Services

CSD3156 Mobile and Cloud Computing Spring 2023

Overview

This lab provides exercises and guidelines to gain familiarity with the Android Service component and its various implementations.

Services
https://developer.android.com/guide/components/services

Coroutines
https://developer.android.com/kotlin/coroutines/

WorkManager
https://developer.android.com/codelabs/android-workmanager#0
https://developer.android.com/courses/pathways/android-basics-kotlin-unit-6-pathway-1

Outcomes

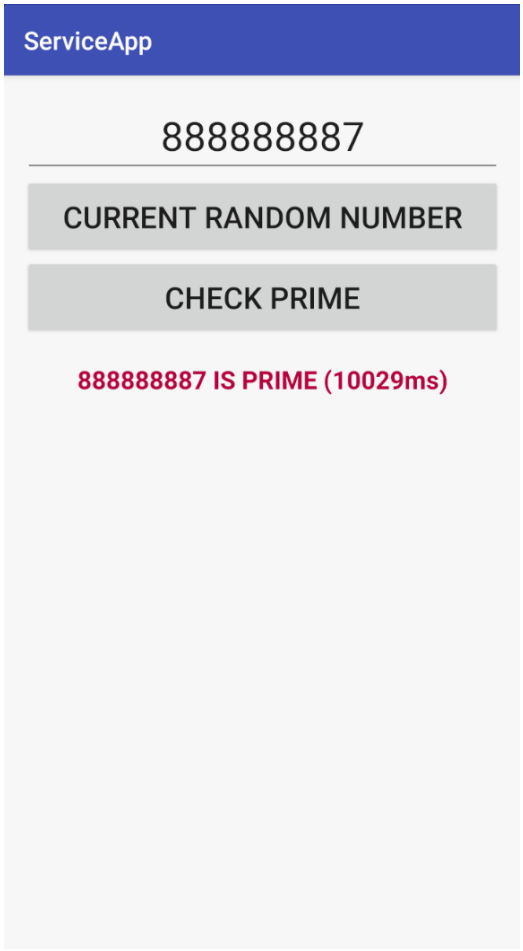
Upon completion of the session, you should be able to:

- Create a started bound Service that serves a method to an Activity
- Create a simple WorkManager with input and output handling

Creating and using Services

The goal of this exercise is to implement some common Android background tasks.

Fork the repo **csd3156-lab4-2023** and inspect the code within the project. This code is incomplete as usual. Examine the skeleton code before working on the lab. Refer to the screenshot on the following page for the descriptions that follow



Note that the top EditText field is editable. As always, you need to recreate a UI similar to the screenshot above.

IMPORTANT:

Ensure that the UiInstrumentedTest.java test file compiles without errors. Do not edit any existing package, class, variable, method or layout file names.

Your first task is to create a second service called **RandomService**:

1. a Started Service that allows local binding
2. the service is started when ServiceActivity starts, and generates a random 9-digit number between 100000000 and 999999999 every one second, and stores it. The generated number is also printed out via logcat in the format: e.g., "RandomService: 888888887 added".
   (PRO TIP: use coroutines in RandomService for l33t-looking code)
3. ServiceActivity also binds to the service upon start, and unbinds when it is stopped

4. the button **CURRENT RANDOM NUMBER** will get the current (last) generated random number from the list within RandomService

Your second task is to create a Scheduled Service that uses a **PrimeWorker** class:

1. The current number shown in the top EditText field will be sent to the PrimeWorker when the user clicks on **CHECK PRIME**
2. PrimeWorker will then test whether the number is a Prime number using the brute force method:

```
private boolean isPrime(long num) {
    for (int i = 2; i < num; ++i)
        if (num % i == 0)
            return false;
    return true;
}
```

3. Use the WorkManager component to process the PrimeWorker to do the required work. Although it is a scheduled service, it will be run ASAP (whenever the system allows) after the user clicks **CHECK PRIME** and will only be scheduled once per click.
4. Additionally, the work will only be done if the user's **device has enough battery**
5. Once the work is done, show the output in the bottom TextView (with the red text) in the format shown in the screenshot. As shown, the time taken to test whether it is a prime number is also shown in milliseconds (ms). Most numbers will be very fast, but for large actual primes, it will take a while.

Lab Exercise 4

Due Date: Feb 12 2023 2359 hrs

Fork the repo **csd3156-lab4-2023** and then clone it. Inspect the code in the project. This code is incomplete obviously.

1. Implement the code needed to create the RandomService Started/Bound Service
2. Implement the code needed for PrimeWorker to work with WorkManager
3. Commit and push all changes to your forked repository.

END OF DOCUMENT