Inter-thread communication in Java

**Inter-thread communication** or **Co-operation** is all about allowing synchronized threads to communicate with each other.

Cooperation (Inter-thread communication) is a mechanism in which a thread is paused running in its critical section and another thread is allowed to enter (or lock) in the same critical section to be executed. It is implemented by following methods of **Object class**:

* wait()
* notify()
* notifyAll()

1) wait() method

Causes current thread to release the lock and wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.

The current thread must own this object's monitor, so it must be called from the synchronized method only otherwise it will throw exception.

| **Method** | **Description** |
| --- | --- |
| public final void wait()throws InterruptedException | waits until object is notified. |
| public final void wait(long timeout)throws InterruptedException | waits for the specified amount of time. |

### 2) notify() method

Wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened. The choice is arbitrary and occurs at the discretion of the implementation. Syntax:

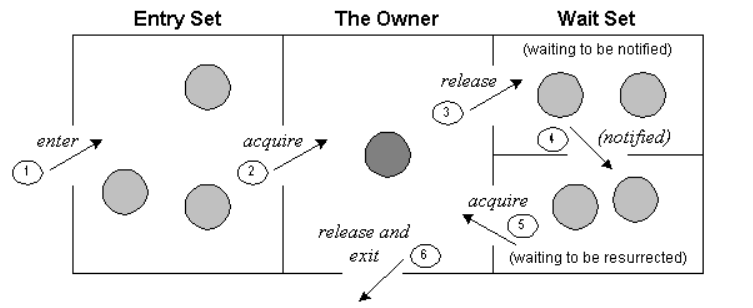
public final void notify()

### 3) notifyAll() method

Wakes up all threads that are waiting on this object's monitor. Syntax:

public final void notifyAll()

### Understanding the process of inter-thread communication



The point to point explanation of the above diagram is as follows:

1. Threads enter to acquire lock.
2. Lock is acquired by on thread.
3. Now thread goes to waiting state if you call wait() method on the object. Otherwise it releases the lock and exits.
4. If you call notify() or notifyAll() method, thread moves to the notified state (runnable state).
5. Now thread is available to acquire lock.
6. After completion of the task, thread releases the lock and exits the monitor state of the object.

### Why wait(), notify() and notifyAll() methods are defined in Object class not Thread class?

It is because they are related to lock and object has a lock.

### Difference between wait and sleep?

Let's see the important differences between wait and sleep methods.

| **wait()** | **sleep()** |
| --- | --- |
| wait() method releases the lock | sleep() method doesn't release the lock. |
| is the method of Object class | is the method of Thread class |
| is the non-static method | is the static method |
| is the non-static method | is the static method |
| should be notified by notify() or notifyAll() methods | after the specified amount of time, sleep is completed. |

**What is Polling and what are problems with it?**  
The process of testing a condition repeatedly till it becomes true is known as polling.

Polling is usually implemented with the help of loops to check whether a particular condition is true or not. If it is true, certain action is taken. This waste many CPU cycles and makes the implementation inefficient.  
For example, in a classic queuing problem where one thread is producing data and other is consuming it.

**How Java multi threading tackles this problem?**  
To avoid polling, Java uses three methods, namely, **wait(), notify() and notifyAll().**  
All these methods belong to object class as final so that all classes have them. They must be used within a synchronized block only.

* **wait()-**It tells the calling thread to give up the lock and go to sleep until some other thread enters the same monitor and calls notify().
* **notify()-**It wakes up one single thread that called wait() on the same object. It should be noted that calling notify() does not actually give up a lock on a resource.
* **notifyAll()-**It wakes up all the threads that called wait() on the same object.

### 

### 

### 

### 

### 

### 

### Example of inter thread communication in java

class InterThread {

public static void main(String args[])throws InterruptedException{

PC pc\_obj = new PC();

Thread t1 = new Thread(new Runnable(){

public void run(){

try{

pc\_obj.produce();

}

catch(InterruptedException e){System.out.println(e);}

}

});

Thread t2 = new Thread(new Runnable(){

public void run(){

try{

pc\_obj.consume();

}

catch(InterruptedException e){System.out.println(e);}

}

});

t1.start();

t2.start();

t1.join();

t2.join();

}

}

class PC{

public void produce()throws InterruptedException{

synchronized(this){

System.out.println("Running in produce");

wait();

System.out.println("woke up again after notify in producer");

}

}

public void consume()throws InterruptedException{

synchronized(this){

Thread.sleep(2000);

System.out.println("After wait from producer inside consumer ");

notify();

Thread.sleep(2000);

}

}

}

### Output

Running in produce

After wait from producer inside consumer

woke up again after notify in producer

### Example of inter thread communication in java

Let's see the simple example of inter thread communication.

class Customer{

int amount=10000;

synchronized void withdraw(int amount){

System.out.println("going to withdraw...");

if(this.amount<amount){

System.out.println("Less balance; waiting for deposit...");

try{wait();}catch(Exception e){}

}

this.amount-=amount;

System.out.println("withdraw completed...");

}

synchronized void deposit(int amount){

System.out.println("going to deposit...");

this.amount+=amount;

System.out.println("deposit completed... ");

notify();

}

}

class Test{

public static void main(String args[]){

final Customer c=new Customer();

new Thread(){

public void run(){c.withdraw(15000);}

}.start();

new Thread(){

public void run(){c.deposit(10000);}

}.start();

}}

Output

going to withdraw...

Less balance; waiting for deposit...

going to deposit...

deposit completed...

withdraw completed...

**Message Passing:** Objects communicate with one another by sending and receiving information to each other. A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results. Message passing involves specifying the name of the object, the name of the function and the information to be sent.