Laboratory – The Fragment Class

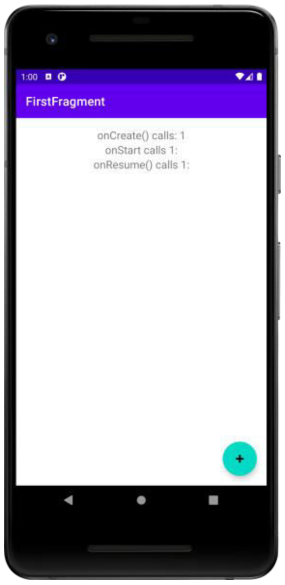
Learn about the Fragments and LifecycleAware components

Objectives

Familiarize yourself with the Fragment class, navigation, and the ViewModel class. Create and monitor a simple application that observes an Activity as it moves through its lifecycle. This app will maintain some Activity-specific information while using two different Fragments to implement its user interface. Once you've completed this Lab you should understand: the Fragment class, how to navigate between Fragments hosted within an Activity, and how to coordinate information across the two Fragments using the ViewModel class.

The Fragment Class

This lab involves an app called FragmentsLab. When it runs, the app displays a user interface like that

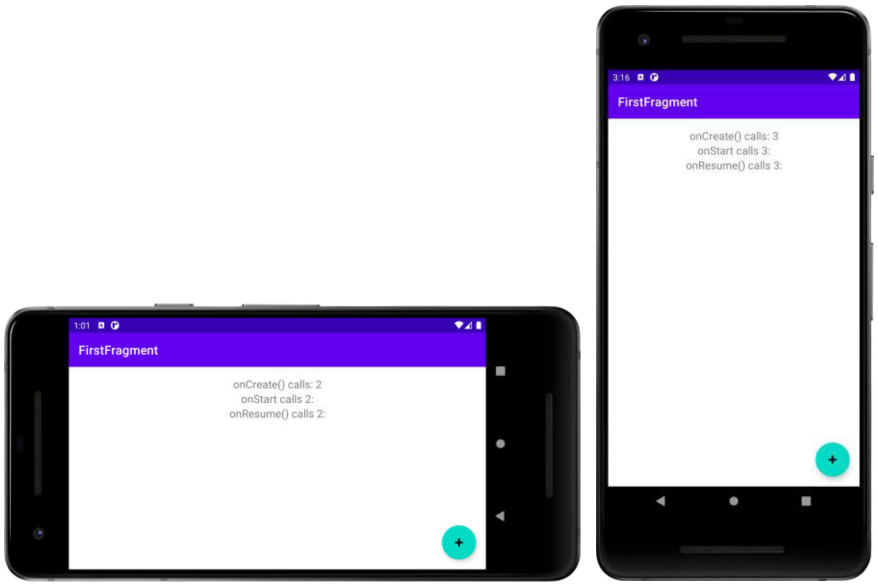


onCreate() calls: 1
onStart calls 1:
onResume() calls 1:

shown below.

This application comprises a single Activity called "MainActivity." MainActivity can display two Fragments. The first Fragment is called "FirstFragment," and the second Fragment is called "SecondFragment." The display shows the number of times certain Activity lifecycle callback methods have been invoked on a MainActivity instance since the app started. These methods are onCreate(), onStart(), and onResume().

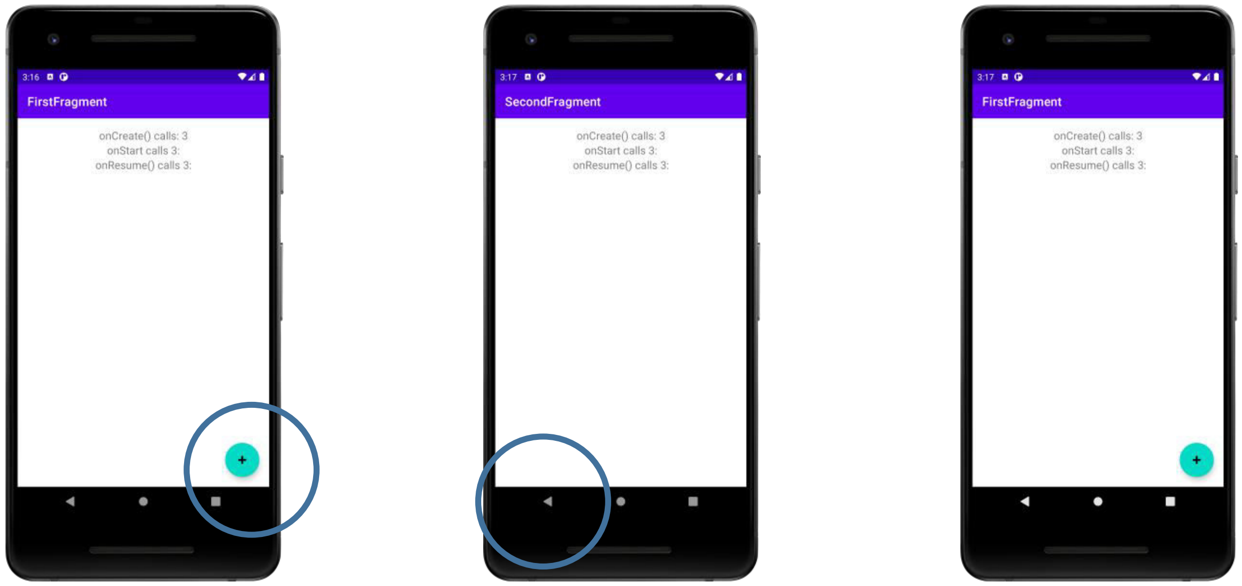
When the user rotates the screen, MainActivity will move through its Activity lifecycle, calling the lifecycle methods mentioned above while displaying up to date counts.



```
onCreate() calls: 2
onStart calls 2:
onResume() calls 2:
```

```
onCreate() calls: 3
onStart calls 3:
onResume() calls 3:
```

In addition, when FirstFragment is active and the user clicks on the floating action button (fab), the button with the "+" symbol, the app responds by navigating to the SecondFragment. The user can then hit the back button to navigate back to FirstFragment. You'll notice that simply moving between the FirstFragment and the SecondFragment does not cause changes to the MainActivity lifecycle.



```
onCreate() calls: 3
onStart calls 3:
onResume() calls 3:
```

```
onCreate() calls: 3
onStart calls 3:
onResume() calls 3:
```

```
onCreate() calls: 3
onStart calls 3:
onResume() calls 3:
```

See the screencast, LabFragments.mp4, that's included in the Lab directory.

Testing

There is one test case with several evaluation points. Each evaluation occurs at a step labelled "evaluation point"

This test case operates as follows:

1. Start the FragmentLab app in portrait mode
2. Check the lifecycle method invocation counts (evaluation point 1)
3. Rotate the device to landscape mode
4. Check the lifecycle method invocation counts (evaluation point 2)
5. Rotate back to portrait mode
6. Check the lifecycle method invocation counts (evaluation point 3)
7. Click on the fab button to navigate to the SecondFragment
8. Check the lifecycle method invocation counts (evaluation point 4)
9. Rotate the device to landscape mode
10. Check the lifecycle method invocation counts (evaluation point 5)
11. Rotate back to portrait mode
12. Check the lifecycle method invocation counts (evaluation point 6)
13. Click on device back button
14. Check the lifecycle method invocation counts (evaluation point 7)

After completing your solution, you will record a screencast while performing a manual test. Afterward, you will submit your code and the screencast via git. You can record a screencast using services available in the Logcat console. See: https://developer.android.com/studio/debug/am-video

Submission

When you are ready just commit your solution to your repo on GitLab by running the following commands:

```
% git add path/to/changed/files
% git commit -m "completed Lab5_Fragments"
% git push origin main
```

Note: if you have not already pushed this branch to your repo on GitLab you will need to make a slight modification for this first time and run this instead:

```
git push –u origin main
```

This sets up tracking between your local branch and a branch with the same name on your repo in GitLab.

Some Implementation Notes

We are providing template code and layout resources for this application. Only modify the areas marked with the word TODO.

We have done our testing on an emulator using a Pixel 5 AVD with API level 31. To limit configuration problems, you should test your app against a similar AVD.