SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

19CS204 OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Topic – Interthread Communication

Interthread Communication

- **Inter-thread communication** is all about allowing synchronized threads to communicate with each other.
- Inter thread communication in Java is a technique through which multiple threads communicate with each other.
- It provides an efficient way through which more than one thread communicate with each other by reducing CPU idle time. CPU idle time is a process in which CPU cycles are not wasted.
- 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()

Interthread Communication

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.

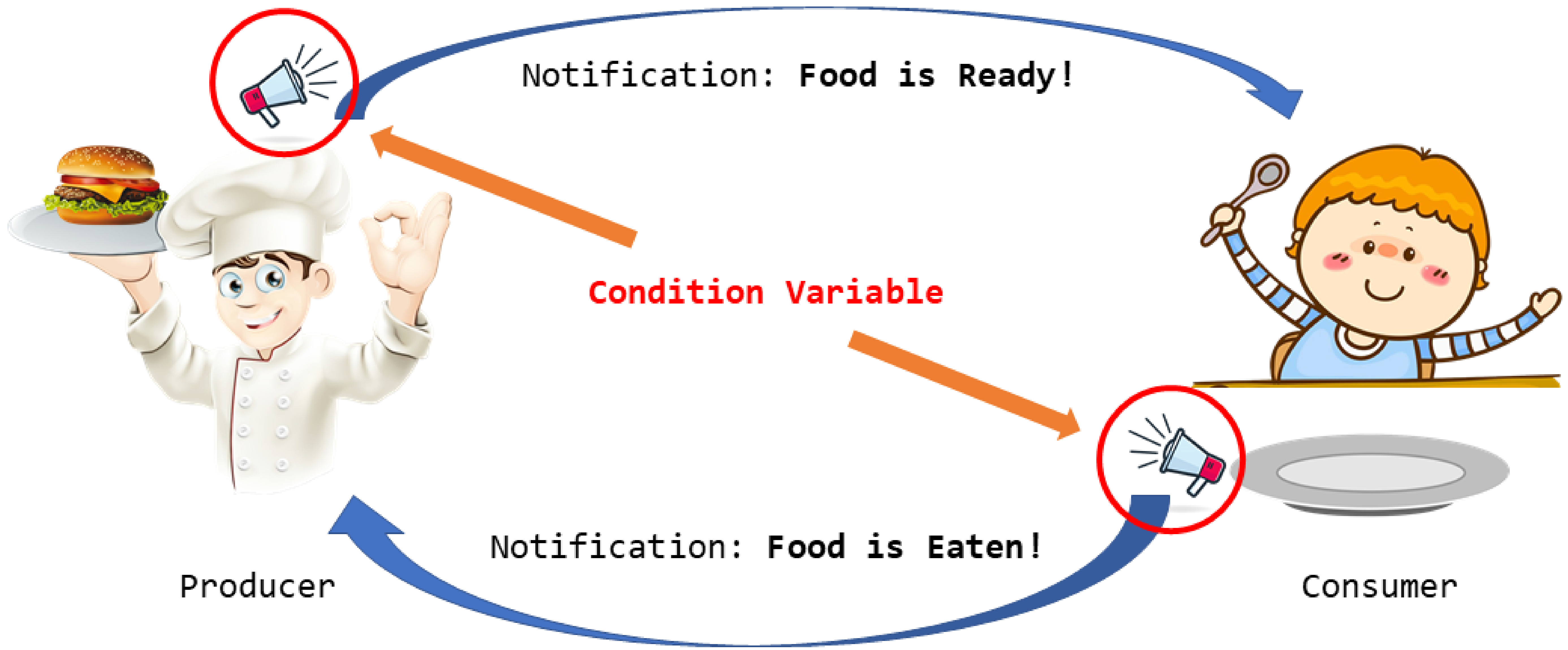
notify() method

- 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.
- public final void notify()

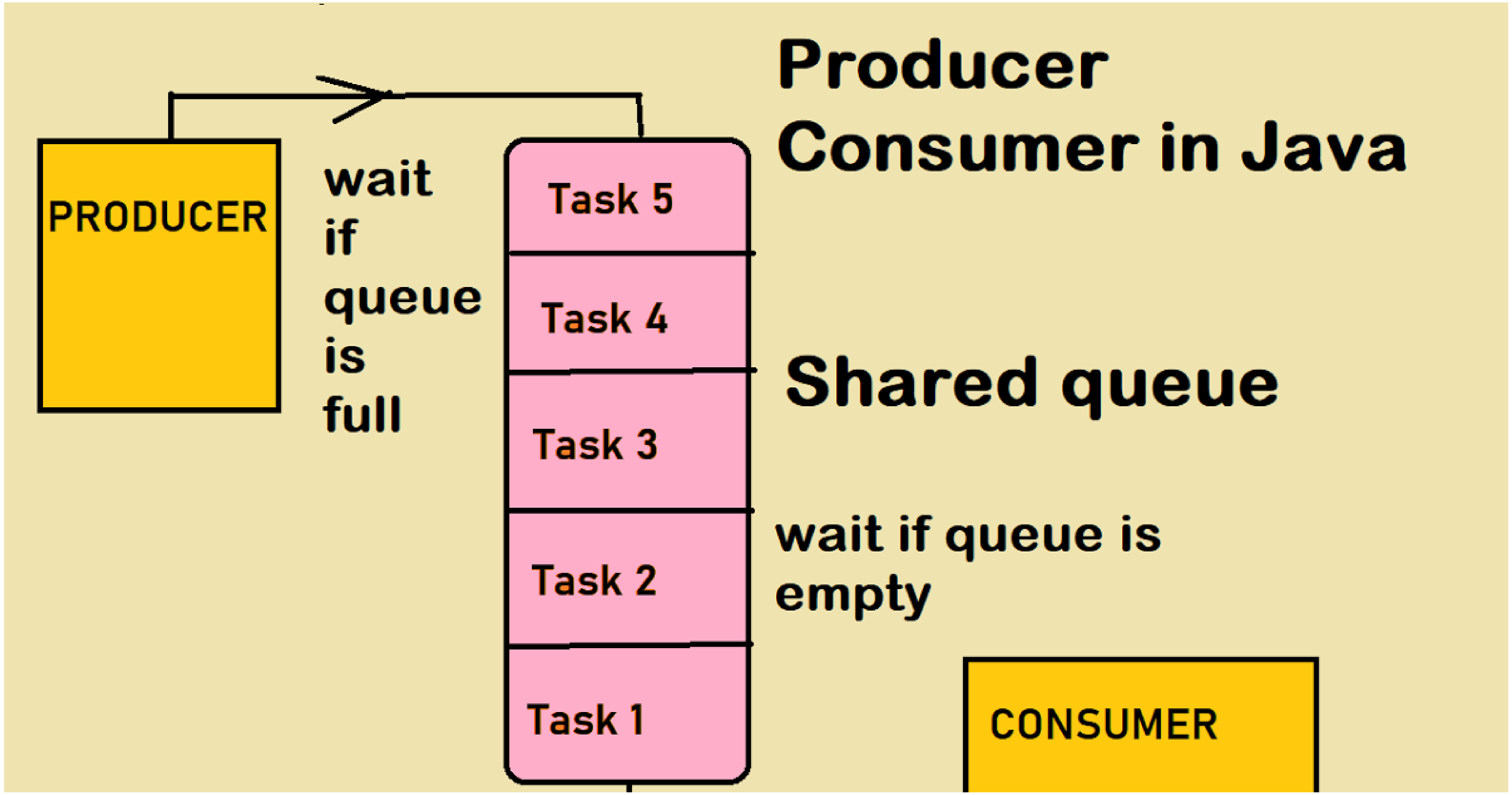
notifyAll() method

- Wakes up all threads that are waiting on this object's monitor. Syntax:
- public final void notifyAll()

Interthread Communication


Notification: **Food is Ready!**
Condition Variable
Notification: **Food is Eaten!**
Producer
Consumer

Interthread Communication



Interthread Communication

```
public class A
{
int i;
 synchronized void deliver(int i)
 {
   this.i = i;
   System.out.println("Data Produced: " +i);
 }
 synchronized int receive()
 {
   System.out.println("Data Consumed: " + i);
   return i;
 } }
public class producer extends Thread
{
 A obj;
 producer(A obj)
 {
  this.obj = obj;
 }
public void run()
{
 for(int j = 1; j <= 5; j++){
   obj.deliver(j);
  } } }
```

```
public class consumer extends Thread
{
A obj;
consumer(A obj)
{
 this.obj = obj;
}
public void run()
{
for(int k = 0; k <= 5; k++){
 obj.receive();
}
 }
}
public class NoCommunication
{
public static void main(String[] args)
{
A obj = new A();
producer t1 = new producer(obj);
consumer t2 = new consumer(obj);
  t1.start();
  t2.start();
 }
}
```

Interthread Communication

```
public class A
{
 int i;
 boolean flag = false; // flag will be true when data production is over.
synchronized void deliver(int i)
{
 if(flag)
 try
 {
  wait(); }
 catch(InterruptedException ie)
 {
  System.out.println(ie);
 }
   this.i = i;
   flag = true; // When data production is over, it will store true into flag.
   System.out.println("Data Produced: " +i);
   notify(); // When data production is over, it will notify Thread2 to use it.
 }
synchronized int receive()
{
if(!flag)
try {
 wait(); // Wait till a notification is received from Thread1.
}
catch(InterruptedException ie){
 System.out.println(ie);
}
 System.out.println("Data Consumed: " + i);
  flag = false; // It will store false into flag when data is received.
  notify(); // When data received is over, it will notify Thread1 to produce next data.
  return i;
 }
}
```

```
public class produced extends Thread
{
 A obj;
 produced(A obj)
 {
  this.obj = obj;
 }
public void run()
{
for(int j = 1; j <= 5; j++){
 obj.deliver(j);
  }
}}
public class consumed extends Thread
{
A obj;
consumed(A obj)
{
 this.obj = obj;
}
public void run()
{
for(int k = 0; k <= 5; k++){
 obj.receive();
}
 }}
```

Interthread Communication

```
public class Communication
{
public static void main(String[] args)
{
 A obj = new A(); // Creating an object of class A.

// Creating two thread objects and pass reference variable obj as parameter to Thread1 and
Thread2.
produced t1 = new produced(obj);
consumed t2 = new consumed(obj);
// Run both threads.
  t1.start();
  t2.start();
 }
}
```

THANK YOU