

2021/12/10 上午2:41	AccountWithoutSync.java	AccountWithoutSync.java	2021/12/10 上午2:41
1 package ex8;			60 }
2			61 }
3			62 }
4 import java.util.concurrent.*;		balance = newBalance;	63 }
5			
6 public class AccountWithoutSync {			
7			
8 private Account account = new Account();			
9			
10 public static void main(String[] args) {			
11 AccountWithoutSync aws = new AccountWithoutSync();			
12 aws.start();			
13 }			
14			
15 public void start() {			
16 ExecutorService executor = Executors.newCachedThreadPool();			
17			
18 // Create and launch 100 threads			
19 for (int i = 0; i < 100; i++) {			
20 executor.execute(new AddAPennyThread());			
21 }			
22			
23 executor.shutdown();			
24			
25 // Wait until all tasks are finished			
26 while (!executor.isTerminated()) {			
27 }			
28			
29 System.out.println("What is the balance ? " + account.getBalance());			
30 }			
31			
32 // An inner class of task for adding a penny to the account			
33 private class AddAPennyThread implements Runnable {			
34			
35 public void run() {			
36 account.deposit(1);			
37 }			
38 }			
39			
40 // An inner class for account			
41 private class Account {			
42			
43 private int balance = 0;			
44			
45 public int getBalance() {			
46 return balance;			
47 }			
48			
49 public void deposit(int amount) {			
50 int newBalance = balance + amount;			
51			
52 // This delay is deliberately added to magnify the			
53 // data-corruption problem and make it easy to see.			
54 try {			
55 Thread.sleep(1);			
56 } catch (InterruptedException ex) {			
57 // do nothing			
58 }			
59			
localhost:4649/?mode=click	1/2	localhost:4649/?mode=click	2/2

2021/12/10 上午2:42

AccountWithSync.java

1 package ex8;

2

3

4 import java.util.concurrent.*;

5

6 public class AccountWithSync {

7

8 private Account account = new Account();

9

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

11 AccountWithSync awsul = new AccountWithSync();

12 awsul.start();

13 }

14

15 public void start() {

16 ExecutorService executor = Executors.newCachedThreadPool();

17

18 // Create and launch 100 threads

19 for (int i = 0; i < 100; i++) {

20 executor.execute(new AddAPennyThread());

21 }

22

23 executor.shutdown();

24

25 // Wait until all tasks are finished

26 while (!executor.isTerminated()) {

27 }

28

29 System.out.println("What is the balance ? " + account.getBalance());

30 }

31

32 // An inner class of task for adding a penny to the account

33 private class AddAPennyThread implements Runnable {

34

35 public void run() {

36 account.deposit(1);

37 }

38 }

39

40 // An inner class for account

41 private class Account {

42

43 private int balance = 0;

44

45 public int getBalance() {

46 return balance;

47 }

48

49 public synchronized void deposit(int amount) {

50

51 int newBalance = balance + amount;

52

53 // This delay is deliberately added to magnify the

54 // data-corruption problem and make it easy to see.

55 try {

56 Thread.sleep(1);

57 } catch (InterruptedException ex) {

58 // do nothing

59 }

2021/12/10 上午2:42

AccountWithSync.java

60

61 balance = newBalance;

62 }

63 }

64 }

65

localhost:4649/?mode=click

1/2

localhost:4649/?mode=click

2/2

```
1 package ex8;
2
3 import java.util.concurrent.*;
4 import java.util.concurrent.locks.*;
5
6 public class AccountWithSyncUsingLock {
7
8     private Account account = new Account();
9
10    public static void main(String[] args) {
11        AccountWithSyncUsingLock awsul = new AccountWithSyncUsingLock();
12        awsul.start();
13    }
14
15    public void start() {
16        ExecutorService executor = Executors.newCachedThreadPool();
17
18        // Create and launch 100 threads
19        for (int i = 0; i < 100; i++) {
20            executor.execute(new AddAPennyThread());
21        }
22
23        executor.shutdown();
24
25        // Wait until all tasks are finished
26        while (!executor.isTerminated()) {
27        }
28
29        System.out.println("What is the balance ? " + account.getBalance());
30    }
31
32    // An inner class of task for adding a penny to the account
33    private class AddAPennyThread implements Runnable {
34
35        public void run() {
36            account.deposit(1);
37        }
38    }
39
40    // An inner class for account
41    private class Account {
42
43        private ReentrantLock lock = new ReentrantLock(); // Create a lock
44        private int balance = 0;
45
46        public int getBalance() {
47            return balance;
48        }
49
50        public void deposit(int amount) {
51            lock.lock(); // Acquire the lock
52
53            try {
54                int newBalance = balance + amount;
55
56                // This delay is deliberately added to magnify the
57                // data-corruption problem and make it easy to see.
58                Thread.sleep(5);
59            }
```

```
60
61         balance = newBalance;
62     } catch (InterruptedException ex) {
63     } finally {
64         lock.unlock(); // Release the lock
65     }
66 }
67 }
68
```

```
1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5 package ex8;
6
7 /**
8  *
9  * @author vanting
10 */
11 class Counter {
12
13     private int c = 0;
14
15     public void increment() {
16         c++;
17     }
18
19     public void decrement() {
20         c--;
21     }
22
23     public int value() {
24         return c;
25     }
26
27 }
```

```
1 package ex8;
2
3 import java.util.concurrent.*;
4
5 public class ExecutorDemo implements Runnable {
6
7     private char c;
8
9     ExecutorDemo(char c) {
10         this.c = c;
11     }
12
13     public void run() {
14         for (int i = 0; i < 10; i++) {
15             System.out.print(c);
16             try {
17                 Thread.sleep(500); // half secs
18             } catch (InterruptedException ex) {
19                 // do nothing
20             }
21         }
22     }
23
24     public static void main(String[] args) {
25
26         // Create a fixed thread pool with maximum three threads
27         ExecutorService pool = Executors.newFixedThreadPool(3);
28
29         // Submit runnable tasks to the executor
30         pool.execute(new ExecutorDemo('a'));
31         pool.execute(new ExecutorDemo('b'));
32         pool.execute(new ExecutorDemo('c'));
33         pool.execute(new ExecutorDemo('d'));
34
35     }
36 }
37
```

2021/12/10 上午2:42

ForkJoinTest.java

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex8;
7
8 import java.util.ArrayList;
9 import java.util.List;
10 import java.util.concurrent.ForkJoinPool;
11 import java.util.concurrent.RecursiveTask;
12
13 /**
14  *
15  * @author wanting
16  */
17 public class ForkJoinTest {
18
19     public static void main(String[] args) {
20         ArrayList<Double> list = new ArrayList<>(1000);
21         for (int i = 0; i < 1000; i++) {
22             list.add(Math.random() * 100.0);
23         }
24
25         ParallelSum task = new ParallelSum(list, 0, list.size());
26
27         // Create a ForkJoinPool to run the task
28         ForkJoinPool pool = new ForkJoinPool();
29
30         // Submit the task to the Fork/Join pool for execution
31         double sum = pool.invoke(task);
32
33         System.out.println("Sum is " + sum);
34     }
35 }
36
37 class ParallelSum extends RecursiveTask<Double> {
38
39     private static final int THRESHOLD = 100;
40     private List<Double> list;
41     private int start, end; // sum up elements from start to end-1
42
43     public ParallelSum(List<Double> a, int s, int e) {
44         list = a;
45         start = s;
46         end = e;
47     }
48
49     @Override
50     protected Double compute() {
51         double result = 0;
52
53         // Base case
54         if (end - start < THRESHOLD) {
55             for (int i = start; i < end; i++) {
56                 result += list.get(i);
57             }
58         } else {
59             // Divide into 2 sub-tasks
```

localhost:4649/?mode=click

1/2

2021/12/10 上午2:42

ForkJoinTest.java

```
60 int mid = (start + end) / 2;
61 ParallelSum task1 = new ParallelSum(list, start, mid);
62 ParallelSum task2 = new ParallelSum(list, mid, end);
63
64 task1.fork(); // Launch the subtasks
65 task2.fork();
66
67 // Wait for subtasks to finish and combine the result
68 result = task1.join()+ task2.join();
69
70 return result;
71 }
72 }
73
```

localhost:4649/?mode=click

2/2

```
1 package ex8;
2
3 /**
4  *
5  * @author vanting
6  */
7 public class HelloRunnable implements Runnable {
8
9     public void run() {
10         System.out.println("Hello from a thread!");
11     }
12
13     public static void main(String args[]) {
14         HelloRunnable task = new HelloRunnable();
15         Thread t = new Thread(task);
16         t.start();
17     }
18 }
19
```

```
1 package ex8;
2
3 /**
4  *
5  * @author vanting
6  */
7 public class HelloThread extends Thread {
8
9     @Override
10     public void run() {
11         System.out.println("Hello from a thread!");
12     }
13
14     public static void main(String args[]) {
15         Thread t = new HelloThread();
16         t.start();
17         //t.run();
18     }
19 }
20
```

```
1 package ex8;
2
3 /**
4  *
5  * @author vanting
6  */
7 public class InterruptDemo1 implements Runnable {
8
9     public void run() {
10         System.err.println("Running");
11         while(true) {
12             if(Thread.interrupted())
13                 break;
14         }
15         System.err.println("Stopped by Interruption.");
16     }
17
18     public static void main(String args[]) throws InterruptedException {
19         Thread t = new Thread(new InterruptDemo1());
20         t.start();
21
22         for(int i=10; i>0; i--) {
23             Thread.sleep(1000); // 1 sec
24             System.out.print(i + " ");
25         }
26         System.out.println("");
27         t.interrupt();
28     }
29 }
30
```

```
1 package ex8;
2
3 /**
4  *
5  * @author vanting
6  */
7 public class InterruptDemo2 implements Runnable {
8
9     public void run() {
10         System.err.println("Start to sleep.");
11         try {
12             Thread.sleep(1000); // 10 secs
13         } catch (InterruptedException ex) {
14             System.err.println("Woken up by Interruption.");
15         }
16     }
17
18     }
19
20     public static void main(String args[]) throws InterruptedException {
21         Thread t = new Thread(new InterruptDemo2());
22         t.start();
23
24         for(int i=5; i>0; i--) {
25             Thread.sleep(1000); // 1 sec
26             System.out.print(i + " ");
27         }
28         System.out.println("");
29         t.interrupt();
30     }
31 }
32
```

```
1 package ex8;
2
3 /**
4  *
5  * @author vanting
6  */
7 public class JoinDemo implements Runnable {
8
9     public void run() {
10         for (int i = 0; i < 1000; i++)
11             System.out.print("T");
12     }
13
14     public static void main(String args[]) throws InterruptedException {
15         Thread t = new Thread(new JoinDemo());
16         t.start();
17         t.join(); // wait for the finish of thread t
18
19         for (int i = 0; i < 1000; i++)
20             System.out.print("W");
21     }
22 }
23
```

```
1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5 package ex8;
6
7 /**
8  *
9  * @author vanting
10  */
11 public class MsLunch {
12
13     private long c1 = 0;
14     private long c2 = 0;
15     private Object lock1 = new Object();
16     private Object lock2 = new Object();
17
18     public void inc1() {
19         synchronized (lock1) {
20             c1++;
21         }
22     }
23
24     public void inc2() {
25         synchronized (lock2) {
26             c2++;
27         }
28     }
29 }
30
```



```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex8;
7
8 import java.util.concurrent.Callable;
9 import java.util.concurrent.ExecutionException;
10 import java.util.concurrent.ExecutorService;
11 import java.util.concurrent.Executors;
12 import java.util.concurrent.Future;
13
14 /**
15  *
16  * @author vanting
17  */
18 public class MultiThreadMaxFinder {
19
20     public static void main(String[] args) throws InterruptedException,
21     ExecutionException {
22
23         int[] array = new int[] {8, 3, 99, 23, 14, 50, 39};
24         System.out.println("The max is: " + max(array));
25     }
26
27     public static int max(int[] data) throws InterruptedException, ExecutionException
28     {
29         if (data.length == 0) {
30             throw new IllegalArgumentException();
31         }
32         if (data.length == 1) {
33             return data[0];
34         }
35
36         // Divide the array into 2 parts.
37         // Use a separate thread to find the max in each part.
38         int mid = data.length / 2;
39         FindMaxTask t1 = new FindMaxTask(data, 0, mid);
40         FindMaxTask t2 = new FindMaxTask(data, mid, data.length);
41
42         // Executors is a factory for creating Executor
43         // ExecutorService, and ScheduledExecutorService
44         ExecutorService s = Executors.newFixedThreadPool(2);
45
46         // Asynchronous tasks
47         Future<Integer> f1 = s.submit(t1);
48         Future<Integer> f2 = s.submit(t2);
49
50         // When f1.get() is called, the main thread is
51         // blocked and waits for f1 to finish.
52         // Similarly for f2.get().
53         int result = Math.max(f1.get(), f2.get());
54
55         s.shutdown(); // Terminate the threads
56         return result;
57     }
```

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex8;
7
8 import java.util.concurrent.locks.Condition;
9 import java.util.concurrent.locks.ReentrantLock;
10 import java.util.logging.Level;
11 import java.util.logging.Logger;
12
13 /**
14  *
15  * @author vanting
16  */
17
18 public class ProducerConsumer {
19
20     public static Account num = new Account(0);
21
22     public static void main(String[] args) {
23
24         Thread add = new Thread(new MyPCThread(true));
25         Thread minus = new Thread(new MyPCThread(false));
26
27         add.start();
28         minus.start();
29
30     }
31
32 }
33
34 class Account {
35
36     private int balance;
37
38     private ReentrantLock lock = new ReentrantLock(); // Create a new lock
39     private Condition newDeposit = lock.newCondition(); // Create a condition
40
41     public Account(int v) {
42         this.balance = v;
43     }
44
45     public int get() {
46         return balance;
47     }
48
49     public int incrementRandomAndGet() {
50         lock.lock();
51         int amount = (int) (Math.random() * 50) + 1;
52         balance += amount;
53         System.out.println("Balance: " + balance + " (+ " + amount + ")");
54         try {
55             newDeposit.signalAll();
56         } finally {
57             lock.unlock();
58         }
59         return balance;
60     }
61
62     public int decrementRandomAndGet() {
63         lock.lock();
64         int amount = (int) (Math.random() * 10) + 1;
65         try {
66             while (balance < amount) {
67                 System.out.println("Not enough balance...");
68                 newDeposit.await();
69                 System.out.println("OK, enough.");
70             }
71             balance -= amount;
72             System.out.println("Balance: " + balance + " (- " + amount + ")");
73         } catch (InterruptedException ex) {
74             Logger.getLogger(Account.class.getName()).log(Level.SEVERE, null, ex);
75         } finally {
76             lock.unlock();
77         }
78         return balance;
79     }
80 }
81
82 class MyPCThread implements Runnable {
83
84     private final boolean isAdd;
85
86     public MyPCThread(boolean isAdd) {
87         this.isAdd = isAdd;
88     }
89
90     @Override
91     public void run() {
92         if (isAdd) {
93             while (true) {
94                 ProducerConsumer.num.incrementRandomAndGet();
95                 try {
96                     Thread.sleep(3000);
97                 } catch (InterruptedException ex) {
98                     Logger.getLogger(MyPCThread.class.getName()).log(Level.SEVERE,
99                         null, ex);
100                 }
101             }
102         } else {
103             while (true) {
104                 try {
105                     // this thread run 10x faster
106                     Thread.sleep(300);
107                 } catch (InterruptedException ex) {
108                     Logger.getLogger(MyPCThread.class.getName()).log(Level.SEVERE,
109                         null, ex);
110                 }
111                 ProducerConsumer.num.decrementRandomAndGet();
112             }
113         }
114     }
115 }
```

```
60 }
61
62 public int decrementRandomAndGet() {
63     lock.lock();
64     int amount = (int) (Math.random() * 10) + 1;
65     try {
66         while (balance < amount) {
67             System.out.println("Not enough balance...");
68             newDeposit.await();
69             System.out.println("OK, enough.");
70         }
71         balance -= amount;
72         System.out.println("Balance: " + balance + " (- " + amount + ")");
73     } catch (InterruptedException ex) {
74         Logger.getLogger(Account.class.getName()).log(Level.SEVERE, null, ex);
75     } finally {
76         lock.unlock();
77     }
78     return balance;
79 }
80 }
81
82 class MyPCThread implements Runnable {
83
84     private final boolean isAdd;
85
86     public MyPCThread(boolean isAdd) {
87         this.isAdd = isAdd;
88     }
89
90     @Override
91     public void run() {
92         if (isAdd) {
93             while (true) {
94                 ProducerConsumer.num.incrementRandomAndGet();
95                 try {
96                     Thread.sleep(3000);
97                 } catch (InterruptedException ex) {
98                     Logger.getLogger(MyPCThread.class.getName()).log(Level.SEVERE,
99                         null, ex);
100                 }
101             }
102         } else {
103             while (true) {
104                 try {
105                     // this thread run 10x faster
106                     Thread.sleep(300);
107                 } catch (InterruptedException ex) {
108                     Logger.getLogger(MyPCThread.class.getName()).log(Level.SEVERE,
109                         null, ex);
110                 }
111                 ProducerConsumer.num.decrementRandomAndGet();
112             }
113         }
114     }
115 }
```

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex8;
7
```

```
8 import java.util.concurrent.atomic.AtomicInteger;
```

```
9 /**
10  *
11  * @author vanting
12  */
13 class NonAtomicNum {
```

```
14     // regular int variables are non-atomic
15     private int value;
```

```
16
17
18
19     public NonAtomicNum(int v) {
20         this.value = v;
21     }
```

```
22
23     public int get() {
24         return value;
25     }
```

```
26
27     public int incrementAndGet() {
28         return ++value;
29     }
```

```
30
31     public int decrementAndGet() {
32         return --value;
33     }
```

```
34
35     /*
36     public synchronized int incrementAndGet() {
37         return ++value;
38     }
```

```
39
40     public synchronized int decrementAndGet() {
41         return --value;
42     }
43     */
```

```
44 }
45
46 class MyRaceThread implements Runnable {
```

```
47     private final boolean isAdd;
```

```
48
49     public MyRaceThread(boolean isAdd) {
50         this.isAdd = isAdd;
51     }
```

```
52
53     @Override
54     public void run() {
55         if (isAdd) {
56             for (int i = 0; i < 10000; i++) {
57                 RaceCondition.num.incrementAndGet();
58             }
59         }
```

```
60     } else {
61         for (int i = 0; i < 10000; i++) {
62             RaceCondition.num.decrementAndGet();
63         }
64     }
65 }
66 }
67
68 public class RaceCondition {
69
70     public static NonAtomicNum num = new NonAtomicNum(0);
71     //public static AtomicInteger num = new AtomicInteger(0);
72
73     public static void main(String[] args) {
74
75         Thread add = new Thread(new MyRaceThread(true));
76         Thread minus = new Thread(new MyRaceThread(false));
77
78         add.start();
79         minus.start();
80
81         /*
82         * wait both threads to finish
83         */
84         try {
85             add.join();
86             minus.join();
87         } catch (InterruptedException e) {
88             e.printStackTrace();
89         }
90
91         // the expected output should be zero
92         System.out.println(num.get());
93     }
94 }
95 }
96 }
```

```
1 package ex8;
2 /**
3  *
4  * @author vanting
5  */
6 public class SleepDemo {
7
8     public static void main(String args[]) throws InterruptedException {
9         System.out.print("Thinking");
10        for(int i=0; i<20; i++) {
11            Thread.sleep(1000); //Pause for 1 seconds
12            System.out.print(".");
13        }
14        System.out.println("Done!");
15    }
16 }
17
18
19
```

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex8;
7
8 /**
9  *
10  * @author vanting
11  */
12 public class StringBufferVsStringBuilder {
13
14     public static void main(String[] args) {
15         int N = 77777777;
16         long t;
17
18         {
19             // thread safe
20             StringBuffer sb = new StringBuffer();
21             t = System.currentTimeMillis();
22             for (int i = N; i-- > 0;) {
23                 sb.append("");
24             }
25             System.out.println(System.currentTimeMillis() - t);
26         }
27
28         {
29             // not thread safe
30             StringBuilder sb = new StringBuilder();
31             t = System.currentTimeMillis();
32             for (int i = N; i-- > 0;) {
33                 sb.append("");
34             }
35             System.out.println(System.currentTimeMillis() - t);
36         }
37     }
38 }
39
```

```
1 package ex8;
2 import java.util.concurrent.*;
3 import java.util.concurrent.locks.*;
4
5 public class ThreadCooperation {
6
7     private Account account = new Account();
8
9     public static void main(String[] args) {
10         ThreadCooperation tc = new ThreadCooperation();
11         tc.start();
12     }
13
14     public void start() {
15         // Create a thread pool with two threads
16         ExecutorService executor = Executors.newFixedThreadPool(2);
17         executor.execute(new DepositTask());
18         executor.execute(new WithdrawTask());
19         executor.shutdown();
20
21         System.out.println("Thread 1\t\tThread 2\t\tBalance");
22     }
23
24     // A task for adding an amount to the account
25     public class DepositTask implements Runnable {
26
27         public void run() {
28             try { // Purposely delay it to let the withdraw method proceed
29                 while (true) {
30                     account.deposit((int) (Math.random() * 10) + 1);
31                     Thread.sleep(1000);
32                 }
33             } catch (InterruptedException ex) {
34                 ex.printStackTrace();
35             }
36         }
37     }
38
39     // A task for subtracting an amount from the account
40     public class WithdrawTask implements Runnable {
41
42         public void run() {
43             while (true) {
44                 account.withdraw((int) (Math.random() * 10) + 1);
45             }
46         }
47     }
48
49     // An inner class for account
50     private class Account {
51         // Create a new lock
52         private ReentrantLock lock = new ReentrantLock(); // Create a
53         condition
54         private Condition newDeposit = lock.newCondition();
55         private int balance = 0;
56
57         public int getBalance() {
58             return balance;
59         }
60     }
61 }
```

```
59     public void withdraw(int amount) {
60         lock.lock(); // Acquire the lock
61         try {
62             while (balance < amount) {
63                 newDeposit.await();
64             }
65             balance -= amount;
66             System.out.println("\t\t\tWithdraw " + amount + "\t\t\t" +
67                 getBalance());
68             } catch (InterruptedException ex) {
69                 ex.printStackTrace();
70             } finally {
71                 lock.unlock(); // Release the lock
72             }
73         }
74
75         public void deposit(int amount) {
76             lock.lock(); // Acquire the lock
77             try {
78                 balance += amount;
79                 System.out.println("Deposit " + amount + "\t\t\t\t\t" +
80                     getBalance());
81             } // Signal thread waiting on the condition
82             newDeposit.signalAll();
83             } finally {
84                 lock.unlock(); // Release the lock
85             }
86         }
87     }
88 }
```

```
1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5  package ex8;
6
7  /**
8   *
9   * @author vanting
10  */
11  public class ThreadSwitching {
12
13      public static void main(String[] args) {
14          MySwitchingThread t1 = new MySwitchingThread(0, 3, 300);
15          MySwitchingThread t2 = new MySwitchingThread(1, 3, 300);
16          MySwitchingThread t3 = new MySwitchingThread(2, 3, 300);
17
18          t1.start();
19          t2.start();
20          t3.start();
21      }
22  }
23
24  class MySwitchingThread extends Thread {
25
26      private int startIdx, nThreads, maxIdx;
27
28      public MySwitchingThread(int s, int n, int m) {
29          this.startIdx = s;
30          this.nThreads = n;
31          this.maxIdx = m;
32      }
33
34      @Override
35      public void run() {
36          for (int i = this.startIdx; i < this.maxIdx; i += this.nThreads) {
37              System.out.println("[ID " + this.getId() + "] " + i);
38          }
39      }
40  }
41
```