT\_URI***, *// Uri of contacts table* **null**, *// null projection returns all columns* **null**, *// null selection returns all rows* **null**, *// no selection arguments* Contact.***COLUMN\_NAME*** + **" COLLATE NOCASE ASC"**); *// sort order* **default**:  
 **return null**;  
 }  
 }  
 *// called by LoaderManager when loading completes* @Override  
 **public void** onLoadFinished(Loader<Cursor> loader, Cursor data) {  
 **contactsAdapter**.swapCursor(data);  
 }  
  
 *// called by LoaderManager when the Loader is being reset* @Override  
 **public void** onLoaderReset(Loader<Cursor> loader) {  
 **contactsAdapter**.swapCursor(**null**);  
 }  
}

*//* ***AddEditFragment.java*** *// Fragment for adding a new contact or editing an existing one***package** com.ex.and.abook;  
  
**import** android.content.ContentValues;  
**import** android.content.Context;  
**import** android.database.Cursor;  
**import** android.net.Uri;  
**import** android.os.Bundle;  
**import** android.support.design.widget.CoordinatorLayout;  
**import** android.support.design.widget.FloatingActionButton;  
**import** android.support.design.widget.Snackbar;  
**import** android.support.design.widget.TextInputLayout;  
**import** android.support.v4.app.Fragment;  
**import** android.support.v4.app.LoaderManager;  
**import** android.support.v4.content.CursorLoader;  
**import** android.support.v4.content.Loader;  
**import** android.text.Editable;  
**import** android.text.TextWatcher;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.view.inputmethod.InputMethodManager;  
  
**import** com.ex.and.abook.data.DatabaseDescription.Contact;  
  
**public class** AddEditFragment **extends** Fragment  
 **implements** LoaderManager.LoaderCallbacks<Cursor> {  
  
 *// defines callback method implemented by MainActivity* **public interface** AddEditFragmentListener {  
 *// called when contact is saved* **void** onAddEditCompleted(Uri contactUri);  
 }  
  
 *// constant used to identify the Loader* **private static final int *CONTACT\_LOADER*** = 0;  
  
 **private** AddEditFragmentListener **listener**; *// MainActivity* **private** Uri **contactUri**; *// Uri of selected contact* **private boolean addingNewContact** = **true**; *// adding (true) or editing  
  
 // EditTexts for contact information* **private** TextInputLayout **nameTextInputLayout**;  
 **private** TextInputLayout **phoneTextInputLayout**;  
 **private** TextInputLayout **emailTextInputLayout**;  
 **private** TextInputLayout **streetTextInputLayout**;  
 **private** TextInputLayout **cityTextInputLayout**;  
 **private** TextInputLayout **stateTextInputLayout**;  
 **private** TextInputLayout **zipTextInputLayout**;  
 **private** FloatingActionButton **saveContactFAB**;  
  
 **private** CoordinatorLayout **coordinatorLayout**; *// used with SnackBars  
  
 // set AddEditFragmentListener when Fragment attached* @Override  
 **public void** onAttach(Context context) {  
 **super**.onAttach(context);  
 **listener** = (AddEditFragmentListener) context;  
 }  
  
 *// remove AddEditFragmentListener when Fragment detached* @Override  
 **public void** onDetach() {  
 **super**.onDetach();  
 **listener** = **null**;  
 }  
  
 *// called when Fragment's view needs to be created* @Override  
 **public** View onCreateView(  
 LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 **super**.onCreateView(inflater, container, savedInstanceState);  
 setHasOptionsMenu(**true**); *// fragment has menu items to display  
  
 // inflate GUI and get references to EditTexts* View view =  
 inflater.inflate(R.layout.***fragment\_add\_edit***, container, **false**);  
 **nameTextInputLayout** =  
 (TextInputLayout) view.findViewById(R.id.***nameTextInputLayout***);  
 **nameTextInputLayout**.getEditText().addTextChangedListener(  
 **nameChangedListener**);  
 **phoneTextInputLayout** =  
 (TextInputLayout) view.findViewById(R.id.***phoneTextInputLayout***);  
 **emailTextInputLayout** =  
 (TextInputLayout) view.findViewById(R.id.***emailTextInputLayout***);  
 **streetTextInputLayout** =  
 (TextInputLayout) view.findViewById(R.id.***streetTextInputLayout***);  
 **cityTextInputLayout** =  
 (TextInputLayout) view.findViewById(R.id.***cityTextInputLayout***);  
 **stateTextInputLayout** =  
 (TextInputLayout) view.findViewById(R.id.***stateTextInputLayout***);  
 **zipTextInputLayout** =  
 (TextInputLayout) view.findViewById(R.id.***zipTextInputLayout***);  
  
 *// set FloatingActionButton's event listener* **saveContactFAB** = (FloatingActionButton) view.findViewById(  
 R.id.***saveFloatingActionButton***);  
 **saveContactFAB**.setOnClickListener(**saveContactButtonClicked**);  
 updateSaveButtonFAB();  
  
 *// used to display SnackBars with brief messages* **coordinatorLayout** = (CoordinatorLayout) getActivity().findViewById(  
 R.id.***coordinatorLayout***);  
  
 Bundle arguments = getArguments(); *// null if creating new contact* **if** (arguments != **null**) {  
 **addingNewContact** = **false**;  
 **contactUri** = arguments.getParcelable(MainActivity.***CONTACT\_URI***);  
 }  
  
 *// if editing an existing contact, create Loader to get the contact* **if** (**contactUri** != **null**)  
 getLoaderManager().initLoader(***CONTACT\_LOADER***, **null**, **this**);  
  
 **return** view;  
 }  
  
 *// detects when the text in the nameTextInputLayout's EditText changes  
 // to hide or show saveButtonFAB* **private final** TextWatcher **nameChangedListener** = **new** TextWatcher() {  
 @Override  
 **public void** beforeTextChanged(CharSequence s, **int** start, **int** count,  
 **int** after) {}  
  
 *// called when the text in nameTextInputLayout changes* @Override  
 **public void** onTextChanged(CharSequence s, **int** start, **int** before,  
 **int** count) {  
 updateSaveButtonFAB();  
 }  
  
 @Override  
 **public void** afterTextChanged(Editable s) { }  
 };  
  
 *// shows saveButtonFAB only if the name is not empty* **private void** updateSaveButtonFAB() {  
 String input =  
 **nameTextInputLayout**.getEditText().getText().toString();  
  
 *// if there is a name for the contact, show the FloatingActionButton* **if** (input.trim().length() != 0)  
 **saveContactFAB**.show();  
 **else  
 saveContactFAB**.hide();  
 }  
  
 *// responds to event generated when user saves a contact* **private final** View.OnClickListener **saveContactButtonClicked** =  
 **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *// hide the virtual keyboard* ((InputMethodManager) getActivity().getSystemService(  
 Context.***INPUT\_METHOD\_SERVICE***)).hideSoftInputFromWindow(  
 getView().getWindowToken(), 0);  
 saveContact(); *// save contact to the database* }  
 };  
  
 *// saves contact information to the database* **private void** saveContact() {  
 *// create ContentValues object containing contact's key-value pairs* ContentValues contentValues = **new** ContentValues();  
 contentValues.put(Contact.***COLUMN\_NAME***,  
 **nameTextInputLayout**.getEditText().getText().toString());  
 contentValues.put(Contact.***COLUMN\_PHONE***,  
 **phoneTextInputLayout**.getEditText().getText().toString());  
 contentValues.put(Contact.***COLUMN\_EMAIL***,  
 **emailTextInputLayout**.getEditText().getText().toString());  
 contentValues.put(Contact.***COLUMN\_STREET***,  
 **streetTextInputLayout**.getEditText().getText().toString());  
 contentValues.put(Contact.***COLUMN\_CITY***,  
 **cityTextInputLayout**.getEditText().getText().toString());  
 contentValues.put(Contact.***COLUMN\_STATE***,  
 **stateTextInputLayout**.getEditText().getText().toString());  
 contentValues.put(Contact.***COLUMN\_ZIP***,  
 **zipTextInputLayout**.getEditText().getText().toString());  
  
 **if** (**addingNewContact**) {  
 *// use Activity's ContentResolver to invoke  
 // insert on the AddressBookContentProvider* Uri newContactUri = getActivity().getContentResolver().insert(  
 Contact.***CONTENT\_URI***, contentValues);  
  
 **if** (newContactUri != **null**) {  
 Snackbar.*make*(**coordinatorLayout**,  
 R.string.***contact\_added***, Snackbar.***LENGTH\_LONG***).show();  
 **listener**.onAddEditCompleted(newContactUri);  
 }  
 **else** {  
 Snackbar.*make*(**coordinatorLayout**,  
 R.string.***contact\_not\_added***, Snackbar.***LENGTH\_LONG***).show();  
 }  
 }  
 **else** {  
 *// use Activity's ContentResolver to invoke  
 // insert on the AddressBookContentProvider* **int** updatedRows = getActivity().getContentResolver().update(  
 **contactUri**, contentValues, **null**, **null**);  
  
 **if** (updatedRows > 0) {  
 **listener**.onAddEditCompleted(**contactUri**);  
 Snackbar.*make*(**coordinatorLayout**,  
 R.string.***contact\_updated***, Snackbar.***LENGTH\_LONG***).show();  
 }  
 **else** {  
 Snackbar.*make*(**coordinatorLayout**,  
 R.string.***contact\_not\_updated***, Snackbar.***LENGTH\_LONG***).show();  
 }  
 }  
 }  
  
 *// called by LoaderManager to create a Loader* @Override  
 **public** Loader<Cursor> onCreateLoader(**int** id, Bundle args) {  
 *// create an appropriate CursorLoader based on the id argument;  
 // only one Loader in this fragment, so the switch is unnecessary* **switch** (id) {  
 **case *CONTACT\_LOADER***:  
 **return new** CursorLoader(getActivity(),  
 **contactUri**, *// Uri of contact to display* **null**, *// null projection returns all columns* **null**, *// null selection returns all rows* **null**, *// no selection arguments* **null**); *// sort order* **default**:  
 **return null**;  
 }  
 }  
  
 *// called by LoaderManager when loading completes* @Override  
 **public void** onLoadFinished(Loader<Cursor> loader, Cursor data) {  
 *// if the contact exists in the database, display its data* **if** (data != **null** && data.moveToFirst()) {  
 *// get the column index for each data item* **int** nameIndex = data.getColumnIndex(Contact.***COLUMN\_NAME***);  
 **int** phoneIndex = data.getColumnIndex(Contact.***COLUMN\_PHONE***);  
 **int** emailIndex = data.getColumnIndex(Contact.***COLUMN\_EMAIL***);  
 **int** streetIndex = data.getColumnIndex(Contact.***COLUMN\_STREET***);  
 **int** cityIndex = data.getColumnIndex(Contact.***COLUMN\_CITY***);  
 **int** stateIndex = data.getColumnIndex(Contact.***COLUMN\_STATE***);  
 **int** zipIndex = data.getColumnIndex(Contact.***COLUMN\_ZIP***);  
  
 *// fill EditTexts with the retrieved data* **nameTextInputLayout**.getEditText().setText(  
 data.getString(nameIndex));  
 **phoneTextInputLayout**.getEditText().setText(  
 data.getString(phoneIndex));  
 **emailTextInputLayout**.getEditText().setText(  
 data.getString(emailIndex));  
 **streetTextInputLayout**.getEditText().setText(  
 data.getString(streetIndex));  
 **cityTextInputLayout**.getEditText().setText(  
 data.getString(cityIndex));  
 **stateTextInputLayout**.getEditText().setText(  
 data.getString(stateIndex));  
 **zipTextInputLayout**.getEditText().setText(  
 data.getString(zipIndex));  
  
 updateSaveButtonFAB();  
 }  
 }  
  
 *// called by LoaderManager when the Loader is being reset* @Override  
 **public void** onLoaderReset(Loader<Cursor> loader) { }  
}

*//* ***DetailFragment.java*** *// Fragment subclass that displays one contact's details***package** com.ex.and.abook;  
  
**import** android.app.AlertDialog;  
**import** android.app.Dialog;  
**import** android.content.Context;  
**import** android.content.DialogInterface;  
**import** android.database.Cursor;  
**import** android.net.Uri;  
**import** android.os.Bundle;  
**import** android.support.v4.app.DialogFragment;  
**import** android.support.v4.app.Fragment;  
**import** android.support.v4.app.LoaderManager;  
**import** android.support.v4.content.CursorLoader;  
**import** android.support.v4.content.Loader;  
**import** android.view.LayoutInflater;  
**import** android.view.Menu;  
**import** android.view.MenuInflater;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.TextView;  
  
**import** com.ex.and.abook.data.DatabaseDescription.Contact;  
  
**public class** DetailFragment **extends** Fragment  
 **implements** LoaderManager.LoaderCallbacks<Cursor> {  
 *// callback methods implemented by MainActivity* **public interface** DetailFragmentListener {  
 **void** onContactDeleted(); *// called when a contact is deleted  
 // pass Uri of contact to edit to the DetailFragmentListener* **void** onEditContact(Uri contactUri);  
 }  
 **private static final int *CONTACT\_LOADER*** = 0; *// identifies the Loader* **private** DetailFragmentListener **listener**; *// MainActivity* **private** Uri **contactUri**; *// Uri of selected contact* **private** TextView **nameTextView**; *// displays contact's name* **private** TextView **phoneTextView**; *// displays contact's phone* **private** TextView **emailTextView**; *// displays contact's email* **private** TextView **streetTextView**; *// displays contact's street* **private** TextView **cityTextView**; *// displays contact's city* **private** TextView **stateTextView**; *// displays contact's state* **private** TextView **zipTextView**; *// displays contact's zip  
 // set DetailFragmentListener when fragment attached* @Override  
 **public void** onAttach(Context context) {  
 **super**.onAttach(context);  
 **listener** = (DetailFragmentListener) context;  
 }  
 *// remove DetailFragmentListener when fragment detached* @Override  
 **public void** onDetach() {  
 **super**.onDetach();  
 **listener** = **null**;  
 }  
 *// called when DetailFragmentListener's view needs to be created* @Override  
 **public** View onCreateView(  
 LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 **super**.onCreateView(inflater, container, savedInstanceState);  
 setHasOptionsMenu(**true**); *// this fragment has menu items to display  
 // get Bundle of arguments then extract the contact's Uri* Bundle arguments = getArguments();  
  
 **if** (arguments != **null**)  
 **contactUri** = arguments.getParcelable(MainActivity.***CONTACT\_URI***);  
 *// inflate DetailFragment's layout* View view =  
 inflater.inflate(R.layout.***fragment\_detail***, container, **false**);  
 *// get the EditTexts* **nameTextView** = (TextView) view.findViewById(R.id.***nameTextView***);  
 **phoneTextView** = (TextView) view.findViewById(R.id.***phoneTextView***);  
 **emailTextView** = (TextView) view.findViewById(R.id.***emailTextView***);  
 **streetTextView** = (TextView) view.findViewById(R.id.***streetTextView***);  
 **cityTextView** = (TextView) view.findViewById(R.id.***cityTextView***);  
 **stateTextView** = (TextView) view.findViewById(R.id.***stateTextView***);  
 **zipTextView** = (TextView) view.findViewById(R.id.***zipTextView***);  
 *// load the contact* getLoaderManager().initLoader(***CONTACT\_LOADER***, **null**, **this**);  
 **return** view;  
 }  
 *// display this fragment's menu items* @Override  
 **public void** onCreateOptionsMenu(Menu menu, MenuInflater inflater) {  
 **super**.onCreateOptionsMenu(menu, inflater);  
 inflater.inflate(R.menu.***fragment\_details\_menu***, menu);  
 }  
 *// handle menu item selections* @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 **switch** (item.getItemId()) {  
 **case** R.id.***action\_edit***:  
 **listener**.onEditContact(**contactUri**); *// pass Uri to listener* **return true**;  
 **case** R.id.***action\_delete***:  
 deleteContact();  
 **return true**;  
 }  
 **return super**.onOptionsItemSelected(item);  
 }  
 *// delete a contact* **private void** deleteContact() {  
 *// use FragmentManager to display the confirmDelete DialogFragment* getActivity().getContentResolver().delete(  
 **contactUri**, **null**, **null**);  
 **listener**.onContactDeleted(); *// notify listener  
// Error DialogFragment API 15 + 18 confirmDelete.show(getFragmentManager(), "confirm delete");* }  
 *//* ***TODO: Error DialogFragment API 15 + 18*** *// Fragments should be static such that they can be re-instantiated by the system,  
 // and anonymous classes are not static  
 // DialogFragment to confirm deletion of contact* **private final** DialogFragment **confirmDelete** = **new** DialogFragment() {  
 *// create an AlertDialog and return it* @Override  
 **public** Dialog onCreateDialog(Bundle bundle) {  
 *// create a new AlertDialog Builder* AlertDialog.Builder builder =  
 **new** AlertDialog.Builder(getActivity());  
  
 builder.setTitle(R.string.***confirm\_title***);  
 builder.setMessage(R.string.***confirm\_message***);  
  
 *// provide an OK button that simply dismisses the dialog* builder.setPositiveButton(R.string.***button\_delete***,  
 **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** button) {  
 *// use Activity's ContentResolver to invoke  
 // delete on the AddressBookContentProvider* getActivity().getContentResolver().delete(  
 **contactUri**, **null**, **null**);  
 **listener**.onContactDeleted(); *// notify listener* }  
 }  
 );  
  
 builder.setNegativeButton(R.string.***button\_cancel***, **null**);  
 **return** builder.create(); *// return the AlertDialog* }  
 };  
 *// called by LoaderManager to create a Loader* @Override  
 **public** Loader<Cursor> onCreateLoader(**int** id, Bundle args) {  
 *// create an appropriate CursorLoader based on the id argument;  
 // only one Loader in this fragment, so the switch is unnecessary* CursorLoader cursorLoader;  
  
 **switch** (id) {  
 **case *CONTACT\_LOADER***:  
 cursorLoader = **new** CursorLoader(getActivity(),  
 **contactUri**, *// Uri of contact to display* **null**, *// null projection returns all columns* **null**, *// null selection returns all rows* **null**, *// no selection arguments* **null**); *// sort order* **break**;  
 **default**:  
 cursorLoader = **null**;  
 **break**;  
 }  
 **return** cursorLoader;  
 }  
 *// called by LoaderManager when loading completes* @Override  
 **public void** onLoadFinished(Loader<Cursor> loader, Cursor data) {  
 *// if the contact exists in the database, display its data* **if** (data != **null** && data.moveToFirst()) {  
 *// get the column index for each data item* **int** nameIndex = data.getColumnIndex(Contact.***COLUMN\_NAME***);  
 **int** phoneIndex = data.getColumnIndex(Contact.***COLUMN\_PHONE***);  
 **int** emailIndex = data.getColumnIndex(Contact.***COLUMN\_EMAIL***);  
 **int** streetIndex = data.getColumnIndex(Contact.***COLUMN\_STREET***);  
 **int** cityIndex = data.getColumnIndex(Contact.***COLUMN\_CITY***);  
 **int** stateIndex = data.getColumnIndex(Contact.***COLUMN\_STATE***);  
 **int** zipIndex = data.getColumnIndex(Contact.***COLUMN\_ZIP***);  
 *// fill TextViews with the retrieved data* **nameTextView**.setText(data.getString(nameIndex));  
 **phoneTextView**.setText(data.getString(phoneIndex));  
 **emailTextView**.setText(data.getString(emailIndex));  
 **streetTextView**.setText(data.getString(streetIndex));  
 **cityTextView**.setText(data.getString(cityIndex));  
 **stateTextView**.setText(data.getString(stateIndex));  
 **zipTextView**.setText(data.getString(zipIndex));  
 }  
 }  
 *// called by LoaderManager when the Loader is being reset* @Override  
 **public void** onLoaderReset(Loader<Cursor> loader) { }  
}

*//* ***ContactsFragment.java*** *// Fragment subclass that displays the alphabetical list of contact names***package** com.ex.and.abook;  
  
**import** android.content.Context;  
**import** android.database.Cursor;  
**import** android.net.Uri;  
**import** android.os.Bundle;  
**import** android.support.design.widget.FloatingActionButton;  
**import** android.support.v4.app.Fragment;  
**import** android.support.v4.app.LoaderManager;  
**import** android.support.v4.content.CursorLoader;  
**import** android.support.v4.content.Loader;  
**import** android.support.v7.widget.LinearLayoutManager;  
**import** android.support.v7.widget.RecyclerView;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
  
**import** com.ex.and.abook.data.DatabaseDescription.Contact;  
  
**public class** ContactsFragment **extends** Fragment  
 **implements** LoaderManager.LoaderCallbacks<Cursor> {  
 *// callback method implemented by MainActivity* **public interface** ContactsFragmentListener {  
 *// called when contact selected* **void** onContactSelected(Uri contactUri);  
 *// called when add button is pressed* **void** onAddContact();  
 }  
 **private static final int *CONTACTS\_LOADER*** = 0; *// identifies Loader  
 // used to inform the MainActivity when a contact is selected* **private** ContactsFragmentListener **listener**;  
 **private** ContactsAdapter **contactsAdapter**; *// adapter for recyclerView  
 // configures this fragment's GUI* @Override  
 **public** View onCreateView(  
 LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 **super**.onCreateView(inflater, container, savedInstanceState);  
 setHasOptionsMenu(**true**); *// fragment has menu items to display  
 // inflate GUI and get reference to the RecyclerView* View view = inflater.inflate(  
 R.layout.***fragment\_contacts***, container, **false**);  
 RecyclerView recyclerView =  
 (RecyclerView) view.findViewById(R.id.***recyclerView***);  
 *// recyclerView should display items in a vertical list* recyclerView.setLayoutManager(  
 **new** LinearLayoutManager(getActivity().getBaseContext()));  
 *// create recyclerView's adapter and item click listener* **contactsAdapter** = **new** ContactsAdapter(  
 **new** ContactsAdapter.ContactClickListener() {  
 @Override  
 **public void** onClick(Uri contactUri) {  
 **listener**.onContactSelected(contactUri);  
 }  
 }  
 );  
 recyclerView.setAdapter(**contactsAdapter**); *// set the adapter  
 // attach a custom ItemDecorator to draw dividers between list items* recyclerView.addItemDecoration(**new** ItemDivider(getContext()));  
 *// improves performance if RecyclerView's layout size never changes* recyclerView.setHasFixedSize(**true**);  
 *// get the FloatingActionButton and configure its listener* FloatingActionButton addButton =  
 (FloatingActionButton) view.findViewById(R.id.***addButton***);  
 addButton.setOnClickListener(  
 **new** View.OnClickListener() {  
 *// displays the AddEditFragment when FAB is touched* @Override  
 **public void** onClick(View view) {  
 **listener**.onAddContact();  
 }  
 }  
 );  
 **return** view;  
 }  
 *// set ContactsFragmentListener when fragment attached* @Override  
 **public void** onAttach(Context context) {  
 **super**.onAttach(context);  
 **listener** = (ContactsFragmentListener) context;  
 }  
 *// remove ContactsFragmentListener when Fragment detached* @Override  
 **public void** onDetach() {  
 **super**.onDetach();  
 **listener** = **null**;  
 }  
 *// initialize a Loader when this fragment's activity is created* @Override  
 **public void** onActivityCreated(Bundle savedInstanceState) {  
 **super**.onActivityCreated(savedInstanceState);  
 getLoaderManager().initLoader(***CONTACTS\_LOADER***, **null**, **this**);  
 }  
 *// called from MainActivity when other Fragment's update database* **public void** updateContactList() {  
 **contactsAdapter**.notifyDataSetChanged();  
 }  
 *// called by LoaderManager to create a Loader* @Override  
 **public** Loader<Cursor> onCreateLoader(**int** id, Bundle args) {  
 *// create an appropriate CursorLoader based on the id argument;  
 // only one Loader in this fragment, so the switch is unnecessary* **switch** (id) {  
 **case *CONTACTS\_LOADER***:  
 **return new** CursorLoader(getActivity(),  
 Contact.***CONTENT\_URI***, *// Uri of contacts table* **null**, *// null projection returns all columns* **null**, *// null selection returns all rows* **null**, *// no selection arguments* Contact.***COLUMN\_NAME*** + **" COLLATE NOCASE ASC"**); *// sort order* **default**:  
 **return null**;  
 }  
 }  
 *// called by LoaderManager when loading completes* @Override  
 **public void** onLoadFinished(Loader<Cursor> loader, Cursor data) {  
 **contactsAdapter**.swapCursor(data);  
 }  
 *// called by LoaderManager when the Loader is being reset* @Override  
 **public void** onLoaderReset(Loader<Cursor> loader) {  
 **contactsAdapter**.swapCursor(**null**);  
 }  
}

*//* ***ContactsAdapter.java*** *// Subclass of RecyclerView.Adapter that binds contacts to RV***import** com.ex.and.abook.data.DatabaseDescription.Contact;  
  
**public class** ContactsAdapter  
 **extends** RecyclerView.Adapter<ContactsAdapter.ViewHolder> {  
 *// interface implemented by ContactsFragment to respond  
 // when the user touches an item in the RecyclerView* **public interface** ContactClickListener {  
 **void** onClick(Uri contactUri);  
 }  
 *// nested subclass of RecyclerView.ViewHolder used to implement  
 // the view-holder pattern in the context of a RecyclerView* **public class** ViewHolder **extends** RecyclerView.ViewHolder {  
 **public final** TextView **textView**;  
 **private long rowID**;  
 *// configures a RecyclerView item's ViewHolder* **public** ViewHolder(View itemView) {  
 **super**(itemView);  
 **textView** = (TextView) itemView.findViewById(android.R.id.***text1***);  
  
 *// attach listener to itemView* itemView.setOnClickListener(  
 **new** View.OnClickListener() {  
 *// executes when the contact in this ViewHolder is clicked* @Override  
 **public void** onClick(View view) {  
 **clickListener**.onClick(Contact.*buildContactUri*(**rowID**));  
 }  
 }  
 );  
 }  
 *// set the database row ID for the contact in this ViewHolder* **public void** setRowID(**long** rowID) {  
 **this**.**rowID** = rowID;  
 }  
 }  
 *// ContactsAdapter instance variables* **private** Cursor **cursor** = **null**;  
 **private final** ContactClickListener **clickListener**;  
 *// constructor* **public** ContactsAdapter(ContactClickListener clickListener) {  
 **this**.**clickListener** = clickListener;  
 }  
 *// sets up new list item and its ViewHolder* @Override  
 **public** ViewHolder onCreateViewHolder(ViewGroup parent, **int** viewType) {  
 *// inflate the android.R.layout.simple\_list\_item\_1 layout* View view = LayoutInflater.*from*(parent.getContext()).inflate(  
 android.R.layout.***simple\_list\_item\_1***, parent, **false**);  
 **return new** ViewHolder(view); *// return current item's ViewHolder* }  
 *// sets the text of the list item to display the search tag* @Override  
 **public void** onBindViewHolder(ViewHolder holder, **int** position) {  
 **cursor**.moveToPosition(position);  
 holder.setRowID(**cursor**.getLong(**cursor**.getColumnIndex(Contact.***\_ID***)));  
 holder.**textView**.setText(**cursor**.getString(**cursor**.getColumnIndex(  
 Contact.***COLUMN\_NAME***)));  
 }  
@Override *// returns the number of items that adapter binds*  
 **public int** getItemCount() {  
 **return** (**cursor** != **null**) ? **cursor**.getCount() : 0;  
 }  
 *// swap this adapter's current Cursor for a new one* **public void** swapCursor(Cursor cursor) {  
 **this**.**cursor** = cursor;  
 notifyDataSetChanged();  
 }  
}

*<?***xml version="1.0" encoding="utf-8"***?>*<**android.support.design.widget.CoordinatorLayout  
 android:id="@+id/coordinatorLayout"  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:fitsSystemWindows="true"  
 tools:context=".MainActivity"**>  
  
 <**android.support.design.widget.AppBarLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:theme="@style/AppTheme.AppBarOverlay"**>  
  
 <**android.support.v7.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="?attr/colorPrimary"  
 app:popupTheme="@style/AppTheme.PopupOverlay"**/>  
 </**android.support.design.widget.AppBarLayout**>  
  
 <**include layout="@layout/content\_main"**/>  
  
</**android.support.design.widget.CoordinatorLayout**>

<**FrameLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**android.support.v7.widget.RecyclerView  
 android:id="@+id/recyclerView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**/>  
  
 <**android.support.design.widget.FloatingActionButton  
 android:id="@+id/addButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="top|end"  
 android:layout\_margin="@dimen/fab\_margin"  
 android:src="@drawable/ic\_add\_24dp"**/>  
</**FrameLayout**>

<**FrameLayout  
 android:id="@+id/fragmentContainer"  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingBottom="@dimen/activity\_vertical\_margin"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="@dimen/activity\_vertical\_margin"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"  
 tools:context=".MainActivity"**/>

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:baselineAligned="false"  
 android:divider="?android:listDivider"  
 android:orientation="horizontal"  
 android:paddingBottom="@dimen/activity\_vertical\_margin"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="@dimen/activity\_vertical\_margin"  
 android:showDividers="middle"  
 android:weightSum="3"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"**>  
  
 <**fragment  
 android:id="@+id/contactsFragment"  
 android:name="com.ex.and.abook.ContactsFragment"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_marginEnd="@dimen/divider\_margin"  
 android:layout\_marginRight="@dimen/divider\_margin"  
 android:layout\_weight="1"  
 tools:layout="@layout/fragment\_contacts"** />  
  
 <**FrameLayout  
 android:id="@+id/rightPaneContainer"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_marginStart="@dimen/divider\_margin"  
 android:layout\_marginLeft="@dimen/divider\_margin"  
 android:layout\_weight="2"**/>  
</**LinearLayout**>