Semaphores Using Java

Last modified on October 17th, 2014 by Joe.

Semaphore is an interesting topic in operating systems and it can be used in a variety of ways to solve challenging problems.

In this article I will explain,

1. what is a semaphore
2. how to implement a semaphore in java
3. an example problem and solution implementation using semaphore.

Following is a challenging programming problem we have in computers masters degree.

Example Problem

Assume there are three processes: Pa, Pb, and Pc. Only Pa can output the letter A, Pb B, and Pc C.
Utilizing only semaphores (and no other variables) the processes are synchronized so that the output satisfies the following conditions:

a) A B must be output before any C's can be output.
b) B's and C's must alternate in the output string, that is, after the first B is output, another B cannot be output until a C is output. Similarly, once a C is output, another C cannot be output until a B is output.
c) The total number of B's and C's which have been output at any given point in the output string cannot exceed the number of A's which have been output up to that point.

Now, just close your eyes and imagine of a solution to this problem. There are different solutions available, just using a constraint solver this can be solved.

This difficult problem can be solved very easily using semaphore's locking mechanism. Before diving into the solution,

What is a Semaphore

Semaphore is a word coined from ancient Greek words (sêma, phoros) meaning 'sign, bearing/bearer'.

Semaphore is a technique used to control access to common resource for competeing multiple processes. Semaphore maintains a counter which keeps track of the number of resources available. When a process requests access to resource, semaphore checks the variable count and if it is less than total count then grants access and subsequently reduces the available count.

Semaphore is just a gatekeeper guarding the resources. If available grants access and otherwise asks the processes to wait.

- When the resource count is arbitrary then this is called counting semaphore.
- If resource count is only one and the state value is restricted to on/off, then it is called binary semaphore. This is slightly different from mutex.

Counting Semaphore Implementation in Java

```
public class CountingSemaphore {
    private int value = 0;
    private int waitCount = 0;
    private int notifyCount = 0;

    public CountingSemaphore(int initial) {
        if (initial > 0) {
            value = initial;
        }
    }

    public synchronized void waitForNotify() {
        if (value <= waitCount) {
            waitCount++;
            try {
                do {
                    wait();
                } while (notifyCount == 0);
            } catch (InterruptedException e) {
                notify();
            } finally {
                waitCount--;
            }
            notifyCount--;
```

This is an implementation of a counting semaphore. It maintains counter variables ‘value’, ‘waitCount’ and ‘notifyCount’. This makes the thread to wait if value is lesser than waitCount and notifyCount is empty.

Binary Semaphore Implementation in Java

```
package com.javapapers.thread;

public class BinarySemaphore {
    private boolean locked = false;

    BinarySemaphore(int initial) {
        locked = (initial == 0);
    }

    public synchronized void waitForNotify() throws InterruptedException {
        while (locked) {
            wait();
        }
        locked = true;
    }

    public synchronized void notifyToWakeup() {
        if (locked) {
            notify();
        }
        locked = false;
    }
}
```

Binary semaphore just acts as an on or off switch. It maintains a boolean lock and on state of the lock it waits or notifies.

Solution to the Problem

```
package com.javapapers.thread;

import java.util.Random;

public class SemaphoreABC {
    protected static final BinarySemaphore binarySemaphore0 = new BinarySemapho
            0);
    protected static final BinarySemaphore binarySemaphore1 = new BinarySemapho
            1);
    protected static final CountingSemaphore countingSemaphore = new CountingSe
            0);

    protected static final Random random = new Random();
```

```java
    public static void main(String args[]) throws InterruptedException {
        new Thread(new ProcessA()).start();
        new Thread(new ProcessB()).start();
        new Thread(new ProcessC()).start();
        Thread.sleep(9000);
        System.exit(0);
    }
}
```

```java
package com.javapapers.thread;

public class ProcessA extends SemaphoreABC implements Runnable {
    public void run() {
        while (true) {
            try {
                Thread.sleep(1 + (int) (random.nextDouble() * 500));
            } catch (InterruptedException e1) {
                e1.printStackTrace();
            }
            System.out.print("A");
            countingSemaphore.notifyToWakeup();
        }
    }
}
```

```java
package com.javapapers.thread;

public class ProcessB extends SemaphoreABC implements Runnable {
    public void run() {
        while (true) {
            try {
                Thread.sleep(1 + (int) (random.nextDouble() * 800));
                binarySemaphore1.waitForNotify();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            countingSemaphore.waitForNotify();
            System.out.print("B");
            binarySemaphore0.notifyToWakeup();
        }
    }
}
```

```java
package com.javapapers.thread;

public class ProcessC extends SemaphoreABC implements Runnable {
    public void run() {
        while (true) {
            try {
                Thread.sleep(1 + (int) (random.nextDouble() * 800));
                binarySemaphore0.waitForNotify();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            countingSemaphore.waitForNotify();
            System.out.print("C");
            binarySemaphore1.notifyToWakeup();
        }
    }
}
```

```
Output: AABCABACAABCAABCAAABCABCABACAABCABACABACABACABAC
```

In solution to the problem, we have used both counting semaphore and binary semaphore.

- Condition b) is met with using the binary semaphore binarySemaphore0 and binarySemaphore1. These two act as alternate switches to the thread and allows for printing.
- Condition c) is met with using the counting semaphore. Both processB and processC calls waitForNotify and processA calls notifyToWakeup indicating that it has printed 'A', thus ensuring the total count of 'A' to be always larger than 'B','C' together.
- We have used a random number generator in three processes to create requests at random interval. Thread.sleep(6000) allows the overall program to run for 6 secs. If you want to see this run for longer duration, increase this number accordingly.

Java has its own implementation of counting semaphore refer API java.util.concurrent.Semaphore

Thank you, hope you have enjoyed reading this article.

> *This is a guest article from Vincy. She was a software engineer and left her fulltime job to pursue higher studies. Presently a student of masters computer science (ME Computer Science) and loves programming using php. You can reach her at giftvincy@gmail.com*

I have been getting requests for contribution to javapapers for quite sometime now. Sure I want to give authors chance to share their knowledge using javapapers as a platform but didn't want to compromise on the style and quality I have built over a period. Vincy is my wife and happy to have her article as first guest article for javapapers.

Get in touch with me if you wish to publish an article but there are two conditions,

1. I have a style of writing and its presenting technology in its simplest form and my readers love it. I will be editing your article 'heavily' to bring it to my own style.

2. Your article will be licensed under “creative commons cc-share with required attribution”.

Popular Articles

serialVersionUID in Java Serialization
Java Serialization
Java Abstraction

Comments on "Semaphores Using Java"

Chetan Jadhav says: *27/02/2012 at 12:03 pm*

Semaphores Concept explained in Plain and Simple way..!!!

---

igor says: *27/02/2012 at 12:31 pm*

What’s the point in re-inventing the weel? What’s wrong with standard java.util.concurrent.Semaphore?

Tena says: *27/02/2012 at 4:11 pm*

Igor, I think that Joe is just explaining the way semaphores work, not re-implementing the standard at all by 2 simple classes.

Very well explained Joe, another way to do it is by an example of few instances using the same object concurrently. And just for adding my 2 cents, another way to solve this kind of problems is by using monitors or messages!

congrats

Joe says: *27/02/2012 at 10:04 pm*

Igor, just trying to explain what a semaphore is. At the end of the article I have even referred java’s semaphore class.

Tena, thanks. I haven’t tried using monitors/messages, thats a nice idea. Let me try that out.

Tarsi says: *02/03/2012 at 4:37 pm*

Nicely Explained.
I love to read your articles.
Can you please explain annotations as well?

Deepak Kumar Shrivastava says: *04/03/2012 at 12:43 pm*

Hi Joe.

I am PG student. I really get benifited by your Papers.

I need your help.
Please help me.

I what brief difference between throw,throws and throwable.

please help any one, if good difference u have.

VckReddy says: *04/03/2012 at 11:27 pm*

Nice explannation on Semaphore with Example.

---

jerry says: *09/03/2012 at 8:00 pm*

Very nice blog,
Congratulation Joe!

---

harish says: *14/03/2012 at 3:59 pm*

it is not clear for me, can u please explore in detail

---

cndhu says: *16/03/2012 at 1:02 pm*

Hi Joe,

Thank you.
Please explain difference between cohesion and coupling in your way. This is my third request

---

Anonymous says: *26/03/2012 at 12:46 pm*

Hi joe,
Nicely Explained.

---

rama says: *27/03/2012 at 12:18 pm*

i want explanation about barber shop problem using semaphore

---

vidyasagar says: *07/04/2012 at 11:52 am*

Hi Joe,
Thanks for nice explanation. I wish someday you will write on Data structures in java. It will be a lot helpful for lots of people. Some of those you have already covered.If possible please write on creation of a data structure without importing any libraries(except java.lang.* which is default).

RAHUL DEB BARMAN says: *16/04/2012 at 12:31 am*

SO GOOD THANKS JOE SIR

E.Prakash says: *18/04/2012 at 1:55 pm*

Hi Joe ,

Really Execellant Information , Thanks a lots for crating Blogs

---

Asraful says: *19/04/2012 at 3:31 pm*

I already read 6-7 article from this site. Will go through to others. Subscribed already via e-mail. The way of writing is too simple but highly easy to understand complex concept like this one and JVM hook. Keep on.

---

karthik says: *02/05/2012 at 6:42 pm*

Hi Joe,

I want use this semaphore in my jsp to limit how many users can see registration page (Like 10 users) …and after that users will get in a Queue ..so that my website performance is not hit.

---

ravi krishnarpan says: *18/06/2012 at 11:44 am*

Sublime article

Amel says: *26/12/2012 at 6:01 pm*

nice really helpful ,thank you.

---

kalai says: *11/01/2013 at 7:25 pm*

Great explanations for wonderful concepts….

---

kalai says: *11/01/2013 at 7:25 pm*

Explantions are So Good…

Swapnil Phatkar says: *17/01/2013 at 8:53 am*

nice explained.

M.fathima sanjas says: *09/02/2013 at 1:44 pm*

it was really helpful..
thank u sir…

---

eswaramoorthy.s says: *19/02/2013 at 2:47 pm*

sir can you please explain this program

Goutam says: *08/04/2013 at 5:55 pm*

Very clearly and simply explained. Thanks Joe.

Concerned says: *17/04/2013 at 11:01 am*

Yes, but he could explain using standard classes. Creating your own is very much reinventing the wheel – and confusing for new learners.

Sagar says: *08/08/2013 at 8:29 pm*

Nice explanation….

J says: *26/08/2013 at 3:39 pm*

Sorry, I'm looking for an explanation of java.util.concurrent.Semaphore. I've got enough to learn without trying to get my head around a 3rd party reinvention of the wheel.

Candice says: *26/11/2013 at 12:58 am*

Thank you so much for this! Everything else I find is for using the built in Semaphore, and that isn't allowed in my assignment. I love the full code, it's so helpful when you are brand new to java. I look forward to viewing the rest of your tutorials.

Ashok says: *17/05/2014 at 2:25 pm*

nice explanation.

Prabhu R Babu says: *18/06/2014 at 10:40 pm*

Hi Joe,

Its a very good article. but why in the method WaitForNotify(), we call notify() in the catch block of InterruptedException. Also, when interruptedException happens, we should not decrement notifyCount. I think we should replace "notify()" with notifyCount++. Let me know what you think

---

alok says: *29/09/2014 at 2:29 pm*

I didnt understood this program. Can Anyone please give step by step explaination for this

---

Comments are closed for "Semaphores Using Java".