Important Methods of Multithreading.

Runnable Interface: -

1. **void run ()**

Thread Class Methods: -

1. **public static void yield ()**
2. **public static void sleep (long millis) throws InterruptedException**
3. **public static void sleep (long millis, int nanos) throws InterruptedException**
4. **public final void join () throws InterruptedException**
5. **public final void join (long millis) throws InterruptedException**
6. **public final void join (long millis, int nanos) throws InterruptedException**
7. **public void run ()**

Object Class Methods: -

1. **public final void wait () throws InterruptedException**
2. **public final void wait (long timeout) throws InterruptedException**
3. **public final void wait (long timeout,int nanos) throws InterruptedException**
4. **public final void notify ()**
5. **public final void notifyAll ()**

Which are methods use for prevent a thread execution?

Answer: - We can prevent a thread execution by using following methods: -

- yield ()
- join ()
- sleep ()

What is the purpose of yield () method?

Answer: -

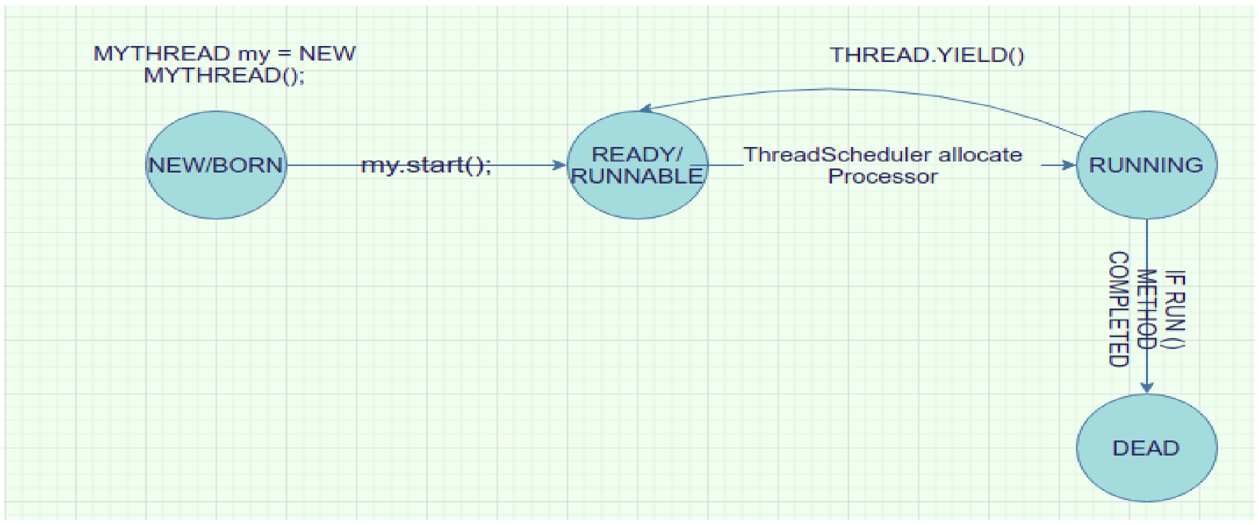
Syntax: - public static **native** void yield ();

A hint to the scheduler that the current thread is **willing to give up its current use of a processor**. The scheduler is free to ignore this hint.

yield () method causes to pass current executing thread to give the chance for waiting thread of same priority. If there is no waiting thread or all waiting have low priority then same thread can continue its execution.

if multiple threads are waiting with same priority, then which waiting thread will get the chance, we can't expect it depends on thread scheduler.

The thread which is yielded, when it will get chance once again it depends on thread scheduler and we can't expect exactly.


MYTHREAD my = NEW MYTHREAD();
THREAD.YIELD()
NEW/BORN
my.start();
READY/ RUNNABLE
ThreadScheduler allocate Processor
RUNNING
IF RUN () METHOD COMPLETED
DEAD

When we go for yield () method?

Answer: - It is rarely appropriate to use this method. It may be useful for **debugging** or **testing purposes**, where it may help to **reproduce bugs due to race conditions.** It may also be useful when designing concurrency control constructs such as the ones in the java.util.concurrent.locks package.

What is the purpose of join () method?

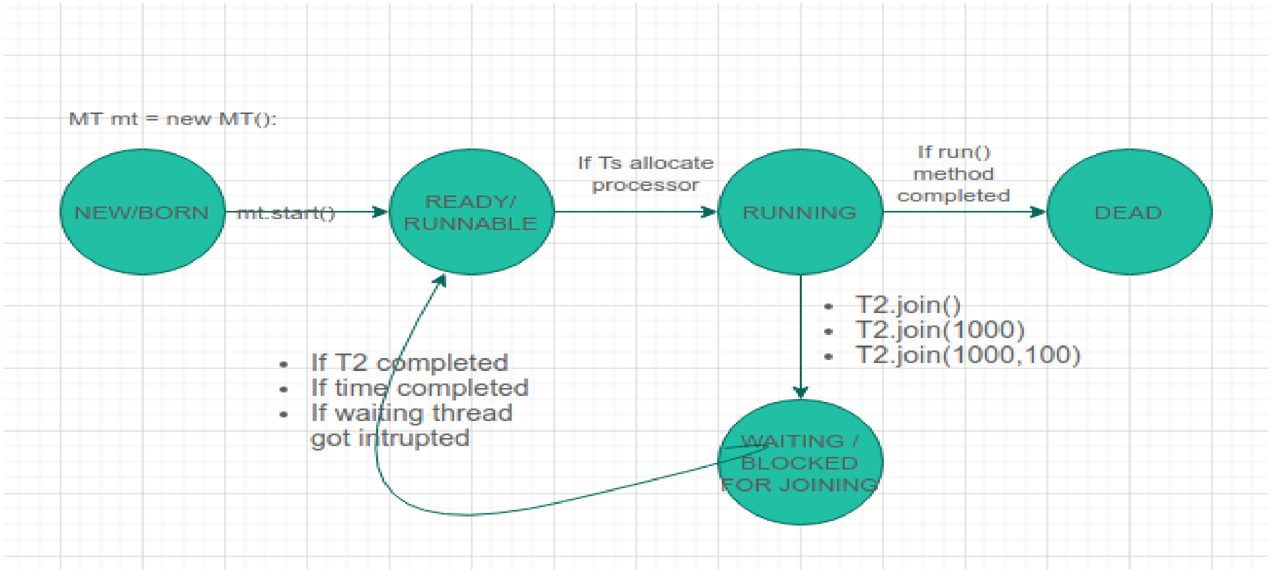
Answer: - If a thread wants to wait until completing some other thread, then we should go for join () method.

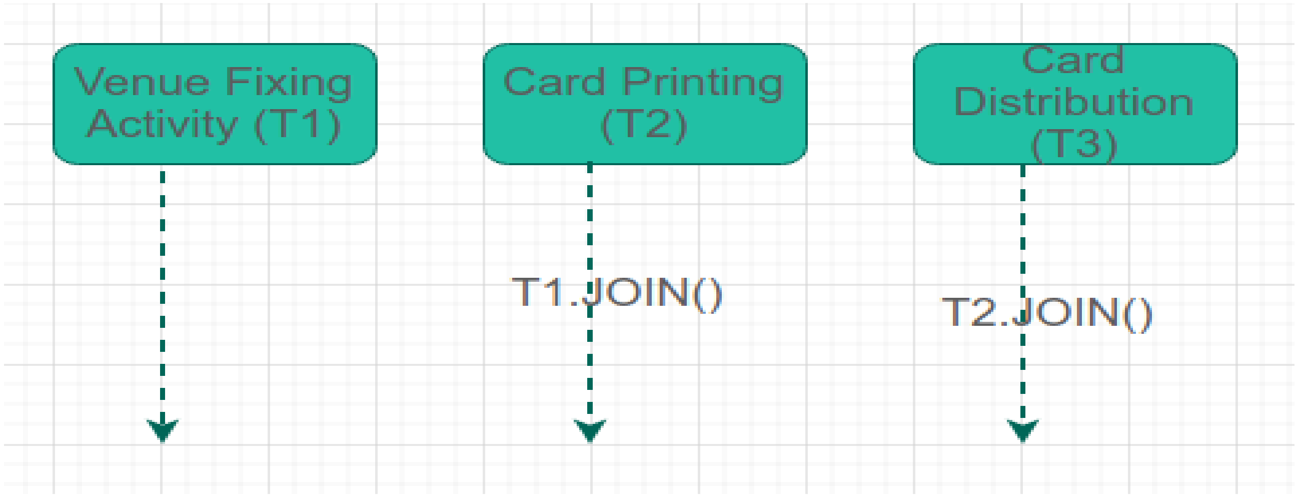
For example, if a thread t1 wants to wait until completing t2 then t1 has to call t2.join (). if t1 execute t2.join () then immediately t1 will be entered into **waiting state** until t2 completes.

Once t2 completes then t1 can continue its execution.

Another example, Wedding card printing thread (T2) has to wait until Venue Fixing thread(T1) completion hence T2 has to call T1.join().

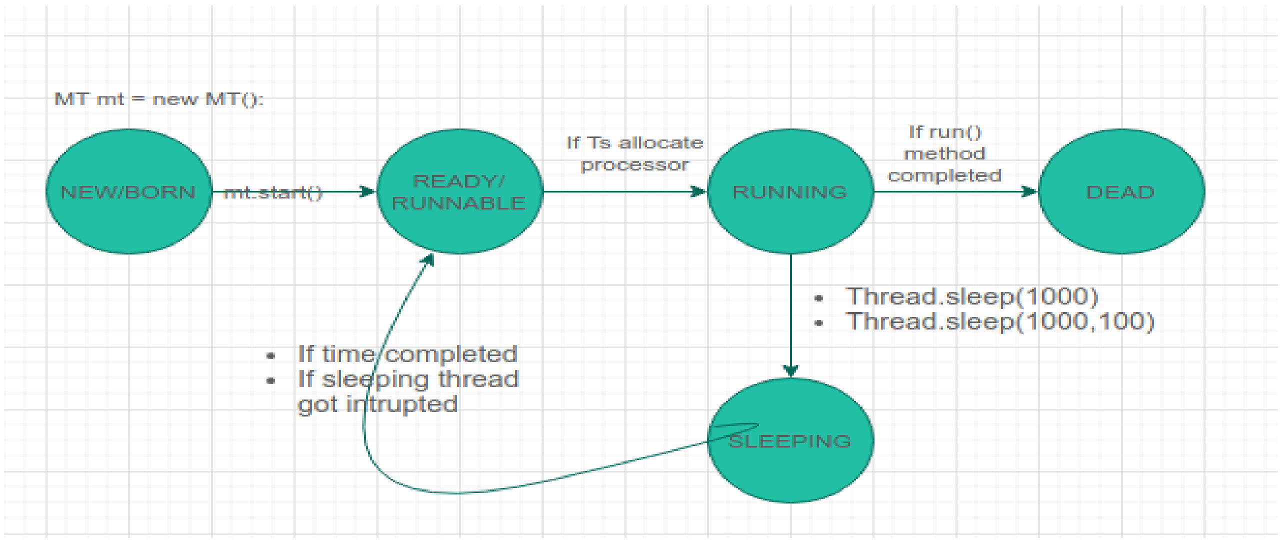
Wedding Card distribution thread (T3) has to wait wedding card printing thread (T2) completion hence T3 has to call T2.join().





What is the purpose of sleep () method?

Answer: - If a thread doesn't want to perform any operation for a particular amount of time, then we should go for sleep () methods. **The thread does not lose ownership of any monitors.**



What is the purpose of interrupt () method?

Answer: - A thread can interrupt a sleeping thread or waiting thread by using interrupt () method of Thread class.

What is the purpose of wait () method?

Answer: - The wait () method causes the current thread to wait indefinitely until another thread either invokes notify () for this object or notifyAll().

What is the purpose of wait (long timeout) method?

Answer: - Using this method, we can specify a timeout after which a thread will be woken up automatically. A thread can be woken up before reaching the timeout using notify () or notifyAll().

Note that calling wait (long timeout)) is the same as calling wait ().

What is the purpose of wait (long timeout,int nanos) method?

Answer: - This is yet another signature providing the same functionality. The only difference here is that we can provide higher precision.

The total timeout period (in nanoseconds) is calculated as 1_000_000*timeout + nanos.

What is the purpose of notify () method?

Answer: - We use the notify () method for waking up threads that are waiting for an access to this object's monitor.

What is the purpose of notifyAll () method?

Answer: - This method simply wakes all threads that are waiting on this object's monitor.

Why Enclose wait () in a while Loop is suggested?

Answer: - Since notify () and notifyAll() randomly wake up threads that are waiting on this object's monitor, it's not always important that the condition is met. Sometimes the thread is woken up, but the condition isn't actually satisfied yet.

We can also define a check to save us from spurious wakeups — where a thread can wake up from waiting without ever having received a notification.

Why wait (), notify () and notifyAll () methods are present in object class but not in thread class?

Answer: - wait (), notify (), notifyall () methods present in object class but not in thread class **because thread can call these methods on any java object**.

Why we have to call wait (), notify () and notifyAll() methods only from synchronized area?

Answer: - To call wait (), notify (), notifyAll () methods on any object, thread should be owner of that object i.e., thread should have lock of that object i.e., thread should be inside synchronized area.

Hence, we can call wait (), notify (), notifyAll () methods only from synchronized area otherwise we will get runtime exception saying **IllegalMonitorStateException**.

Which method release lock?

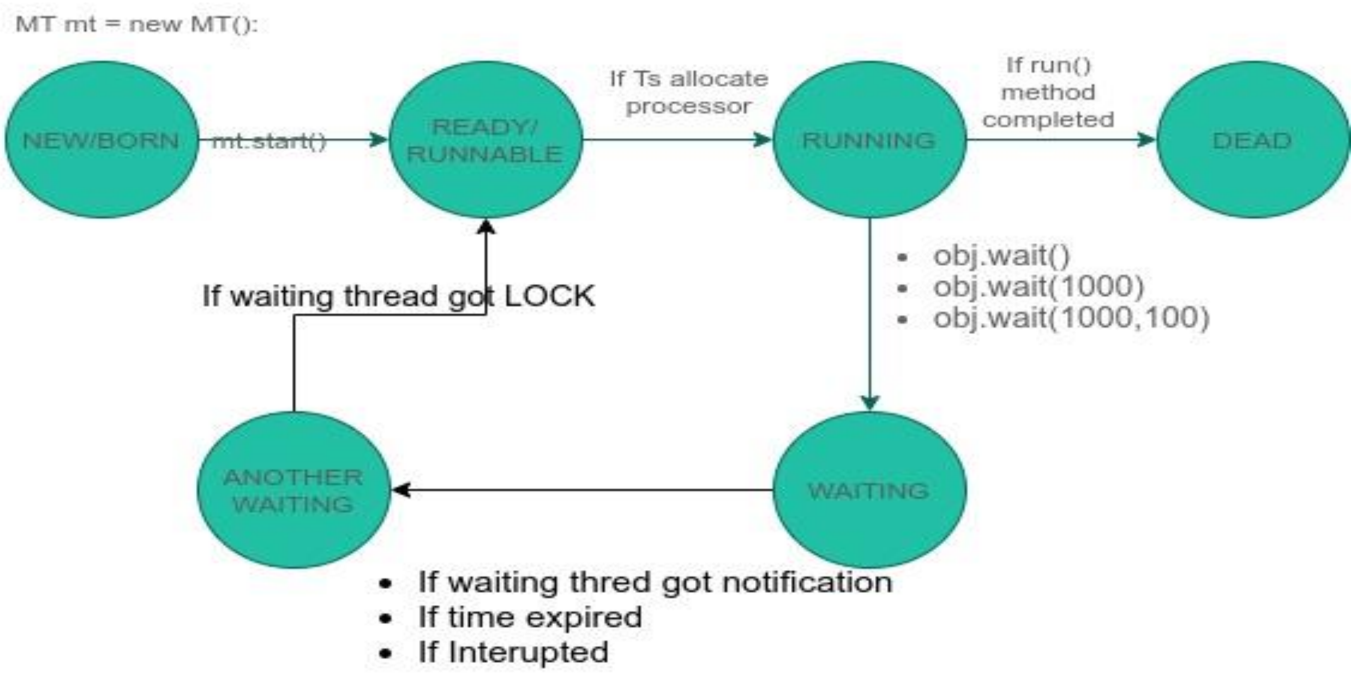
Answer: - Except wait (), notify (), notifyAll() methods there is no other methods where thread release lock.

Release Lock	**Does not Release Lock**
wait ()	sleep ()
notify ()	yield ()
notifyAll ()	join ()

List down the important point of wait (), notify (), notifyAll() methods.

Answer: -

- Two threads can communicate with each other by using wait (), notify () and notifyAll() methods.
- The thread which is expecting updation is responsible to call wait () method then immediately thread will enter into waiting state.
- The thread which is responsible to perform updation, after performing updation it is responsible to call notify method then waiting thread will get that notification and continue its execution with those updated items.
- To call wait (), notify (), notifyAll () methods on any object, thread should be owner of that object i.e., thread should have lock of that object i.e., thread should be inside synchronized area.
- Hence, we can call wait (), notify (), notifyAll () methods only from synchronized area otherwise we will get runtime exception saying **IllegalMonitorStateException**.
- If thread call wait () method on any object it immediately releases lock of that particular object and entered into waiting state.
- If that particular thread contains for example 10 locks and call wait () method on any object it immediately releases lock of that particular object and entered into **waiting state**, so it only releases that particular object lock, i.e., only one lock not all lock, so 9 locks still thread have.
- if a thread call notify () method on any object it releases lock of that object may not immediately. But it will release lock is sure.
- if waiting thread got notification it will come to another waiting state for waiting for lock and if it got lock it will move to ready or runnable state.



Important Differences

What is difference between Runnable.run() and Thread.run() methods?

Answer: -

Runnable.run()	Thread.run()
If this thread was constructed using a separate Runnable run object, then that Runnable object's run method is called; otherwise, this method does nothing and returns. Subclasses of Thread should override this method.	When an object implementing interface **Runnable** is used to create a thread, starting the thread causes the object's run method to be called in that separately executing thread. The general contract of the method run is that it may take any action whatsoever.

What is difference between yield () and sleep ()?

Answer: -

public static void yield ()	**public static void sleep (long millis) throws InterruptedException**
yield () method pauses the currently executing thread temporarily for giving a chance to the remaining waiting threads of the same priority to execute. If there is no waiting thread or all the waiting threads have a lower priority then the same thread will continue its execution. The yielded thread when it will get the chance for execution is decided by the thread scheduler whose **behavior is vendor dependent** (i.e., yield () can only make a heuristic attempt to suspend the execution of the current thread with no guarantee of when will it be scheduled back).	sleep () method **can force** the scheduler to suspend the execution of the current thread for a specified period of time as its parameter. if no other thread or process needs to be run, the CPU will be idle.

What is difference between yield () and join ()?

Answer: -

public static void yield ()	**public final void join () throws InterruptedException**
yield () method pauses the currently executing thread temporarily for giving a chance to the remaining waiting threads of the same priority to execute. If there is no waiting thread or all the waiting threads have a lower priority then the same thread will continue its execution.	If a thread wants to wait until completing some other thread, then we should go for join () method. **For example,** if a thread t1 wants to wait until completing t2 then t1 has to call t2.join (). if t1 execute t2.join () then immediately t1 will be entered into **waiting state** until t2 completes.

The yielded thread when it will get the chance for execution is decided by the thread scheduler whose **behavior is vendor dependent** (i.e., yield () can only make a heuristic attempt to suspend the execution of the current thread with no guarantee of when will it be scheduled back).	Once t2 completes then t1 can continue its execution.

Comparison Between yield (), join () and sleep () Method

Property	Yield ()	Join ()	Sleep ()
Purpose	To pause the current executing thread for giving the chance of remaining threads of same priority.	If a thread wants to wait until completing thread some other threads, then we should go for join.	If a thread doesn't want to perform any operation for a particular amount of time, then we should go for sleep () method.
Is it static?	yes	no	yes
Is it final?	No	yes	no
Is it overloaded?	no	yes	yes
Is it throws Interrupted Exception?	no	yes	yes
Is its native method	yes	no	These are variety here. Sleep (long ms):- native Sleep (long ms, int ns) : non-native

What is difference between yield () and wait ()?

Answer: -

public static void yield ()	**public final void wait () throws InterruptedException**

Yield is declared on java.lang.Thread class	wait () is declared in java.lang.Object class
yield is not overloaded.	wait is an overloaded method and has two versions of wait, normal and timed wait.
yield is a static method and works on the current thread	wait is an instance method
yield is better to be called outside of the loop	It's advised to call the wait method inside the loop
There is no such requirement for the Yield method.	wait () method must be called from either synchronized block or synchronized method,
	Use wait () for inter-thread communication while yield is not just reliable enough even for the mentioned task. prefer Thread.sleep(1) instead of yield.
	When a Thread call waits it releases the monitor.
	With *wait ()* it is also possible to wake the thread anytime through an invocation of *notify ()* or *notifyAll()* on the concerned lock object

What is difference between sleep () and wait ()?

Answer: -

public final void wait () throws InterruptedException	**public static void sleep (long millis) throws InterruptedException**
The main *difference between wait and sleep is that wait ()* method **releases the acquired monitor** when the thread is waiting.	while `Thread.sleep()` method **keeps the lock** or monitor even if the thread is waiting.
wait is called from synchronized context only.	while sleep can be called without synchronized block.

waiting thread can be awake by calling notify and notifyAll	while sleeping thread cannot be awakened by calling notify method.
wait is normally done on the condition; Thread waits until a condition is true	while sleep is just to put your thread on sleep.
wait () method releases the lock of the object on which it has called, it does release other locks if it holds any.	while the sleep method of the Thread class does not release any lock at all.
The wait () method is called on an object on which the synchronized block is locked,	while sleep is called on the Thread.

What is difference between notify () and notifyAll()?

Answer: -

public final void notify ()	**public final void notifyAll ()**
If multiple threads are waiting on any locks in Java, notify method to send a notification to **only one** of the waiting thread.	If multiple threads are waiting on any locks in Java, notifyAll informs all threads waiting on that lock.
If you use notify method, it's not guaranteed which thread will be informed,	if you use notifyAll since all thread will be notified, they will compete for lock, and the lucky thread which gets lock will continue. In a way, the notifyAll method is safer because it sends a notification to all threads, so if any thread misses the notification, there are other threads to do the job, while in the case of notify () method if the notified thread misses the notification, then it could create subtle, hard to debug issues.