SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23CSB101- OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Unit III – EXCEPTION HANDLING AND MULTITHREADING

Topic : INTER-THREAD COMMUNICATION

INTER THREAD COMMUNICATION

- INTER THREAD COMMUNICATION : 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.

INTER THREAD COMMUNICATION

Difference between critical section and deadlock

Aspect	Critical Section	Deadlock
Definition	A part of the program that accesses shared resources and needs to be executed by only one thread at a time to ensure correctness.	A situation where two or more threads are stuck, waiting on each other to release resources, preventing any progress.
Issue Addressed	Ensures mutual exclusion to prevent race conditions.	Prevents threads from getting stuck in a circular waiting state.
Cause	Multiple threads trying to access a shared resource simultaneously.	Circular waiting between threads for resources they hold.
Solution	Synchronization (locks, synchronized, etc.).	Avoid circular waits, use timeout mechanisms, proper resource ordering.
Impact	Leads to race conditions if not handled properly.	Leads to threads becoming stuck and never completing their task.
Example	Using synchronized to ensure only one thread accesses the resource at a time.	Two threads waiting for each other to release locks, resulting in a freeze.

INTER THREAD COMMUNICATION

It is implemented by following methods of Object class and all these methods can be called only from within a synchronized context.

S.No.	Method & Description
1	**public final void wait() throws InterruptedException** Causes the current thread to wait until another thread invokes the notify().
2	**public final void wait(long timeout) throws InterruptedException** Causes current thread to wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed. **Parameters:** **timeout** – *the maximum time to wait in milliseconds.*
3	**public final void notify()** Wakes up a single thread that is waiting on this object's monitor.
4	**Public final void notifyAll()** Wakes up all the threads that called wait( ) on the same object.

INTER THREAD COMMUNICATION

Difference between wait() and sleep()		
Parameter	**wait()**	**sleep()**
Synchonized	wait should be called from synchronized context i.e. from block or method, If you do not call it using synchronized context, it will throw IllegalMonitorStateException	It need not be called from synchronized block or methods
Calls on	wait method operates on Object and defined in Object class	Sleep method operates on current thread and is in java.lang.Thread
Release of lock	wait release lock of object on which it is called and also other locks if it holds any	Sleep method does not release lock at all
Wake up condition	until call notify() or notifyAll() from Object class	Until time expires or calls interrupt()
static	wait is non-static method	sleep is static method

Java Program: Inter-thread Communication, Suspending, Resuming, Stopping

```
class MyThread extends Thread {
    private boolean suspendFlag = false;
    private boolean stopFlag = false;

    public synchronized void customSuspend() {
        suspendFlag = true;
    }
    public synchronized void customResume() {
        suspendFlag = false;
        notify(); // Wake up the thread
    }
```

```
public synchronized void customStop() {
    stopFlag = true;
    notify(); // Wake up the thread if it's waiting
}

@Override
public void run() {
    int i = 1;
    while (true) {
        synchronized (this) {
            while (suspendFlag) {
```

Cont…

```
try {
                wait(); // Pause the thread
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted.");
            }
        }

        if (stopFlag) {
            break; // Exit the loop
        }
    }
```

Cont…

```
            System.out.println("Thread running: " + i++);
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                System.out.println("Sleep interrupted.");
            }
        }

        System.out.println("Thread stopped.");
    }
}
```

Cont…

```
public class ThreadControlDemo {
    public static void main(String[] args) {
        MyThread t = new MyThread();
        t.start();
        try {
            Thread.sleep(2000);
            t.customSuspend();
            System.out.println("Thread suspended.");
            Thread.sleep(2000);
            t.customResume();
            System.out.println("Thread resumed.");
            Thread.sleep(2000);
            t.customStop();
            System.out.println("Thread stopped request sent.");
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }
    }
}
```

Output:

Thread running: 1
Thread running: 2
Thread running: 3
Thread running: 4
Thread suspended.
Thread resumed.
Thread running: 5
Thread running: 6
Thread running: 7
Thread running: 8
Thread stopped request sent.
Thread stopped.

EXAMPLE:

simple bank transaction operations with inter-thread communication:

```
class Customer{
int Balance=10000;
synchronized void withdraw(int amount)
{
System.out.println("going to withdraw..."+amount);
if(Balance<amount)
{
System.out.println("Less balance; Balance = Rs. "+Balance+"\nWaiting for deposit...\n");
```

CONT…

EXAMPLE:

```
try
{
wait();
}
catch(Exception e){}
}
Balance-=amount;
System.out.println("withdraw completed...");
}
```

CONT…

EXAMPLE:

```
synchronized void deposit(int amount)
{
System.out.println("going to deposit... Rs. "+amount);
Balance+=amount;
System.out.println("deposit completed... Balance = "+Balance);
notify();
}
}
```

CONT…

EXAMPLE:

```
class ThreadCommn
{
public static void main(String args[]) {
Customer c=new Customer();
new Thread()
{
public void run(){c.withdraw(20000);}
}.start();
new Thread(){
public void run(){c.deposit(15000);}
}.start();
}
}
```

EXAMPLE:

```
class ThreadCommn
{
public static void main(String args[]) {
Customer c=new Customer();
new Thread()
{
public void run(){c.withdraw(20000);}
}.start();
new Thread(){
public void run(){c.deposit(15000);}
}.start();
}
}
```

Output:

```
going to withdraw...20000
```

EXAMPLE:

Output:

going to withdraw...20000
Less balance; Balance = Rs. 10000
Waiting for deposit...
going to deposit... Rs. 15000
deposit completed... Balance = 25000
withdraw completed...


Thank
you