**Sleep() in Java**

**Definition of sleep method :**

"To suspend execution of the current thread for a specified period."

This is an efficient means of making processor time available to the other threads of an application or other applications that might be running on a computer system.

**Example :**

import java.lang.Thread;

import java.io.\*;

public class TestSleepMethod2

{ // main method

public static void main(String argvs[])

{

try {

for (int j = 0; j < 5; j++)

{

// The main thread sleeps for the 1000 milliseconds, which is 1 sec whenever the loop runs

Thread.sleep(1000);

System.out.println(j); // displaying the value of the variable

}

}

catch (Exception expn)

{

// catching the exception

System.out.println(expn);

} } }

**Output:**

0

1

2

3

4

**Yield() in Java**

**Definition of yield method:**

A yield() method is a static method of Thread class and it can stop the currently executing thread and will give a chance to other waiting threads of the same priority.

If in case there are no waiting threads or if all the waiting threads have low priority then the same thread will continue its execution.

**Example :**

public class JavaYieldExp extends Thread {

public void run() {

for (int i=0; i<3 ; i++)

System.out.println(Thread.currentThread().getName() + " in control");

}

public static void main(String[]args) {

JavaYieldExp t1 = new JavaYieldExp();

JavaYieldExp t2 = new JavaYieldExp();

// this will call run() method

t1.start();

t2.start();

for (int i=0; i<3; i++) {

// Control passes to child thread

t1.yield();

System.out.println(Thread.currentThread().getName() + " in control");

}

}

}

Output:

main in control

main in control

main in control

Thread-0 in control

Thread-0 in control

Thread-0 in control

Thread-1 in control

Thread-1 in control

Thread-1 in control

**Wait() in Java**

**Definition of wait method:**

wait() causes current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object.

In other words, this method behaves exactly as if it simply performs the call wait(0).

The current thread must own this object's monitor.

Example :

class GunFight {

private int bullets = 40;

// This method fires the number of bullets that are

// passed it. When the bullet in magazine becomes zero,

// it calls the wait() method and releases the lock.

synchronized public void fire(int bulletsToBeFired) {

for (int i = 1; i <= bulletsToBeFired; i++) {

if (bullets == 0) {

System.out.println(i - 1

+ " bullets fired and "

+ bullets + " remains");

System.out.println(

"Invoking the wait() method");

try { wait();

}

catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(

"Continuing the fire after reloading");

} bullets--;

}

System.out.println(

"The firing process is complete");

}

// reload() increases the bullets by 40 everytime it is invoked and calls the notify() method which wakes up the thread that was sent to sleep using wait() inside of fire() method

synchronized public void reload() {

System.out.println(

"Reloading the magazine and resuming "

+ "the thread using notify()");

bullets += 40;

notify();

} }

public class WaitDemo extends Thread {

public static void main(String[] args){

GunFight gf = new GunFight();

// Creating a new thread and invoking our fire() method on it

new Thread() {

@Override public void run() { gf.fire(60); }

}.start();

// Creating a new thread and invoking our reload method on it

new Thread() {

@Override public void run() { gf.reload(); }

}.start() } }

**Output :**

40 bullets fired and 0 remains

Invoking the wait() method

Reloading the magazine and resuming the thread using notify()

Continuing the fire after reloading

The firing process is complete

**Interview Questions and Answers**

1. Does sleep block a process?

Ans : Sleep blocks all execution, but only in the thread from which you call it.

1. Does sleep block all threads?

Ans : Sleep method causes the current thread to immediately block for the number of milliseconds or the time interval you pass to the method, and yields the remainder of its time slice to another thread.

Once that interval elapses, the sleeping thread resumes execution.

1. When should you use yield?

Ans : We should use yield when we want to iterate over a sequence, but don't want to store the entire sequence in memory.

1. Does wait () return?

Ans : If successful, wait() returns a value that is the process ID (PID) of the child whose status information has been obtained. If unsuccessful, wait() returns -1 and sets errno to one of the following values:

1. Why avoid thread sleep Java?

Ans : Thread. sleep is bad! It blocks the current thread and renders it unusable for further work.