# Inter thread Communication:

**The process of providing communication between more than one thread is called as " Inter Thread Communication".**

**To perform Inter Thread Communication we have to use the following methods.**

**1.wait()**

**2.notify()**

**3.notifyAll()**

**Where wait() method can be used to keep a thread in waiting state.**

**Where notify() method can be used to give a notification to a thread which is available in waiting state.**

**Where notifyAll() method can be used to give a notification to all the threads which are available in waiting state.**

**The above methods are provided by JAVA in java.lang.Object class.**

**If we want to use these methods in java applications then we must provide "Synchronization".**

**Note : In general, Inter Thread Communication will provide solutions for the problems like "Producer-Consumer" problem.**

**In Producer-Consumer problem, producer and cosumer are two threads, where producer has to produce an item and consumer has to consume that item, the same sequence has to be providedfor infinite no. of times, where Producer must not produce an item with out consuming previous item by consumer and consumer must not consume an item with out producing an item by producer.**

**Code:**

class A

{

boolean flag=true;

int count=0;

public synchronized void produce()

{

try

{

while(true)

{

if(flag == true)// **producer is producing an item so consumer must //wait till that time**

{

count=count+1;

System.out.println("Producer Produced Item"+count);

flag=false;// **now not producing an item, the producer**

notify();// **for consumer to start consuming that produced item**

wait();// **producer is in wait state**

}

else// **consumer is in the process of consuming an item.So wait for the //producer thread to produce an item**

{

wait();

}

}

}

catch (Exception e)

{

e.printStackTrace();

}

}

public synchronized void consume()

{

try

{

while(true)

{

if(flag == true)// **producer is active in producing an item so consumer //must be in waiting state**

{

wait();

}

else // **consumer is in the state of consuming an item**

{

System.out.println("Consumer Consumed Item"+count);

flag=true;

notify();// **to producer for item production now**

wait();

}

}

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

class Producer extends Thread

{

A a;

Producer(A a)

{

this.a=a;

}

public void run()

{

a.produce();

}

}

class Consumer extends Thread

{

A a;

Consumer(A a)

{

this.a=a;

}

public void run()

{

a.consume();

}

}

class Test

{

public static void main(String[] args)

{

A a=new A();

Producer producer=new Producer(a);

Consumer consumer=new Consumer(a);

producer.start();

consumer.start();

}

}

**Note: Automatically now threads will be created,that new thread will come to rspective class run() method and inside this method a.produce() and a. consume() is identified/found.**

**THEREFORE ITC[INTER THREAD COMMUNICATION] is providing solution for Producer-Consumer problem AS** EXPLAINED ABOVE.