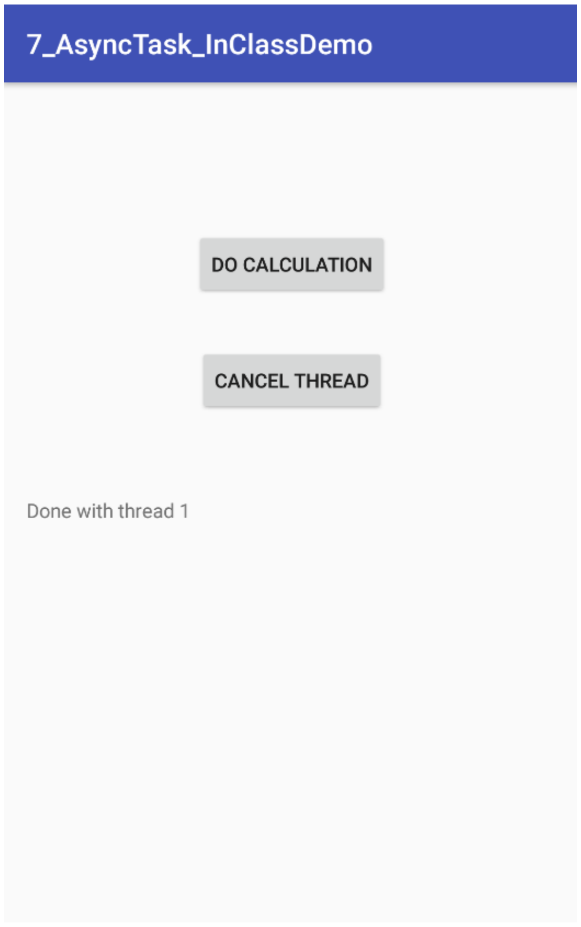
AsyncTask - In class Lab and Project

I am demoing a concept so I am not going to worry about handling phone rotations



Build app with a 2 buttons above
useCalculationResult
and a text working

have some constants

```java
private static final int NUMBER_UPDATES = 2;
private static final int ONE_SECOND = 500;
private static final String RUNNING_CALC = "Running Calculation for thread ";
private static final String DONE = "Done with thread ";
```

add associated button handlers

```
public void doCalculation(View view) {
    //run calculation
    myTextView.setText(RUNNING_CALC);
    runCalcs(NUMBER_UPDATES);
    myTextView.setText(DONE);
}
```

create a long running function

```
void runCalcs(Integer numb_updates)
{
    for (int i = 0; i <= NUMBER_UPDATES; i++) {
        try {
            Thread.sleep(ONE_SECOND);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Now click the do calc button (nothing happens, no text change either) that’s cause the main thread cant change the text until we return from this function, bummer so lets put it in a thread.

```
//notice also that this is static, so it does not hold an implicit referenidce to
enclosing
//activity, rotate the phone and activity is GCed

private static class MyTask extends AsyncTask<Integer,Void,Void>{

    private final MainActivity act;
    private static int numberInstances=0;   //how many threads are running

    public MyTask(MainActivity act){
        //want to be able to modify UI
        //in parent so save parent
        this.act = act;
    }

    @Override
    protected Void doInBackground(Integer... params) {
        act.runCalcs(params[0]);
        return (null);
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        act.myTextView.setText(RUNNING_CALC + Integer.toString(++numberInstances));
    }

    @Override
    protected void onPostExecute(Void aVoid) {
        super.onPostExecute(aVoid);
        act.myTextView.setText(DONE + Integer.toString(numberInstances--));
    }
}
```

change the main code

```
public void doCalculation(View view) {
    doThreadedCalculation();
}

private void doThreadedCalculation() {
    MyTask task = new MyTask(this);
    task.execute(NUMBER_UPDATES);
}
```

But all the UI is still available, and I want the user to only be able to run 1 thread at a time?
Progress bar? Nah does not solve the multiple click problem

How about if we go and disable all the elements while we run the calculations?
PITB. Have to get a reference to each and set enabled to false.

How about if we pop a Dialog that indicates that we are busy so that the user cant touch other buttons? When thread is finished it stops that UI?

Add private variable

```
private ProgressDialog myProgressDialog;
```

and a couple of methods to start and stop it

```
private void progressDialog_start() {
        myProgressDialog = new ProgressDialog(this);
        myProgressDialog.setTitle("Please wait");
        myProgressDialog.setMessage("Notice user cannot interact with rest of
UI\nincluding starting additional threads");
        myProgressDialog.setCancelable(false);
        myProgressDialog.show();
    }

    private void progressDialog_stop(){
        myProgressDialog.dismiss();
    }
```

start and stop in onPre and onPost in thread

so what if the thread goes on forever and we want to cancel it?
        Make the async a member variable
        Add a cancel button to progress dialog
        Add an onclick handler, if user clicks call async.cancel(true)

```
      myProgressDialog.setButton(DialogInterface.BUTTON_NEGATIVE, "Cancel",
              new DialogInterface.OnClickListener() {
                  @Override
                  public void onClick(DialogInterface dialog, int which) {
                      dialog.dismiss();
                      task.cancel(true);
                      //myProgressDialog = null;
                  }
              });
```

        tidy up doInBackground so it checks for canceled

```
protected Void doInBackground(Integer... params) {

    for (int i = 0; i <= params[0]; i++) {
        act.runCalcs(params[0]);
        if (this.isCancelled());
            break;
    }
    return (null);
}
```

add a canceled method

```
protected void onCancelled(Void aVoid) {
    super.onCancelled(aVoid);
    numberInstances--;
    act.myTextView.setText(USER_CANCELED);
    act.progressDialog_stop();
}
```

Finished this 2/25/16