# Activities, Fragments, Menus & ToolBars

## Task, Activities and the Back Stack

An application usually contains multiple activities, not necessarily from the same application; an activity can run another activity created in another application, in the case the latter advertised that it can handle a specific type of intent.

A task is a collection of activities that users interact with when performing a certain job. The activities are arranged in a stack (the *back stack*), in the order in which each activity is opened. So, everytime you run an application by clicking on an icon on the application launcher, a task (and its corresponding backstack) is created and maintained by the system.

The corresponding stack holds all the activities created during the lifetime of the task in a LIFO order.

When the back button is used, the top activity is destroyed and  the system *does not* retain the activity's state (does not call onSaveInstanceState as there is no need to !). To save the activity’s content, it’s important to do it using « onPause() » (onSaveInstanceState is called for example when the screen rotates; either onRestoreInstanceState is used to restore the data or you can put the corresponding code inside onCreate as both receive the bundle).

## Processes and Activities

In most cases, every Android application runs in its own Linux process. This process is created for the application when some of its code needs to be run, and will remain running until it is no longer needed *and* the system needs to reclaim its memory for use by other applications.

## Fragments

The user interface can be divided into logical sections, with these sections represented by fragments. A fragment can then be reused in more than one screen within the application. A Fragment is modeled as a subdivision of an Activity. In other words, a Fragment is integrated into an Activity; therefore, it needs an Activity to run.

Like an Activity, a Fragment has its own lifecycle, as well as its own user interface. The Fragment’s lifecycle is connected to the activity that owns it. Each fragment has its own callback methods in the standard Activity lifecycle.

### Additional Callbacks for Fragments

| | |
|---|---|
| onCreateView | This is a core lifecycle method that is called to bring a fragment up to a resumed state, interacting with the user. This callback method creates and returns the view hierarchy associated with the fragment. |
| onInflate | Called every time the fragment is inflated. |
| onActivityCreated | Called when the fragment’s activity has been created and this fragment’s view hierarchy is instantiated. Final initializations, such as restoring a state or retrieving a view, are often implemented here. |
| onAttach | Called after the fragment has been attached (associated) to an activity. |
| onDestroyView | Informs the fragment that its view is being destroyed so that it can clean up any associated resources. This is called after onStop() and before onDestroy(). |
| onDetach | Called when the fragment is no longer attached to its activity, after onDestroy(). This is the final call before the fragment object is released to the garbage collector. |

The most significant difference in lifecycle between an activity and a fragment is how one is stored in its respective back stack. An activity is placed into a back stack of activities that's managed by the system when it's stopped, by default. However, a fragment is placed into a back stack managed by the host activity only when you explicitly request that the instance be saved by calling addToBackStack() during a transaction that removes the fragment.

## Fragments by example

We will be creating a main UI and add to it dynamically a fragment one « Button 1 » is clicked. Using the back button, we’re able to navigate through the previous fragments. To start off, I’ve created in Android Studio a project with a blank activity with a fragment, hence the comments generated automatically in the code below.

### activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="com.example.cyk.myfragapp.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

## Content_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    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"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.cyk.myfragapp.MainActivity"
    tools:showIn="@layout/activity_main">

    <!--
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
        -->
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Replace Fragment"
        />

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/myFrame"
        android:layout_below="@+id/button2" />
</RelativeLayout>
```

## fragment_first.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".firstFragment">

        <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button"
        android:layout_gravity="left|top" />
</FrameLayout>
```

```
public class MainActivity extends AppCompatActivity implements
firstFragment.OnFragmentInteractionListener {

    private static int i;
    private Button b;
    private firstFragment first;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });

        first = firstFragment.newInstance("first fragment", "you");

        getSupportFragmentManager().beginTransaction().add(R.id.myFrame,
first).commit();

        b = (Button) findViewById(R.id.button2);
        b.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                i++;
                firstFragment second = firstFragment.newInstance("second
fragment " + i, "you");
```

```
                FragmentTransaction fgt =
getSupportFragmentManager().beginTransaction();
                fgt.addToBackStack("new fragment");
                fgt.replace(R.id.myFrame, second).commit();

            }
        });

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onFragmentInteraction(Object) {
        System.out.println("Fragment Test");
    }
}
```

// here starts the fragment class code

```
public class firstFragment extends Fragment {
    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";
    // The request code must be 0 or greater.
    private static final int PLUS_ONE_REQUEST_CODE = 0;
    // The URL to +1.  Must be a valid URL.
    private final String PLUS_ONE_URL = "http://developer.android.com";
    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    private Button b1;

    private OnFragmentInteractionListener mListener;

    public firstFragment() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment firstFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static firstFragment newInstance(String param1, String param2) {
        firstFragment fragment = new firstFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_first, container,
false);


        b1 = (Button) view.findViewById(R.id.button);
        b1.setText(mParam1);
        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mListener != null) {
                    mListener.onFragmentInteraction(null);
                }

            }
        });

        return view;
    }

    @Override
    public void onResume() {
        super.onResume();

        // Refresh the state of the +1 button each time the activity
receives focus.
        //mPlusOneButton.initialize(PLUS_ONE_URL, PLUS_ONE_REQUEST_CODE);
    }
    // TODO: Rename method, update argument and hook method into UI event
    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof OnFragmentInteractionListener) {
            mListener = (OnFragmentInteractionListener) context;
        } else {
            throw new RuntimeException(context.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    /**
     * This interface must be implemented by activities that contain this
     * fragment to allow an interaction in this fragment to be communicated
     * to the activity and potentially other fragments contained in that
     * activity.
         */
    public interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        void onFragmentInteraction(Object ref);
    }

}
```

Add « Toasts » or messages to the different callbacks of the fragment to see when they are called !

## Toolbars

The *toolbar bar* is represented as of Android 5.0 via the Toolbar view group and can be placed into your layout file. It can display the activity title, icon, actions which can be triggered, additional views and other interactive items. It can also be used for navigation in your application.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
```

```
    <item
        android:id="@+id/action_refresh"
        android:orderInCategory="100"
        android:showAsAction="always"
        android:icon="@drawable/ic_media_play"
        android:title="Refresh"/>

<item
        android:id="@+id/action_settings"
        android:title="Settings"
       android:orderInCategory="110"
        android:showAsAction="ifRoom" />


<item
        android:id="@+id/action_third"
        android:title="Settings"
       android:orderInCategory="120"
        android:showAsAction="ifRoom" />


</menu>
```

The MenuInflator class allows to inflate actions defined in an XML file and adds them to the action bar. MenuInflator can get accessed via the getMenuInflator() method from your *activity*. The following example code demonstrates the creation of actions.

```
@Override
  public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.mainmenu, menu);
    return true;
  }
```

If an action is selected, the onOptionsItemSelected() method in the corresponding activity is called. It receives the selected action as parameter. Based on this information, you can decide what to do. The usage of this method is demonstrated in the following code snippet.

```
@Override
  public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    // action with ID action_refresh was selected
    case R.id.action_refresh:
      Toast.makeText(this, "Refresh selected", Toast.LENGTH_SHORT)
          .show();
      break;
    // action with ID action_settings was selected
    case R.id.action_settings:
      Toast.makeText(this, "Settings selected", Toast.LENGTH_SHORT)
          .show();
      break;
    default:
      break;
    }
    return true;
  }
```

If you want to change the menu later, you have to call the `invalidateOptionsMenu()` method. Afterwards this `onCreateOptionsMenu()` method is called again.