

JAVA.LANG.OBJECT.NOTIFY METHOD

http://www.tutorialspoint.com/java/lang/object_notify.htm

Copyright © tutorialspoint.com

Description

The **java.lang.Object.notify** 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. A thread waits on an object's monitor by calling one of the wait methods.

This method should only be called by a thread that is the owner of this object's monitor. A thread becomes the owner of the object's monitor in one of three ways:

- By executing a synchronized instance method of that object.
- By executing the body of a synchronized statement that synchronizes on the object.
- For objects of type Class, by executing a synchronized static method of that class.

Only one thread at a time can own an object's monitor.

Declaration

Following is the declaration for **java.lang.Object.notify** method

```
public final void notify()
```

Parameters

- NA

Return Value

This method does not return a value.

Exception

- **IllegalMonitorStateException** --if the current thread is not the owner of this object's monitor.

Example

The following example shows the usage of lang.Object.notify method.

```
package com.tutorialspoint;

import java.util.Collections;
import java.util.LinkedList;
import java.util.List;

public class ObjectDemo extends Object {

    private List synchedList;

    public ObjectDemo() {
        // create a new synchronized list to be used
        synchedList = Collections.synchronizedList(new LinkedList());
    }

    // method used to remove an element from the list
    public String removeElement() throws InterruptedException {
        synchronized (synchedList) {

            // while the list is empty, wait
```

```

while (synchedList.isEmpty()) {
    System.out.println("List is empty...");
    synchedList.wait();
    System.out.println("Waiting...");
}
String element = (String) synchedList.remove(0);

return element;
}
}

// method to add an element in the list
public void addElement(String element) {
    System.out.println("Opening...");
    synchronized (synchedList) {

        // add an element and notify all that an element exists
        synchedList.add(element);
        System.out.println("New Element:" + element + "");

        synchedList.notifyAll();
        System.out.println("notifyAll called!");
    }
    System.out.println("Closing...");
}

public static void main(String[] args) {
    final ObjectDemo demo = new ObjectDemo();

    Runnable runA = new Runnable() {

        public void run() {
            try {
                String item = demo.removeElement();
                System.out.println("" + item);
            } catch (InterruptedException ix) {
                System.out.println("Interrupted Exception!");
            } catch (Exception x) {
                System.out.println("Exception thrown.");
            }
        }
    };

    Runnable runB = new Runnable() {

        // run adds an element in the list and starts the loop
        public void run() {
            demo.addElement("Hello!");
        }
    };

    try {
        Thread threadA1 = new Thread(runA, "A");
        threadA1.start();

        Thread.sleep(500);

        Thread threadA2 = new Thread(runA, "B");
        threadA2.start();

        Thread.sleep(500);

        Thread threadB = new Thread(runB, "C");
        threadB.start();

        Thread.sleep(1000);

        threadA1.interrupt();
        threadA2.interrupt();
    }
}

```

```
    } catch (InterruptedException x) {  
    }  
  }  
}
```

Let us compile and run the above program, this will produce the following result:

```
List is empty...  
List is empty...  
Opening...  
New Element:'Hello!'  
notifyAll called!  
Closing...  
Waiting...  
Hello!  
Waiting...  
List is empty...  
Interrunted Exception!
```

Loading [Mathjax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js