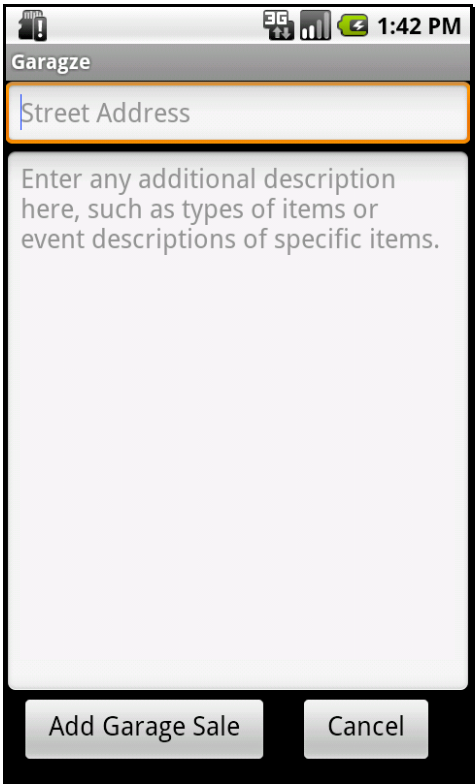


Lab 6.1 – New Activity

Overview:

In this lab you will create a new activity to allow a user to enter a new event item. This activity will be initiated by an Intent that will be created when the user taps on the appropriate menu button.

Step	Description
1.	Create a new layout for entering event information.
a.	<p>Create a new layout using the Android XML wizard. Call the layout “res/layout/add_event.xml”.</p> <p>Add the following view elements</p> <ul style="list-style-type: none">– EditText view for entering address– EditText view for entering description– Button for adding event– Button for cancelling activity <p>Although you do not need to match this layout exactly, yours should look something like the following:</p>  <p>Use the hint attribute to show text in the edit box.</p> <pre>android:hint="@string/event_description_hint"</pre> <p>You can use the following layout or create your own:</p>

```

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

    <EditText
        android:id="@+id/address"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:ems="10"
        android:hint="@string/address_hint">
        <requestFocus />
    </EditText>

    <EditText
        android:id="@+id/description"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/add_garage_sale"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/address"
        android:ems="10"
        android:hint="@string/desc_hint"/>

    <Button
        android:id="@+id/add_garage_sale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="25dp"
        android:text="@string/add_button_text" />

    <Button
        android:id="@+id/cancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/add_garage_sale"
        android:layout_alignBottom="@+id/add_garage_sale"
        android:layout_alignParentRight="true"
        android:text="@string/cancel_button_text" />

</RelativeLayout>

```

- b. Externalize string values in the layout as much as possible. For example, the text for the view showing the event name could have a default value. Rather than hard coding this in the layout, reference it in the layout and put the value in “res/values/strings.xml” as shown below:

In “res/layout/add_event.xml” replace direct string references such as the following:

```
android:text="Add Garage Sale"
```

with an external string reference such as the following:

```
android:text="@string/add_button_text"
```

	<p>and then add the value for “add_button_text” to “res/values/strings.xml as follows:</p> <pre><string name="add_button_text">Add Garage Sale</string></pre> <p>Apply this technique to all the strings in the XML layout files.</p>
2.	<p>Create an activity to display the layout for a new event and save the event data using the event service.</p>
a.	<p>Create a new activity called “AddEventActivity.java” in “com.garagze” by right clicking on the “com.garagze” package in the Package Explorer, and selecting “New > Class” from the popup menu.</p> <p>Fill in “AddEventActivity” for the “Name” field in the dialog.</p> <p>In the “Superclass” field, enter “android.app.Activity” (you can also just type Activity and hit Ctrl-Space on Windows and Linux or Cmd-Space on the Mac, to invoke code assist and find the right package and class).</p> <p>Click the “Finish” button.</p> <p>In the resulting “AddEventActivity” class, right click in the editor window and select “Source > Override/Implement Methods:</p> <p>Scroll down through the checklist in the dialog until you see “onCreate(Bundle)” and check the box next to it.</p> <p>Click OK.</p>
b.	<p>Over-ride the “onCreate” method. Modify the code to make it look like the following:</p> <pre>public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.add_event); }</pre>
3.	<p>Declare the new activity in the application manifest.</p>
a.	<p>Open the file “AndroidManifest.xml”</p> <p>Add the following element to the application tag.</p> <pre><activity android:name=".AddEventActivity"> </activity></pre>
4.	<p>Call the activity from the main menu in “MainActivity.java”</p>
a.	<p>Add the following code to the “addEvent” method in the “MainActivity.java” file.</p> <pre>Intent intent = new Intent(this, AddEventActivity.class); startActivity(intent);</pre> <p>Run the application. Press the Menu button and select “Add Event”. The add event activity should appear.</p>
5.	<p>Add an event handler to the button. This code should be added to the end of the “AddEventActivity.java” file. Make sure you are editing the correct file.</p>

a.	<p>The “Add Garage Sale” button should create an event object and run the <code>EventService.addEvent</code> method. Add the following code to the end of the “onCreate” method in “AddEventActivity.java”. Be sure that the id for the view elements correspond to what you named them in the “add_event.xml” file.</p> <pre> final Button addButton = (Button) findViewById(R.id.add_button); Log.v("AddEvent", "Declaring button"); addButton.setOnClickListener(new View.OnClickListener() { public void onClick(View view) { final EditText addressView = (EditText) findViewById(R.id.event_address); String address = addressView.getText().toString(); final EditText descriptionView = (EditText) findViewById(R.id.event_description); String description = descriptionView.getText().toString(); Event event = new Event(); event.setStreet(address); event.setDescription(description); event.setRating(3.0F); event.setCity("Naperville"); EventService.addEvent(event); finish(); } }); </pre>
b.	<p>Add an event handler to the cancel button also. The cancel button should just call the activity “finish()” method.</p> <p>Optional: There is another technique for assigning event handlers to buttons. Declare an “onClick” attribute for the button in the XML layout file and reference a method name in the activity. The method in the activity should then contain the event handler code. The instructor will demonstrate this technique during the review of the exercise.</p>
6.	<p>When the “AddEventActivity” finishes, Android automatically returns to the item on the stack just below it which is the “MainActivity”. As described in the activity lifecycle, when an item on the stack comes to the top, it’s “onResume” method is called. You need to provide an implementation of this method which will create a new “EventArrayAdapter” to recognize the update to the events data.</p>
a.	<p>Add the following method to the “MainActivity.java” file.</p> <pre> @Override protected void onResume() { super.onResume(); Log.v("MainActivity", "Running onResume method"); displayListView(); } </pre> <p>Optional: Explore the error you get if you do not provide an “onResume” method.</p>
7.	<p>Run the application and add a new event. Verify that the event is now appearing in the list view.</p> <p>The event is only persisted while the application is active. If you run the application from Eclipse</p>

	again, the new events will be gone.
8.	Optional: Modify the "AddEventActivity" to allow the editing of existing events.
a.	Create an event handler in "MainActivity" on the ListView view to be called when an item is selected. Research "setOnItemClickListener"
b.	In the event handler just created in "MainActivity" save the event (or event id) to an intent before starting the "AddEventActivity". Research saving data in intents.
c.	In the event handler just created in "MainActivity" start the "AddEventActivity"
d.	In "AddEventActivity" get the event data from the intent.
e.	In "AddEventActivity" display the event data.
f.	In "AddEventActivity" change the button label to "Update".
g.	When the button is clicked, update the event. You may have to create additional methods in "EventService".