CSCI 4100 Laboratory 8

More Bird Sighting

Introduction

In the previous laboratory we produced two versions of the bird sighting application. In this laboratory we will produce the final version of the application that allows us to add new bird sightings. This version is based on the second version from the previous laboratory and you can use it as a starting point. The main addition in this version is a fragment that allows the user to enter a new bird sighting. This fragment is shown in figure 1 and the updated main screen is shown in figure 2.

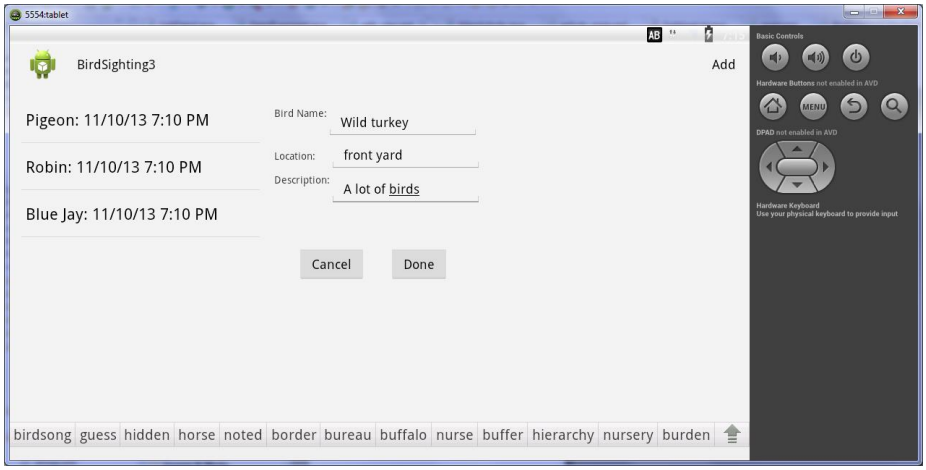

Figure 1 Adding a new sighting

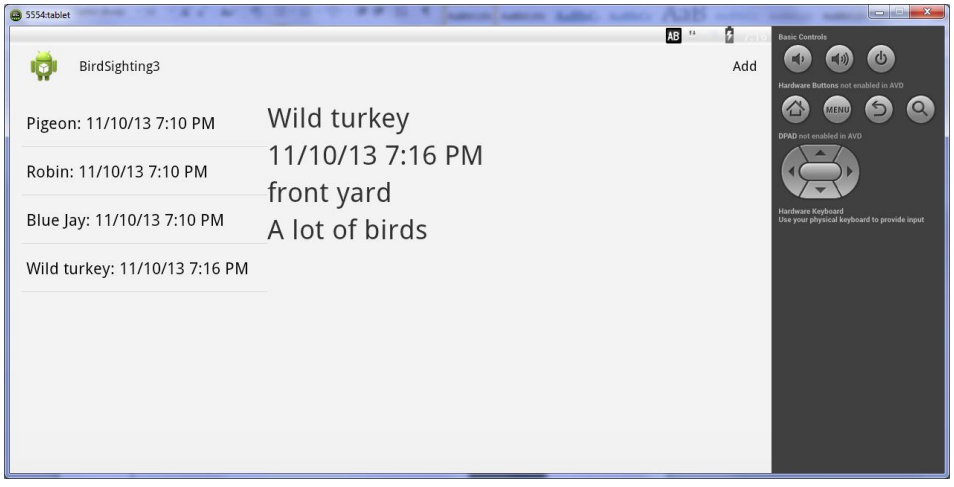

Figure 2 Updated main screen

Building the New Application

You can start this laboratory by making a copy of your second application from the previous laboratory. You can call this new application BirdSighting3. If you do a copy of the project you will need to change all the references to birdsighting2 to birdsighting3. Eclipse will do some of this for you using refactoring, but you will need to do some of it yourself. Check the manifest file and the package name in each file.

Our new fragment will alternate with our previous detail fragment. This is the first major change that we need to make in our previous version of the application. Previously the detail fragment was created automatically when the layout file was read by the main activity. Unfortunately fragments that are created in this way cannot be replaced, which is what we need to do. To handle this problem the main_activity.xml file must be modified to replace the second fragment by a FrameLayout, which gives:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    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"
    tools:context=".MainActivity" >

    <fragment
        android:id="@+id/sighting_fragment"
        android:name="ca.uoit.MarkGreen.birdsighting3.SightingList"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        tools:layout="@layout/sighting_list" />

    <FrameLayout
        android:id="@+id/detail_fragment"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="2"
        tools:layout="@layout/detail_view" />

</LinearLayout>
```

The other major change that we need to make involves the sightings array. In the previous version we used a standard Java array for this. Unfortunately, this array is of fixed size and we can't add new sightings to the end of it. The easiest solution to this problem is to replace the Java array with an ArrayList. The declaration of this list is:

```
ArrayList<Sighting> sightings;
```

Note, you will also need to make this change in the other files that use the sightings array. We can no longer initialize this array in its declaration and instead its initialization is moved to the start of the onCreate() method. The new version of the onCreate() method is:

```
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		sightings = new ArrayList<Sighting>();
		sightings.add(new Sighting("Pigeon", "everywhere","An ugly bird"));
		sightings.add(new Sighting("Robin", "back yard", "The early bird gets
the worm"));
		sightings.add(new Sighting("Blue Jay", "AC Centre", "Let's play ball"));
		setContentView(R.layout.activity_main);
		myDetailFragment = new DetailFragment();
		FragmentTransaction transaction =
getFragmentManager().beginTransaction();
		transaction.replace(R.id.detail_fragment, myDetailFragment);
		transaction.commit();
	}
```

Note that the initialization must occur before the layout is read in, since the SightingList fragment uses this array to fill the list view when it's created.

As you can see we have added an item to the Action Bar, the "Add" item on the right of figures 1 and 2. This is a two step process. First, the main.xml file in the menu folder needs to be modified to include the new item. Second, we need to add code to the main activity to respond to the item being selected. Start by double clicking main.xml in the menu folder. This will give you something like figure 3. This figure shows the resulting menu file.

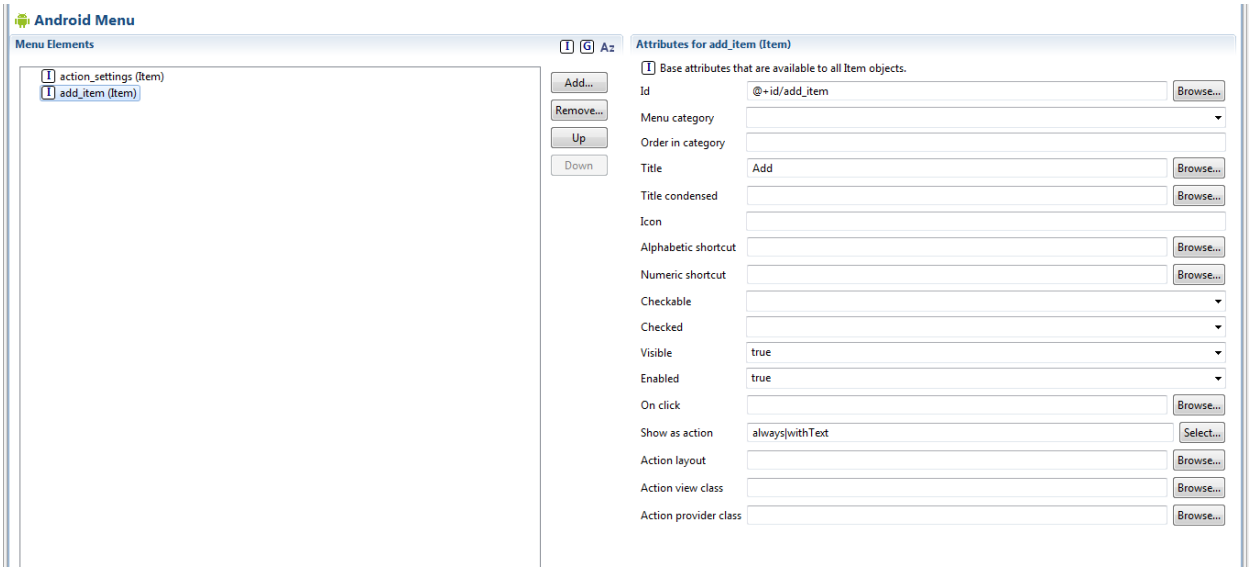

Figure 3 Menu editing

To create a new menu item press the "Add…" button and a window like the one in figure 4 will be shown. Make sure that the top radio button is selected and the double click Item and you will

get a window like figure 3. Make sure that "visible" and "enabled" are true or the menu item won't work.

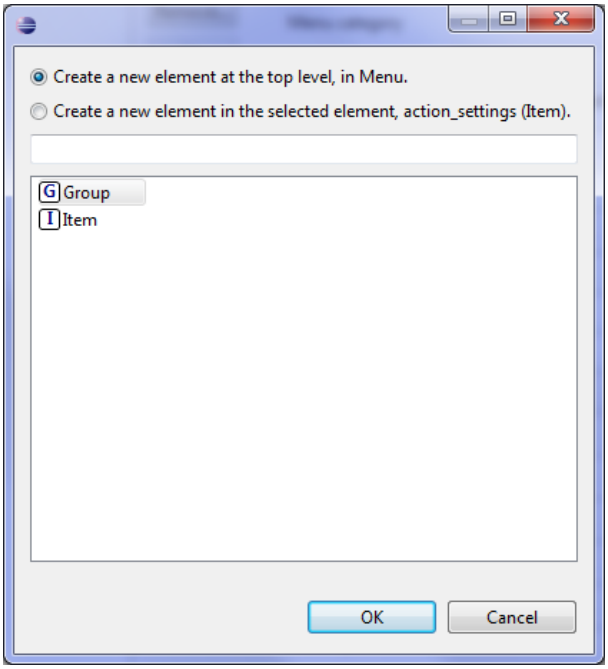

Figure 4 Adding a new menu item

Now add the following code the main activity:

```
        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            // Handle presses on the action bar items
            switch (item.getItemId()) {
               case R.id.add_item:
                     myEditDetail = new EditDetail();
                     FragmentTransaction transaction =
getFragmentManager().beginTransaction();
                     transaction.replace(R.id.detail_fragment, myEditDetail);
                     transaction.commit();
                     return true;
                 default:
                    return super.onOptionsItemSelected(item);
            }
        }
```

In response to the button press a new EditDetail fragment is created and then a fragment transaction is created to replace the current detail fragment with this new fragment.

Now let's turn to our new fragment, the EditDetail fragment. We will start by doing the screen layout, so create a new layout called edit_view.xml. You can use the relative layout as the root view for this layout. The layout is shown in figure 5. It consists of three text views to label the information to be entered, three plain edit text widgets to enter the text and two buttons to signal

cancel and done. Be careful when you are using the relative layout and pay attention to the feedback as you position widgets.

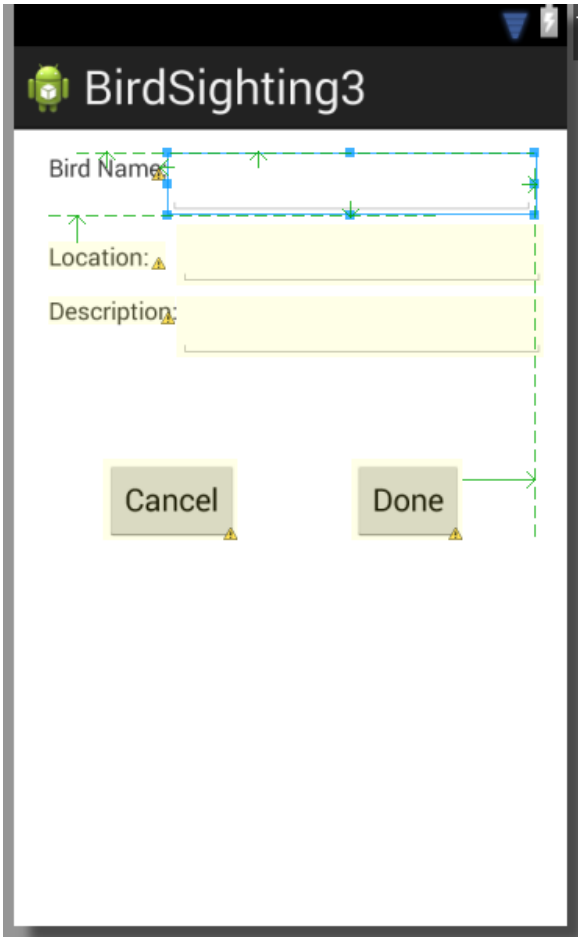

Figure 5 New fragment screen layout

Now we turn to the fragment program code. There is a lot of similarity between this fragment and our detail fragment, so you might want to review it. There is one complication here; for the buttons we cannot set the onClick method through the interface builder. When we do this Android assumes that the onClick methods are in the activity and not in the fragment. To solve this problem we set the event handlers using program code. First, the event handlers are:

```
	private OnClickListener myDoneListener = new OnClickListener() {
		public void onClick(View view) {
			Sighting sighting = new Sighting(view1.getText().toString(),
view2.getText().toString(), view3.getText().toString());
			((MainActivity) myActivity).editDone(sighting);
		}
	};


	private OnClickListener myCancelListener = new OnClickListener() {
		public void onClick(View view) {
```

```
                ((MainActivity) myActivity).editCancel();
            }
        };
```

Both of these event handlers call back to the main activity to signal that entering the new sighting has been completed. For the done button we also create a new Sighting object that we pass to the main activity.

The code for the onCreateView() method is shown below. There is nothing really new in this method.

```
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
           View view;
        view = inflater.inflate(R.layout.edit_view, container, false);
        myActivity = getActivity();
        sightings = ((MainActivity) myActivity).sightings;
        view1 = (EditText) view.findViewById(R.id.editText1);
        view2 = (EditText) view.findViewById(R.id.editText2);
        view3 = (EditText) view.findViewById(R.id.editText3);
        Button cancel = (Button) view.findViewById(R.id.button1);
        cancel.setOnClickListener(myCancelListener);
        Button done = (Button) view.findViewById(R.id.button2);
        done.setOnClickListener(myDoneListener);
        return view;
    }
```

Now all we need to do is wire everything together. Back in the main activity; we add the two methods that are called by the new fragment:

```
    public void editCancel() {
            FragmentTransaction transaction =
getFragmentManager().beginTransaction();
            transaction.replace(R.id.detail_fragment, myDetailFragment);
            transaction.commit();
            myDetailFragment.displayDetail(0);
    }

    public void editDone(Sighting sighting) {
            mySightingList.addSighting(sighting);
            FragmentTransaction transaction =
getFragmentManager().beginTransaction();
            transaction.replace(R.id.detail_fragment, myDetailFragment);
            transaction.commit();
    }
```

The editCancel method replaces the fragment used to enter a new sighting with the detail fragment, and then we have the detail fragment display the first sighting. The editDone method does the same thing and in addition informs the sighting list that a new sighting has been created.

The sighting list fragment has been changed slightly to accommodate the addition of new sightings. We start by making the array adapter a class variable:

ArrayAdapter<Sighting> adapter;

Then we add the addSighting() method:

```
public void addSighting(Sighting sighting) {
	adapter.add(sighting);
}
```

This method lets the adapter do all the work in updating our underlying model.

Because we have change the array to an ArrayList we need to make some changes in DetailFragment where we referenced this array. The following modified displayDetail() method shows how this is done. You will also need to change the onCreateView() method in the same way.

```
public void displayDetail(int p) {
	Sighting sighting;
	sighting = sightings.get(p);
	view1.setText(sighting.bird);
	view2.setText(df.format(sighting.when));
	view3.setText(sighting.location);
	view4.setText(sighting.description);
}
```

Laboratory Report

Make all of the modifications and show the correctly functioning application to the TA.