

CONTENTS

wait, notify and notifyAll in Java

// TUTORIAL //

Java Thread wait, notify and notifyAll Example

Published on August 4, 2022

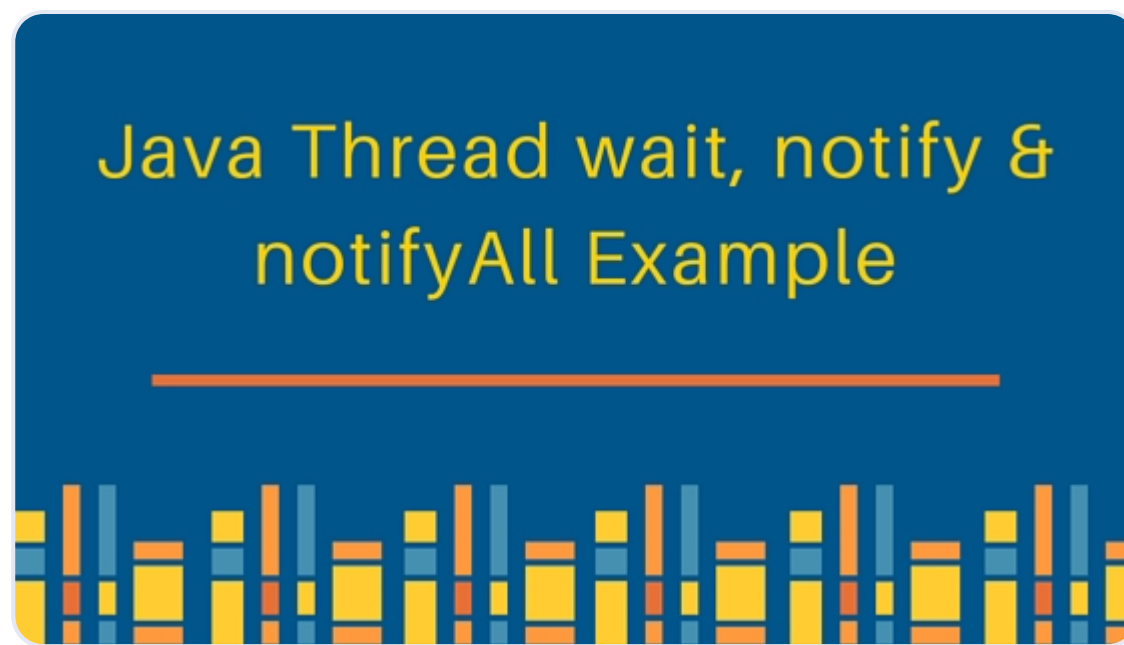
Java

 Pankaj



The Object class in java contains three final methods that allows threads to communicate about the lock status of a resource. These methods are **wait()**, **notify()** and **notifyAll()**. So today we will look into wait, notify and notifyAll in java program.

wait, notify and notifyAll in Java



The current thread which invokes these methods on any object should have the object **monitor** else it throws **java.lang.IllegalMonitorStateException** exception.

wait

Object wait methods has three variance, one which waits indefinitely for any other thread to call notify or notifyAll method on the object to wake up the current thread. Other two variances puts the current thread in wait for specific amount of time before they wake up.

notify

notify method wakes up only one thread waiting on the object and that thread starts execution. So if there are multiple threads waiting for an object, this method will wake up only one of them. The choice of the thread to wake depends on the OS implementation of thread management.

notifyAll

notifyAll method wakes up all the threads waiting on the object, although which one will process first depends on the OS implementation. These methods can be used to implement [producer consumer problem](#) where consumer threads are waiting for the objects in Queue and producer threads put object in queue and notify the waiting threads. Let's see an example where multiple threads work on the same object and we use wait, notify and notifyAll methods.

Message

A java bean class on which threads will work and call wait and notify methods.

```
package com.journaldev.concurrency;

public class Message {
    private String msg;

    public Message(String str){
        this.msg=str;
    }

    public String getMsg() {
        return msg;
    }

    public void setMsg(String str) {
        this.msg=str;
    }

}
```

Waiter

A class that will wait for other threads to invoke notify methods to complete it's processing. Notice that Waiter thread is owning monitor on Message object using synchronized block.

```
package com.journaldev.concurrency;

public class Waiter implements Runnable{

    private Message msg;

    public Waiter(Message m){
        this.msg=m;
    }

    @Override
    public void run() {
        String name = Thread.currentThread().getName();
        synchronized (msg) {
            try{
                System.out.println(name+" waiting to get notified at time:"+System.currentTimeMillis());
                msg.wait();
            }catch(InterruptedException e){
                e.printStackTrace();
            }
            System.out.println(name+" waiter thread got notified at time:"+System.currentTimeMillis())
            //process the message now
            System.out.println(name+" processed: "+msg.getMsg());
        }
    }

}
```

Notifier

A class that will process on Message object and then invoke notify method to wake up threads waiting for Message object. Notice that synchronized block is used to own the monitor of Message object.

```
package com.journaldev.concurrency;

public class Notifier implements Runnable {

    private Message msg;


    public Notifier(Message msg) {
        this.msg = msg;
    }

    @Override
    public void run() {
        String name = Thread.currentThread().getName();
        System.out.println(name+" started");
        try {
            Thread.sleep(1000);
            synchronized (msg) {
                msg.notify();
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

}
```

Reserve Capacity Now: Bare Metal NVIDIA HGX H200 GPUs →

Blog Docs Get Support Contact Sales

 **DigitalOcean**

Products ▾ Solutions ▾ Developers ▾ Partners ▾ Pricing


Log in ▾

Sign up ▾

Tutorials Questions Product Docs Cloud Chats

Q

Search Community



```
}
```

WaitNotifyTest

Test class that will create multiple threads of Waiter and Notifier and start them.

```
package com.journaldev.concurrency;

public class WaitNotifyTest {
```

```
public static void main(String[] args) {
    Message msg = new Message("process it");
    Waiter waiter = new Waiter(msg);
    new Thread(waiter,"waiter").start();

    Waiter waiter1 = new Waiter(msg);
    new Thread(waiter1, "waiter1").start();

    Notifier notifier = new Notifier(msg);
    new Thread(notifier, "notifier").start();
    System.out.println("All the threads are started");
}
}
```

When we will invoke the above program, we will see below output but program will not complete because there are two threads waiting on Message object and notify() method has wake up only one of them, the other thread is still waiting to get notified.

```
waiter waiting to get notified at time:1356318734009
waiter1 waiting to get notified at time:1356318734010
All the threads are started
notifier started
waiter waiter thread got notified at time:1356318735011
waiter processed: notifier Notifier work done
```

If we comment the notify() call and uncomment the notifyAll() call in Notifier class, below will be the output produced.

```
waiter waiting to get notified at time:1356318917118
waiter1 waiting to get notified at time:1356318917118
All the threads are started
notifier started
waiter1 waiter thread got notified at time:1356318918120
waiter1 processed: notifier Notifier work done
waiter waiter thread got notified at time:1356318918120
waiter processed: notifier Notifier work done
```

Since notifyAll() method wake up both the Waiter threads and program completes and terminates after execution. That’s all for wait, notify and notifyAll in java.

Thanks for learning with the DigitalOcean Community. Check out our offerings for compute, storage, networking, and managed databases.

[Learn more about our products →](#)

About the author(s)


Pankaj

See author profile

Category: Tutorial

Tags: Java





While we believe that this content benefits our community, we have not yet thoroughly reviewed it. If you have any suggestions for improvements, please let us know by clicking the “report an issue” button at the bottom of the tutorial.

Still looking for an answer?

Ask a question

Search for more help

Was this helpful?

Yes

No



Comments

JournalDev  • August 21, 2013



Why wait(),notify(),notifyAll() are in object class?please reply...Thanks

- Nitya

[Show replies](#) 

JournalDev  • October 11, 2013



The program is good but in that on miss take we have to set the wait time in waiter class other wise it will go for idel sleep time. so in that we have to set the time.

- sandeep

JournalDev  • November 8, 2013



thank you very [much.It](#) is really helpful

- Rajendra Verma

JournalDev  • November 19, 2013



i have a doubt here... In waiter class,waiter got a lock on msg object using synchronized(msg).Now waiter1 has been started also...How can waiter1 get the lock again on msg object using synchronized(msg) when waiter is already holding lock on msg object

- Ruhina

[Show replies](#) 

JournalDev  • December 13, 2013



Thanks sir...it was realyyyyyyy helpful...god bless u for ur effort

- Jaykishan

JournalDev  • January 20, 2014



Thanks a lot for making it simple to understand

- Rudi

JournalDev  • April 26, 2014



Thanks a lot... It really cleared my confusion

- Anind

JournalDev  • May 10, 2014



Excellent examples... continue more on collections...if you have blogs already please give link here Thanks & Regards, Vignesh Kanna M S/w Eng, Chennai, India.

- Vignesh Kanna M

[Show replies](#) 

JournalDev  • May 13, 2014



Thanks for this article. Really helpful for freshers and experienced people. In your next article please come up with some real time scenarios which you have faced in development in the next article through which increases in clarity of concept.

- Nanda

JournalDev  • June 6, 2014



Hi pankaj and All, i have one requirement in threads. Requirement: i have three [threads.so](#) how to make sure that one thread is exeucted after [another.ie](#) thread1 followed bye thread two ⇒is followed by threade three. please help me

- chandu

[Show replies](#) 

Load more comments



This work is licensed under a Creative Commons Attribution-NonCommercial- ShareAlike 4.0 International License.

Try DigitalOcean for free

Click below to sign up and get **\$200 of credit** to try our products over 60 days!



Popular Topics

- AI/ML
- Ubuntu
- Linux Basics
- JavaScript
- Python
- MySQL
- Docker
- Kubernetes

[All tutorials →](#)

[Talk to an expert →](#)



Become a contributor for community

Get paid to write technical tutorials and select a tech-focused charity to receive a matching donation.

Sign Up →



DigitalOcean Documentation

Full documentation for every DigitalOcean product.

Learn more →



Resources for startups and SMBs

The Wave has everything you need to know about building a business, from raising funding to marketing your product.

Learn more →

Get our newsletter

Stay up to date by signing up for DigitalOcean’s Infrastructure as a Newsletter.

Email address

Submit

New accounts only. By submitting your email you agree to our [Privacy Policy](#)

The developer cloud

Scale up as you grow — whether you're running one virtual machine or ten thousand.

View all products

Get started for free

Sign up and get \$200 in credit for your first 60 days with DigitalOcean.*

Get started

*This promotional offer applies to new accounts only.

Company

- About
- Leadership
- Blog
- Careers
- Customers
- Partners
- Referral Program
- Affiliate Program
- Press
- Legal
- Privacy Policy
- Security
- Investor Relations
- DO Impact
- Nonprofits

Products

- Overview
- Droplets
- Kubernetes
- Functions
- App Platform
- GPU Droplets
- 1-Click Models
- GenAI Platform
- Bare Metal GPUs
- Load Balancers
- Managed Databases
- Spaces
- Block Storage
- API
- Uptime
- Identity Access Management
- Cloudways

Resources

- Community Tutorials
- Community Q&A
- CSS-Tricks
- Write for DOnations
- Currents Research
- Hatch Startup Program
- Wavemakers Program
- Compass Council
- Open Source
- Newsletter Signup
- Marketplace
- Pricing
- Pricing Calculator
- Documentation
- Release Notes
- Code of Conduct
- Shop Swag

Solutions

- Website Hosting
- VPS Hosting
- Web & Mobile Apps
- Game Development
- Streaming
- VPN
- SaaS Platforms
- Cloud Hosting for Blockchain
- Startup Resources

Contact

- Support
- Sales
- Report Abuse
- System Status

Share your ideas



© 2025 DigitalOcean, LLC. Sitemap. Cookie Preferences

