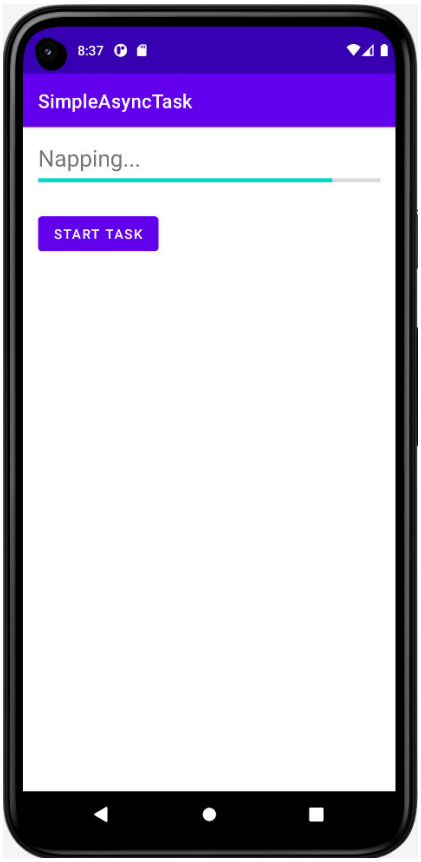
Homework 3

Problem 1

Complete the Codelabs for Android Developer fundamentals Lesson 7.1: AsyncTask.

https://developer.android.com/codelabs/android-training-create-asynctask?index=..%2F..%2Fandroid-training#2

Do the Coding challenge in step 6 where you'll add a progress bar to the app and display the progress of the napping time.



You can read about working with the ProgressBar here https://abhiandroid.com/ui/progressbar.

Some changes that you'll need to make in **doInBackground** and **onProgressUpdate** are shown below.

```
@Override
protected String doInBackground(Void... voids) {

    // Generate a random number between 0 and 10
    Random r = new Random();
    int n = r.nextInt(11);
```

```
    // Make the task take long enough that we have
    // time to rotate the phone while it is running
    int s = n * 20;

    // Sleep for the random amount of time
    mProgressBar.get().setMax(s);
    try {
        for (int i = 0; i < s; i++) {
            Thread.sleep(1);
            publishProgress((Integer) i);
            if (isCancelled()) break;
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    // Return a String result
    return "Awake at last after sleeping for " + s + " milliseconds!";
}

@Override
protected void onProgressUpdate(Integer... values) {
    mProgressBar.get().setProgress(values[0]);
}
```

Make sure you also declare the variable mProgressBar in both the MainActivity class and the SimpleAsyncTask class just like with the mTextView variable, and assign the correct values to it.

Problem 2

Continuing with Problem 1 above, instead of putting the thread to sleep to simulate the time delay, actually do an audio file download. You can download an audio file from the hwang.lasierra.edu server under the csmajor account (password: the same one) in the public_html directory. You can browse to it to see the files at http://hwang.lasierra.edu/~csmajor/

You'll need to do a FTP download in the doInBackground method. If you've not used FTP to download files before, you can try it by running an Open Source app called FileZilla https://filezilla-project.org/download.php. Below is the code for my Hymnal app for downloading a song from my server. You'll need to make the appropriate changes.

```
// http://android-er.blogspot.com/2015/02/progressdialog-and-asynctask.html

// in the following three parameter types <String, Integer, Integer>
// the first is the value that is passed into here from the
myAsyncTask.execute(audioFilename) call
//    this in turn is passed to the doInBackground(String... path)
// the second value is passed to onProgressUpdate(Integer... values) for updating
the progress
// the third is the return value to myAsyncTask.execute(audioFilename).get() after
completion
//    this needs to match the return type of the doInBackground() and the type
passed to onPostExecute(Integer value)
public class AsyncDownload extends AsyncTask<String, Integer, Integer> {
    Context context;
```

```
    File mySongsFolderPath;
    //    boolean running;
    //    private final AsyncDownloadInterface asyncDownloadInterface;  // for
returning selected song number to MainActivity
    ProgressDialog progressDialog;

    public AsyncDownload(Context context, File mySongsFolderPath) {
        //    public AsyncDownload(Context context, File mySongsFolderPath,
MainActivity activity) {
        this.context = context;
        this.mySongsFolderPath = mySongsFolderPath;
        //        this.asyncDownloadInterface = activity; // still for testing
    }

    @Override
        protected Integer doInBackground(String... path) {
        // path[0] format: "piano2/"+songName+".mp3" or "organ2/"+songName+".mp3"
        Log.i("myTag", "in doInBackground path=" + path[0]);
        final String host = "hwang.lasierra.edu";
        String username = "replace with csmajor account";
        String password = "replace with password for csmajor";
        final int port = 21;
        File localPath = mySongsFolderPath;   // doesn't work for sdcard
        //        File localPath = context.getExternalFilesDir(null);
        //        File localPath = context.getExternalCacheDir();
        try {
            //onProgressUpdate("Testing");
            FTPClient ftpClient = new FTPClient();
            String directory = path[0].substring(0, path[0].indexOf('/'));   //
extract the directory portion of the path (either piano2 or organ2)
            String filename = path[0].substring(path[0].indexOf('/') + 1);    //
extract the filename portion of the path
            File localFile;
            //            ftpClient.setDefaultTimeout(5000);  // different timeouts.
Don't know which to use
            ftpClient.setConnectTimeout(5000);
            //            ftpClient.setDataTimeout(5000);
            ftpClient.connect(host, port);         // connect to FTP server
            //            ftpClient.setSoTimeout(5000);
            ftpClient.login(username, password);   // login
            ftpClient.enterLocalPassiveMode();
            ftpClient.setFileType(FTP.ASCII_FILE_TYPE);
            ftpClient.changeWorkingDirectory("public_html");  // go to directory on
the server
            localFile = new File(localPath, directory + "/" + filename);
            OutputStream outputStream = new BufferedOutputStream(new
FileOutputStream(localFile)); // set up output file
            boolean success = ftpClient.retrieveFile(filename, outputStream);   //
download file
            outputStream.close();
            if (success) {
                //                asyncDownloadInterface.stopProgressBar();
                return 0;   // file downloaded successfully
            }
            else {
                localFile.delete(); // the file is created locally even though it
does not exists on the server
                return 1;   // file does not exist
```

```
            }
        }
        catch (IOException e) {
            e.printStackTrace();
            Log.i("myTag", "doInBackground error: " + e);
            return 2;   // no network connection error
        }
        //return true;
    }

    @Override
        protected void onProgressUpdate(Integer... values) {
        super.onProgressUpdate(values);
        //        progressDialog.setMessage(String.valueOf(values[0]));
    }

    @Override
        protected void onPreExecute() {
        //        Log.i("myTag", "in onPREExecute");
        super.onPreExecute();
        //        progressDialog = ProgressDialog.show(context, "Downloading file",
"Please wait...");
        //        progressDialog.setCanceledOnTouchOutside(true);
        //        progressDialog.setOnCancelListener(new
DialogInterface.OnCancelListener() {
        //            @Override
        //            public void onCancel(DialogInterface dialog) {
        // //                running = false;
        //            }
        //        });
    }

    @Override
        protected void onPostExecute(Integer value) {
        //        Log.i("myTag", "in onPOSTExecute");
        super.onPostExecute(value);
        //        progressDialog.dismiss();
        //        running = false;
        //        SharedPreferences sharedPreferences =
context.getSharedPreferences("shared preferences", MODE_PRIVATE);
        //        SharedPreferences.Editor editor = sharedPreferences.edit();
        //        editor.putBoolean("downloading", false);
        //        editor.apply();
    }

}
```