Lab 4.4 – Fragments

Overview:

In this lab you will convert activities to fragments. You will utilize the Android compatability package to use classes that were introduced in Android 3.0 and have been made available to Android 1.5 through 2.3. The purpose of fragments is to allow the UI to display effectively in both small screens (where a single fragment will be shown at one time) and on larger screens where multiple fragments will be shown at the same time. The Fragments API is Androids suggested technique for developing apps.

Section 1 – Create a new Event detail view activity.

Step	Description
1.	Add an additional activity for showing Event details so that we have multiple activities (fragments) to display at one time on larger screens. You will create a new activity for viewing event detail information. This activity will be shown when the user clicks on an item in the event list view.

a.	Create a new layout called “res/layout/event_detail.xml” for displaying the detail of an event item.
	
	Display data from the Event object. Following are some suggestions for the data to display.
	
	– Street
	– City
	– Description
	– Rating
	– Distance
	
	The TextView for description should be nested inside a “ScrollView” since this field may contain a large amount of text that would extend beyond the boundaries of the screen.
	
	The layout should look similar to the following:
	
	`<?xml version="1.0" encoding="utf-8"?>`
	`<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"`
	`android:layout_width="fill_parent"`
	`android:layout_height="fill_parent"`
	`android:layout_margin="10sp"`
	`android:orientation="vertical" >`
	
	`<TextView`
	`android:id="@+id/event_street"`
	`android:layout_width="fill_parent"`
	`android:layout_height="wrap_content"`
	`android:layout_margin="10dip"`
	`android:padding="6dip" />`
	
	`<ScrollView`
	`android:layout_width="fill_parent"`
	`android:layout_height="wrap_content" >`
	
	`<TextView`

```
                android:id="@+id/event_description"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_gravity="clip_vertical"
                android:layout_weight="1"
                android:gravity="top"
                android:padding="6dip" />

        </ScrollView>

    </LinearLayout>
```

b. Create a new Activity for displaying the detail of an event item. The activity should be called "com.garagze.EventDetailActivity.java".

Implement the method "onCreate" and add code to the method to display the view.

```
super.onCreate(bundle);
this.setContentView(R.layout.event_detail);
```

c. Add additional code to the "onCreate" method of "EventDetailActivity.java" to display the event item by using data passed with the intent. You will create the calling intent in a subsequent step.

```
// get references to the view elements in the detail layout
TextView street = (TextView) findViewById(R.id.event_street);
TextView description = (TextView) findViewById(R.id.event_description);

Intent intent = getIntent();

// Get the event data from the intent and update view
street.setText(intent.getStringExtra("street"));
description.setText(intent.getStringExtra("description"));
```

d. Add Activity to the Android Manifest.

Open the file "AndroidManifest.xml" and add the following element to the application tag.

```
<activity
    android:name=".EventDetailActivity">
</activity>
```

e. Update the "MainActivity.java" to call activity for displaying the detail of an event item.

The "EventDetailActivity" must display an individual event selected from the prior list which requires that some kind of data be passed from the list. Two primary alternatives are to pass the data for the event or to pass an id from which the event object can be acquired.

The ListView itself saves a map of the data which we can then use so we will take the approach of passing all the data through an intent.

Add the following code to "MainActivity.java". Place it an the end of the "displayListView" method. This code overrides the method that gets called when a user selects an item from the ListView.

```
final Context context = this;

listView.setOnItemClickListener(new OnItemClickListener() {
```

```
        public void onItemClick
           (AdapterView<?> parent, View view, int position, long id) {
           Log.v("MainActivity ", "Calling EventDetailActivity.");
           Intent intent = new Intent(context, EventDetailActivity.class);

           Event event = events.get(position);
           intent.putExtra("street", event.getStreet());
           intent.putExtra("description", event.getDescription());

           startActivity(intent);
        }
     });
```

Note: You may instead use the following method to get the context when creating the new Intent: `getBaseContext()`

f. View that the application works when changing from portrait to landscape. You may also view the app on a device with a larger screen. Rotate the screen to see what happens. You can rotate the emulator by typing the following

KEYPAD_9 or CTRL-F12

Or on the Mac

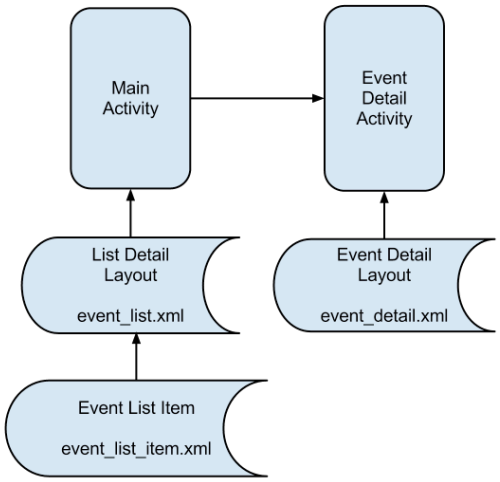
Ctrl + fn + f12

Section 2 – Convert Activities to Fragments

Convert the list and detail activities to fragments. This is a multistep process that involves the following individual steps.

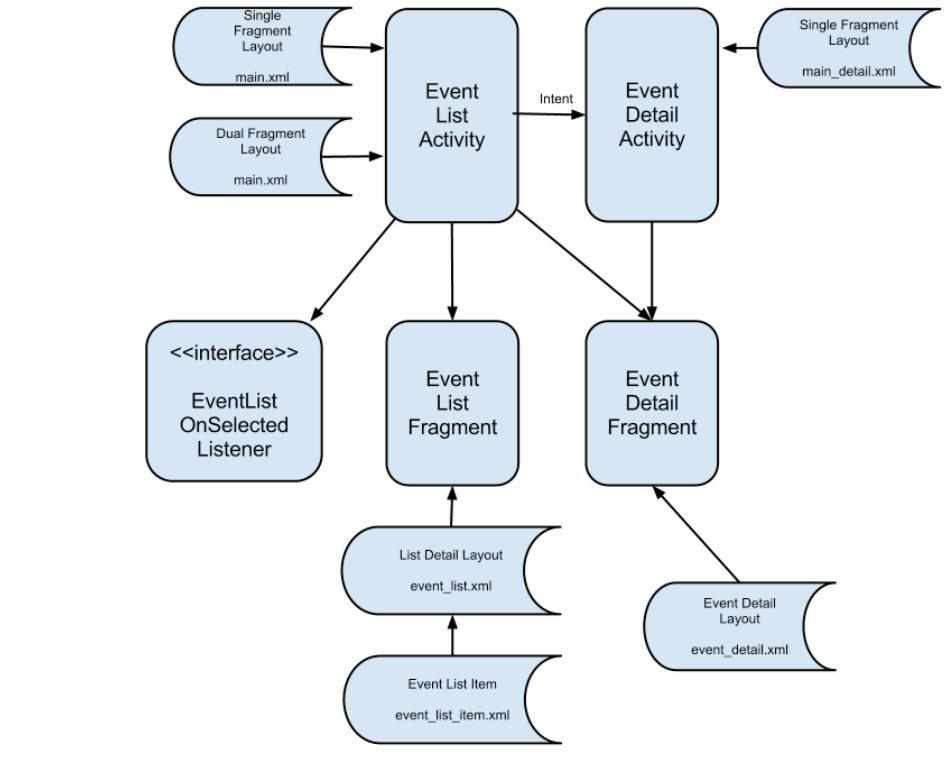
1. Convert the list activty (MainActivity) to an Activity and Fragment
2. Convert Detail activity (EventDetailActivity) to Activity and Fragment
3. Run new activities using fragments
4. Create Fragment layout for multiple fragments
5. Convert EventListActivity to call the detail fragment properly

This diagram depicts the components and their relationship before the conversion:



Google Docs: Android Fragment Conversion

This diagram depicts the components and their relationship after the conversion:



Google Docs: Android Fragment Conversion

Note: Prepare the project by adding the jar file for the compatability package.

1.	Convert the list activity (MainActivity) to an Activity using a Fragment

a. Convert the list activity to a fragment. Create a new fragment called "EventListFragment" which extends "android.app.Fragment". If there are compiler errors for "Fragment" then the compatability library was probably not added properly.

Override the "onCreateView" method. Add code to do the following:

- Create a LinearLayout view by inflating the "list_event" layout
- Get the list of events from EventService
- Create a new EventArrayAdapter
- Attach the adapter to the ListView element

Most of this code can be harvested from "MainActivity.java"

The following code can be used to over-ride "onCreateView" in "EventListFragment.java":

```
List<Event> events;

public View onCreateView
   (LayoutInflater inflater, ViewGroup container, Bundle bundle) {

   LinearLayout view =
      (LinearLayout) inflater.inflate(R.layout.list_event, container, false);

   events = EventService.getAllEvents(this.getActivity());

   final ArrayAdapter<Event> arrayAdapter =
      new EventArrayAdapter
         (this.getActivity(), R.layout.event_list_item, events);

   ListView listView = (ListView) view.findViewById(R.id.eventlistview);
   listView.setAdapter(arrayAdapter);

   return view;
}
```

You will receive an error in the code due to an invalid reference. Copy "res/layout/main.xml" to "event_list.xml"

b. Create a new layout for the activity with fragment. Call it "main.xml" and place it in the "res/layout" directory. The layout contains an element to declare the fragment. The fragment will be created when an activity using this layout inflates the layout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <fragment
        android:id="@+id/list_fragment"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:name="com.garagze.EventListFragment" >
    </fragment>

</LinearLayout>
```

c. Create a new activity to manage the fragment. Create a new class called " EventListActivity" which

extends "FragmentActivity". This activity should be registered in the AndroidManifest. Also, assign it to the launcher intent so that it becomes the entry point for the application. It can not just extend "Activity" since it needs to create a fragment automatically. Over-ride the "onCreate" method and add code to inflate the "main" layout.

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
}
```

All the activity does is inflate the layout which builds the fragment.

2. Convert the detail activity (EventDetailActivity) to an Activity calling a Fragment

a. Convert the detail activity to a fragment. Create a new fragment called "EventDetailFragment" which extends "import android.support.v4.app.Fragment". Override the "onCreateView" method. Inflate the layout "event_detail.xml".

```
LinearLayout view;

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {

  view = (LinearLayout)
      inflater.inflate(R.layout.event_detail, container, false);

  // get references to the view elements in the detail layout
  TextView street = (TextView)

  view.findViewById(R.id.event_street);
  TextView description = (TextView)
      view.findViewById(R.id.event_description);

  Intent intent = getActivity().getIntent();

  // Get the event data from the intent and update view
  street.setText(intent.getStringExtra("street"));
  description.setText(intent.getStringExtra("description"));

  return view;
}
```

b. Create a new layout for the activity with fragment. Call it "main_detail.xml" and place it in the "res/layout" directory. The layout contains an element to declare the fragment. The fragment will be created when an activity using this layout inflates the layout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <fragment
        android:id="@+id/list_fragment"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
```

```
            android:name="com.garagze.EventDetailFragment" >
        </fragment>

    </LinearLayout>
```

c. Create a new activity to manage the fragment. Create a new class called " EventDetailActivity" which extends "FragmentActivity". This activity should be registered in the AndroidManifest. . Over-ride the "onCreate" method and add code to inflate the "main_detail" layout.

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main_detail);
}
```

All the activity does is inflate the layout which builds the fragment.

3. Run the application and verify that it runs the same as before.

4. Create a layout for showing both a list and detail fragments.

a. Create a layout called "main.xml" for showing both fragments at the same time when the screen is large enough. Place this is a new directory called "res/layout-land".

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <fragment
        android:id="@+id/list_fragment"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:name="com.garagze.EventListFragment" />

    <fragment
        android:id="@+id/detail_fragment"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:name="com.garagze.EventDetailFragment" />

</LinearLayout>
```

This layout will be used when the device is in landscape orientation.

5. Modify "EventListActivity" to show both fragments and to display the event detail when an item is selected in the list.

a. Add a method to the detail fragment "EventDetailFragment" to display a specific event.

```
public void update(Event event) {

  // get references to the view elements in the detail layout
  TextView street = (TextView) view.findViewById(R.id.detail_event_street);
```

```
  TextView description = (TextView)
      view.findViewById(R.id.detail_event_description);

  street.setText(event.getStreet());
  description.setText(event.getDescription());
}
```

b. Add listeners to the list fragment "EventListFragment" in the "onCreateView" method. Add listeners for both "onItemSelected" and "onItemClick":

```
listView.setOnItemSelectedListener(new ListView.OnItemSelectedListener() {
==
  @Override
  public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    Event event = events.get(position);
    Log.v(TAG, "Event has been selected: " + event.getStreet());
    eventSelectedListener.onEventSelected(event);
  }

  @Override
  public void onNothingSelected(AdapterView<?> parent) {
    // do nothing
  }


});

listView.setOnItemClickListener(new ListView.OnItemClickListener() {

  @Override
  public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    Event event = events.get(position);
    Log.v(TAG, "Event has been clicked: " + event.getStreet());
    eventSelectedListener.onEventSelected(event);
  }

});
```

Create an "OnEventSelectedInterface" in "EventListFragment"

```
public interface OnEventSelectedListener {
      public void onEventSelected(Event event);
}
```

Add the following code to "EventListFragment" to register the "EventListActivity" as a listener

```
@Override
public void onAttach(Activity activity) {
      super.onAttach(activity);
      try {
            eventSelectedListener = (OnEventSelectedListener) activity;
      } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString() +
            " must implement OnEventSelectedListener");
      }
}
```

Note: Remember to create an instance variable for "eventSelecteListener" of type "OnEventSelectedListener".

c. When an item is selected from the list, communicate through the activity to display the item. In

"EventListActivity" implement the "OnEventSelectedListener" and implement the "onEventSelected" method:

```
@Override
public void onEventSelected(Event event) {

      EventDetailFragment viewer = (EventDetailFragment)
        getSupportFragmentManager().
        findFragmentById(R.id.detail_fragment);

      if (viewer == null || !viewer.isInLayout()) {
            Intent showContent =
               new Intent(getApplicationContext(),
               EventDetailActivity.class);
            showContent.putExtra("street", event.getStreet());
            showContent.putExtra("description", event.getDescription());
            startActivity(showContent);
      } else {
            viewer.update(event);
      }
}
```

Run the application in portrait mode. Only the list should display. Switch the device to landscape mode. The list and detail fragments should appear at the same time.

Optional:

Refactor the code so that it will work on version below 3.0. Use the compatability library.

When the screen orientation changes the performance is a little slow. Can this be improved?

See the following link to explore one technique for this:

http://android-developers.blogspot.com/2009/02/faster-screen-orientation-change.html

References:

http://android-developers.blogspot.com/2011/02/android-30-fragments-api.html

http://android-developers.blogspot.com/2011/09/preparing-for-handsets.html