**Fun with alarms Part 1: Alarms without services**

**Overview (Video) (Source code for first part)**

* Android allows you to set alarms
* When the alarm goes off, the OS sends your application an intent.
* However, receiving these intents is a bit tricky.
* Consequently, this approach is probably not sufficient for most applications.
* We’ll need to use services, which is Part 2
* Note that Handler is also used to make things happen in the future. Thus, one should consider using it instead of AlarmManager.
  + Note: The Alarm Manager is intended for cases where you want to have your application code run at a specific time, even if your application is not currently running. For normal timing operations (ticks, timeouts, etc) it is easier and much more efficient to use Handler.

**Starting**

* Start with a new Android project
  + called FunWithAlarms
  + with a button and button listener, and
  + TextView, called textView
* Make a new class, we’ll call OneShotAlarmReceiver, derived from android.content.BroadcastReceiver
  + Note, it must be a new class. It cannot be a local class of FunWithAlarms.
  + As we will see, some BroadcastReceivers can be local classes.
* In OneShotAlarmReceiver, add function onReceive(Context context, Intent intent)
* In this function, add Log.d("OneShotAlarmReceiver ","Alarm Expired");
* In the manifest, with the application section (as far as I know, everything must be in the application section), add
  + <receiver android:name=". OneShotAlarmReceiver "> </receiver>, alternatively, add a “receiver” node under the applications tab
  + Note, this is the name we gave our broadcast receiver

**Setting the alarm**

* In FunWithAlarms, within the button onClick listener add
  + Intent intent = new Intent(FunWithAlarms.this, OneShotAlarmReceiver.class); // again, our broadcast receiver
  + PendingIntent pendingIntent = PendingIntent.getBroadcast(FunWithAlarms.this, 0, intent, PendingIntent.FLAG\_ONE\_SHOT); // other flags make other types of alarms, like periodic alarms
  + AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM\_SERVICE);
  + alarmManager.set(AlarmManager.RTC\_WAKEUP, System.currentTimeMillis() + (5 \* 1000), pendingIntent); // (5\*1000) so the alarm will expire in 5 seconds
  + Log.d("FunWithAlarms1","Alarm is set");

**Run program**

* Press button
* Note that all log shows things worked as expected.
* Try setting the alarm and closing the app (by pressing the back button or the home button). The alarm still expires as expected.
* Try setting alarm and then sleeping the phone. The alarm still expires as expected.

**Problems (Video) (Source code for second part)**

* How to run the main activity?
* We can’t directly access the FunWithAlarms object.
* Instead we’ll use another intent to trigger a BroadcastReceiver that is part of FunWithAlarms
  + Why not make OneShotAlarmReceiver part of FunWithAlarms?
  + Maybe one can, but I couldn’t get it to work.

**Add BroadcastReceiver to FunWithAlarms**

* In FunWithAlarms, add local class
  + public BroadcastReceiver receiver=new BroadcastReceiver() {
  + public void onReceive(Context context, Intent intent) {
  + Log.d("FunWithAlarms ","Received TimesUp message");
  + }
  + };
* Register this BroadcastReceiver
  + In FunWithAlarms.onCreate add
    - registerReceiver(receiver, new IntentFilter("edu.udel.eleg454.FunWithAlarms.TIMESUP"));// note that this name com. FunWithAlarms.TIMESUP must be the same as the name below (It is better to use a single string constant.).
* We must also unregister this broadcast receiver in onPause
  + In FunWithAlarms, add
    - @Override
    - public void onPause () {
    - super.onPause();
    - Log.d(“FunWithAlarms1”,”onPause”);
    - unregisterReceiver(receiver);
    - }
* And reregister in onResume
  + In FunWithAlarms, add
    - @Override
    - **public** **void** onResume () {

**super**.onResume();

Log.*d*("FunWithAlarms1","”onResume");

registerReceiver(receiver, **new** IntentFilter("edu.udel.eleg454.FunWithAlarms.TIMESUP")); // that same constant again!

* Call this BroadcastReceiver from OneShotAlarmReceiver
  + In OneShotAlarmReceiver.onReceive add
    - Intent broadcast=new Intent("edu.udel.eleg454.FunWithAlarms.TIMESUP"); // the same name again, we really should use a constant!
    - context.sendBroadcast(broadcast); // context is the first argument of onReceive. Change the argument name from arg0 to context
* Note: now we have two BroadcastReceivers. One is registered in the manifest and one is registered in the code.

**Run program**

* The log shows that things run as expected
* Now test what happens if the phone sleeps before the alarm goes off. The alarm expires so that first BroadcastReceiver is called.
* But the activity is not running, so it is not called.
* In some cases, this behavior might be fine.
* However, if you want to catch this event, you must use a service