

NEW state

```
Thread thread = new Thread();
```

NEW → RUNNABLE

```
thread.start();
```

RUNNABLE → WAITING

Object.wait(); // wait() is not a static method. This is an instance method. This method comes from the Object class.

// Goes into WAITING for an infinite time until notify() or notifyAll() is called by another thread on the Monitor object.

```
thread.join();
```

// thread that calls the **join()** goes into WAITING state until the thread on which the join() is called dies out.

WAITING → RUNNABLE

WAITING_wait → RUNNABLE

// **notify()** or **notifyAll()** is called by another thread on the monitor object on which the thread went into the WAITING state.

WAITING_join → RUNNABLE

// thread on which the join() called completes the run() (thread dies out)

RUNNABLE → BLOCK

// failed to acquire the lock

Ex:- In the Mother – Child – Plate Producer Consumer example, if the Mother calls putFood() , as a result she calls the lock.lock(). Therefore, she acquires the lock on the plate.

While the Mother is holding the lock on the plate if Child tries to call the getFood() child also will call lock.lock() . Since Mother already has a lock on the plate, Child call to lock() will fail.

BLOCKED → RUNNABLE

//When the lock is released, the thread that is BLOCKED will come to the RUNNABLE state.

Ex: When the Mother completes the putFood(), as the last statement she calls lock.unlock() . When the unlock() is called, Mother releases the lock. Now the Child, which was blocked can come out of the BLOCKED state .

RUNNABLE → TERMINATED

// when the run() of the thread completes, running goes into TERMINATED state.

RUNNABLE → TIMED_WAITING

1. RUNNABLE → TIMED_WAITING_sleep(ms)

This happens when you call the sleep method

Thread.sleep(ms);

2. RUNNABLE → TIMED_WAITING_wait(ms)

This happens when the wait(ms) is called.

Object.wait(ms)

3. RUNNABLE → TIMED_WAITING_join(ms)

This happens when the join(ms) is called.

Thread.join(ms);

TIMED_WAITING → RUNNABLE

1. TIMED_WAITING_sleep(ms) → RUNNABLE

Timeout – Once the time has elapsed, the Thread will come back to the RUNNABLE state to TIMED_WAITING state.

2. TIMED_WAITING_wait(ms) → RUNNABLE

Either Timeout or notify() / notifyAll() is called.

Whichever happens first.

3. TIMED_WAITING_join(ms) → RUNNABLE

Either Timeout or the Thread on which the join() is called completes the execution (dies out).

Whichever happens first.