Lecture 9

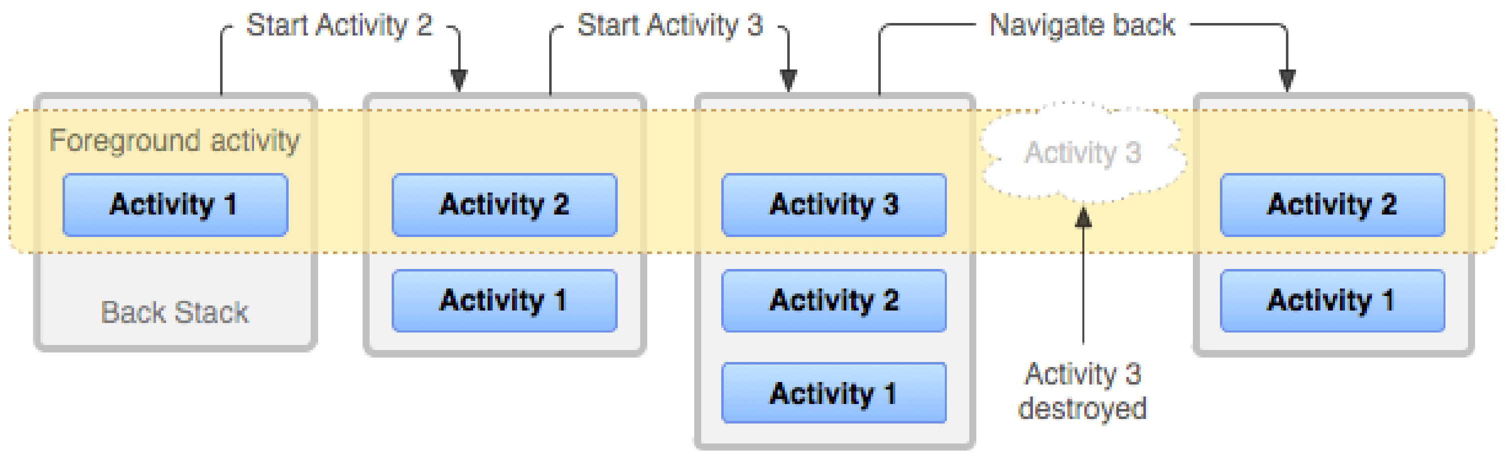
- Back stack
- Animating Views
- Animating Fragments

Back Stack

When you start an application, its Activity (Activity 1) is placed on top of a stack: the back stack.

If Activity 1 launches Activity 2, then this new Activity is placed on top of the back stack.

When the user taps the Back button, the top Activity is removed from the back stack, being replaced by the former one.



Fragments and the Back Stack

It is possible to include Fragment transitions in the back stack flow, so that tapping back will navigate back to the previous Fragment or Activity.

This is done by calling

`addToBackStack(null);`

at a Fragment Transaction.

```
FragmentTransaction ft = fm.beginTransaction();
ft.addToBackStack(null);
ft.replace(R.id.fragment_container, fragment);
ft.commit();
```

Fragments on the same level

Normally, the Activity switches among a set of Fragments, displaying only one of them at the same time.

You can add all fragments to the Activity at the start, assigning a unique tag to each of them, making all of them invisible except for the first one.

```
if(savedInstanceState==null) {
    first = new FirstFragment();
    FragmentTransaction ft = fm.beginTransaction();
    ft.add(R.id.fragment_container, first, "first");
    ft.commit();
} else {
    first = (FirstFragment)fm.findFragmentByTag("first");
}
```

Fragments on the same level

Instead of replacing the current Fragment by a new one, hide it and show the new one.

Find Fragments using their tag. Keep track of the current Fragment’s tag.

```
public void showFragment(String tag) {
    Fragment fragment = fm.findFragmentByTag(tag);
    Fragment currentFragment = fm.findFragmentByTag(currentTag);
    if(fragment!=null) {
        FragmentTransaction ft = fm.beginTransaction();
        if(currentFragment!=null) {
            ft.hide(currentFragment);
        }
        ft.show(fragment);
        ft.commit();
        currentTag = tag;
    }
}
```

FragmentViewer

It is quite useful to implement a class (FragmentViewer) that contains and manages a FragmentManager, taking care of adding and showing Fragments:

```
public class FragmentViewer {
    private int container; // att placera fragments i
    private FragmentManager fm;
    private String currentTag; // fragment som visas
    // konstruktor, setCurrentTag, getCurrentTag och show-metod

    public void add(Fragment fragment, String tag) {
        FragmentTransaction ft = fm.beginTransaction();
        if(!fragment.isAdded())
            ft.add(container, fragment, tag);
        ft.hide(fragment);
        ft.commit();
    }

    public String show(String tag) {
        Fragment fragment = fm.findFragmentByTag(tag);
        Fragment currentFragment = fm.findFragmentByTag(currentTag);
        if(fragment!=null) {
            FragmentTransaction ft = fm.beginTransaction();
            if(currentFragment!=null)
                ft.hide(currentFragment);
            ft.show(fragment);
            ft.commit();
            currentTag = tag;
        }
        return currentTag;
    }
}
```

FragmentContainer

Then code a Fragment that contains a FragmentViewer. Handle Fragment switching here.

```
public class ContainerFragment extends Fragment {
    private FragmentViewer viewer;
    // konstruktor, onCreateView

    public void onResume() {
        super.onResume();
        viewer.show();
    }

    public void onSaveInstanceState(Bundle outState) {
        outState.putString("currentTag",viewer.getCurrentTag());
        super.onSaveInstanceState(outState);
    }

    public void add(Fragment fragment, String tag) {
        viewer.add(fragment,tag);
    }
    public void setCurrentTag(String tag) {
        viewer.setCurrentTag(tag);
    }
    public String getCurrentTag() {
        return viewer.getCurrentTag();
    }
    public String show() {
        return viewer.show();
    }

    public String show(String tag) {
        return viewer.show(tag);
    }
}
```

Animation - ObjectAnimator

There are several ways to animate Views in Android. An easy way is using ObjectAnimator. Several properties in a View can be animated sequentially or at the same time.

With the methods onFloat, onInt, and, onObject (among others) we get an ObjectAnimator object. The input arguments define the View to be animated, its target property, and the animation itself.

E.g.: rotate view clockwise once in 2 seconds

```
private void rotate(View view) {
    ObjectAnimator animator =
        ObjectAnimator.ofFloat(view,"rotation”,0f,360f);
    animator.setDuration(2000);
    animator.start();
}
```

Animation - ObjectAnimator

E.g: rotate a View once while moving it horizontally:

```
private void rotateAndMove(View view, float startRot,
                           float stopRot, float toX) {
    float x = view.getX();
    ObjectAnimator rotate = ObjectAnimator.ofFloat(view,
                                "rotation",startRot,endRot);
    ObjectAnimator move = ObjectAnimator.ofFloat(view,"x",x,toX);
    AnimatorSet set = new AnimatorSet();
    rotate.setDuration(2000);
    move.setDuration(4000);
    set.playTogether(rotate, move);
    set.start();
}
```

E.g: Change a View’s alpha:

```
private void fade(View view) {
    ObjectAnimator fadeOut = ObjectAnimator.ofFloat(view,"alpha",
                                                   1f,0f);
    ObjectAnimator fadeIn = ObjectAnimator.ofFloat(view,"alpha",
                                                   0f,1f);
    fadeOut.setDuration(2000);
    fadeIn.setDuration(1000);
    AnimatorSet set = new AnimatorSet();
    set.playSequentially(fadeOut, fadeIn);
    set.start();
}
```

Animation – ObjectAnimator via xml

It is possible to define an animation in an XML file. This way the animation is reusable, though it is not possible to change it.

Place Animations under the res/animator folder.

The root element for the Animation is a **set**, that may contain several **objectAnimator**.

```
<!-- rotate_pos.xml -->
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:propertyName="rotation"
        android:duration="2000"
        android:valueFrom="360"
        android:valueTo="0" />
</set>
```

You get a AnimatorSet by inflating the XML file.

```
private void rotateWithXml(View view) {
    AnimatorSet set = (AnimatorSet)AnimatorInflater.loadAnimator(
                           this,R.animator.rotate_pos);
    set.setTarget(view);
    set.setDuration(2000);
    set.start();
}
```

Animation – AnimationDrawable

You can display animations made up of a sequence of images using an XML file. The image sequence is defined in an XML file which will be saved in the drawable directory.

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true">
    <item android:drawable="@drawable/new1" android:duration="100" />
    <item android:drawable="@drawable/new2" android:duration="100" />
    <item android:drawable="@drawable/new3" android:duration="100" />
    : osv
</animation-list>
```

Then, retrieve the animation when initializing the View, placing it as a BackgroundResource first.

```
private void initAnimation() {
    ImageView ivDrawable = (ImageView)findViewById(R.id.imageView2);
    ivDrawable.setBackgroundResource(R.drawable.new_animation);
    animDrawable = (AnimationDrawable)ivDrawable.getBackground();
}

private void drawableAnim() {
    animDrawable.stop();
    animDrawable.start();
}
```

Animating Fragment Transactions

It is possible to animate the transition from one fragment to another.

Define animations in XML files placed under the res/animator folder.

Load those animations (in and out) before committing the Fragment Transaction.

```
public String show(String tag) {
    Fragment fragment = fm.findFragmentByTag(tag);
    Fragment currentFragment = fm.findFragmentByTag(currentTag);
    if(fragment!=null) {
        FragmentTransaction ft = fm.beginTransaction();
        if(!tag.equals(currentTag)) {
            ft.setCustomAnimations(R.animator.slide_in_left,
                                   R.animator.slide_out_right);
        }
        if(currentFragment!=null)
            ft.hide(currentFragment);
        ft.show(fragment);
        ft.commit();
        currentTag = tag;
    }
    return currentTag;
}
```

Animating Fragment Transactions

slide_in_left.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set>
    <objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
        android:propertyName="x"
        android:valueType="floatType"
        android:valueFrom="@integer/nWidth"
        android:valueTo="0"
        android:duration="@integer/animationTime"/>
</set>
```

slide_out_right.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set>
    <objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
        android:propertyName="x"
        android:valueType="floatType"
        android:valueFrom="0"
        android:valueTo="@integer/width"
        android:duration="@integer/animationTime"/>
</set>
```