

# Lifecycle-Aware Components Lab

---

## *Lifecycle-Aware Components*

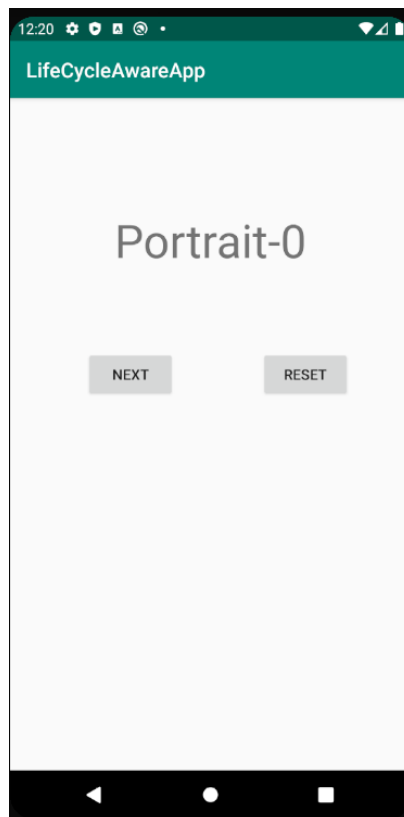
### Objectives:

This week's Lab explores the use of Lifecycle aware components, view model and live data. This lab is a little different from the previous labs in the sense that we have given you a very minimal skeleton code and you are expected to implement your own logic to make the app work in the way described ahead. There will be minimal TODOs given to you for this app and these are to be treated **only as suggestions**. This will give you more freedom to design your own workflow and logic for the app. This will be helpful in preparing you for final projects.

### App Behavior:

This Lab involves an app which will show your current orientation of the phone and a counter which can be incremented by pressing the next button. Whenever you change the orientation of the phone, the current instance of your activity is destroyed, and a new instance is started. You need to add the information of the list below to the buttons.

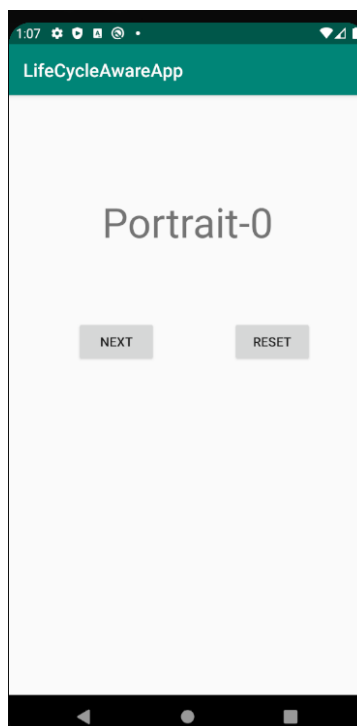
When you open the app following screen will show up with the screen orientation and counter value.



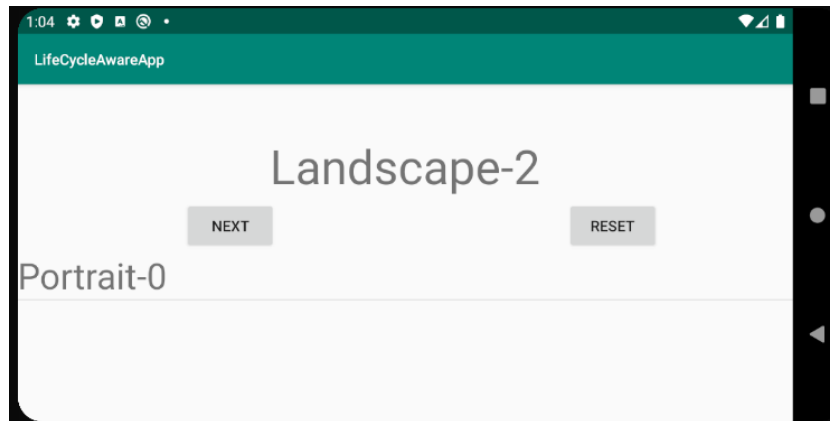
If you press the “Next” button 3 times following will be the result:



If you press the “Reset” button, the counter will reset, and it will go back to the first screen.

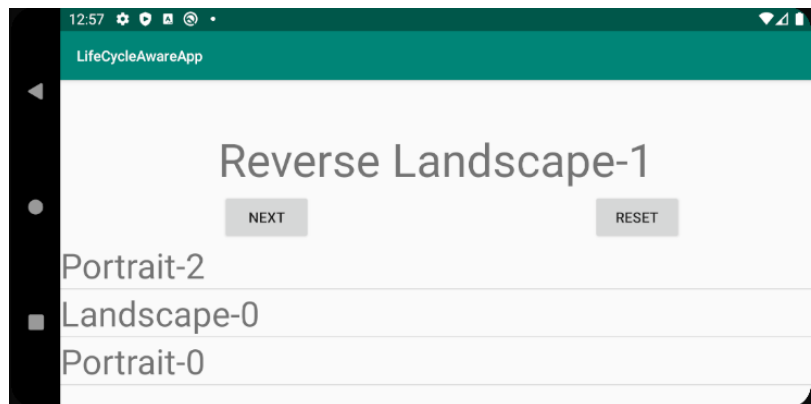


Now when you rotate the device counterclockwise and press the “Next” button 2 times following will be the output:



Notice “Portrait-0”. Each time you rotate the phone, the last value on the main text field should be added to the list. For example, the resultant of following actions after opening the app:

1. Press “NEXT” button 2 times.
2. Rotate phone counterclockwise.
3. Press “NEXT” button 1 time
4. Press “RESET” button
5. Rotate phone clockwise
6. Rotate phone clockwise
7. Press “NEXT” button 1 time



Please see the video included in the lab to see the output at each intermediate step

### Implementation Notes:

1. Download the application skeleton files and import them into your IDE.
2. You can modify, add, and delete any of the classes or function with some restrictions:
  - i) The entry point of app should be *LifecycleMainActivity*.
  - ii) You must use the included layout files only.
3. The displayed text should not have any space between the screen orientation, “-” and the counter value. For example: “REVERSE LANDSCAPE-0”.
4. We have implemented a supporting function called “getScreenOrientation” in *LifecycleMainActivity* and other functions in *HistoryListAdapter* for you.

### Testing:

The test cases for this Lab are in the Lab5\_LifecycleAware\app\src\androidTest\java\course\labs\lab5\_lifecycle\_aware project. You can run the test cases either all at once, by right clicking the test package and then selecting Run ‘Tests in ‘course.lab...’, or one at a time, by right clicking on an individual test case class and then continuing as before. The test classes are Robotium test cases. As you implement various steps of the Lab, run the test cases every so often to see if you are making progress toward completion of the Lab.

### Warnings:

These test cases have been tested on a Pixel 3 AVD emulator with API level 29. To limit configuration problems, you should test your app against a similar AVD. Also if you feel that you made no change since your last test of your code but the tests suddenly start failing, give the emulator some time to refresh memory. If the test case still fails wipe the data from your emulator and run the test case again

Once you’ve passed all the test cases, submit your project to GitLab.

### Submission

To submit your work, you will need to commit your solution to your repo on GitLab by running the following command: `git push origin master`.

### Additional Notes and Tips:

- 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 master`. This sets up tracking between your local branch and a branch with the same name on your repo in GitLab.
- Thinking for yourself : Sometimes you might encounter problems which you might have already solved while doing previous labs. Learn to debug your errors and make sure to check the test files and figuring out why you might be failing the tests.