

GO Lang course is LIVE.

study tonight Learn to Code Library Tests Forum Tech Blog

Search Login

## Basics of Java

- OOPS Concepts
- String Handling
- Exception Handling
- Java Multithreading
- Introduction to Multithreading
- Thread Class
- Creating a thread
- Joining threads
- Sleeping Thread in Java
- Naming Thread in Java
- Thread Priority in Java
- Daemon Thread in Java
- Synchronization
- Interthread Communication

## Advanced Topics

- Collection Framework
- Java GUI
- Reflection API
- RMI Application
- Inner Class
- Wrapper Class
- File Handling
- List
- Set
- Map
- Queue & Deque
- JDBC
- Layout Managers

# Java Interthread Communication

ADVERTISEMENT

Java provides benefits of avoiding thread pooling using inter-thread communication. The `wait()`, `notify()`, and `notifyAll()` methods of Object class are used for this purpose. These methods are implemented as **final** methods in Object, so that all classes have them. All the three methods can be called only from within a **synchronized** context.

- `wait()` tells calling thread to give up monitor and go to sleep until some other thread enters the same monitor and calls `notify()`.
- `notify()` wakes up a thread that called `wait()` on same object.
- `notifyAll()` wakes up all the threads that called `wait()` on same object.

## Difference between `wait()` and `sleep()`

<code>wait()</code>	<code>sleep()</code>
called from synchronized block	no such requirement
monitor is released	monitor is not released
gets awake when <code>notify()</code> or <code>notifyAll()</code> method is called	does not get awake when <code>notify()</code> or <code>notifyAll()</code> method is called
not a static method	static method
<code>wait()</code> is generally used on condition	<code>sleep()</code> method is simply used to put your thread on sleep.

## Thread Pooling

Pooling is usually implemented by loop i.e. to check some condition repeatedly. Once condition is true appropriate action is taken. This wastes CPU time.

## Thread Deadlock in Java

Deadlock is a situation of complete Lock, where no thread can complete its execution because of lack of resources. In the above picture, Thread 1 is holding resource R1 and needs another resource R2 to finish execution, but R2 is locked by Thread 2, which needs resource R3, which in turn is locked by Thread 3. Hence none of them can finish and are stuck in a deadlock.

### Example

In this example, multiple threads are accessing the same method that leads to deadlock condition. When a thread holds the resource and does not release it, then other threads will wait and in deadlock condition, wait time is never ending.

```

class Pen{}
class Paper{}

public class Write {
    public static void main(String[] args) {
        final Pen pn = new Pen();
        final Paper pr = new Paper();

        Thread t1 = new Thread() {
            public void run() {
                synchronized(pn) {
                    System.out.println("Thread1 is holding Pen");
                    try{
                        Thread.sleep(1000);
                    }
                }
            }
        };
    }
}

```

**OUTPUT:**  
Thread1 is holding Pen  
Thread2 is holding Paper

[← Prev](#) [Next →](#)

studytonight.com

About Us  
Testimonials  
Privacy Policy  
Terms  
Contact Us  
Suggest  
We are Hiring!  
© 2022 Studytonight Technologies Pvt. Ltd.

Learn Coding (for beginners)  
Tutorial Library  
Interview Tests  
Curious  
Practice Coding

Coding Courses  
Learn Go Lang  
Learn Java Script  
Learn CSS  
Learn HTML  
Resources  
C Language  
C++/STL  
Java  
DBMS  
Python  
PHP  
Android  
Game Development  
Data Structure & Alog.  
Operating System  
Computer Network  
Computer Architecture  
Docker  
GO Language  
GIT Guide  
Linux Guide  
More...

Interview Tests  
Java Interview Tests  
Python Interview Tests  
DBMS Interview Tests  
Linux Interview Tests  
Aptitude Tests  
GATE 2022 Tests  
More...

Projects/Programs  
Python Projects  
C Projects  
Python Programs  
C Programs  
C++ Programs  
Java Programs