[Building H2O](https://leetcode.com/problems/building-h2o)



class H2O {

private final CyclicBarrier barrier = new CyclicBarrier(3);

private final Semaphore semaphoreForHydrogen = new Semaphore(2);

public H2O() {

}

public void hydrogen(Runnable releaseHydrogen) throws InterruptedException {

semaphoreForHydrogen.acquire();

// releaseHydrogen.run() outputs "H". Do not change or remove this line.

releaseHydrogen.run();

await();

semaphoreForHydrogen.release();

}

public synchronized void oxygen(Runnable releaseOxygen) throws InterruptedException {

// releaseOxygen.run() outputs "O". Do not change or remove this line.

releaseOxygen.run();

await();

}

private void await() {

try {

barrier.await();

} catch (InterruptedException | BrokenBarrierException e) {

throw new RuntimeException(e);

}

}

}



class H2O {

private final CyclicBarrier barrier = new CyclicBarrier(3);

private final Semaphore semaphoreForHydrogen = new Semaphore(2);

private final Semaphore semaphoreOxygen = new Semaphore(1);

public H2O() {

}

public void hydrogen(Runnable releaseHydrogen) throws InterruptedException {

semaphoreForHydrogen.acquire();

// releaseHydrogen.run() outputs "H". Do not change or remove this line.

releaseHydrogen.run();

await();

semaphoreForHydrogen.release();

}

public void oxygen(Runnable releaseOxygen) throws InterruptedException {

semaphoreOxygen.acquire();

// releaseOxygen.run() outputs "O". Do not change or remove this line.

releaseOxygen.run();

await();

semaphoreOxygen.release();

}

private void await() {

try {

barrier.await();

} catch (InterruptedException | BrokenBarrierException e) {

throw new RuntimeException(e);

}

}

}

[Print Zero Even Odd](https://leetcode.com/problems/print-zero-even-odd)



class ZeroEvenOdd {

private int n;

private final CyclicBarrier barrierZero = new CyclicBarrier(2);

private final CyclicBarrier barrierEven = new CyclicBarrier(2);

private final CyclicBarrier barrierOdd = new CyclicBarrier(2);

public ZeroEvenOdd(int n) {

this.n = n;

}

// printNumber.accept(x) outputs "x", where x is an integer.

public void zero(IntConsumer printNumber) throws InterruptedException {

for (int i=1; i<=n; i++) {

printNumber.accept(0);

if (i % 2 == 0) {

await(barrierEven);

} else {

await(barrierOdd);

}

await(barrierZero);

}

}

public void even(IntConsumer printNumber) throws InterruptedException {

for (int i=2; i<=n; i+=2) {

await(barrierEven);

printNumber.accept(i);

await(barrierZero);

}

}

public void odd(IntConsumer printNumber) throws InterruptedException {

for (int i=1; i<=n; i+=2) {

await(barrierOdd);

printNumber.accept(i);

await(barrierZero);

}

}

private static void await(CyclicBarrier barrier) {

try {

barrier.await();

} catch (InterruptedException | BrokenBarrierException e) {

throw new RuntimeException(e);

}

}

}

[Print FooBar Alternately](https://leetcode.com/problems/print-foobar-alternately)

**1**.

class FooBar {

private int n;

private final CyclicBarrier barrier1 = new CyclicBarrier(2);

private final CyclicBarrier barrier2 = new CyclicBarrier(2);

public FooBar(int n) {

this.n = n;

}

public void foo(Runnable printFoo) throws InterruptedException {

for (int i = 0; i < n; i++) {

// printFoo.run() outputs "foo". Do not change or remove this line.

printFoo.run();

await(barrier2);

await(barrier1);

}

}

public void bar(Runnable printBar) throws InterruptedException {

for (int i = 0; i < n; i++) {

await(barrier2);

// printBar.run() outputs "bar". Do not change or remove this line.

printBar.run();

await(barrier1);

}

}

private static void await(CyclicBarrier barrier) {

try {

barrier.await();

} catch (InterruptedException | BrokenBarrierException e) {

throw new RuntimeException(e);

}

}

}



class FooBar {

private int n;

private CountDownLatch latch1 = new CountDownLatch(1);

private CountDownLatch latch2 = new CountDownLatch(1);

public FooBar(int n) {

this.n = n;

}

public void foo(Runnable printFoo) throws InterruptedException {

for (int i = 0; i < n; i++) {

// printFoo.run() outputs "foo". Do not change or remove this line.

printFoo.run();

latch2.countDown();

latch1.await();

latch1 = new CountDownLatch(1);

}

}

public void bar(Runnable printBar) throws InterruptedException {

for (int i = 0; i < n; i++) {

latch2.await();

latch2 = new CountDownLatch(1);

// printBar.run() outputs "bar". Do not change or remove this line.

printBar.run();

latch1.countDown();

}

}

}

[Print in Order](https://leetcode.com/problems/print-in-order)



class Foo {

private final Phaser phaser1 = new Phaser(1);

private final Phaser phaser2 = new Phaser(1);

public Foo() {

}

public void first(Runnable printFirst) throws InterruptedException {

// printFirst.run() outputs "first". Do not change or remove this line.

printFirst.run();

phaser1.arrive();

}

public void second(Runnable printSecond) throws InterruptedException {

phaser1.awaitAdvance(0);

// printSecond.run() outputs "second". Do not change or remove this line.

printSecond.run();

phaser2.arrive();

}

public void third(Runnable printThird) throws InterruptedException {

phaser2.awaitAdvance(0);

// printThird.run() outputs "third". Do not change or remove this line.

printThird.run();

}

}



class Foo {

private final CyclicBarrier barrier1 = new CyclicBarrier(2);

private final CyclicBarrier barrier2 = new CyclicBarrier(2);

public Foo() {

}

public void first(Runnable printFirst) throws InterruptedException {

// printFirst.run() outputs "first". Do not change or remove this line.

printFirst.run();

await(barrier1);

}

public void second(Runnable printSecond) throws InterruptedException {

await(barrier1);

// printSecond.run() outputs "second". Do not change or remove this line.

printSecond.run();

await(barrier2);

}

public void third(Runnable printThird) throws InterruptedException {

await(barrier2);

// printThird.run() outputs "third". Do not change or remove this line.

printThird.run();

}

private static void await(CyclicBarrier barrier) {

try {

barrier.await();

} catch (InterruptedException e) {

throw new RuntimeException(e);

} catch (BrokenBarrierException e) {

throw new RuntimeException(e);

}

}

}



class Foo {

private final CyclicBarrier barrier1 = new CyclicBarrier(2);

private final CyclicBarrier barrier2 = new CyclicBarrier(2);

public Foo() {

}

public void first(Runnable printFirst) throws InterruptedException {

// printFirst.run() outputs "first". Do not change or remove this line.

printFirst.run();

await(barrier1);

}

public void second(Runnable printSecond) throws InterruptedException {

await(barrier1);

// printSecond.run() outputs "second". Do not change or remove this line.

printSecond.run();

await(barrier2);

}

public void third(Runnable printThird) throws InterruptedException {

await(barrier2);

// printThird.run() outputs "third". Do not change or remove this line.

printThird.run();

}

private static void await(CyclicBarrier barrier) {

try {

barrier.await();

} catch (InterruptedException e) {

throw new RuntimeException(e);

} catch (BrokenBarrierException e) {

throw new RuntimeException(e);

}

}

}



class Foo {

private final CountDownLatch latch1;

private final CountDownLatch latch2;

public Foo() {

this.latch1 = new CountDownLatch(1);

this.latch2 = new CountDownLatch(1);

}

public void first(Runnable printFirst) throws InterruptedException {

// printFirst.run() outputs "first". Do not change or remove this line.

printFirst.run();

latch1.countDown();

}

public void second(Runnable printSecond) throws InterruptedException {

latch1.await();

// printSecond.run() outputs "second". Do not change or remove this line.

printSecond.run();

latch2.countDown();

}

public void third(Runnable printThird) throws InterruptedException {

latch2.await();

// printThird.run() outputs "third". Do not change or remove this line.

printThird.run();

}

}