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

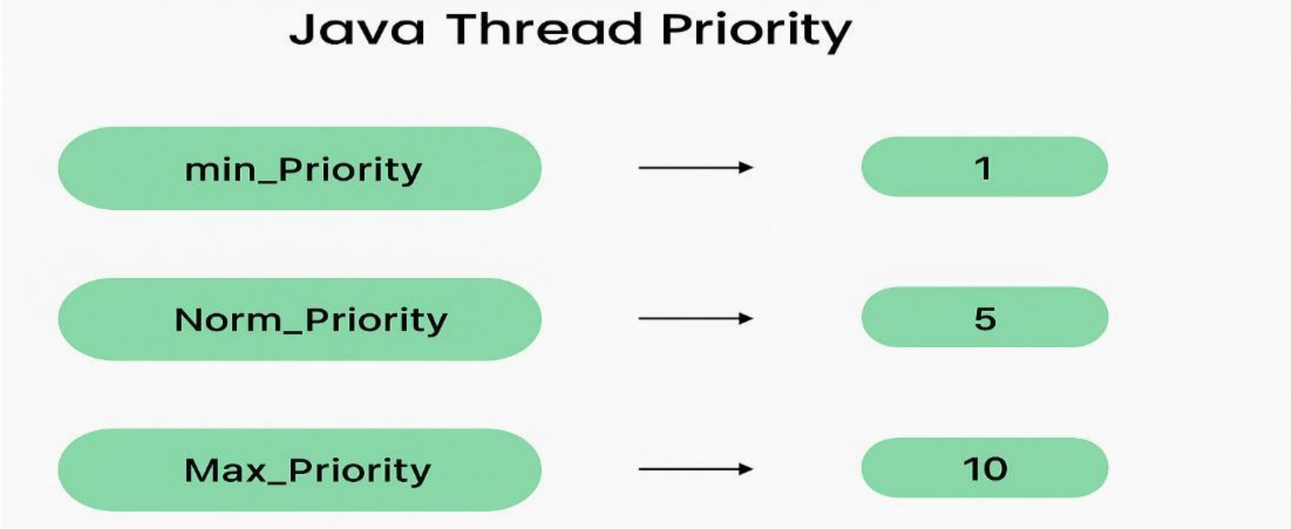
TOPIC 7 - Thread control and priorities

1. Thread Control in Java

- **Thread control** refers to methods used to **manage the execution of threads**.

These methods help to:

- Start, pause, resume, and stop threads
- Control execution order
- Coordinate between threads



2. Important Thread Control Methods

✅ 1. start()

- Starts a new thread
- Internally calls the run() method

```
Thread t = new Thread();
t.start();
```

✅ 2. run()

- Contains the code executed by the thread

```
public void run() {
    System.out.println("Thread running");
}
```

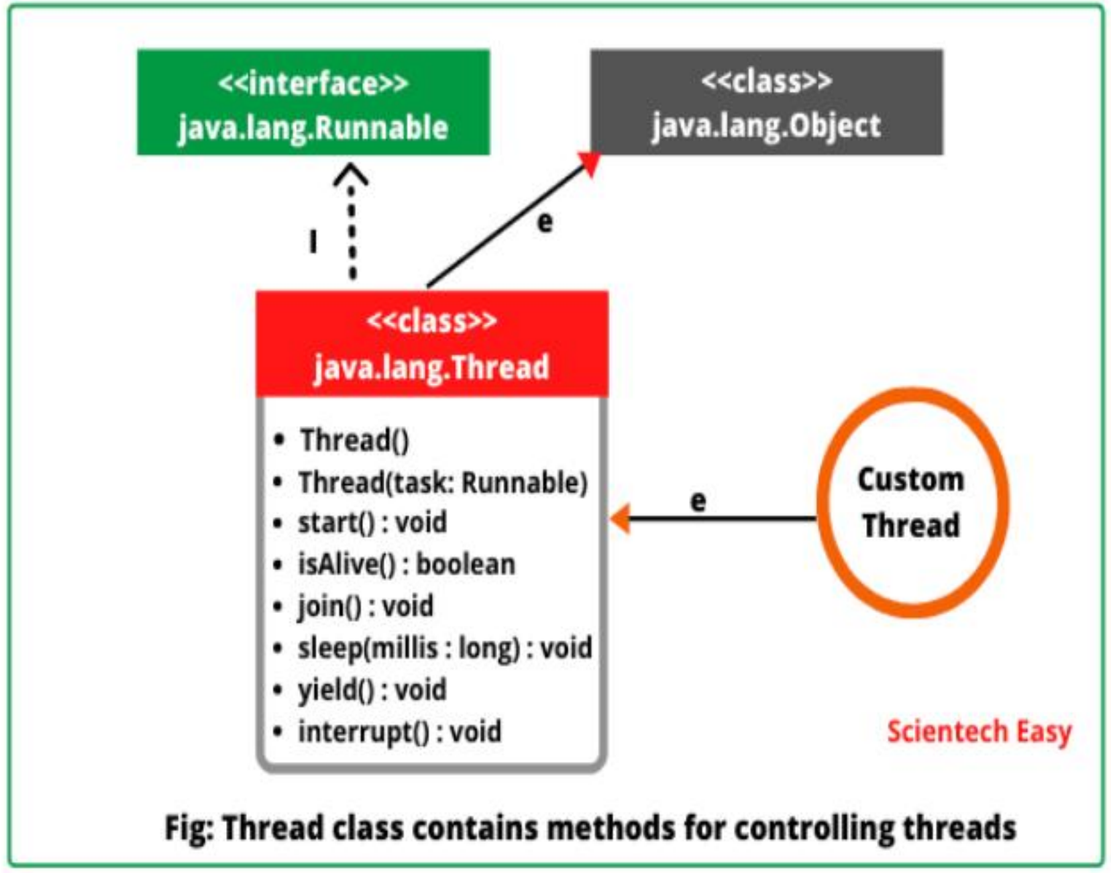


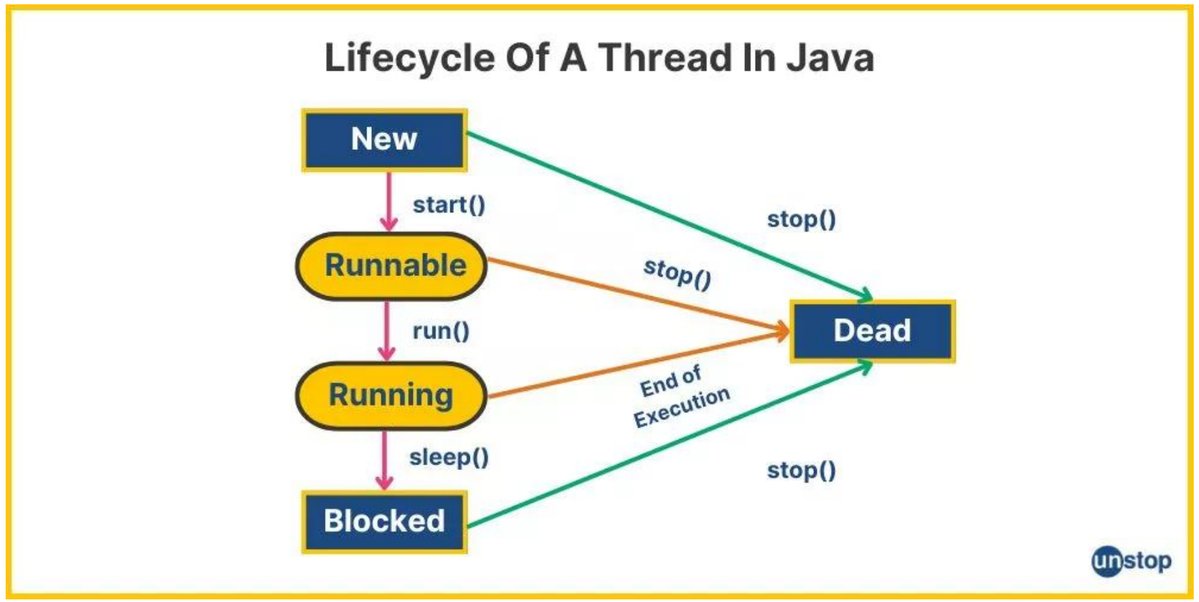
Fig: Thread class contains methods for controlling threads

✅ 3. sleep()

- Pauses the thread for a specified time (in milliseconds)

```
try {
    Thread.sleep(1000); // 1 second
} catch (InterruptedException e) {}
```

👉Used for delays, animations, etc.



✅ 4. join()

- Makes one thread **wait** until another thread finishes

```
t1.join(); // main thread waits for t1
```

Example:

```
class Test extends Thread {
    public void run() {
        for(int i=1; i<=3; i++) {
            System.out.println(i);
        }
    }
    public static void main(String[] args)
throws Exception {
        Test t1 = new Test();
        Test t2 = new Test();
        t1.start();
        t1.join();  // wait for t1 to finish
        t2.start();
    }
}
```

✅ 5. yield()

- Causes the current thread to **pause temporarily**
- Gives chance to other threads of same priority

```
Thread.yield();
```

✅ 6. isAlive()

- Checks whether a thread is still running

```
System.out.println(t1.isAlive());
```

✅ 7. interrupt()

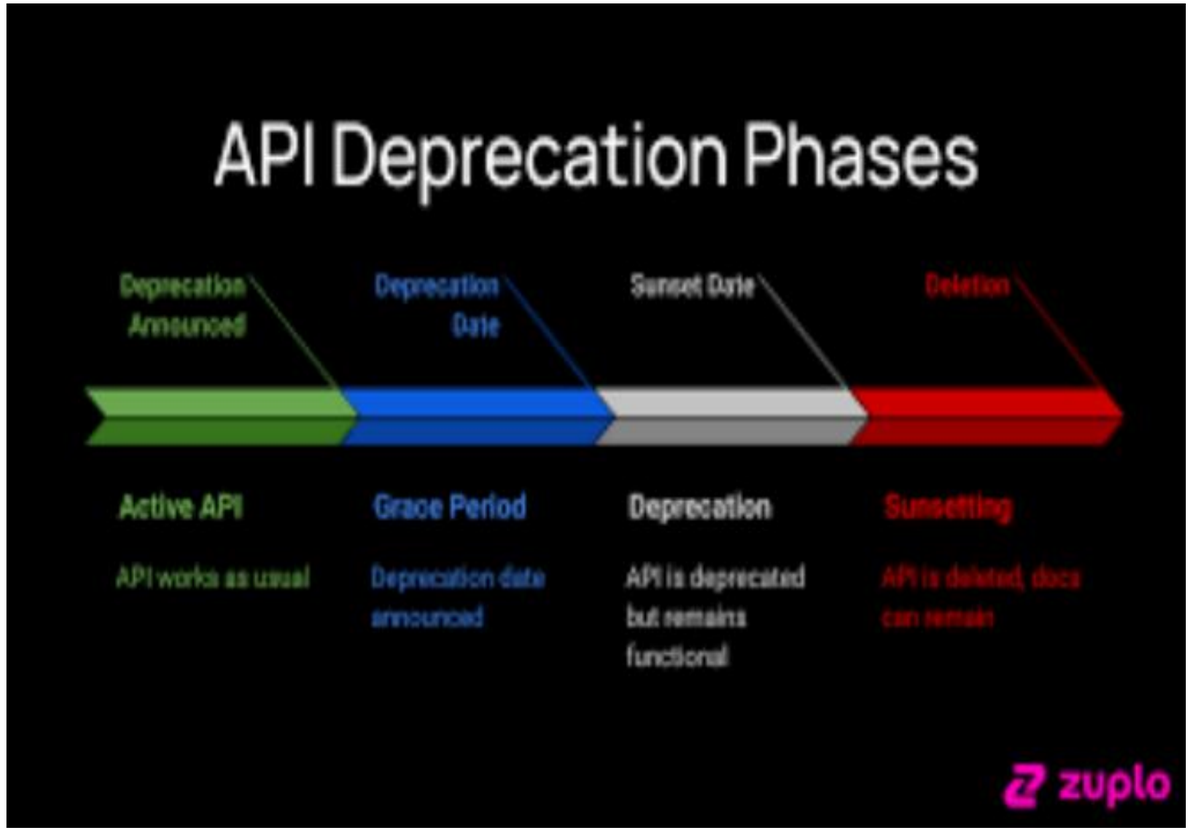
- Interrupts a sleeping or waiting thread

t1.interrupt();

❌ Deprecated Methods (Avoid using)

- stop()
- suspend()
- resume()

These are unsafe and removed in modern Java



3. Thread Priorities in Java

Every thread has a **priority** that helps the **thread scheduler** decide execution order.

Priority Range

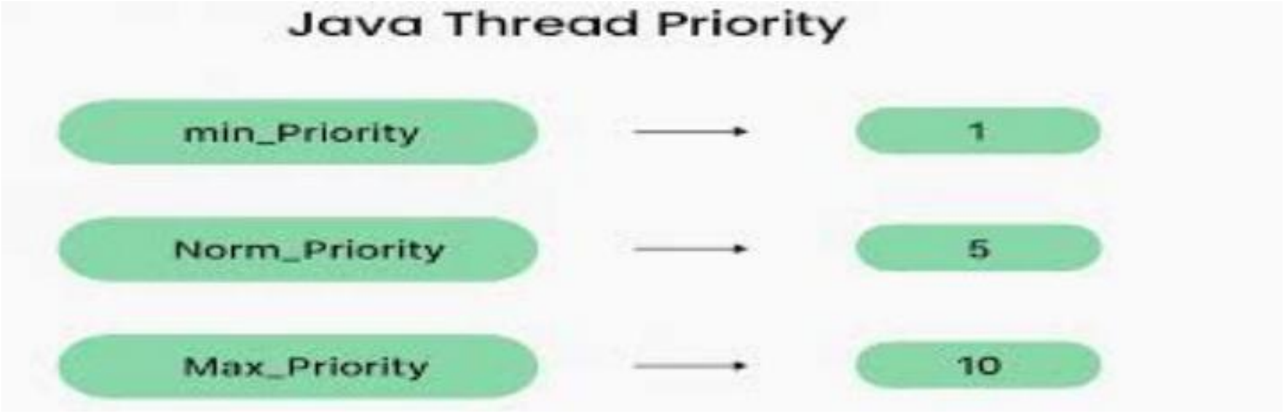
Constant	Value
MIN_PRIORITY	1
NORM_PRIORITY	5 (default)
MAX_PRIORITY	10

Setting Priority

```
t1.setPriority(Thread.MAX_PRIORITY);
t2.setPriority(Thread.MIN_PRIORITY);
```

Getting Priority

```
System.out.println(t1.getPriority());
```



4. Example of Thread Priority

```
class MyThread extends Thread {
    public void run() {
        System.out.println(Thread.currentThread().getName());
    }
    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        MyThread t2 = new MyThread();
        t1.setPriority(Thread.MIN_PRIORITY);
        t2.setPriority(Thread.MAX_PRIORITY);
        t1.start();
        t2.start();
    }
}
```

Output may vary depending on scheduler

5. Important Points About Priorities

✓ Higher priority → More chance to execute first

✓ Not guaranteed (depends on OS scheduler)

✓ Default priority = 5

✓ Threads inherit priority from parent thread

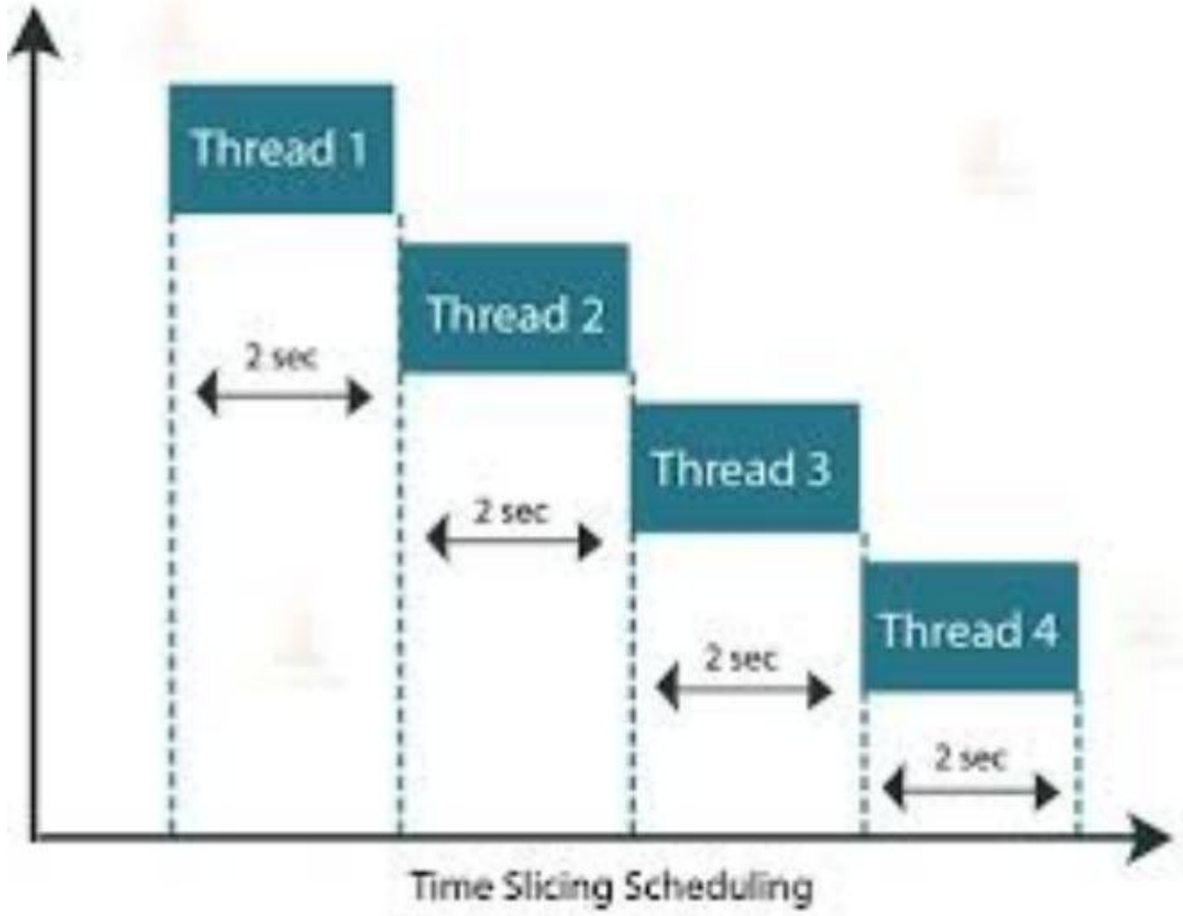
6. Thread Scheduler

The **thread scheduler** decides:

- Which thread runs
- When it runs

It uses:

- Priority
- Time slicing



7. Example Combining Control + Priority

```
class Example extends Thread {
    public void run() {
        for(int i=1; i<=3; i++) {
            System.out.println(Thread.currentThread().getName());
            try {
                Thread.sleep(500);
            } catch(Exception e) {}
        }
    }
            public static void main(String[] args) {
                    Example t1 = new Example();
                    Example t2 = new Example();
                    t1.setPriority(1);
                    t2.setPriority(10);
                    t1.start();
                    t2.start();
                }
            }
```

8. Advantages of Thread Control

✓ Better coordination

✓ Avoids conflicts

✓ Improves performance

✓ Helps synchronization

9. Disadvantages

✗ Complex to manage

✗ May cause deadlock

✗ Hard to debug

10. Key Exam Points

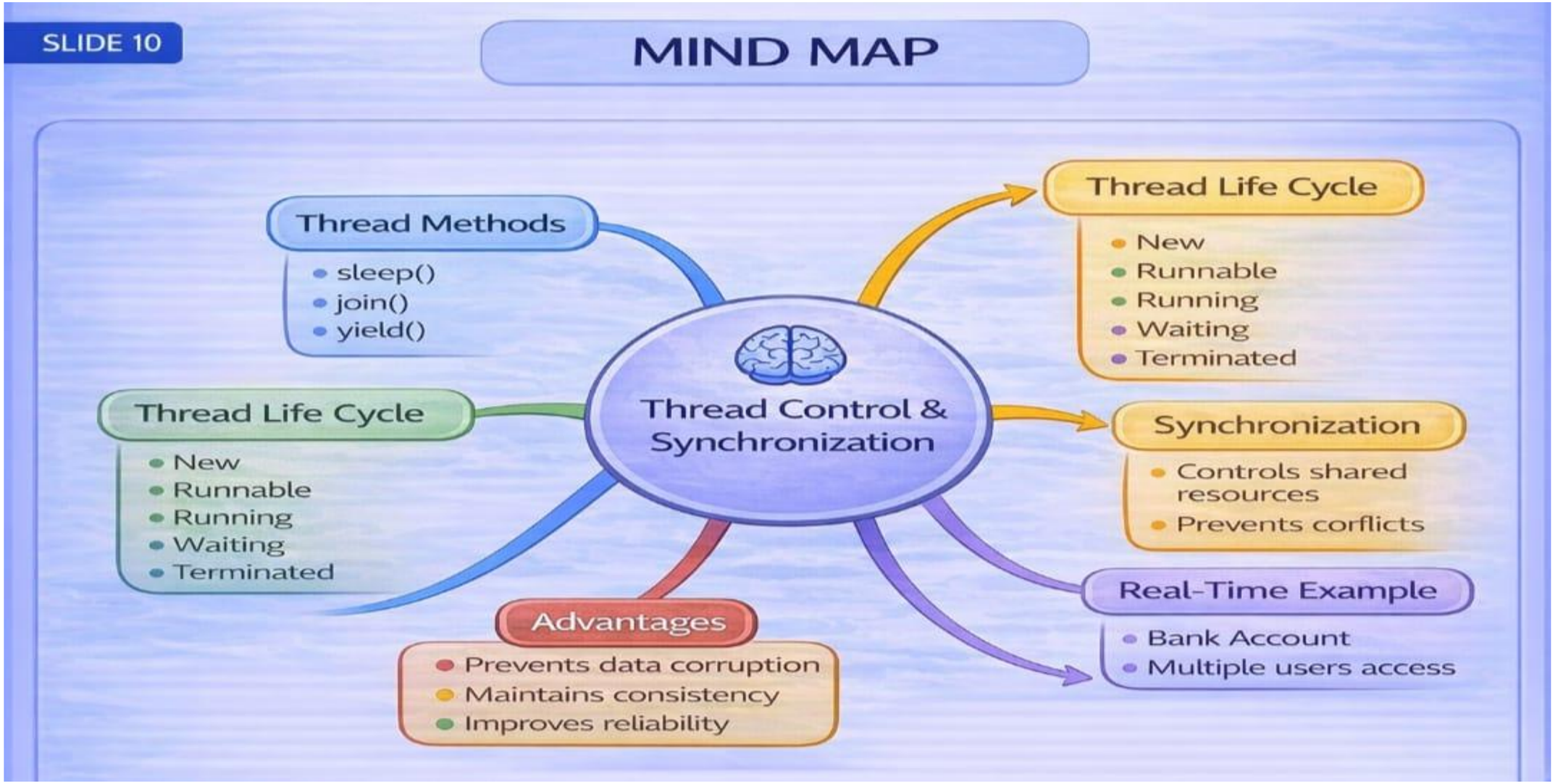
- sleep() pauses thread
- join() waits for another thread
- yield() gives chance to others
- isAlive() checks thread status
- Priority range = **1 to 10**
- Default priority = **5**

11. Summary

- Thread control methods manage execution
- Priorities influence scheduling
- Scheduler decides actual execution
- Proper use improves performance

SLIDE 10

MIND MAP



Assessment

1. What are the tasks of the start() method?
2. What is the difference between start() and run()?
3. What is the difference between sleep() and yield()?
4. What is the role of join()?

THANK YOU