# Pi Calculation

<http://stackoverflow.com/questions/41094056/how-to-write-pi-calculation-program-in-java-using-multi-thread>

Bài giải 1:

**package** stackoverflow.com.pi;

**public** **class** Excute {

**public** **static** **void** main(String[] args) **throws** InterruptedException {

**int** threadCount = 4;

**int** N = 100;

Pithread[] threads = **new** Pithread[threadCount];

**for** (**int** i = 0; i < threadCount; i++) {

threads[i] = **new** Pithread(threadCount, i, N);

threads[i].start();

}

**for** (**int** i = 0; i < threadCount; i++) {

threads[i].join();

}

**double** pi = 0;

**for** (**int** i = 0; i < threadCount; i++) {

pi += threads[i].getSum();

}

System.***out***.print("PI/4 = " + pi);

}

**static** **class** Pithread **extends** Thread {

**private** **final** **int** threadCount;

**private** **final** **int** threadRemainder;

**private** **final** **int** N;

**private** **double** sum = 0;

**public** Pithread(**int** threadCount, **int** threadRemainder, **int** n) {

**this**.threadCount = threadCount;

**this**.threadRemainder = threadRemainder;

N = n;

}

@Override

**public** **void** run() {

System.***out***.println(threadRemainder);

**for** (**int** i = 0; i < N; i++) {

sum += Math.*pow*(-1, i) / (2\*i + 1);

}

}

**public** **double** getSum(){

**return** sum;

}

}

}

Chia ra 4 thread: thread 1 chạy 1,5,9,…

Thread 2 chạy 2, 6, 10, …

Thread 3 chạy 3, 7, 11, …

Thread 4 chạy 4, 8, 12, …

Bài giải 2:

**package** stackoverflow.com.pi;

**import** java.util.concurrent.atomic.AtomicLong;

**public** **class** Excute {

**public** **static** **volatile** **boolean** *running* = **true**;

**public** **static** **void** main(String[] args) **throws** InterruptedException {

**int** threadCount = 4;

**long** timeoutMs = 5\_000;

**final** AtomicLong counter = **new** AtomicLong(0);

Pithread[] threads = **new** Pithread[threadCount];

**for** (**int** i = 0; i < threadCount; i++) {

threads[i] = **new** Pithread(counter);

threads[i].start();

}

Thread.*sleep*(timeoutMs);

*running* = **false**;

**for** (**int** i = 0; i < threadCount; i++) {

threads[i].join();

}

**double** sum = 0;

**for** (**int** i = 0; i < threadCount; i++) {

sum += threads[i].getSum();

}

System.***out***.print("counter = " + counter.get());

System.***out***.print("PI = " + 4\*sum);

}

**static** **class** Pithread **extends** Thread {

**private** AtomicLong counter;

**private** **double** sum = 0;

**public** Pithread(AtomicLong counter) {

**this**.counter = counter;

}

@Override

**public** **void** run() {

**long** i;

**while** (*running* && isValidCounter(i = counter.getAndAdd(1))) {

sum += Math.*pow*(-1, i) / (2 \* i + 1);

}

}

**private** **boolean** isValidCounter(**long** value) {

**return** value >= 0 && value < Long.***MAX\_VALUE***;

}

**public** **double** getSum() {

**return** sum;

}

}

}

Cho chương trình chạy vô tận, sau khi 5s sẽ dừng việc tính toán và xuất ra màn hình

# 2. Các thuật ngữ:

## start:

bắt đầu cho 1 thread.

## Join:

chờ cho thread đó chết, nghĩa là thread đó chạy xong rồi.

## Volatile:

Biến volatile trong Java có tác dụng thông báo sự thay đổi giá trị của biến tới các thread khác nhau nếu biến này đang được sử dụng trong nhiều thread.

## AtomicLong

Na ná như biến volatile, nhưng sử dụng bên trong class thread luôn, nhiều thread chỉ sử dụng chung 1 AtomicLong có cùng name (hình như vậy).

# 3. Chạy 2 thread có hàm sleep

**package** stackoverflow.com.pi;

**class** ThreadDemo **extends** Thread {

**private** Thread t;

**private** String threadName;

ThreadDemo(String name) {

threadName = name;

System.***out***.println("Creating " + threadName);

}

**public** **void** run() {

System.***out***.println("Running " + threadName);

**try** {

**for** (**int** i = 4; i > 0; i--) {

System.***out***.println("Thread: " + threadName + ", " + i);

// Let the thread sleep for a while.

Thread.*sleep*(50);

}

} **catch** (InterruptedException e) {

System.***out***.println("Thread " + threadName + " interrupted.");

}

System.***out***.println("Thread " + threadName + " exiting.");

}

**public** **void** start() {

System.***out***.println("Starting " + threadName);

**if** (t == **null**) {

t = **new** Thread(**this**, threadName);

t.start();

}

}

**public** **static** **void** main(String[] args) {

ThreadDemo T1 = **new** ThreadDemo("Thread-1");

T1.start();

ThreadDemo T2 = **new** ThreadDemo("Thread-2");

T2.start();

}

}