IT332: Mobile Application Development

Lecture # 15 : Using ViewPager

Muhammad Imran

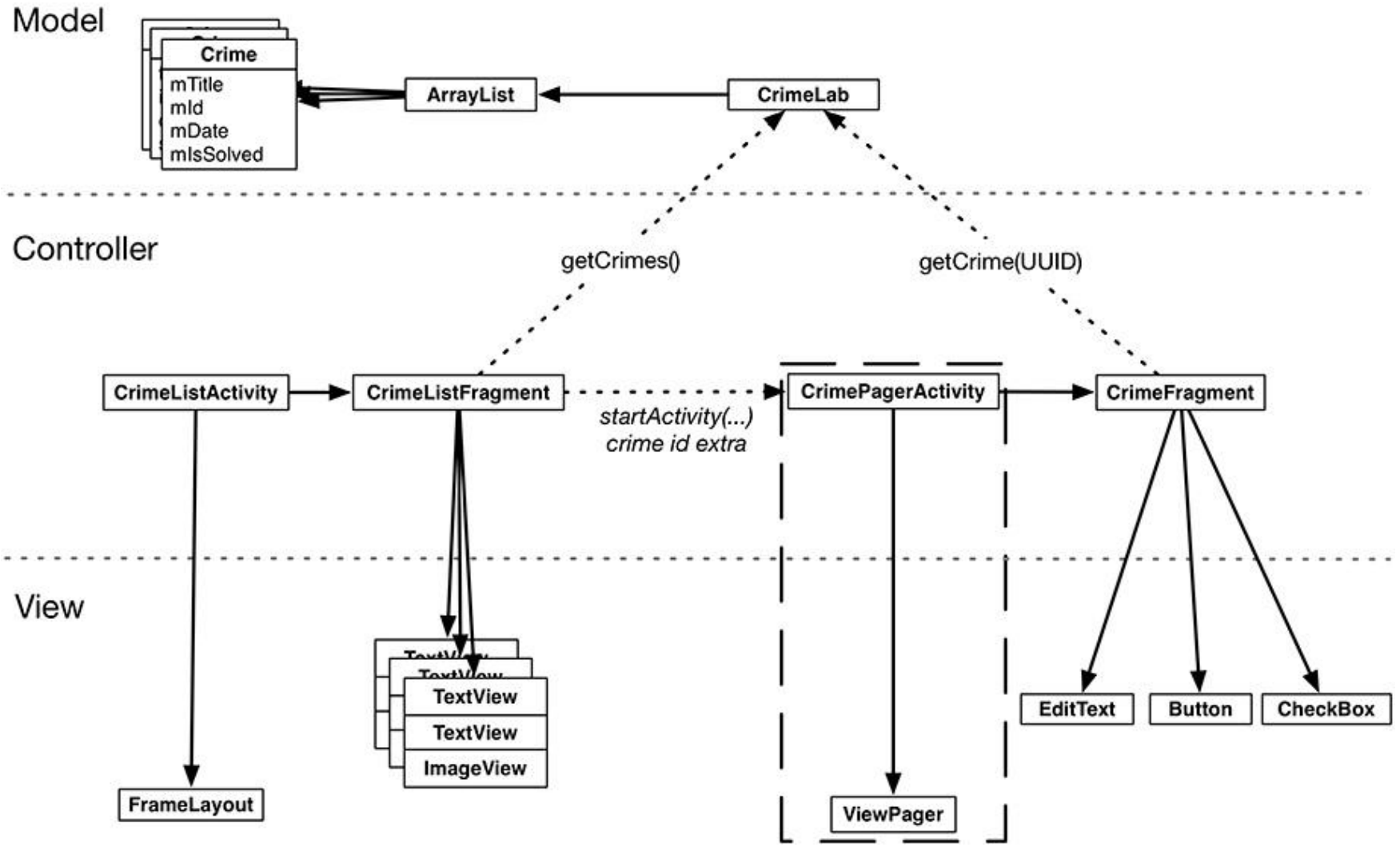
Outline

- Final Objective Today
- Object diagram for CrimePagerActivity
- Creating CrimePagerActivity
- ViewPager and PagerAdapter
- Setting up PagerAdapter
- ViewPager Default Behaviour
- Setting the initial pager item.
- FragmentStatePagerAdapter vs FragmentPagerAdapter
- Which Adapter Shall we Use?

Final Objective Today

- We will create a new activity to host **CrimeFragment**.
- This activity's layout will consist of an instance of **ViewPager**.
- Adding a ViewPager to UI will let users navigate between list items by swiping across the screen to "page" forward or backward through the crimes

Object diagram for CrimePagerActivity



- The new activity will be named **CrimePagerActivity** and will take the place of CrimeActivity

Tasks to be Completed Today

1. create the **CrimePagerActivity** class
2. define a view hierarchy that consists of a **ViewPager**
3. wire up the **ViewPager** and its **adapter** in CrimePagerActivity
4. modify **CrimeHolder.onClick(…)** to start CrimePagerActivity instead of CrimeActivity

Creating CrimePagerActivity

- **CrimePagerActivity** will be a subclass of AppCompatActivity. It will create and manage the **ViewPager**.

```
public class CrimePagerActivity extends AppCompatActivity {

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

Layout for CrimePagerActivity

```
android.support.v4.view.ViewPager
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/crime_view_pager"
android:layout_width="match_parent"
android:layout_height="match_parent"
```

- We use ViewPager's full package name when adding it to the layout file because the ViewPager class is from the **support library**.
- Unlike Fragment, ViewPager is only available in the support library; there is not a "standard" ViewPager class in SDK.

ViewPager and PagerAdapter

- A ViewPager is like a RecyclerView in some ways.
- A **RecyclerView** requires an **Adapter** to provide views.
- A **ViewPager** requires a **PagerAdapter**.

- However, the conversation between ViewPager and PagerAdapter is much **more involved** than the conversation between RecyclerView and Adapter.
- Luckily, we can use **FragmentStatePagerAdapter**, a subclass of PagerAdapter, to take care of many of the details.

ViewPager and PagerAdapter

- **FragmentStatePagerAdapter** will boil down the conversation to two simple methods: **getCount()** and **getItem(int)**
- When your **getItem(int)** method is called for a position in your array of crimes, it will return a **CrimeFragment** configured to display the crime at that position.

Setting up pager adapter (CrimePagerActivity.java)

- Setting up the ViewPager's pager adapter and implementing **getCount()** and **getItem(int)**.

```
private ViewPager mViewPager;
private List<Crime> mCrimes;

    mViewPager = (ViewPager) findViewById(R.id.crime_view_pager);

    mCrimes = CrimeLab.get(this).getCrimes();
    FragmentManager fragmentManager = getSupportFragmentManager();
    mViewPager.setAdapter(new FragmentStatePagerAdapter(fragmentManager) {

        @Override
        public Fragment getItem(int position) {
            Crime crime = mCrimes.get(position);
            return CrimeFragment.newInstance(crime.getId());
        }

        @Override
        public int getCount() {
            return mCrimes.size();
        }
    });
}
```

Integrating CrimePagerActivity

- **FragmentStatePagerAdapter** will boil down the conversation to two simple methods: **getCount()** and **getItem(int)**
- When your **getItem(int)** method is called for a position in your array of crimes, it will return a **CrimeFragment** configured to display the crime at that position.

ViewPager Default Behaviour

- By default, **ViewPager** loads the item currently onscreen plus one neighboring page in each direction so that the response to a swipe is immediate.
- You can tweak how many neighboring pages are loaded by calling **setOffscreenPageLimit(int).**
- If we Press the Back button to return to the list of crimes and press a different item. We will see the **first crime** displayed again instead of the crime that we asked for.
- **By default**, the ViewPager shows the first item in its **PagerAdapter**.
- We can set the ViewPager's **current item** to the index of our choice.

Setting the initial pager item (CrimePagerActivity.java)

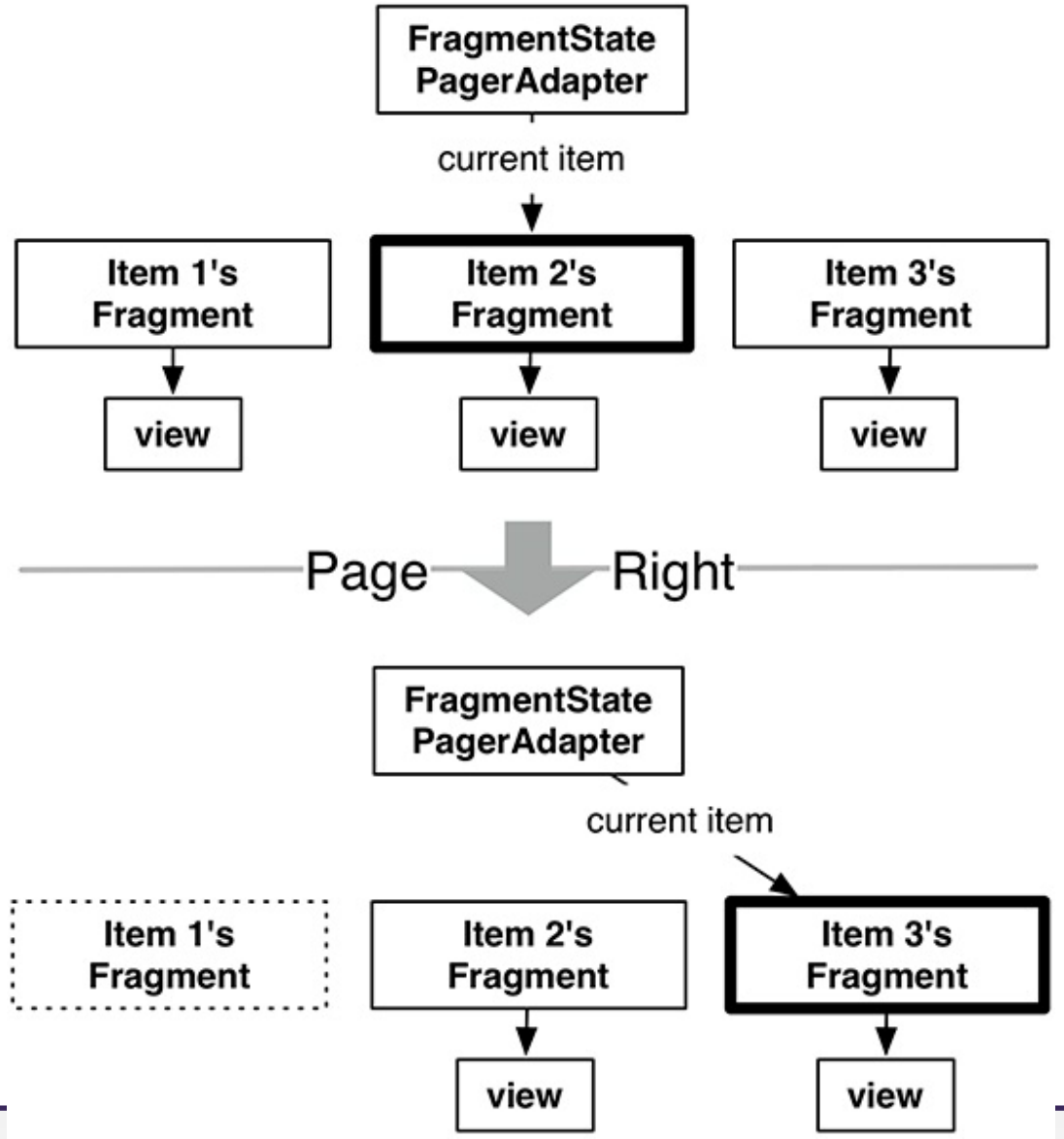
- At the end of **CrimePagerActivity.onCreate()**, find the index of the crime to display by looping through and checking each crime's ID.
- When you find the Crime instance whose mId matches the crimeId in the intent extra, **set the current item** to the index of that Crime.

```
for (int i = 0; i < mCrimes.size(); i++) {
    if (mCrimes.get(i).getId().equals(crimeId)) {
        mViewPager.setCurrentItem(i);
        break;
    }
}
```

FragmentStatePagerAdapter vs FragmentPagerAdapter

- There is **another** PagerAdapter type called FragmentPagerAdapter.
- FragmentPagerAdapter only **differs** in how it **unloads the fragments** when they are no longer needed.
- With **FragmentStatePagerAdapter**, the **unneeded fragment** is **destroyed**.
- A **transaction is committed** to completely remove the fragment from the activity's FragmentManager.
- The "**state**" in FragmentStatePagerAdapter comes from the fact that it will **save out** your fragment's Bundle from onSaveInstanceState(Bundle) when it is destroyed.
- When the user **navigates back**, the new fragment will be **restored** using that instance state.

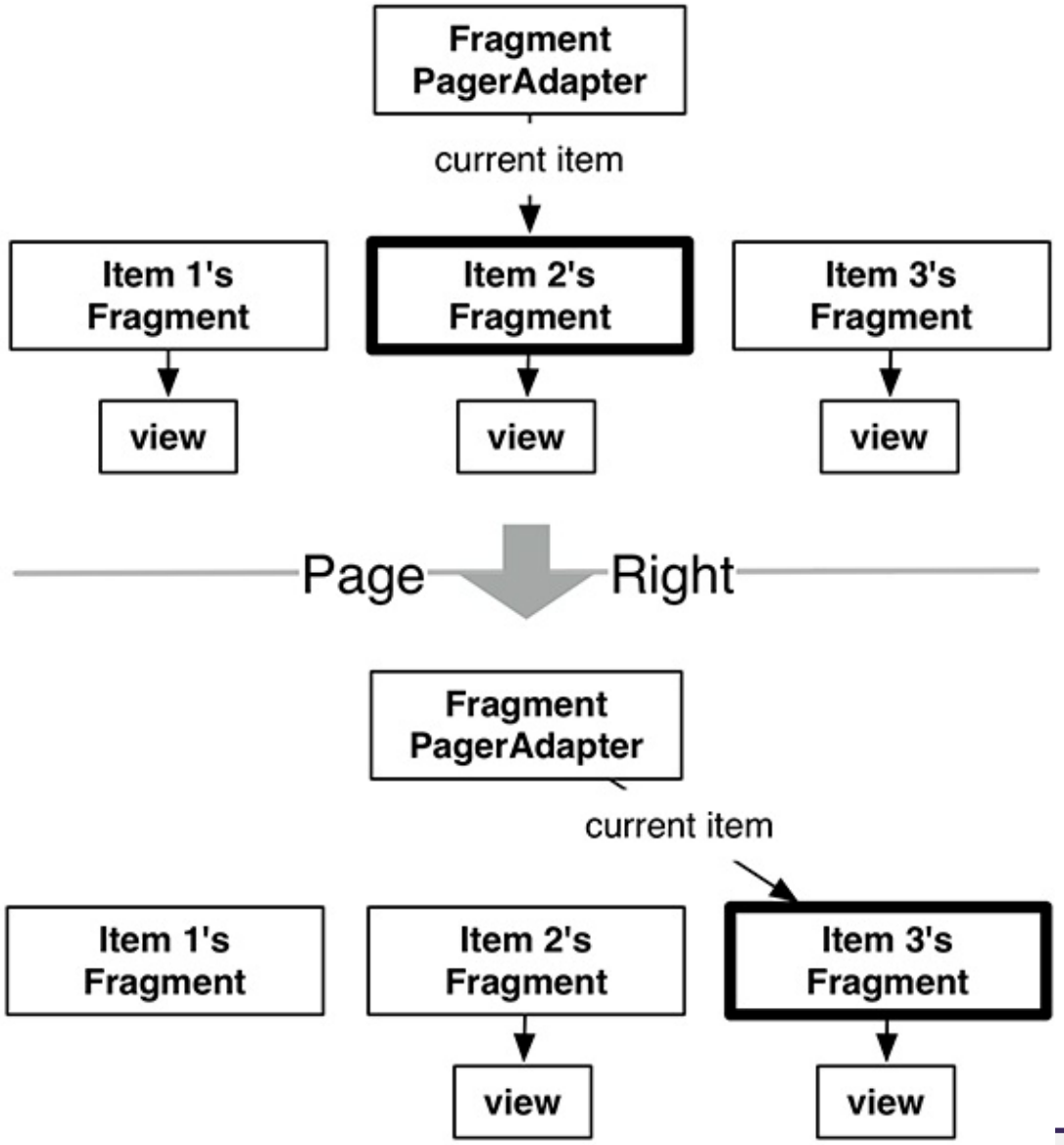
FragmentStatePagerAdapter's fragment management



FragmentStatePagerAdapter vs FragmentPagerAdapter

- FragmentPagerAdapter handles things differently.
- When the fragment is no longer needed, FragmentPagerAdapter calls **detach(Fragment)** on the transaction, **instead** of **remove(Fragment).**
- This **destroys** the fragment's **view**, but leaves the **fragment** instance **alive** in the FragmentManager.
- So the fragments created by FragmentPagerAdapter are **never destroyed**

FragmentPagerAdapter's fragment management



Which Adapter Shall we Use?

- The kind of adapter we should use depends on application.
- **FragmentStatePagerAdapter** is generally more **economical** with memory.
- CriminalIntent is displaying what could be a **long list** of crimes, each of which will eventually include a photo.
- We **do not** want to keep all that information in memory, so we use **FragmentStatePagerAdapter**.

Which Adapter Shall we Use?

- On the other hand, if our interface has a **small**, **fixed number** of fragments, **FragmentPagerAdapter** would be safe and appropriate.
- The most common example of this scenario is a **tabbed interface**.
- Some detail views have enough details to require two screens, so the details are split across multiple tabs.
- Adding a **swipeable ViewPager** to this interface makes the **app concrete**.
- Keeping these fragments in memory can make your **controller code easier** to manage.
- Plus, because this style of interface usually has only two or three fragments per activity, there is little danger of running low on memory.

Hands-on Videos

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

https://youtu.be/bx9tpGNKzoo

InClass Task 08 (Restoring CrimeFragment's Margins)

- You may have noticed that your CrimeFragment has mysteriously lost its margins. In fragment_crime.xml, you specified a margin of 16dp.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="16dp"
    android:orientation="vertical">
```

- That margin no longer shows up. So, what gives? ViewPager's layout params do not support margins.
- Update fragment_crime.xml and restore your margins.

InClass Task 09 (Adding First and Last Buttons)

- Add two buttons to **CrimePagerActivity** that allow jumping the **ViewPager** to the first or last crime instantly.
- As a bonus, disable “jump to first” when on the first page and “jump to last” when on the last page.

Recommended Readings

- Page # 217 to 226, Chapter 11: Using ViewPager from Android Programming: The Big Nerd Ranch Guide, 3rd Edition by Bill Phillips, 2017