🚀 Java Multithreading – Well-Structured Notes

🔹 1. What is a Thread in Java?

📘 Definition

A thread is a lightweight sub-process. It is the smallest unit of execution within a process and shares memory with other threads of the same process.

📝 Short Answer

A thread is a small, independent task within a program.

💻 Example

```java
public class MyThread extends Thread {
    public void run() {
        System.out.println("Thread is running...");
    }
    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        t1.start();  // starts the thread
    }
}
```

🔹 2. How to create a thread in Java?

📘 Definition

There are two ways:

1. By extending the `Thread` class.

2. By implementing the `Runnable` interface.

📝 Short Answer

Use Thread or Runnable.

💻 Example (Runnable)

```java
public class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Runnable thread running...");
    }

    public static void main(String[] args) {
        Thread t = new Thread(new MyRunnable());
        t.start();
    }
}
```

🔹 3. Difference between `start()` and `run()`

📘 Definition

`start()` creates a new thread and calls `run()`.
Calling `run()` directly runs it like a normal method in the main thread.

📝 Short Answer

`start()` → new thread | `run()` → current thread

💻 Example

```java
public class Test extends Thread {
    public void run() {
```

```
        System.out.println("Running in: " +
Thread.currentThread().getName());
    }
    public static void main(String[] args) {
        Test t = new Test();
        t.run();   // current thread
        t.start(); // new thread
    }
}
```

🔹 4. Thread Lifecycle

📘 Definition

States: New → Runnable → Running → Blocked/Waiting → Terminated

📝 Short Answer

New → Ready → Running → Dead

💻 Example

Java

```
Thread t = new Thread(() -> System.out.println("Hi"));
System.out.println(t.getState()); // NEW
t.start();
System.out.println(t.getState()); // RUNNABLE or TERMINATED
```

🔹 5. Difference between `sleep()` and `wait()`

📘 Definition

`sleep()` pauses thread execution (Thread class).
`wait()` releases lock and waits (Object class).

📝 Short Answer

`sleep()` → no lock release | `wait()` → releases lock

💻 Example

```java
Thread.sleep(1000);
```

```java
lock.wait();
lock.notify();
```

Output:

```
WaitingThread: Waiting...
NotifyingThread: Sending notification...
WaitingThread: Got notified!
```

🔹 6. Thread Synchronization

📘 Definition

Controls access to shared resources to avoid thread interference.

📝 Short Answer

Prevents multiple threads from accessing shared data at the same time.

💻 Example

```java
public synchronized void print() {
    System.out.println("Synchronized method");
```

```
}
```

🔹 7. Daemon Thread

📘 Definition

Runs in background and stops when user threads finish.

📝 Short Answer

Background thread.

💻 Example

```java
Thread t = new Thread(() -> System.out.println("Daemon"));
t.setDaemon(true);
t.start();
```

🔹 8. Thread Priority

📘 Definition

Determines scheduling order.

📝 Short Answer

Higher priority → higher chance.

💻 Example

```java
Thread t = new Thread();
t.setPriority(Thread.MAX_PRIORITY);
```

🔹 9. Deadlock

📘 Definition

Two threads waiting on each other's locks.

📝 Short Answer

Threads wait forever.

💻 Example

```java
synchronized(obj1) {
   synchronized(obj2) {
   }
}
```

🔹 10. `wait()` vs `notify()` vs `notifyAll()`

📘 Definition

- `wait()` → makes thread wait
- `notify()` → wakes one
- `notifyAll()` → wakes all

📝 Short Answer

Thread communication methods.

💻 Example

```java
synchronized(obj) {
    obj.wait();
```

```
        obj.notify();
    }
```

🔹 11. volatile Keyword

📘 Definition

Ensures visibility of variable updates across threads.

📝 Short Answer

Always read latest value.

💻 Example

Java

```
volatile boolean flag = true;
```

🔹 12. join() Method

📘 Definition

Makes one thread wait for another to finish.

📝 Short Answer

Wait till thread completes.

💻 Example

Java

```
t1.join();
```

🔹 13. yield() Method

📘 Definition

Gives execution chance to other threads.

📝 Short Answer

Pause current thread.

💻 Example

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

🔹 14. Runnable vs Callable

📘 Definition

Runnable → no return
Callable → returns value

📝 Short Answer

Callable returns result.

💻 Example

```java
Callable<Integer> task = () -> 42;
```

🔹 15. ExecutorService

📘 Definition

Framework for managing async tasks.

📝 Short Answer

Thread pool manager.

💻 Example

```java
ExecutorService es = Executors.newFixedThreadPool(2);
es.submit(() -> System.out.println("Running"));
```

🔹 16. Thread Pool

📘 Definition

Reusable worker threads.

📝 Short Answer

Pre-created threads.

💻 Example

```java
ExecutorService pool = Executors.newCachedThreadPool();
```

🔹 17. Concurrency vs Parallelism

📘 Definition

Concurrency → multiple tasks
Parallelism → simultaneous execution

📝 Short Answer

Concurrency ≠ Parallelism

🔹 18. Race Condition

📘 Definition

Multiple threads modifying shared data.

📝 Short Answer

Unpredictable result.

💻 Example

```java
count++; // race condition
```

🔹 19. Avoid Race Condition

📝 Answer

Use synchronization or locks.

💻 Example

```java
synchronized(this) {
    count++;
}
```

🔹 20. Thread-Safe Classes

📝 Answer

`Vector`, `Hashtable`, `StringBuffer`
