SNS COLLEGE OF TECHNOLOGY

Kurumbapalayam (Po), Coimbatore – 641 107

AN AUTONOMOUS INSTITUTION

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

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

23CSB201-OBJECT ORIENTED PROGRAMMING

UNIT IV - EXCEPTION AND MULTITHREADING

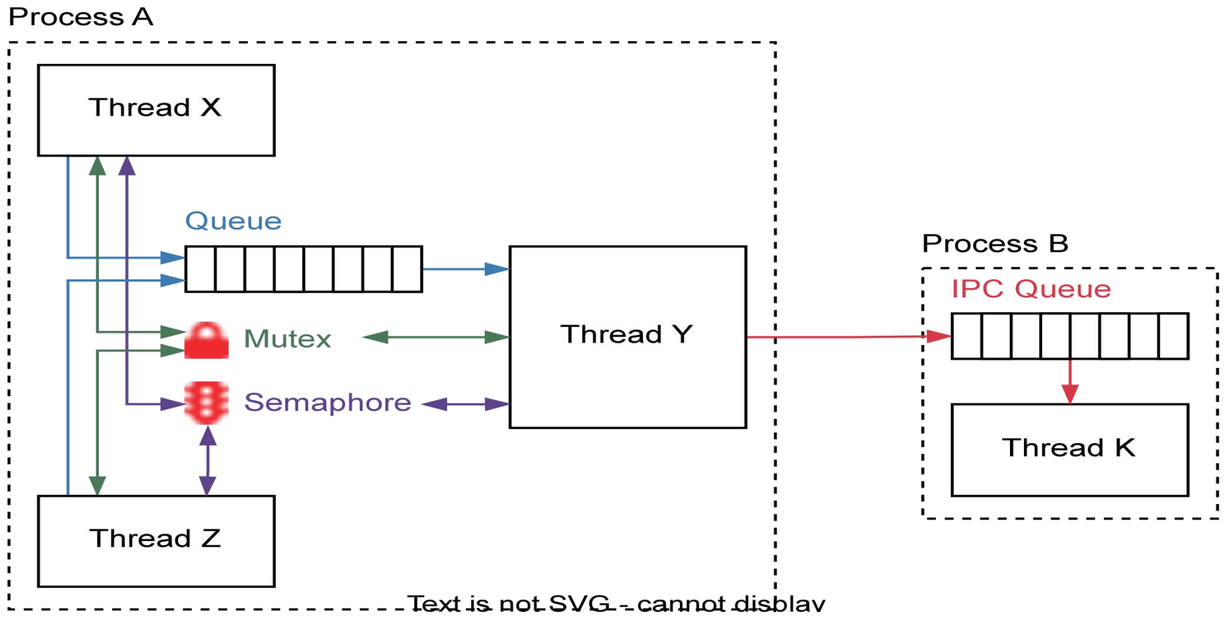
TOPIC 9 - Inter thread communication

1. What is inter Thread Communication?

Thread Communication (or **Inter-thread communication**) is a mechanism by which **multiple threads communicate and coordinate with each other**.

It is used when:

- One thread depends on the result of another
- Threads must work in a **proper sequence**



2. Why inter Thread Communication is Needed?

Without communication:

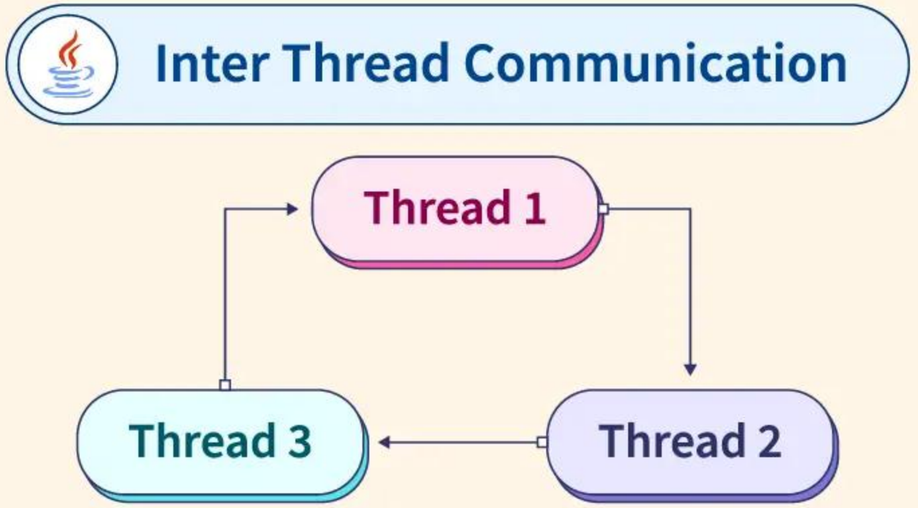
❌ Threads may execute in random order

❌ Data inconsistency may occur

❌ CPU time may be wasted (busy waiting)

Thread communication helps to:

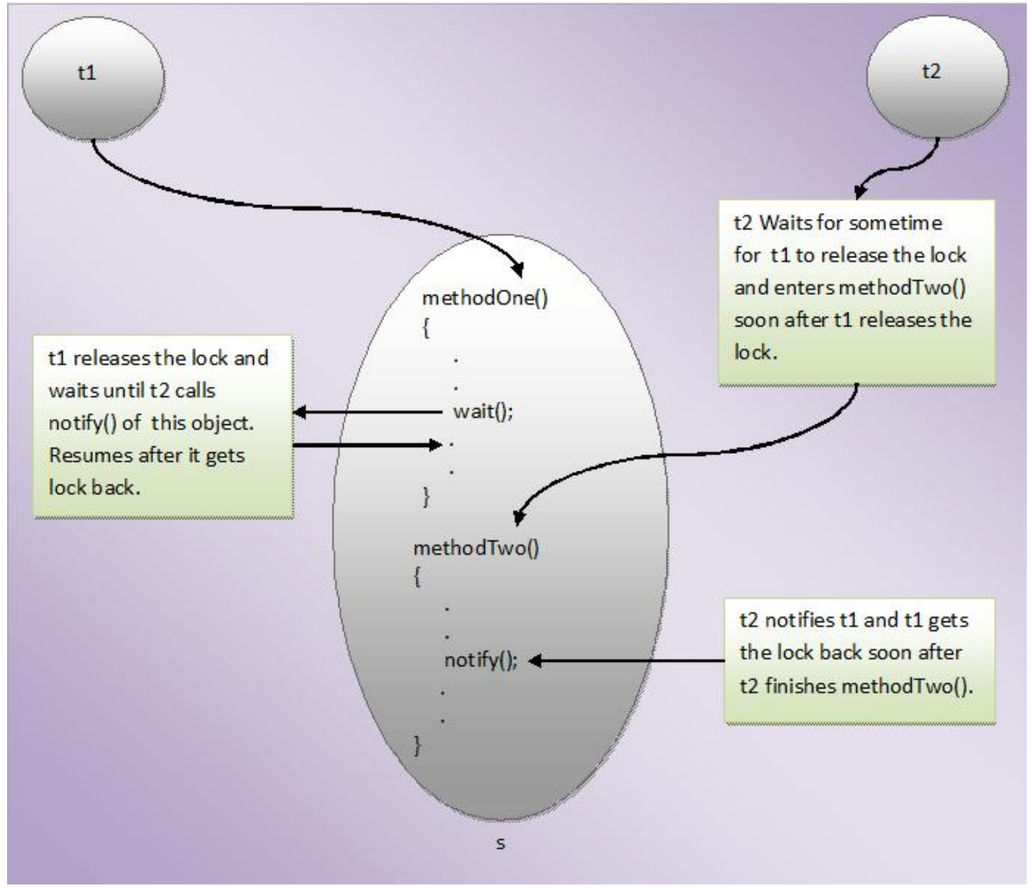
- Avoid unnecessary execution
- Improve efficiency
- Maintain proper coordination



3. Methods Used in Thread Communication

These methods belong to the Object **class**:

Method	Description
wait()	Makes thread wait
notify()	Wakes up one waiting thread
notifyAll()	Wakes up all waiting threads



4. How These Methods Work

✓ wait()

- Causes the current thread to **pause execution**
- Releases the lock on the object
- Moves thread to **waiting state**

```
wait();
```

✓ notify()

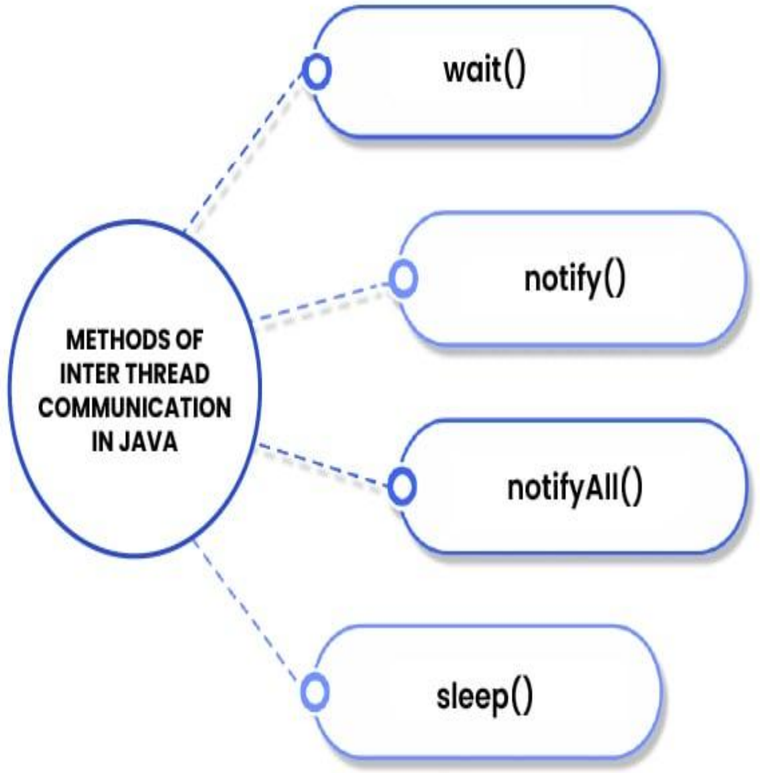
- Wakes up **one** thread waiting on the object

```
notify();
```

✓ notifyAll()

- Wakes up **all waiting threads**

```
notifyAll();
```



5. Important Rules

✓ Must be used inside **synchronized method/block**

✓ Always use on **shared object**

✓ Use **while loop** instead of if

✓ Handle InterruptedException

6. Example: Producer–Consumer Problem

- One thread produces data, another consumes it.

Program:

```
class Shared {
    int data;
    boolean available = false;
    synchronized void produce(int value)
throws Exception {
        while (available) {
            wait(); // wait if data already exists
        }
        data = value;
        available = true;
        System.out.println("Produced: " + data);
        notify(); // notify consumer
    }
    synchronized void consume()
throws Exception {
        while (!available) {
            wait(); // wait if no data
        }
        System.out.println("Consumed:
" + data);
        available = false;
        notify(); // notify producer
    }
}
class Producer extends Thread {
    Shared s;
    Producer(Shared s) {
```

```
    this.s = s;
    }
    public void run() {
        try {
            for (int i = 1; i <= 5; i++) {
                s.produce(i);
            }
        } catch (Exception e) {}
    }
}
class Consumer extends Thread {
    Shared s;
    Consumer(Shared s) {
        this.s = s;
    }

    public void run() {
        try {
            for (int i = 1; i <= 5; i++) {
                s.consume();
            }
        } catch (Exception e) {}
    }
}
public class Main {
    public static void main(String[] args)
{
        Shared s = new Shared();
        Producer p = new Producer(s);
        Consumer c = new Consumer(s);
        p.start();
        c.start();
    }
}
```

Sample Output:

Produced: 1

Consumed: 1

Produced: 2

Consumed: 2

...

- Threads run in a coordinated way

7. Explanation of Flow

1.Producer produces data

2.Consumer waits until data is available

3.Producer calls notify()

4.Consumer wakes and consumes

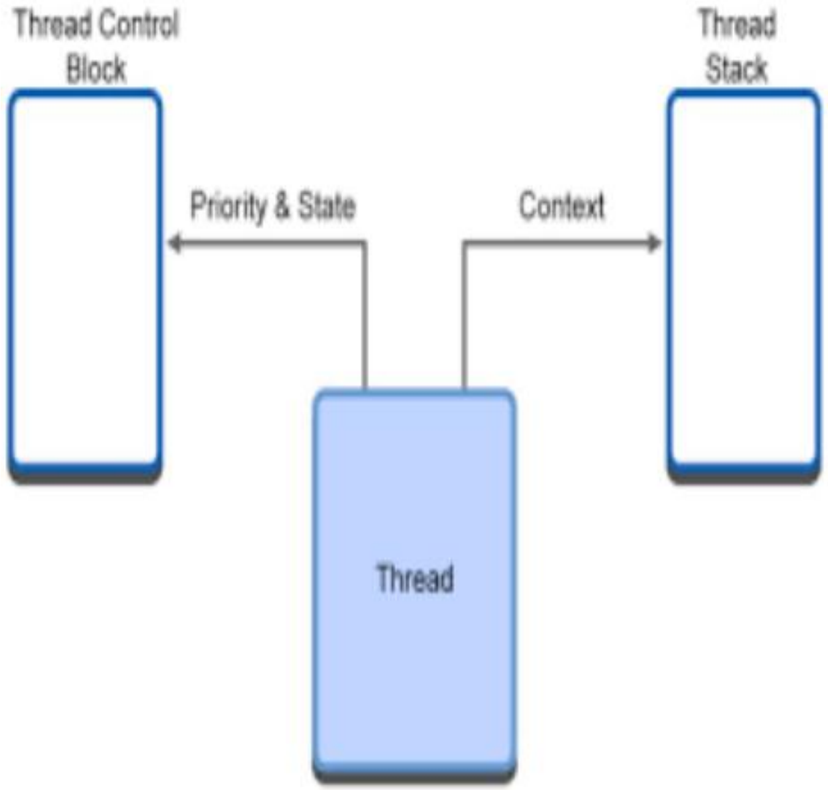
5.Consumer calls notify()

6.Producer resumes

8. Without Thread Communication (Problem)

```
while(true) {
    if(condition) {
        // do something
    }
}
```

✕ This causes **busy waiting** (wastes CPU)



9. Advantages of Thread Communication

✓ Efficient CPU usage

✓ Proper coordination

✓ Avoids busy waiting

✓ Improves performance

10. Disadvantages

✗ Complex to implement

✗ Risk of deadlock

✗ Hard to debug

11. Real-Life Example

Restaurant system:

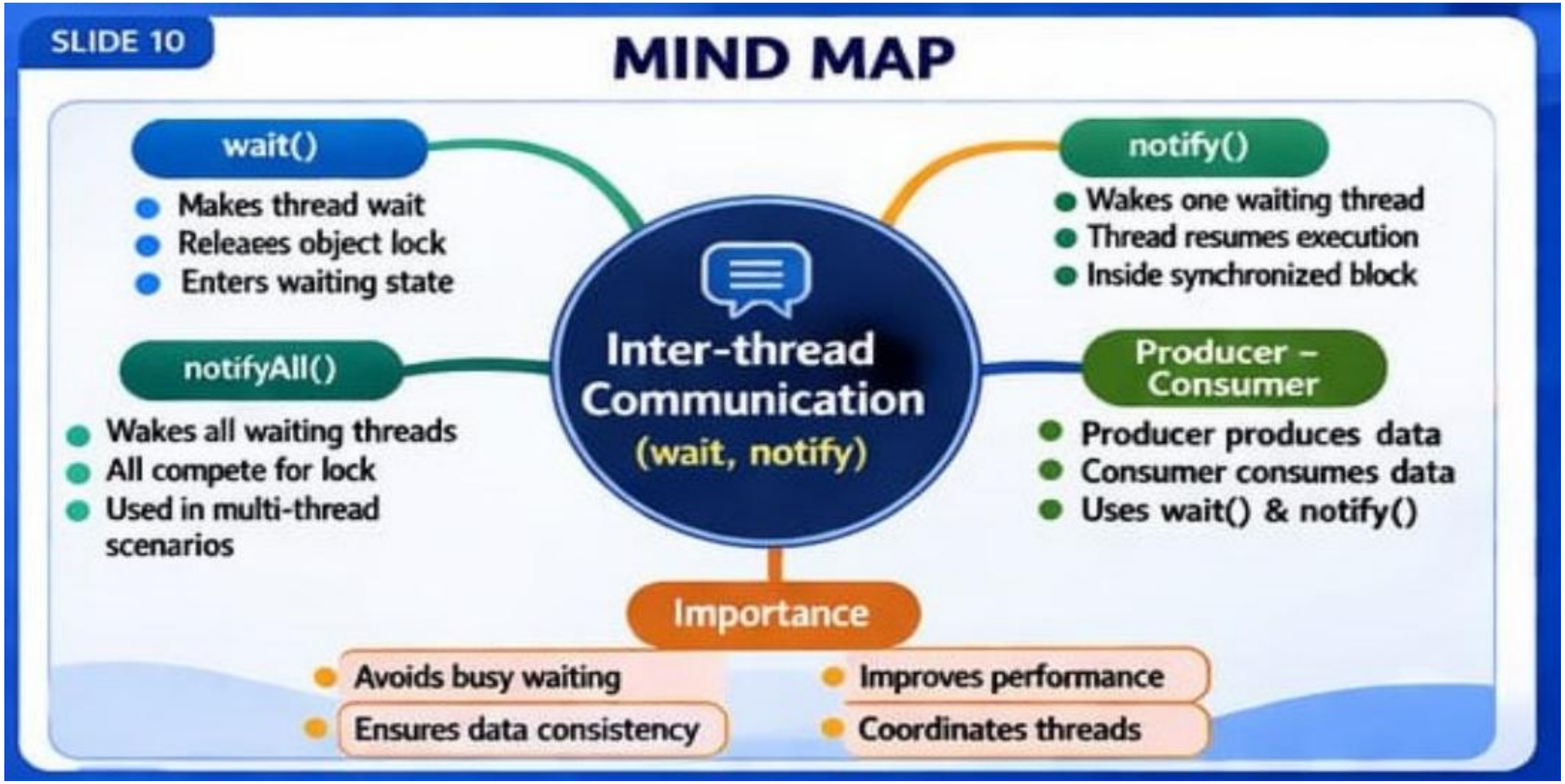
- Chef (Producer) prepares food
- Waiter (Consumer) serves food
- Waiter waits if food not ready
- Chef waaits if no orders

12. Key Exam Points

- Uses wait(), notify(), notifyAll()
- Must be inside **synchronized block**
- Avoid if, use **while loop**
- Prevents **busy waiting**
- Used in **Producer-Consumer problem**

13. Summary

- Thread communication = coordination between threads
- Uses Object class methods
- Works with synchronization
- Essential for efficient multithreading


SLIDE 10
MIND MAP
wait()
Makes thread wait
Releaees object lock
Enters waiting state
notifyAll()
Wakes all waiting threads
All compete for lock
Used in multi-thread scenarios
Inter-thread Communication (wait, notify)
notify()
Wakes one waiting thread
Thread resumes execution
Inside synchronized block
Producer – Consumer
Producer produces data
Consumer consumes data
Uses wait() & notify()
Importance
Avoids busy waiting
Ensures data consistency
Improves performance
Coordinates threads

Assessment

1. Which class contains the methods used for inter-thread communication?

a) Thread

b) Object

c) Runnable

d) System

2. What is the effect of calling wait() on an object?

a) The thread terminates immediately.

b) The thread pauses but keeps the lock on the object.

c) The thread pauses and releases the lock on the object.

d) The thread pauses for a specified time and then resumes.

Assessment

3. Which method should be used to wake up all threads currently waiting on an object's monitor?

a) notify()

b) wakeAll()

c) resume()

d) notifyAll()

4. Why must wait(), notify(), and notifyAll() be called from a synchronized context?

a) To improve performance.

b) To ensure the thread holds the monitor lock for the object.

c) Because they are static methods.

d) To prevent the thread from terminating prematurely.

THANK YOU