IT332: Mobile Application Development

Lecture # 14 : Using Fragment Arguments

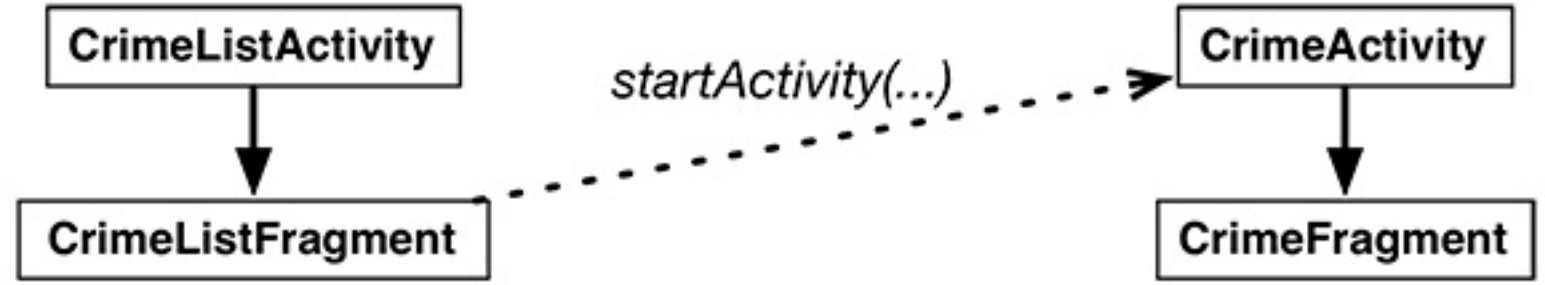
Muhammad Imran

Outline

- Final Objective Today
- Starting an Activity from a Fragment
- The downside to direct retrieval
- Fragment Arguments
- Retrieving arguments
- Reloading the List (RecyclerView)
- Getting Results with Fragments

Final Objective Today

- We will get the list and the detail parts of CrimeReporting App working together
- When a user presses an item in the list of crimes, a new CrimeActivity hosting a CrimeFragment will appear and display the details for that instance of Crime



- We will have CrimeListFragment start an instance of CrimeActivity.

Starting an Activity from a Fragment

- You call the **Fragment.startActivity(Intent)** method, which calls the corresponding Activity method behind the scenes

```
Intent intent = new Intent(getActivity(), CrimeActivity.class);
startActivity(intent);
```

- CrimeListFragment can create an explicit intent that names the CrimeActivity class
- getActivity() method can be used to get the hosting activity as the Context object

Putting an extra data to Intent

- After creating an explicit intent, you call putExtra(…) and pass in a string key and the value the key maps to (the crimeId).
- UUID is a Serializable object.

The downside to direct retrieval

- Having the fragment access the intent that belongs to the hosting activity makes for simple code.
- However, it costs you the **encapsulation** of your fragment.
- CrimeFragment is no longer a reusable building block because it expects that it will always be hosted by an activity whose Intent defines an extra named com.nomadlearner.criminalintent.crime_id
- CrimeFragment with this dependency, cannot be used with just any activity.

A better Solution

- A better solution is to supply the crime ID someplace that belongs to CrimeFragment rather than keeping it in CrimeActivity's personal space.
- The CrimeFragment could then retrieve this data without relying on the presence of a particular extra data in the activity's intent.
- The "**someplace**" that belongs to a fragment is known as its **arguments bundle.**

Fragment Arguments

- Every fragment instance can have a Bundle object attached to it.
- This bundle contains key-value pairs that work just like the intent extras of an Activity.
- Each pair is known as an argument.
- To create fragment arguments, we can create a Bundle object and place data using “put” methods

```
Bundle args = new Bundle();
args.putSerializable(ARG_MY_OBJECT, myObject);
args.putInt(ARG_MY_INT, myInt);
args.putCharSequence(ARG_MY_STRING, myString);
```

Attaching arguments to a fragment

- To attach the arguments bundle to a fragment, you call Fragment.setArguments(Bundle).
- Attaching arguments to a fragment must be done after the fragment is created but before it is added to an activity.
- To hit this window, Android programmers follow a convention of adding a static method named **newInstance()** to the Fragment class.
- This method creates the fragment instance and bundles up and sets its arguments.

Attaching arguments to a fragment

- When the hosting activity needs an instance of that fragment, you have it call the newInstance(…) method rather than calling the constructor directly.
- The activity can pass in any required parameters to newInstance(…) that the fragment needs to create its arguments.

- In CrimeFragment, we will write a newInstance(UUID) method that
  - accepts a UUID,
  - creates an arguments bundle,
  - creates a fragment instance, and then
  - attaches the arguments to the fragment.

newInstance(UUID) method of CrimeFragment

```
public static CrimeFragment newInstance(UUID crimeId) {
    Bundle args = new Bundle();
    args.putSerializable(ARG_CRIME_ID, crimeId);

    CrimeFragment fragment = new CrimeFragment();
    fragment.setArguments(args);
    return fragment;
}
```

- Now, CrimeActivity should call CrimeFragment.newInstance(UUID) when it needs to create a CrimeFragment.
- It will pass in the UUID it retrieved from its extra.

The Activity & Fragment Dependency

- The need for independence does not go both ways.
- CrimeActivity has to know plenty about CrimeFragment, including that it has a newInstance(UUID) method.
- Hosting activities should know the specifics of how to host their fragments, but fragments should not have to know specifics about their activities.
- At least, not if you want to maintain the flexibility of independent fragments.

Retrieving arguments

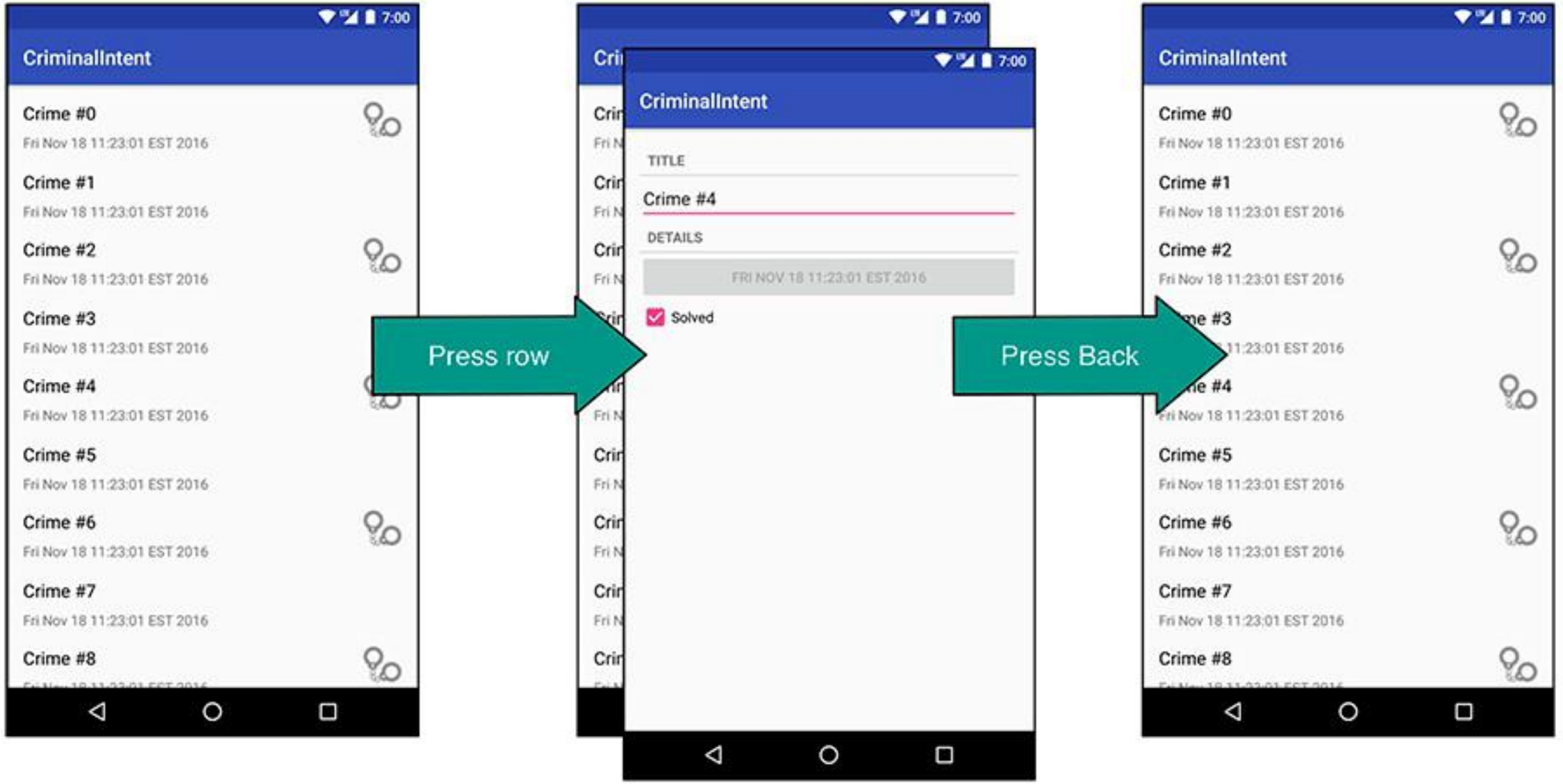
- When a fragment needs to access its arguments, it calls the Fragment method getArguments() and then one of the type-specific “get” methods of Bundle.

```
UUID crimeId = (UUID) getArguments().getSerializable(ARG_CRIME_ID);
```

Reloading the List (RecyclerView)

- As of now, if we modify the Crime's details, these changes are saved to the model, but when you return to the list, the RecyclerView is unchanged.
- The RecyclerView's Adapter needs to be informed that the data has changed so that it can refetch the data and reload the list.
- We can work with the ActivityManager's back stack to reload the list at the right moment.

Application's Back Stack



Application's Back Stack

- When CrimeListFragment starts an instance of CrimeActivity, the CrimeActivity is put on top of the stack.
- This pauses and stops the instance of CrimeListActivity that was initially on top.
- When the user presses the Back button to return to the list, the CrimeActivity is popped off the stack and destroyed.
- At that point, the CrimeListActivity is started and resumed
- When the CrimeListActivity is resumed, it receives a call to onResume() from the OS.
- When CrimeListActivity receives this call, its FragmentManager calls onResume() on the fragments that the activity is currently hosting.
- In this case, the only fragment is CrimeListFragment.

Reloading the List (RecyclerView)

- In CrimeListFragment, override onResume() and trigger a call to updateUI() to reload the list.
- Modify the updateUI() method to call notifyDataSetChanged() if the CrimeAdapter is already setup.

Reloading the List (RecyclerView)

- Why override onResume() to update the RecyclerView and not onStart()?
- You cannot assume that your activity will be stopped when another activity is in front of it.
- If the other activity is transparent, your activity may just be paused.
- If your activity is paused and your update code is in onStart(), then the list will not be reloaded.
- In general, onResume() is the safest place to take action to update a fragment's view.

Getting Results with Fragments

```
public class CrimeListFragment extends Fragment {

    private static final int REQUEST_CRIME = 1;
    ...
    private class CrimeHolder extends RecyclerView.ViewHolder
            implements View.OnClickListener {
        ...
        @Override
        public void onClick(View view) {
            Intent intent = CrimeActivity.newIntent(getActivity(), mCrime.getId());
            startActivityForResult(intent, REQUEST_CRIME);
        }
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == REQUEST_CRIME) {
            // Handle result
        }
    }
    ...
}
```

Hands-on Videos

- You can watch the complete implementation example at the following link

https://youtu.be/791RVxBhRyg

InClass Task 06

- The notifyDataSetChanged method on your Adapter is a handy way to ask the RecyclerView to reload all of the items that are currently visible.

- The use of this method in CriminalIntent is wildly inefficient because at most one Crime will have changed when returning to the CrimeListFragment.

- Use the RecyclerView.Adapter's notifyItemChanged(int) method to reload a single item in the list. Modifying the code to call that method is easy. The challenge is discovering which position has changed and reloading the correct item.

InClass Task 07

- CrimeLab's get(UUID) method works, but checking each crime's ID against the ID you are looking for one at a time can be improved upon.

- Improve the performance of the lookup, making sure that CriminalIntent's existing behavior remains unchanged as you refactor.

Recommended Readings

- Page # 205 to 216, Chapter 10: Using Fragment Arguments from Android Programming: The Big Nerd Ranch Guide, 3rd Edition by Bill Phillips, 2017