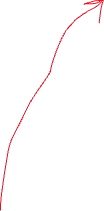
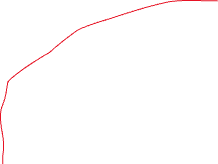
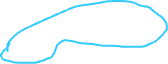
Thread life cycle

A diagram of a computer program

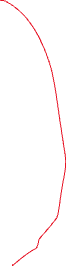
Description automatically generated



NEW state

new Thread(…) when you create an instance of a Thread it enters a NEW state

Thread thread1 = new Thread(t1, "My Thread 01");



When you call the start() method a thread instance it will enter into RUNNABLE state

thread1.start();

when the current thread on which sleep(ms) method is called it will go into TIMED\_WAITING state for so many ms milliseconds



Thread.*sleep*(i); // Currently running Thread will be put on TIMED\_WAITING state



If the thread failed to acquire the lock or blocked for I/O it will go into BLOCKED state



thread1.join()



Puts the other threads into WAITING state until its execution is over and once it completes the execution other threads can start the execution

My Thread 01 is current status NEW

My Thread 01 current value of i is 0

My Thread 01 current value of i is 1

My Thread 01 current value of i is 2

My Thread 01 current value of i is 3

My Thread 01 current value of i is 4

My Thread 01 current value of i is 5

My Thread 01 current value of i is 6

My Thread 01 current value of i is 7

My Thread 01 current value of i is 8

My Thread 01 current value of i is 9

My Thread 01 current value of i is 10

My Thread 01 current value of i is 11

My Thread 01 current value of i is 12

My Thread 01 current value of i is 13

My Thread 01 current value of i is 14

My Thread 01 current value of i is 15

My Thread 01 current value of i is 16

My Thread 01 current value of i is 17

My Thread 01 current value of i is 18

My Thread 01 current value of i is 19

My Thread 01 current value of i is 20

My Thread 01 current value of i is 21

My Thread 01 current value of i is 22

My Thread 01 current value of i is 23

My Thread 01 current value of i is 24

My Thread 02 current value of i is 0

My Thread 02 current value of i is 1

My Thread 02 current value of i is 2

My Thread 02 current value of i is 3

My Thread 02 current value of i is 4

My Thread 02 current value of i is 5

My Thread 02 current value of i is 6

My Thread 02 current value of i is 7

My Thread 02 current value of i is 8

My Thread 02 current value of i is 9

My Thread 02 current value of i is 10

My Thread 02 current value of i is 11

My Thread 02 current value of i is 12

My Thread 02 current value of i is 13

My Thread 02 current value of i is 14

My Thread 02 current value of i is 15

My Thread 02 current value of i is 16

My Thread 02 current value of i is 17

My Thread 02 current value of i is 18

My Thread 02 current value of i is 19

My Thread 02 current value of i is 20

My Thread 02 current value of i is 21

My Thread 02 current value of i is 22

My Thread 02 current value of i is 23

My Thread 02 current value of i is 24

My Thread 03 current value of i is 0 => My Thread 03 started so join(10) methods says to main thread which is only other thread that is in the RUNNABLE state to go into TIMED\_WAITING for 10 milliseconds

My Thread 03 current value of i is 1

My Thread 03 current value of i is 2

My Thread 03 current value of i is 3

My Thread 03 current value of i is 4

My Thread 03 current value of i is 5 // 10 millisecond is over so now both main thread and My Thread 03 both comes into RUNNABLE state because of the TIMEOUT though My Thread 03 is still in the RUNNABLE state

My Thread 01 is current status TERMINATED

My Thread 02 is current status TERMINATED

My Thread 03 is current status TIMED\_WAITING

main is current status RUNNABLE

My Thread 03 current value of i is 6

My Thread 03 current value of i is 7

My Thread 03 current value of i is 8

My Thread 03 current value of i is 9

My Thread 03 current value of i is 10

My Thread 03 current value of i is 11

My Thread 03 current value of i is 12

My Thread 03 current value of i is 13

My Thread 03 current value of i is 14

My Thread 03 current value of i is 15

My Thread 03 current value of i is 16

My Thread 03 current value of i is 17

My Thread 03 current value of i is 18

My Thread 03 current value of i is 19

My Thread 03 current value of i is 20

My Thread 03 current value of i is 21

My Thread 03 current value of i is 22

My Thread 03 current value of i is 23

My Thread 03 current value of i is 24

But when I made the TIMEOUT to 1000ms

thread3.join(1000);

My Thread 03 current value of i is 0

My Thread 03 current value of i is 1

My Thread 03 current value of i is 2

My Thread 03 current value of i is 3

My Thread 03 current value of i is 4

My Thread 03 current value of i is 5

My Thread 03 current value of i is 6

My Thread 03 current value of i is 7

My Thread 03 current value of i is 8

My Thread 03 current value of i is 9

My Thread 03 current value of i is 10

My Thread 03 current value of i is 11

My Thread 03 current value of i is 12

My Thread 03 current value of i is 13

My Thread 03 current value of i is 14

My Thread 03 current value of i is 15

My Thread 03 current value of i is 16

My Thread 03 current value of i is 17

My Thread 03 current value of i is 18

My Thread 03 current value of i is 19

My Thread 03 current value of i is 20

My Thread 03 current value of i is 21

My Thread 03 current value of i is 22

My Thread 03 current value of i is 23

My Thread 03 current value of i is 24 // with the My Thread 03 only other thread in the RUNNABLE is main thread and because of 10000/1000ms which is TIMEOUT is so big the main THREAD went into TIMED\_WAITING for 10s/1s but before TIMEOUT it resumed because the currently executed thread which was My Thread 03 died down before TIMEOUT

My Thread 01 is current status TERMINATED

My Thread 02 is current status TERMINATED

My Thread 03 is current status TERMINATED

main is current status RUNNABLE

Exception in thread "Producer Thread" Exception in thread "Consumer Thread" java.lang.IllegalMonitorStateException: current thread is not owner

1. java.lang.IllegalMonitorStateException
2. why I get the above error because the producer thread doesn’t own the shared object
   1. current thread hasn’t acquired the lock
   2. if the current thread has acquired the lock only it can call wait() to enter WAITING state or notify() / notifyAll() method can be called to inform other threads in the waitset on this particular shared object (mailbox obj )

Mailbox obj = new Mailbox();

* 1. wait() / notify() / notifyAll() can only be called within synchronized block because then only producer or consumer can acquire the lock then it owns the object so it can call wait() / notify() / notifyAll()

If you look the out though it doesn’t look in a proper order

Producer produces the value

Producer Thread produced the value 1

Then Consumer consumes the value

Consumer Thread consumes the value 1

There is a inter thread communication and coordination though output not necessarily reflects

But you can clearly see there no double consumption or value loss due to double production one after the other before consumption

Producer Thread produced the value 1 => 1st produces value 01

Producer Thread produced the value 2 => 3rd

Consumer Thread consumes the value 1 => 2nd consumes value 01

Consumer Thread consumes the value 2 => 4th

Producer Thread produced the value 3

Producer Thread produced the value 4

Consumer Thread consumes the value 3

Consumer Thread consumes the value 4

Consumer Thread consumes the value 5

Producer Thread produced the value 5

Producer Thread produced the value 6

Consumer Thread consumes the value 6

Producer Thread produced the value 7

Consumer Thread consumes the value 7

Producer Thread produced the value 8

Consumer Thread consumes the value 8

Producer Thread produced the value 9

Producer Thread produced the value 10

Consumer Thread consumes the value 9

Consumer Thread consumes the value 10

Just demonstrate the current state:

for(int i=0; i < 5; i++) {

System.***out***.println(i+".consumerThread current state is "+consumerThread.getState());

System.***out***.println(i+".producerThread current state is "+producerThread.getState());

}

You can see the producer thread is gone into WAITING state

Consumer thread is because “failed acquire the lock” on the mailbox object therefore BLOCKED

0.consumerThread current state is BLOCKED

0.producerThread current state is WAITING

1.consumerThread current state is BLOCKED

1.producerThread current state is WAITING

2.consumerThread current state is BLOCKED

2.producerThread current state is WAITING

3.consumerThread current state is BLOCKED

3.producerThread current state is WAITING

4.consumerThread current state is BLOCKED

4.producerThread current state is WAITING