# COSC 2P13
# Project - Step 1

**Due date:** May 24th, 2018 at 23:55 (11:55pm)
**Delivery method:** the student needs to delivery the assignment only through Sakai.
**Delivery contents**: a document with a description and Java codes (see Submission instructions).
**Attention**: check the Late Assignment Policy.

## Project Overview

The final output of this project is expected to be a totally functional distributed system (application), which is a byproduct of designing and implementing a solution for a problem associated to each of its four incremental steps. This project thus involves implementation where all its steps must be coded using Java. For each of the steps, the students are expected to employ the concepts learned in class properly.

## Step 1 - Description

This step 1 consists of designing and implementing a thread-based concurrent application. The application needs to make use of the concurrency tools covered in class to organize the execution of the threads so that they can produce coherent output.

## Step 1 - Specifics

At this step, the students must design a solution for a concurrency problem using Java threads. The is described as follows. Assume a set W={$w_1$, $w_2$, …, $w_k$} of workers (threads) that need to perform the calculation of Pi collaboratively. Each thread is responsible for executing a given term of the Gregory-Leibniz Series to calculate Pi, a shown in the formula below. Each term is related of a thread, so we have a set of terms T={$t_1$, $t_2$, …, $t_k$} associated to each thread. The precision of the calculations directly relates to the number of elements in this slow-converging series. These threads are supposed to be independent of each other, without requiring any sort of synchronization. However, our whole application cannot allow full "parallelism" of these threads. A way chosen to organize the execution of these threads consists of the use of control variables, or control locks; consequently, assume a set R = {$r_1$, $r_2$, …, $r_k$} of variables. Therefore, associated to each thread, there is a control variable $r_i$ that contains the respective term of the thread $w_i$. In order for a thread $w_i$ to execute for some iterations, it needs to reserve (or lock) two control variables: $r_i$ and $r_{i-1}$ ($r_i$ and $r_k$ in case $w_i$ is the 1st thread worker – i = 0).

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots = \frac{\pi}{4}.$$

Assuming that Pi will be calculated with reasonably high precision, we want this algorithm to have very large n, with value over 100,000,000. A thread $t_i$ is then responsible for calculating a term $t_i$ which corresponds to a sum of n/k elements out of the whole n. This calculation should occur within a critical region, with mutual exclusion guaranteed through the reservation of the two control variables.
As a result, since you have to control the execution of the threads using restrictive access to variables/objects, Java **synchronized** directive are suggested when reserving/acquiring/locking each of the variables. Also, beore reserving/locking each variable and after releasing each of them, the following *bogus* **doAction** Java method should be called.

```
                    private void doAction() {
```

```
    calculate( (int)(Math.random() * 4 + 36) );
}

private static long calculate(int n) {
    if (n <= 1) return n;
    else return calculate(n-1) + calculate(n-2);
}
```

## Submission Material

The submission for this assignment will consist of two parts:

1. The Java code of your implementation, well divided and commented. The code should compile and should run properly, as described in the Assignment. Please provide instructions on how to compile your code in your description document. As recommended, make your project **compilible** and **runnable** from command line for marking purposes, defining all setup steps, such as classpath definition, to make your code run in any “foreign” environment.
2. A description document that explains your implementation decisions, justifications, issues present in the code, and challenges. The comments you added in your code should serve as a good start point to generate this document. You should describe how to compile and run your code, as part of your explanation. Word processing (MS Word) or PDF are acceptable file formats for this document.

You must guarantee that your code is legible, clear, and succinct. Keep in mind that any questionable implementation decision or copy from any source might have a negative effect on your mark.
If you still have any questions regarding which file types are acceptable, please inquire prior to submission. Note that it is not the fault of the markers if they are unable to mark your work due to submitting an unreasonable or uncommon file format.

## Marking Scheme

The work produced in this project step will not receive any mark but it will count later when grading the performance of your project in the end of the course. Thus, keep in mind the following marking scheme.
Marks will be awarded for completeness and demonstration of understanding of the material. It is important that you fully show your knowledge when providing solutions in a concise manner. Quality and conciseness of solutions are considered when awarding marks. Every code added to the originals should be well commented and explicitly indicated in the Java files; lack of clarity may lead you to loose marks, so keep it simple and clear.

## Submission

The submission is expected to contain two parts: the proper Java code well commented and a word processed document. The document can be in either DOC or PDF format; it should be single column, at least single spaced, and at least in font 11. All content files should be organized and put together in a ZIP for the submission through Sakai.

## Late Submissions Policy

A penalty of 25% will be applied on late submissions. Late submissions are accepted until the Late Submission Date, three days after the Project Step 1 Due Date. No excuses are accepted for missing these deadlines.

## Plagiarism

Students are expected respect academic integrity and deliver evaluation materials that are only produced by themselves. Any copy of content, text or code, from other students, books, web, or any other source is not tolerated. If there is any indication that an activity contains any part copied from any source, a case will be open and brought to a plagiarism committee's attention. In case plagiarism is determined, the activity will be cancelled, and the author(s) will be subject to the university regulations. For further information on this sensitive subject, please refer to the document below:

https://brocku.ca/node/10909