Generalitat de Catalunya
Departament d'Educació
Institut Baix Camp

c/Jacint Barrau,1 – 43201 Reus
Tf.977310953 / Fax.977314721
http://www.insbaixcamp.org
email: e3002594@xtec.cat

CFGS - Desenvolupament d'aplicacions multiplataforma
Mòdul 9 – Programació de serveis i processos
UF2 – Processos i fils

Per l'activitat heu de lliurar un PDF amb el següent contingut:

- Nom o noms dels membres que fan l'activitat (2 alumnes, no s'admeten grups més grans i només s'admetrà un grup individual per classe).
- URL del repositori on hi ha els fonts (github, bitbucket,...).
- Noms d'usuari utilitzats per cada membre del grup del repositori del núvol.
- Explicacions, respostes, línies de codi creades/modificades i bolcats de pantalla de l'execució dels diferents punts que es demanen en la tasca i que serveixin per comprovar la correcta realització de la mateixa.

El pdf s'ha d'anomenar seguint el format NomCognom1Cognom2.pdf del membre que fa el lliurament. Si un grup està format per dos membres NOMÉS UN l'ha de lliurar.

Tasques entregades fora de termini i/o on els fonts del repositori del núvol hagin estat creats/ modificats fora de termini tindran, com a màxim un 5 sempre que el lliurament i/o la modificació del repositori sigui de, com a molt, 3 dies després del venciment. Passats 3 dies no s'avaluarà la tasca i es qualificarà com a 0.

No s'acceptarà tampoc cap tasca que no es lliure en el format especificat i/o que no contingui de forma clara els punts que es demanen.

Enllaços d'interès:
https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html

Objectius:
Identificar les paraules reservades del llenguatge Java per gestió de processos/fils

Activitat 2.
Aquesta activitat conté dos parts. La primera l'ha de fer tothom, la segona part és opcional per l'alumnat que estudia en modalitat DUAL.

c/Jacint Barrau,1 – 43201 Reus
Tf.977310953 / Fax.977314721
http://www.insbaixcamp.org
email: e3002594@xtec.cat

Primera part

L'algoritme de cerca de numeros de la série de fibonacci s'implementa recursivament d'aquesta manera:

```
public static long calculaFibonacci(long numero) {
        double calcul = java.lang.Math.cos(54879854);
        if (numero==0) { return 0; }
        else if (numero==1) { return 1; }
        else {
            return (calculaFibonacci(numero-2) + calculaFibonacci(numero-1));
        }
    }
```

Si utilitzem fils, una possible implementació seria aquesta:

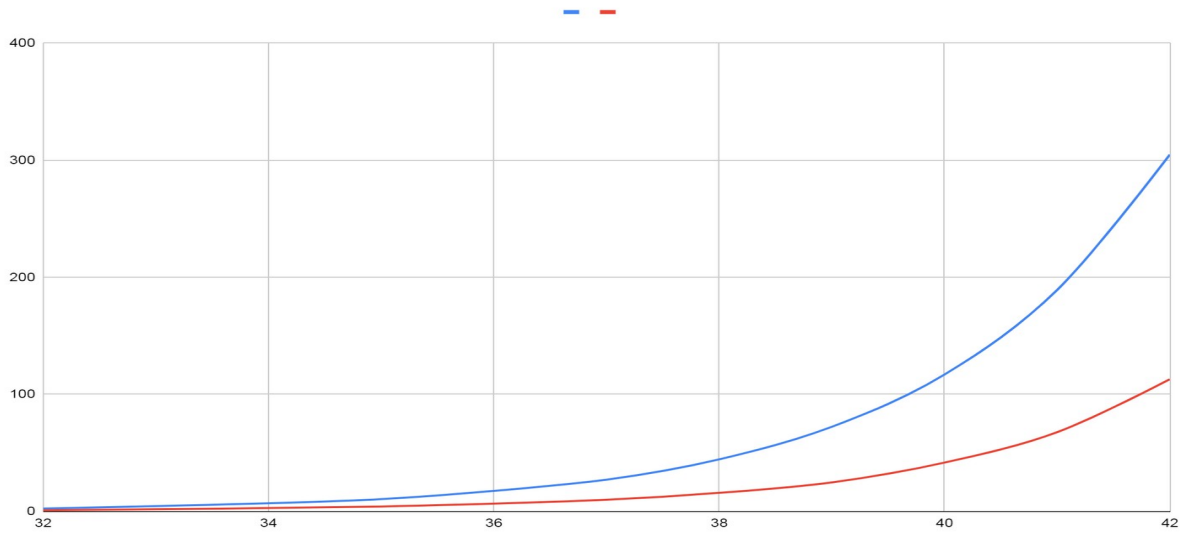
```
public class A_fibonacci_forkJoin extends RecursiveTask<Long> {
    long numero;
    public A_fibonacci_forkJoin(long numero){
        this.numero=numero;
    }
    @Override
    protected Long compute() {
        // ATENCIO **1** double calcul = java.lang.Math.cos(54879854);
        if(numero <= 1) return numero;
        A_fibonacci_forkJoin fib1 = new A_fibonacci_forkJoin(numero-1);
        //fib1.fork();
       A_fibonacci_forkJoin fib2 = new A_fibonacci_forkJoin(numero-2);
        fib2.fork();
         return fib1.compute()+ fib2.join();
         }
    public static void main(String[] args){
        ForkJoinPool pool = new ForkJoinPool();
        System.out.println("Calculat:  " + pool.invoke(new A_fibonacci_forkJoin(35)));
    }
}
```

Sobre els dos algoritmes:

- Comprova en diferents nombres quina de les dues versions del algoritme funciona més ràpid i dibuixa una gràfica comparativa dels temps d'execució en els dos algoritmes i diferents valors (interessa veure a partir de quin valor és més eficient un algoritme o l'altre)

- Treu el comentari de la línia ATENCIO 1 i torna a comprovar, en diferents nombres quina versió dels dos algoritmes funciona millor. Dibuixa una gràfica amb els diferents temps d'execució dels dos algoritmes i amb diferents valors (interessa veure a partir de quin valor és més eficient un algoritme o l'altre)

numero fibonacci	normal(ns)	Jork Join(ns)
1	195001	3885464
2	316348	3625567
6	322385	3540142
12	601302	3333674
24	52531505	71245669
30	940079889	425097220
32	2467012953	984070265
35	10511860594	4231354410
37	27140160725	9982301401
38	44515176231	15869117057
39	72343130850	24809656065
40	116981512986	41749262043
41	189262488484	67725070868
42	304883106475	112969575241



- Descriu amb les teves paraules la utilitat de la classe RecursiveTask

  RecursiveTask lo que hace es que por cada tarea hace diversas tareas para dividir el problema principal y así se pueda realizar mas rápido.

- Que signifiquen els mètodes compute, fork i join en referència als fils?

  -Ell metodo compute es un equivalente al run o al call y son para que se realice la tarea con diferentes hilos de forma recursiva

c/Jacint Barrau,1 – 43201 Reus
Tf.977310953 / Fax.977314721
http://www.insbaixcamp.org
email: e3002594@xtec.cat

-El fork y el join son opuestos, el fork lanza un hilo para una tarea y el join para decir que ya ha acabado el hilo.

Segona Part

Al programa MaximTask (podeu trobar en el pdf del moodle) cal fer les modificacions necessàries per què mostri per terminal el nombre de vegades que s'executa compute() i la divisió que fa del array cada vegada que es crida MaximTask de manera semblant a la que es pot veure a la següent sortida:

```
Inici càlcul
Comptador  1     Inici 0          Fi    100000000
Comptador  2     Inici 50000001   Fi    100000000
Comptador  3     Inici 0          Fi    50000001
Comptador  4     Inici 50000001   Fi    75000001
Comptador  5     Inici 50000001   Fi    62500002
Comptador  6     Inici 50000001   Fi    56250002
.....
Comptador  30    Inici 43750002   Fi    50000001
Temps utilitzat 65
Màxim es 1005
```

Que cal fer:

- Expliqueu amb les vostres paraules què fa l'algoritme.
- Que el programa mostri el nombre de vegades que es crida a si mateix.
- Que mostri el fragment d'array que es passa a MaximTask en cada crida.
- Provar amb diferents llindars i dir quin donar millor resultats i per què us sembla això
- Comentar tots els canvis introduïts al codi.