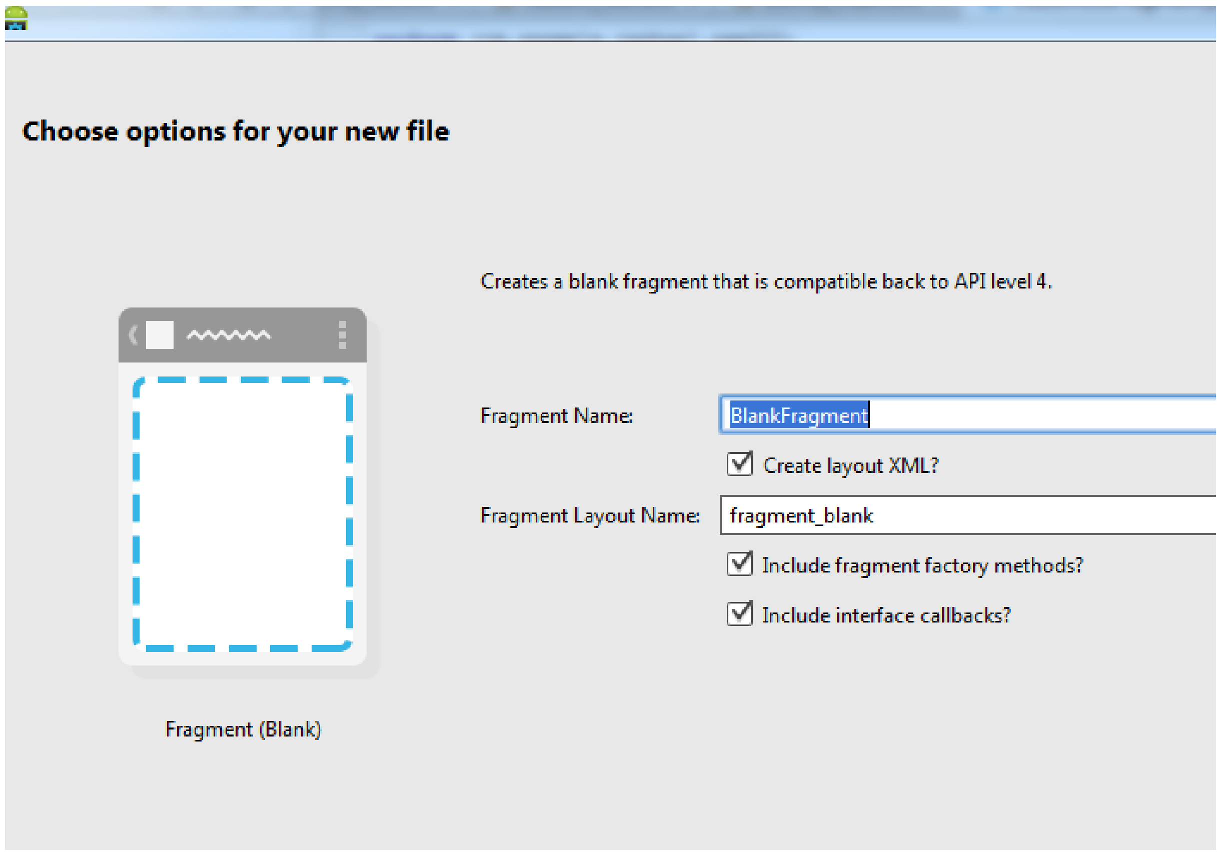
How to change an Activity to make use of Fragments

1.) Create a new Class that extends Fragment (The IDE can do that for you)



Choose options for your new file

Creates a blank fragment that is compatible back to API level 4.

Fragment Name: BlankFragment

[✓] Create layout XML?

Fragment Layout Name: fragment_blank

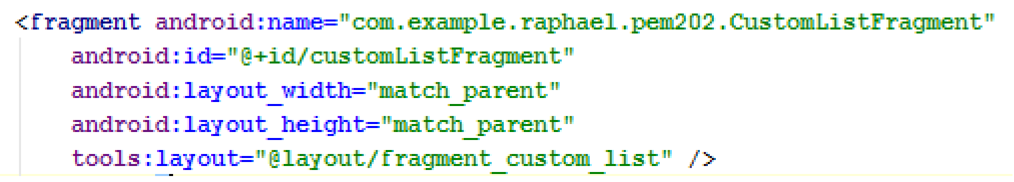
[✓] Include fragment factory methods?

[✓] Include interface callbacks?

2.)Create a layout xml file for the new Fragment (if the IDE hasn't already done it)

3.)Copy your layout from your Activity to your Fragment layout xml file

4.)Replace the layout in your Activity with a Fragment tag that contains the name of the Fragment. (This should only be done if this Fragment is permanent as it can't be removed at runtime)



```
<fragment android:name="com.example.raphael.pem202.CustomListFragment"
    android:id="@+id/customListFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:layout="@layout/fragment_custom_list" />
```

5.)In the OnCreateView(…) method of the Fragment inflate your layout (If you chose to create a layout file when creating the Fragment earlier this code is already there. (We need the View later so it needs to be saved)

```java
View view = inflater.inflate(R.layout.fragment_custom_list, container, false);
```

6.)Move the code that is used to do things in your Activity to the Fragment. In this example there is only code in the onCreate() method of the Activity. Copy that Code to the onCreateView(…) method in the Fragment.

```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_list);

    customListAdapter=new CustomListAdapter(getApplicationContext());
    listView = (ListView)findViewById(R.id.listViewFragment);
    listView.setAdapter(customListAdapter);
}
```

```java
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {



    // Inflate the layout for this fragment
    System.out.println("onCreateView()");
    View view = inflater.inflate(R.layout.fragment_custom_list, container, false);


    customListAdapter=new CustomListAdapter(getApplicationContext());
    listView = (ListView)findViewById(R.id.listViewFragment);
    listView.setAdapter(customListAdapter);
```

This will cause an error as some methods don't work in a Fragment like they did in an Activity.

First getApplicationContext() needs to be changed to getActivity().getApplicationContext()

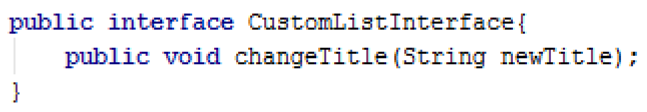
Then the reason we saved the View becomes clear as we need it to change findViewById() to:

```java
listView = (ListView)view.findViewById(R.id.listViewFragment);
```

This should now work like the Activity did before.

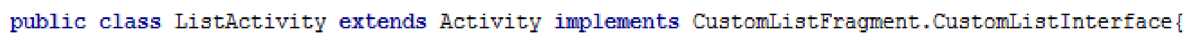
How to connect a Fragment with an Activity via Callback

1.)Create an Interface in your Fragment. This interface should define Methods that you need to achieve your goal. In our case we want to change the ActionBar title in the Activity to the TextElement of a chosen ListElement.

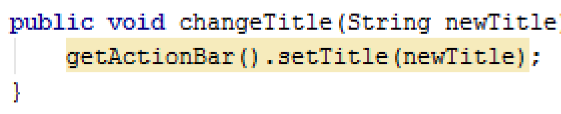


```
public interface CustomListInterface{
    public void changeTitle(String newTitle);
}
```

2.) Implement the interface in your Activity

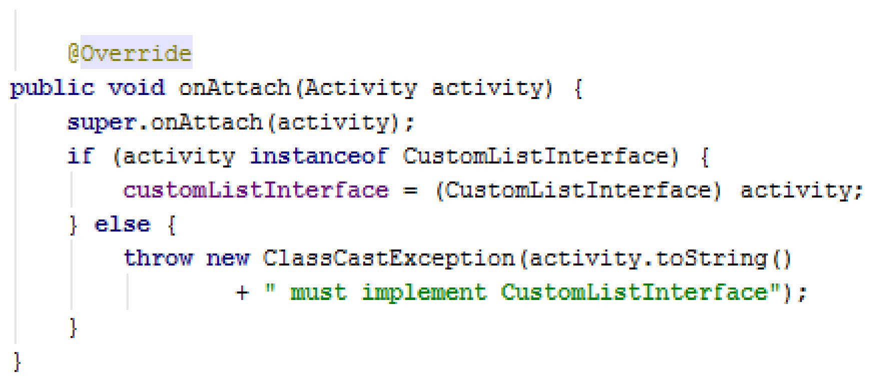


```
public class ListActivity extends Activity implements CustomListFragment.CustomListInterface{
```



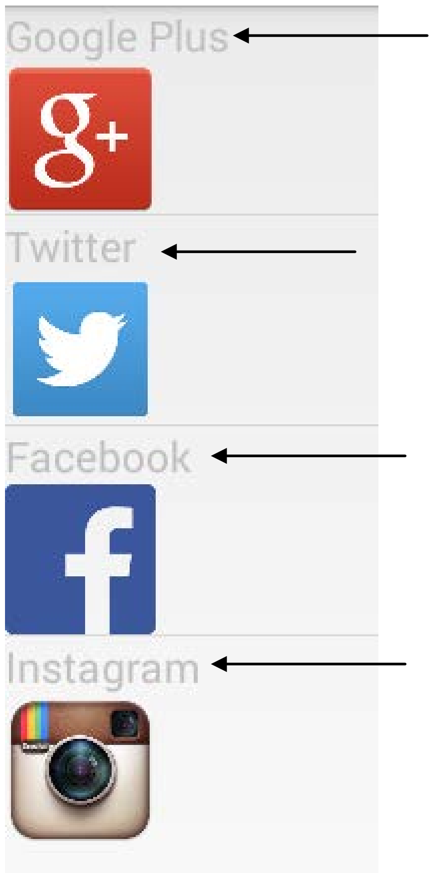
```
public void changeTitle(String newTitle){
    getActionBar().setTitle(newTitle);
}
```

3.) Override the OnAttach() Method of the Fragment to make sure that the Activity that attaches our Fragment implemented our Interface.
Otherwise our Callback will fail.



```
    @Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    if (activity instanceof CustomListInterface) {
        customListInterface = (CustomListInterface) activity;
    } else {
        throw new ClassCastException(activity.toString()
                + " must implement CustomListInterface");
    }
}
```

4.) Get the data you need to be passed to the Activity.
In our Case we need the Text of the TextElement of the chosen ListElement



For this we have to first add an OnClickListener() to our ListView. Then we need to Override the OnItemClick Method. Here we get the text of our TextElement via the getView() Method of our CustomList Adapter that returns a View from which the Text can be accessed. Lastly we make use of a Callback and pass the Text to the Activity.

```java
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {
        System.out.println("position: "+position);
        View nView = customListAdapter.getView(position,view,viewGroup);
        TextView textView=(TextView) nView.findViewById(R.id.text);
        String title = textView.getText().toString();
        customListInterface.changeTitle(title);
    }
});
```