1. <https://java2blog.com/print-even-odd-numbers-threads-java/>

Print Odd Even Using Threads

Solution 1 : Using wait and notify

We will use wait and notify to solve how to print even and odd numbers using threads in java.

* Use a variable called boolean [**odd**](https://java2blog.com/even-odd-program-java/). If you want to print odd number, it’s value should be true and vice versa for even number.
* Create two methods(printOdd and printEven), one will print odd numbers and other will print even numbers.
* Create two threads, one for odd and one for even.
* t1 will call  printEven method and t2 will call printOdd method simultaneously.
* If boolean odd is true in printEven method, t1 will wait.
* If boolean odd is false in printOdd method, t2 will wait.

**package** org.arpit.java2blog;

**public** **class** OddEvenPrintMain {

**boolean** odd;

**int** count = 1;

**int** MAX = 20;

**public** **void** printOdd() {

**synchronized** (**this**) {

**while** (count < MAX) {

// System.out.println("Checking odd loop");

**while** (!odd) {

**try** {

// System.out.println("Odd waiting : " + count);

wait();

// System.out.println("Notified odd :" + count);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

System.***out***.println("Odd Thread :" + count);

count++;

odd = **false**;

notify();

}

}

}

**public** **void** printEven() {

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException e1) {

e1.printStackTrace();

}

**synchronized** (**this**) {

**while** (count < MAX) {

// System.out.println("Checking even loop");

**while** (odd) {

**try** {

// System.out.println("Even waiting: " + count);

wait();

// System.out.println("Notified even:" + count);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

System.***out***.println("Even thread :" + count);

count++;

odd = **true**;

notify();

}

}

}

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

OddEvenPrintMain oep = **new** OddEvenPrintMain();

oep.odd = **true**;

Thread t1 = **new** Thread(**new** Runnable() {

@Override

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

oep.printEven();

}

});

Thread t2 = **new** Thread(**new** Runnable() {

@Override

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

oep.printOdd();

}

});

t1.start();

t2.start();

**try** {

t1.join();

t2.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

Solution 2 : Using remainder

You can use concept of remainder here.

* If number%2==1 then Odd will print the number and increment it else will go in the wait state.
* If number%2==0 then Even will print the number and increment it else will go in the wait state.

**package** org.arpit.java2blog;

**public** **class** OddEvenRunnable **implements** Runnable{

**public** **int** PRINT\_NUMBERS\_UPTO=10;

**static** **int** *number*=1;

**int** remainder;

**static** Object *lock*=**new** Object();

OddEvenRunnable(**int** remainder)

{

**this**.remainder=remainder;

}

@Override

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

**while** (*number* < PRINT\_NUMBERS\_UPTO) {

**synchronized** (*lock*) {

**while** (*number* % 2 != remainder) { // wait for numbers other than remainder

**try** {

*lock*.wait();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

System.***out***.println(Thread.*currentThread*().getName() + " " + *number*);

*number*++;

*lock*.notifyAll();

}

}

}

}

**package** org.arpit.java2blog;

**public** **class** PrintOddEvenMain {

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

OddEvenRunnable oddRunnable = **new** OddEvenRunnable(1);

OddEvenRunnable evenRunnable = **new** OddEvenRunnable(0);

Thread t1 = **new** Thread(oddRunnable, "Odd");

Thread t2 = **new** Thread(evenRunnable, "Even");

t1.start();

t2.start();

}

}