Lab.03 Threads and Services

1. Local service

Create a local service that counts from a value (1st parameter) to zero, and waits a time (2nd parameter) between 2 iterations.

1. Create a new project *Lab03.Surname* with the *ServActiv* activity.
   - Add 2 EditText *et_time_wait* and *et_count* (default values 500 and 20, respectively).
   - Add a ToggleButton *tb_localService* which (de)activate a local service, sending as parameters the values of the two EditText.
     Do not forget to show each action/event with a log.
2. Create a new Android Service with the name and label *LocalServ*.
   - Collect the 2 integer parameters *count* and *time_wait*. Show these values in a log.
   - The service task is to count in a loop from the value *count* to 0. At each iteration, **wait** *time_wait* milliseconds and display the iteration number in a log.
     Use *wait()*, *Thread.sleep()* within a *try, catch*, or *SystemClock.sleep()*.
   - Remember that the service runs **on the main thread**, so avoid frezeen UI.
3. Check that the application works by looking at the system log:
   - Is the service stopped at the end of the account?
   - What happens if you stop the Service before it ends?

2. Remote Service

Create a remote service (or adapt the previous local service) so the activity can interact with it.

1. Implement the necessary method for a remote Service. Same two parameters as before. Do not forget to display in a log when the service is started/stopped, received parameters and the value of the iteration each time the loop is executed.
2. Add to the activity *ServActiv* a new ToggleButton *tb_remoteService*:
3. Add a TextView (*tv*) and an EditText (*et* (default value 20).
4. Add 2 buttons (*but_get*, *but_send*):
   a) *but_get* requests the current value of the internal counter of the remote service loop and displays it in *tv*.
   b) *but_send* sends the value of *et* to the remote service and replaces the internal counter of the loop in service (ie, the number of iterations executed).
   If remote service has finished, it starts a new loop.

3. Run background tasks and display data on screen

We want to run one or more tasks in background: count in a loop, show the iterator variable in the main activity and wait a specific time between each iteration (value of EditText *et_time_wait*). The loops begin on the value of EditText *et_count* and end on zero.

1. In the main activity layout (*ServActiv*), add the following elements:
   a) A *but_task* button with text *Long Task*, to launch one AsyncTask.
   b) A TextView *tv_result*, to show the results of all bakcground tasks.
   c) A *but_thread* button with text *Thread*, to launch one Thread.
   d) A *ProgressBar* for **each** background task, which show the iterator of each loop.
2. The maximum value of a *ProgressBar* depends on *et_count* of its task. When a background task ends, its associated ProgressBar is deleted from the main activity's layout.
3. Try to launch as many background tasks as possible.
4. Try to show the ProgressBars when activity is recreated (e.g. screen rotation).

4. Broadcast receiver

We will use Broadcast Receivers in our application to respond to certain events:

1. Each time the screen is turned off/on, the remote service is launched.
2. Each time the USB cable is inserted/removed, the local service is launched.
3. At the end of the service that has been launched through the main activity (both local and remote), change the state of the corresponding ToggleButton (i.e., switch from ON to OFF).