Lab2 – Asynchronous Executions

Threads & Handlers

The UI runs in an activity object, which is itself integrated in a low level linux process. The latter is responsible for refreshing the output and interacting with the user; it's like having in the main thread a loop doing that all the time. So, if the programmer decides to do something that needs time to run, this cannot be done on the main thread for the simple fact that it will freeze the UI!

Therefore, in such cases, asynchronous tasks (threads) should be used. However, if they tend to interact with the UI, then this could cause race problems which have to be solved appropriately by the UI thread. For this purpose, a set of objects, classes (Handler, AsyncTask) have been added to the Android framework which we're going to cover in this lab.

Handler Example

Create a project "fr.android.moi.handler" with the following activity "ProgressTestActivity". Create the following layout and associate it to the previous activity :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <ProgressBar android:id="@+id/progressBar1"
                style="?android:attr/progressBarStyleHorizontal"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:indeterminate="false"
                android:max="10" android:padding="4dip">
      </ProgressBar>
        <Button android:text="Start Progress"
           android:onClick="startProgress"
                android:id="@+id/button1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content">
      </Button>
</LinearLayout>
```

Change the content of your activity to look similar to the following code (no need to change the “extends” class which could also be AppCompatActivity) :

```
public class ProgressTestActivity extends Activity {
        private Handler handler;
        private ProgressBar progress;


/** Called everytime the activity is created */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                progress = (ProgressBar) findViewById(R.id.progressBar1);
                handler = new Handler();
        }

        public void startProgress(View view) {
                Runnable runnable = new Runnable() {
                        @Override
                        public void run() {
                                for (int i = 0; i <= 10; i++) {
                                        final int value = i;
                                        // simulate a slow network !
                                        try {
                                                Thread.sleep(2000);
                                        } catch (InterruptedException e) {
                                                e.printStackTrace();
                                        }
                                        handler.post(new Runnable() {
                                                @Override
                                                public void run() {

        progress.setProgress(value);
                                                }
                                        });
                                }
                        }
                };
                new Thread(runnable).start();
        }
}
```

Run your application, test it and understand how it's coded and what happens when it runs in detail.

What happens when your click multiple times on the button ? You should be able to explain why it's running the way it is !

AsyncTask

I want you to use this special class to do exactly the same thing as in the previous exercise. Asynctask is a wrapper class around handlers and threads that can be used too to run asynchronous tasks.

In the same project, comment out the previous code and replace it with the following wherever it's necessary.

doInBackground is used to run the asynchronous code
publishProgress is used to publish the progress on the UI

And maybe some other functions to code ! Refer to the following page for information and an example on how to use this class (read up to "cancelling a task").

https://developer.android.com/reference/android/os/AsyncTask

Mainly, understand the following code and modify it to run the same code as in the previous exercise !

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
            // Escape early if cancel() is called
            if (isCancelled()) break;
        }
        return totalSize;
    }

    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}
```