CSC 214 ASSIGNMENT 05

The goal of this assignment is to get more practice with fragments, especially sending data into fragments, getting data out of fragments, and moving data from one fragment to another. You will experiment with two different ways of passing data into fragments: using **`setArguments(Bundle)`** on a new Fragment, and defining a special API to use (and reuse) on an existing Fragment. You will also log the lifecycle of multiple Fragments.

Assignment Instructions

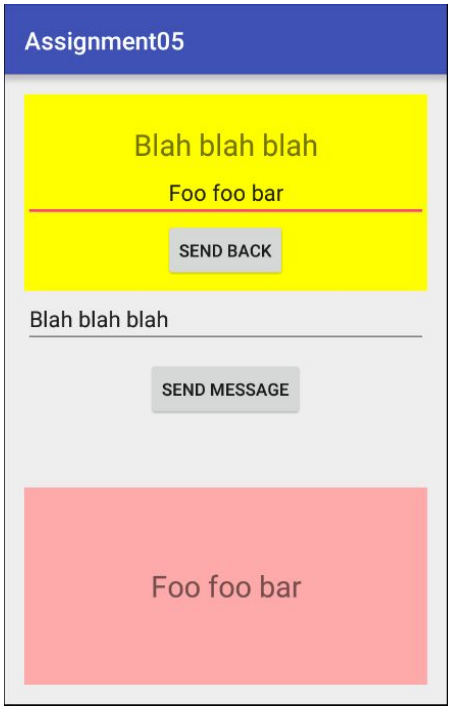
1. Create a new Application with a main activity with a vertical **`LinearLayout`** that is divided into three equal parts. The top third should contain an empty **`FrameLayout`** with a unique ID, the middle third should contain an **`EditText`** and at least one **`Button`** labeled "Send Message", and the bottom third should contain another empty **`FrameLayout`** with a unique ID.

2. Refer to the slides for Lecture 9 regarding the Fragment Lifecycle. Create a new class that extends **`Fragment`** called **`FragmentLifecycleLogger`**. *Uncheck* the **Create layout XML** checkbox, (or delete the layout it if you do create one). Add the following field to the class: **`protected final String TAG = getClass().getName();`** Add a comment describing what you think the value of this String will be. Next, override each of the lifecycle methods (there are 11 in all) such that they call the **`super`** method with the same name and use TAG to log a message indicating that the lifecycle method was called. Remove the **`onCreateView`** method.

3. Create a new fragment called **`TopFragment`** and modify it so that it extends **`FragmentLifecycleLogger`**. **`TopFragment`** should contain a **`TextView`**, an **`EditText`** and a **`Button`** labeled "Send Back". The background color of **`TopFragment`** should not be the default color (you may set this on the root layout of the fragment's layout file but **not** the main activity's layout file; the point here is to be able to tell when the fragment is added to the activity). By default the **`TextView`** should display "No message received" and the **`EditText`** should be empty (but should display a hint). Update the main activity so that whenever the "Send Message" **`Button`** is pressed, The **`FragmentManager`** is used to <u>**`replace`**</u> the current **`TopFragment`** with a **new** **`TopFragment`** in the top third of the screen that displays whatever message has been typed into the main activity's **`EditText`**. You **must** use the **`setArguments(Bundle)`** method to pass the message into the new fragment. You may want to use a factory method to implement this. Make sure to log a message using the **`TAG`** in the parent class in the **`onCreateView`** method.

4. Create a fragment called **`BottomFragment`** that extends **`FragmentLifecycleLogger`**. The **`BottomFragment`** should contain a single **`TextView`**, and the background color should not be the default color (you may set this on the root layout of the fragment's layout but **not** in the main activity's layout). By default, the **`TextView`** should display the message "No message sent back." **`BottomFragment`** should be added to the bottom third of the screen. **`BottomFragment`** should define a method with the signature **`public void messageSentBack(CharSequence message)`**. When the "Send Message Back" button is pressed in the **`TopFragment`**, a callback interface must be used to send whatever value is typed into the **`TopFragment's EditText`** to the **main activity**. You **must not** use **`setArguments(Bundle)`** to pass the message into the fragment. Fragments **never** communicate directly with each other. The main activity will then call the **`messageSentBack`** method on the **`BottomFragment`**. The **`BottomFragment`** should update its **`TextView`** to display the message. Make sure to log a message using the **`TAG`** in the parent class in the **`onCreateView`** method.

   A Note about onAttach(): You will need to override the **`onAttach`** method in **`TopFragment`** to initialize your callback in part 4. The **`onAttach(Activity)`** method will be called by API 22. Many of you are using API 23 to develop and test on your virtual devices. The **`onAttach(Activity)`** method was deprecated in API 23 and the **`onAttach(Context)`** method will be called instead on devices running API 23. This means that an app that works on API 22 may not work on API 23 and vice versa. To be safe and insure that your app runs on BOTH API levels, override BOTH methods (and don't forget to call the **`super`** version of the same method).



Your user interface should look something like this after a message has been sent to `TopFragment` and a message sent back to `BottomFragment`.

HAND IN

Before handing in, create two additional files in your lab directory:

1. Create a README that contains the following:
    a. Your contact information, TA name, and assignment number.
    b. A brief (one paragraph at most) description of the assignment.
2. Create a directory titled "SampleOutput." This directory should contain:
    a. A file called "logs.txt" that contains examples any relevant logs generated by your application. Remember to use LogCat filters to show *only* your log messages before copying them into the file.
    b. Screenshots (in PNG or JPG format) taken from an Android Virtual Device that show examples of your application's user interface.
3. Name your file using your last name and the assignment number. For example: "stjacques_assignment01.zip". This makes it easier for the TAs to organize when grading multiple students.

Hand in by uploading the compressed (i.e. "zipped") folder containing your Android Studio project and the required additional files to Blackboard. Remember that **late submissions are not accepted**.