**ISTE456 – Using Fragments in the List UI**

**The purpose of this app is to learn how to make a LIST pattern UI using fragments.**

1. Create a new **EmptyActivity** project (don’t use Blank Activity with Fragment) call it **FragmentExample**
2. Open the **activity\_fragment\_example.xml** (in the layout folder) and

<LinearLayout 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"

android:orientation="vertical"

tools:context="com.packagename.fragmentexample.FragmentExample">

**NOTE: replace “com.packagename” with the name of the package for your app.**

<TextView

android:text="Fragment Example"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:padding="10dp"

android:textSize="20dp"

android:textStyle="bold"

/>

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="List Fragment"

android:id="@+id/listButton"/>

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Fragment Two"

android:id="@+id/fragmentTwoButton"/>

<FrameLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:id="@+id/the\_info"

android:layout\_margin="10dp">

</FrameLayout>

</LinearLayout>

1. Under the “**layout**” folder, create a new “**Layout Resource File**” called “**list\_item.xml**” with a root element of “**TextView**” and make it:

<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/text1"

android:textSize="23sp"

android:gravity="center\_vertical|left"

android:padding="10dp"

android:minHeight="?android:attr/listPreferredItemHeight"

android:textColor="@android:color/black">

</TextView>

1. Under the “**layout**” folder, create a new “**Layout Resource File**” called “**detail\_fragment.xml**” with a root element of “**LinearLayout**” and make it:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical" android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge"

android:text="Information About:"

android:id="@+id/textViewLabel"

android:layout\_margin="10dp"

android:padding="10dp" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="New Text"

android:id="@+id/textViewName"

android:layout\_gravity="center"

android:textStyle="bold"

android:textSize="20dp"

android:padding="5dp" />

</LinearLayout>

1. Under the “**layout**” folder, create a new “**Layout Resource File**” called “**fragment\_two.xml**” with a root element of “**RelativeLayout**” and make it:

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:gravity="center\_vertical|center\_horizontal">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="50dp"

android:text="Fragment Two"

android:id="@+id/textView"

android:layout\_centerVertical="true"

android:layout\_centerHorizontal="true" />

</RelativeLayout>

1. Create a new “**anim**” **Android Resource Directory** of **type** “**anim**” under “**res**”
2. Under the “**anim**” folder, create a new “**Animation Resource File**” called “**fragment\_animation\_fade\_in.xml**” and make it:

<?xml version="1.0" encoding="utf-8"?>

<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/linear\_interpolator"

android:valueFrom="0"

android:valueTo="1"

android:propertyName="alpha"

android:duration="1500" />

1. Under the “**anim**” folder, create a new “**Animation Resource File**” called “**fragment\_animation\_fade\_out.xml**” and make it:

<?xml version="1.0" encoding="utf-8"?>

<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/linear\_interpolator"

android:valueFrom="1"

android:valueTo="0"

android:propertyName="alpha"

android:duration="1500"

/>

1. Create a new “**Java Class**” under the **java->package** folder and call it “**MyListFragment**” and paste this code in, then use the “option>return” to include the import statements needed.

public class MyListFragment extends ListFragment {

public ArrayList<String> itemsArrayList; //list of items

private ArrayAdapter<String> itemsArrayAdapter;

private ItemChangedListener itemChangedListener;

//interface describing listener for changes to selected item

public interface ItemChangedListener {

//the selected item is changed

public void onSelectedItemChanged(String itemNameString);

}

//set the ItemChangedListener

public void setItemChangedListener(ItemChangedListener listener) {

itemChangedListener = listener;

}

***(this is continued on the next page!)***

@Override

public void onActivityCreated(Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

//create ArrayList to save item names

itemsArrayList = new ArrayList<String>() {{ //can use command-D to duplicate each line below to change

add("A"); **//NOTE: after typing, command-D will duplicate the line, then change each letter**

add("B");

add("C");

add("D");

add("E");

add("F");

add("G");

add("H");

add("I");

add("J");

}};

//set the Fragment's ListView adapter

setListAdapter(new ItemsArrayAdapter<String>(

getActivity(),

R.layout.list\_item,

itemsArrayList));

ListView thisListView = getListView(); //get the Fragment's listview

//allow one item to be selected at a time

thisListView.setChoiceMode(ListView.CHOICE\_MODE\_SINGLE);

thisListView.setBackgroundColor(Color.WHITE);

thisListView.setOnItemClickListener(itemsOnItemClickListener);

}

//define the GUI components for each ListView item

private static class ViewHolder {

TextView itemTextView; //refers to the ListView item's TextView

}

//customer ArrayAdapter

private class ItemsArrayAdapter<T> extends ArrayAdapter<String> {

private Context context; //this Fragment's Activity's Context

private LayoutInflater inflater;

private List<String> items; //list of items

//public constructor

public ItemsArrayAdapter(Context context, int textViewResourceId,

List<String> objects) {

super(context,-1,objects); //-1 indicates we're customizing view

this.context = context;

this.items = objects;

inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

}

//get ListView item for the given position

@Override

public View getView(int position, View convertView, ViewGroup parent) {

ViewHolder viewHolder; //holds reference to current item's GUI

//if convertView is null, inflate GUI and create ViewHolder otherwise, get an existing ViewHolder

if(convertView == null) {

convertView = inflater.inflate(R.layout.list\_item,null);

//set up ViewHOlder for this item

viewHolder = new ViewHolder();

viewHolder.itemTextView = (TextView)convertView.findViewById(R.id.text1);

convertView.setTag(viewHolder);

} else {

//get the ViewHolder from the convertView's tag

viewHolder = (ViewHolder)convertView.getTag();

}

String item = items.get(position); //get current item

viewHolder.itemTextView.setText(item);

return convertView;

} //getView

}//arrayadapter

private AdapterView.OnItemClickListener itemsOnItemClickListener =

new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent,

View view,

int position,

long id) {

itemChangedListener.onSelectedItemChanged(

((TextView)view).getText().toString()

);

}

};

}//fragment

1. Create a new “**Java Class**” under the **java->package** folder and call it “**FragmentTwo**” and paste this code in, then use the “option>return” to include the import statements needed.

public class FragmentTwo extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater,

ViewGroup container, Bundle savedInstanceState){

//inflate the layout for this fragment

return inflater.inflate(R.layout.fragment\_two,container,false);

}

}

1. Create a new “**Java Class**” under the **java->package** folder and call it “**DetailFragment**” and paste this code in, then use the “option>return” to include the import statements needed.

public class DetailFragment extends Fragment{

private String itemName;

public static DetailFragment newInstance(String itemName) {

DetailFragment myFragment = new DetailFragment();

Bundle args = new Bundle();

args.putString("ITEM",itemName);

myFragment.setArguments(args);

return myFragment;

}

@Override

public View onCreateView(LayoutInflater inflater,

ViewGroup container,

Bundle savedInstanceState){

Bundle args = getArguments();

//itemName = args.getString("ITEM","Unknown");

itemName = (String) args.get("ITEM");

//inflate the layout for this fragment

View myView =

inflater.inflate(R.layout.detail\_fragment,container,false);

((TextView)myView.findViewById(R.id.textViewName)).setText(itemName);

return myView;

}

}

1. Open “FragmentExample.java”. Add the following imports:

import android.app.FragmentTransaction;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.os.Build;

import android.widget.Button;

import android.app.Activity;

1. On the FragmentExample class definition, replace “extends AppCompatActivity” with ‘extends’ to “Activity”
2. Now add the following instance variables

private MyListFragment myListFragment;

private FragmentTwo fragmentTwo;

private Button listButton;

private Button fragmentTwoButton;

1. Add the following to the onCreate method after setting the content view:

listButton = (Button)findViewById(R.id.listButton);

listButton.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick(View v) {

loadFragment("List");

}

});

fragmentTwoButton = (Button)findViewById(R.id.fragmentTwoButton);

fragmentTwoButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

loadFragment("Two");

}

});

//would need to keep track of which fragment is being displayed for

//rotation if desired in the appropriate methods

loadFragment("List");

1. Add the following method:

private void loadFragment(String which) {

if (which.equals("List")) {

//get the MyListFragment

myListFragment = new MyListFragment();

//set the change listener

myListFragment.setItemChangedListener(itemChangedListener);

//note: no transition or backstack - but clear backstack

getFragmentManager().popBackStack(null,

getFragmentManager().POP\_BACK\_STACK\_INCLUSIVE);

FragmentTransaction ft = getFragmentManager().beginTransaction();

ft.replace(R.id.the\_info,myListFragment);

ft.commit();

} else if (which.equals("Two")) {

//get the Fragment Two Fragment

fragmentTwo = new FragmentTwo();

//note: no transition or backstack - but clear backstack

getFragmentManager().popBackStack(null,

getFragmentManager().POP\_BACK\_STACK\_INCLUSIVE);

FragmentTransaction ft = getFragmentManager().beginTransaction();

ft.replace(R.id.the\_info,fragmentTwo);

ft.commit();

}

}

***(Continued on the next page)***

//listener for list fragment

private MyListFragment.ItemChangedListener itemChangedListener =

new MyListFragment.ItemChangedListener() {

@Override

public void onSelectedItemChanged(String itemNameString) {

//create and show the fragment

DetailFragment details = DetailFragment.newInstance(itemNameString);

FragmentTransaction ft = getFragmentManager().beginTransaction();

ft.setCustomAnimations(R.anim.fragment\_animation\_fade\_in,

R.anim.fragment\_animation\_fade\_out);

ft.replace(R.id.the\_info, details);

ft.addToBackStack(null);

//optional name for this backstack state or null – needs to be just before the commit –

//if multiple “transactions” (add/remove/replace in any combination) happen before “addToBackStack” ,

// they ALL are undone with one back button. if you use addToBackStack, the fragment is stopped and //resumed when the user navigates to it, if you don’t use addToBackStack it is destroyed.

ft.commit();

}

};

1. **Build and run the app**