



Start streaming in minutes
**A ready-to-use video toolkit
engineered for growth**

Get Started



< By Developers/For Developers >

VIBLO PARTNER



Do Hung @hungtdo

[Follow](#)

★ 589

• 36

✍ 17

Published Jul 31st, 2015 2:02 AM

① 5.6K

3

4

TABLE OF CONTENTS

H
T
C

▲
+7

Java Synchronized Blocks

Java

...



Khi chúng ta bắt đầu 2 hay nhiều Thread trong cùng 1 chương trình, có thể xảy ra tình huống nhiều Thread cố gắng truy cập vào cùng 1 file hay 1 đối tượng nhất định gây ra tình trạng xung đột dữ liệu, mất dữ liệu.

Ví dụ, nếu nhiều thread đồng thời chỉ vào cùng một tệp tin, vì tệp tin có thể bị hỏng dữ liệu và

cùng lúc với mục đích khác nhau có thể gây ra lỗi.

▲
+7

Vì vậy, sinh ra nhu cầu để đồng bộ hóa các hành động của nhiều thread khác nhau (multi-thread) và chắc chắn rằng chỉ có một thread có thể truy cập các tài nguyên tại một thời điểm nào đó. Điều này được thực hiện bằng cách sử dụng một khái niệm gọi là monitors (giám sát).



Ngôn ngữ lập trình Java cung cấp một cách rất tiện lợi cho việc tạo ra Thread và đồng bộ hóa các nhiệm vụ bằng cách sử dụng các khối **synchronized** (đồng bộ). Bạn giữ các nguồn tài nguyên được chia sẻ trong khối này.

Một khối Synchronized block đánh dấu một phương thức hay một khối mã là được đồng bộ. Sử dụng khối đồng bộ trong Java có thể tránh xảy ra xung đột.

Chạy nhiều thread bên trong cùng một ứng dụng không tự gây ra vấn đề. Các vấn đề chỉ phát sinh khi nhiều Thread cùng truy cập vào các nguồn tài nguyên. Ví dụ cùng một bộ nhớ

(biến, mảng, hay đối tượng), các hệ thống (cơ sở dữ liệu, dịch vụ web, vv) hoặc các tập tin.

Trong thực tế, vấn đề chỉ phát sinh nếu một hoặc nhiều hơn các thread cùng ghi tới các tài nguyên này. Nó là an toàn để cho nhiều Thread cùng đọc cùng một tài nguyên, miễn là không thay đổi chúng.

VIBLO

Posts Questions Discussions

Search Viblo



Sign In/Sign up

```
synchronized(objectidentifier) {  
    // Access shared variables and other shared resources  
}
```

+7

Ở đây, **objectidentifier** là một tham chiếu đến một đối tượng và tuyên bố đại diện cho đồng bộ



Một khối đồng bộ trong Java đồng bộ được truy cập trên một số đối tượng. Tất cả các khối đồng bộ đồng bộ trên cùng một đối tượng chỉ có thể cùng một lúc có một thread thực thi bên trong chúng. Tất cả các thread khác cố gắng để nhập khối đồng bộ bị chặn cho đến khi thread bên trong khối đồng bộ thoát khỏi.



Các từ khóa synchronized có thể được sử dụng để đánh dấu 4 kiểu khác nhau của các khối:

1. Phương thức thông thường
2. Phương thức tĩnh (static method)
3. Khối code (Synchronized blocks) được đồng bộ bên trong phương thức
4. Khối code được đồng bộ bên trong phương thức tĩnh

Các khối này được đồng bộ trên các đối tượng khác nhau. Các loại hình khối đồng bộ bạn cần phụ thuộc vào tình hình cụ thể để sử dụng.

VIBLO

Posts Questions Discussions

Search Viblo



Sign In/Sign up

Đây là ví dụ dùng bộ phương thức:

```
public synchronized void add(int value){  
    this.count += value;  
}
```

+7

Chú ý việc sử dụng các từ khóa synchronized (đồng bộ) để khai báo phương thức. Điều này thông báo cho Java rằng phương thức này sẽ được đồng bộ.



Một phương thức được đồng bộ trong Java được đồng bộ trên thể hiện của đối tượng sở hữu

chúng. Vì thế mỗi thể hiện của đối tượng có các phương thức đồng bộ riêng và được đồng bộ riêng rẽ trên từng thể hiện (instance) của chúng. Chỉ có một thread có thể thực hiện bên trong một phương thức đồng bộ. Nếu có nhiều hơn 1 thể hiện (instance object) tồn tại, sau đó chỉ có 1 thread tại 1 thời điểm có thể thực thi bên trong phương thức được đồng bộ cho mỗi thể hiện. Vì vậy “một thread trên một thể hiện”.

2. Đồng bộ các phương thức tĩnh (Static)

Phương thức tĩnh được đánh dấu là đồng bộ có cú pháp cũng giống như phương thức thông thường bằng cách sử dụng từ khóa synchronized. Dưới đây là một ví dụ phương thức tĩnh được đồng bộ

The screenshot shows a Viblo post interface. At the top, there's a navigation bar with 'VIBLO' (highlighted), 'Posts', 'Questions', 'Discussions', a search bar ('Search Viblo'), and a sign-in button ('Sign In/Sign up'). Below the navigation is a post card. The post has a score of '+7' and a title: 'Cũng như vậy sử dụng synchronized ở đây tuyên bố phương thức này là đồng bộ.' Below the title is a note: 'Lưu ý: nếu bạn thực hiện đánh dấu phương thức tĩnh là đồng bộ, phương thức sẽ được khóa (lock) trên class không phải object. Tức là "tại 1 thời điểm chỉ 1 thread được chạy trên 1 class".' There are social sharing icons for Facebook and Twitter on the left. The code block contains Java code for a static synchronized method and two thread classes.

Cũng như vậy sử dụng synchronized ở đây tuyên bố phương thức này là đồng bộ.

Lưu ý: nếu bạn thực hiện đánh dấu phương thức tĩnh là đồng bộ, phương thức sẽ được khóa (lock) trên class không phải object. Tức là "*tại 1 thời điểm chỉ 1 thread được chạy trên 1 class*".

Ví dụ:

```
class Table {  
  
    synchronized static void printTable(String name, int n) {  
        for (int i = 1; i <= 5; i++) {  
            System.out.println(name + ":" + (n * i));  
            try {  
                Thread.sleep(100);  
            } catch (Exception e) {}  
        }  
    }  
  
    class MyThread1 extends Thread {  
        private String name;  
        public MyThread1(String name) {  
    }
```

The screenshot shows a continuation of the Viblo post. It features the same header and post structure. The code block continues from the previous part, showing the implementation of the MyThread1 and MyThread2 classes. The MyThread1 class calls the static printTable method.

```
        Table.printTable(name, 1);  
    }  
  
    class MyThread2 extends Thread {  
        private String name;  
        public MyThread2(String name) {  
    }  
  
        Table.printTable(name, 2);  
    }
```

```
private String name;
public MyThread2(String name) {
    this.name = name;
}

public void run() {
    Table.printTable(name, 10);
}

public class JavaThreadSynchronized1 {

    public static void main(String[] args) {
        MyThread1 t1 = new MyThread1("Thread 1");
        MyThread2 t2 = new MyThread2("Thread 2");
        MyThread1 t3 = new MyThread1("Thread 1(2)");
        t1.start();
        t2.start();
        t3.start();
    }
}
```

VIBLO

Posts

Questions

Discussions

Search Viblo



Sign In/Sign up

+7
Thread 1: 1
Thread 1: 2
Thread 1: 3
Thread 1: 4
Thread 1: 5
Thread 2: 10
Thread 2: 20
Thread 2: 30
Thread 2: 40
Thread 2: 50
Thread 1(2): 1
Thread 1(2): 2
Thread 1(2): 3
Thread 1(2): 4
Thread 1(2): 5

Nếu không khai báo đồng bộ (Synchronized):

Thread 2: 10
Thread 1: 1
Thread 1(2): 1
Thread 2: 20
Thread 1: 2
Thread 1(2): 2
Thread 2: 30
Thread 1: 3
Thread 1(2): 3

Thread 1: 5
Thread 1(2): 5
Thread 2: 50

+7

▼

3. Khối code (Synchronized blocks) được đồng bộ bên trong phương thức

Nếu bạn không cần đồng bộ toàn bộ phương thức và chỉ muốn đồng bộ từng phần nào đó trong phương thức, Java cung cấp cho bạn khối đồng bộ (Synchronized blocks) để làm điều đó.



Đây là 1 khối đồng bộ bên trong 1 phương thức không đồng bộ (unsynchronized):

```
public void add(int value){  
    //TODO: Do something else  
  
    // Synchronized block  
    synchronized(this){  
        this.count += value;  
    }  
}
```

Chú ý: Khối mã được đồng bộ phải được đặt trong dấu ngoặc nhọn. Biến "this" thể hiện khối mã sẽ được đồng bộ trên đối tượng (object) sở hữu phương thức chứa chúng. Các đối tượng được đồng bộ được gọi là các đối tượng giám sát (**Monitor Object**).

Chỉ một Thread được thực thi bên trong 1 Synchronized block trên cùng 1 đối tượng giám sát.

+7

▼

Ví dụ sau đây có thể được gọi là tương đồng với nhau khi sử dụng:

```
public class MyClass {  
  
    public synchronized void log1(String msg1, String msg2){  
        log.writeln(msg1);  
        log.writeln(msg2);  
    }  
  
    public void log2(String msg1, String msg2){  
        synchronized(this){  
            log.writeln(msg1);  
            log.writeln(msg2);  
        }  
    }  
}
```

4. Khối code được đồng bộ bên trong phương thức tĩnh

2 ví dụ sau đây là tương tự như trên nhưng là đồng bộ với phương thức tĩnh. Khác với khối Synchronized blocks ở phương thức thông thường, khi khai báo đồng bộ trên phương thức tĩnh thì khi đó cùng 1 thời điểm chỉ có 1 thread được thực thi trên 1 class (k phải trên thể hiện của class)

A screenshot of a Viblo post page. The title is "VIBLO". Below it are tabs for "Posts", "Questions", and "Discussions". A search bar contains "Search Viblo". On the right are icons for a magnifying glass, a person, and "Sign In/Sign up".
The post content shows Java code:

```
public class MyClass {  
    public static synchronized void log1(String msg1, String msg2){  
        log.writeln(msg1);  
        log.writeln(msg2);  
    }  
  
    public static void log2(String msg1, String msg2){  
        synchronized(MyClass.class){  
            log.writeln(msg1);  
            log.writeln(msg2);  
        }  
    }  
}
```

With social sharing icons for upvote (+7), downvote, save, share, and tweet.

5. Ví dụ DEMO

```
class MyClass5 {  
  
    private int a = 0;  
  
    public synchronized void log1(String msg1, String msg2) {  
  
        for (int i = 1; i <= 5; i++) {  
            a++;  
            System.out.println(msg1 + "-" + msg2 + " > A: " + a);  
  
            try {  
                Thread.sleep(100);  
            } catch (Exception e) {}  
        }  
    }  
  
    public void log2(String msg1, String msg2) {  
        synchronized (this) {  
            for (int i = 1; i <= 5; i++) {  
                a++;  
                System.out.println(msg1 + "-" + msg2 + " > A: " + a);  
  
                try {  
                    Thread.sleep(100);  
                } catch (Exception e) {}  
            }  
        }  
    }  
}
```

A screenshot of a Viblo post page. The title is "VIBLO". Below it are tabs for "Posts", "Questions", and "Discussions". A search bar contains "Search Viblo". On the right are icons for a magnifying glass, a person, and "Sign In/Sign up".
The post content shows Java code:

```
class MyClass5 {  
  
    private int a = 0;  
  
    public synchronized void log1(String msg1, String msg2) {  
  
        for (int i = 1; i <= 5; i++) {  
            a++;  
            System.out.println(msg1 + "-" + msg2 + " > A: " + a);  
  
            try {  
                Thread.sleep(100);  
            } catch (Exception e) {}  
        }  
    }  
  
    public void log2(String msg1, String msg2) {  
        synchronized (this) {  
            for (int i = 1; i <= 5; i++) {  
                a++;  
                System.out.println(msg1 + "-" + msg2 + " > A: " + a);  
  
                try {  
                    Thread.sleep(100);  
                } catch (Exception e) {}  
            }  
        }  
    }  
}
```

With social sharing icons for upvote (+7), downvote, save, share, and tweet.

```
        }
    }

}

public class JavaThreadSynchronized5 {

    public static void main(String[] args) {
        MyClass5 obj1 = new MyClass5();

        Thread t1 = new Thread() {
            public void run() {
                obj1.log1("Thread 1", "Log 1");
                // obj1.log2("Thread 1", "Log 2");
            }
        };

        Thread t2 = new Thread() {
            public void run() {
                obj1.log1("Thread 2", "Log 1");
                // obj1.log2("Thread 2", "Log 2");
            }
        };

        Thread t3 = new Thread() {
            public void run() {
                obj1.log1("Thread 3", "Log 1");
                // obj1.log2("Thread 3", "Log 2");
            }
        };

        t1.start();
        t2.start();
        t3.start();
    }
}
```

VIBLO

Posts Questions Discussions

Search Viblo



Sign In/Sign up

+7
▼



Output:

```
Thread 1-Log 1 > A: 1
Thread 1-Log 1 > A: 2
Thread 1-Log 1 > A: 3
Thread 1-Log 1 > A: 4
Thread 1-Log 1 > A: 5
```

VIBLO

Posts Questions Discussions

Search Viblo



Sign In/Sign up

+7
▼

```
Thread 3-Log 1 > A: 8
Thread 3-Log 1 > A: 9
Thread 3-Log 1 > A: 10
Thread 2-Log 1 > A: 11
Thread 2-Log 1 > A: 12
Thread 2-Log 1 > A: 13
```

Thread 2-Log 1 > A: 13
Thread 2-Log 1 > A: 14
Thread 2-Log 1 > A: 15

**6. Tổng kết **

Như vậy sử dụng Java Synchronized Blocks là rất hữu ích trong trường hợp có nhiều luồng (thread) cùng truy cập tới 1 đối tượng, 1 class, hay 1 tài nguyên nào đó. Việc này giúp ta kiểm soát được luồng truy cập cũng như giúp luồng an toàn và tính chính xác của dữ liệu.

Tham khảo:

- <http://www.javatpoint.com/static-synchronization-example>
- <http://tutorials.jenkov.com/java-concurrency/synchronized.html>
- http://www.tutorialspoint.com/java/java_thread_synchronization.htm



VIBLO Posts Questions Discussions Search Viblo Sign In/Sign up

Tuốt tuồn tuột về JDBC

Kien Nguyen

939 views 4 comments 8 likes

Đinh Công Ngọc

509 views 1 comment 4 likes

Design pattern: Tìm hiểu St...

Tran Anh Vu

485 views 3 comments 3 likes

Ký hiệu Big O – Sử dụng to...

Cuong Bui

460 views 4 comments 9 likes

More from Do Hung

Hướng dẫn cơ bản về Gradl...

Do Hung

5996 views 5 comments 4 likes

Framework Test UI cho ứng...

Do Hung

1073 views 3 comments 2 likes

Android Library: Tìm hiểu R...

Do Hung

11501 views 7 comments 4 likes

10 Bài Hướng Dẫn Cho Ngư...

Do Hung

772 views 3 comments 1 like

Clips

Comments

Login to comment

Trần Quang Định @quangdinhcr Dec 15th, 2015 2:19 PM

Mình thấy cách hiểu đồng bộ theo đoạn code hay phương thức rất dễ gây hiểu nhầm. Minh thì hiểu đồng bộ chia làm 2 cấp, đầu tiên là ở cấp đối tượng, 2 là ở cấp class(tức là khác đối tượng cũng đồng bộ được)

có hai trường hợp phải dùng các câu lệnh này method có từ khóa synchronized.

Điều quan trọng thứ 2 là khi synchronized(object) tức là object đó đang bị khóa bởi luồng(tạm gọi ThreadA) chứa câu lệnh đó. Và bất kỳ thread nào làm việc gì liên quan đến synchronized đối với object này, mà trong thời gian thằng ThreadA đang lock object đó đều phải đợi cho đến khi thằng ThreadA xong việc với object.

Tương tự với cấp độ Class.

0 | Reply Share

Trần Quang Định @quangdinhcr

Dec 15th, 2015 2:30 PM

Về mặt mình nói kĩ một chút, tức là đồng hồ chỉ chổi với sốt nhanh hella. Một thread A muốn sốt nhanh hella một lần với object của ThreadB thì cả hai phải rã giải đoạn khóa

mang cùng ý nghĩa như trên.

0 | Reply Share

Do Hung @hungtdo

Dec 16th, 2015 5:14 AM

@quangdinhcr Like & Thanks (y)

0 | Reply Share

RESOURCES

Posts	Questions
Videos	Tags
Authors	Discussions
Tools	Machine Learning

LINKS

Facebook
GitHub
Browser extension
Atom plugin

MOBILE APP

