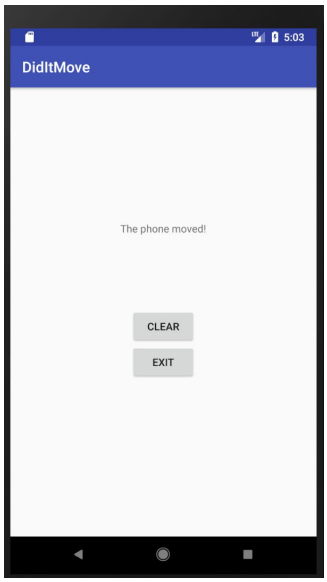
CMPS 121 - Assignment 3

Spring 2019

Submission deadline: June 2nd, 2019 at 11:59 pm

Description:

In this assignment, you need to build an application that detects whether someone moved your phone while you left it somewhere. The UI is as below:



The app tells you if the phone was moved. When you start the app, it waits 30 seconds before starting to detect motion. Then, it remembers if someone has moved the phone. When you get back and use the phone again, it tells you whether the phone has been moved.

Details

1. The service:

When you start the app, it should create a background service running on its own thread. You might want to take the code of ServiceExample which will be provided in lecture 13 as a starting

point for your implementation. The background service remembers the time T_0 when it has been started, and listen to the accelerations. For simplicity, we don't really detect movement, but only whether the phone was moved away from a flat position, that is, we detect if there is a significant acceleration in the X or Y directions. Read just the acceleration in the X and Y axes, so you don't have to worry about removing the effect of gravity; in any case, it's unlikely that a curious housemate manages to pick up the phone and keep it completely horizontal.

The service remembers in a variable `first_accel_time` the time when the first significant acceleration occurred. When you start the service, you should set `first_accel_time = null`. Whenever the service detects that there is acceleration above the significance threshold at time T_1, it does as follows:

- If $T_1 - T_0 > 30$ seconds, `first_accel_time =` T_1 (here, I assume T_1 is an object e.g. a Date; this is just an idea).

The service has a method, that can be called from the activity. The method is used to check if someone moved the phone at least 30 seconds ago, to give you time to pick the phone up and check. The method does something like this (this is not code, it's just to give you the idea):

```
public boolean didItMove() {
   Date d = new Date();
   boolean moved = false;
   synchronized(myLock) {
        if(first_accel_time != null && d - first_accel_time > 30
seconds)
         moved = true;
    }
   return moved;
}
```

In the above, `myLock` is used because `first_accel_time` is an object (might be of type Date for instance) or a long, and those types do not have an atomic update in Java. Another alternative would be to do without the lock and declare `first_accel_time` to be of type `AtomicLong`.

2. The activity:

When the activity executed `onResume()`, it must do the following:

- Starts the service.
- Binds to it.
- Calls `didItMove()` to the Service, which calls `didItMove()` in the ServiceTask (you need to relay the call from Activity to Service to ServiceTask). This tells the activity if the phone was moved; update the UI accordingly.

When someone presses the CLEAR button, the activity should call the service, and the service should:

- Set T_0 to the current time
- Set `first_accel_time` to `null`.

When someone presses the EXIT button, the activity should unbind and stop from the process, then exit. You can exit from an activity by calling `finish()`.

Note: When the activity executes `onPause()`, it should unbind from the process, but it should NOT stop it! If you stop the service in `onPause()`, `onStop()`, or `onDestroy()`, then the phone cannot detect whether it's being moved if the screen goes off, or if the person moving it exits from the app by pressing the Home button.

The only thing that should stop the service is the EXIT button.

3. Keeping the phone awake:

On most phones, the above will work fine. On some phones, though, the sensor updates will cease after a while after the phone goes to sleep. To force the phone to keep the CPU on, we acquire a `WakeLock`. Use these lines:

➔ In the `onStartCommand()` method of the service:

```
PowerManager powerManager =
(PowerManager)getSystemService(POWER_SERVICE);
PowerManager.WakeLock wakeLock=
powerManager.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,"MyTag"
);
wakeLock.acquire();
```

➔ In the `onDestroy()` of the Service:

```
wakeLock.release();
```

Note that you need to ask permission in `AndroidManifest.xml` in order to use the `wakeLock`.

Testing:

You can test your code on an Android device or using the virtual emulator by feeding the acceleration to the emulator. In order to do that, go to the setting of the emulator, choose virtual sensors, and they try to either move the device shown on the view or feed the accelerations you want.