New Thread life Cycle :
----
* Thread class has provided an enum called State which is defined inside Thread class. It is used to represent the Universal Constants of Thread State.

* This new life cycle is available from JDK 1.5V.

* It contains the following state.

1) NEW state (Only Thread Object is created, Thread not started yet)

2) RUNNABLE State (Ready to run, Waiting for processor time)

3) BLOCKED State (Waiting for Object lock to enter/re-enter inside synchronized area)

4) WAITING State (Thread is waiting for another thread to complete OR provide notification [Without Timeout time])

5) TIMED_WAITING State (Thread will wait for a particular time)

6) TERMINATED State (Thread has succesfully executed run() method)

NEW :

Whenever we create a thread instance(Thread Object) a thread comes to new state OR born state. New state does not mean that the Thread has started yet only the object or instance of Thread has been created.

RUNNABLE :

Whenever we call start() method on thread object, A thread moves to Runnable state i.e Ready to run state. Here the thread is considered "alive," but it doesn't immediately start execution unless the CPU scheduler assigns it time.

BLOCKED :

If a thread is waiting for object lock OR monitor to enter inside synchronized area OR re-enter inside synchronized area then it is in blocked state.

WAITING :

A thread in the waiting state is waiting for another thread to perform a particular action but WITHOUT ANY TIMEOUT period. A thread that has called wait() method on an object is waiting for another thread to call notify() or notifyAll() on the same object OR A thread that has called join() method is waiting for a specified thread to terminate.

TIMED_WAITING :

A thread in the timed_waiting state, if we call any method which put the thread into temporarly timed_waiting state but WITH POSITIVE TIMEOUT period like sleep(lons ms), join(long ms), wait(long ms) then the Thread is considered as Timed_Waiting state.

TERMINATED :

The thread has successuflly completed it's execution in the separate stack memory that means run() method is successfully completed.

==

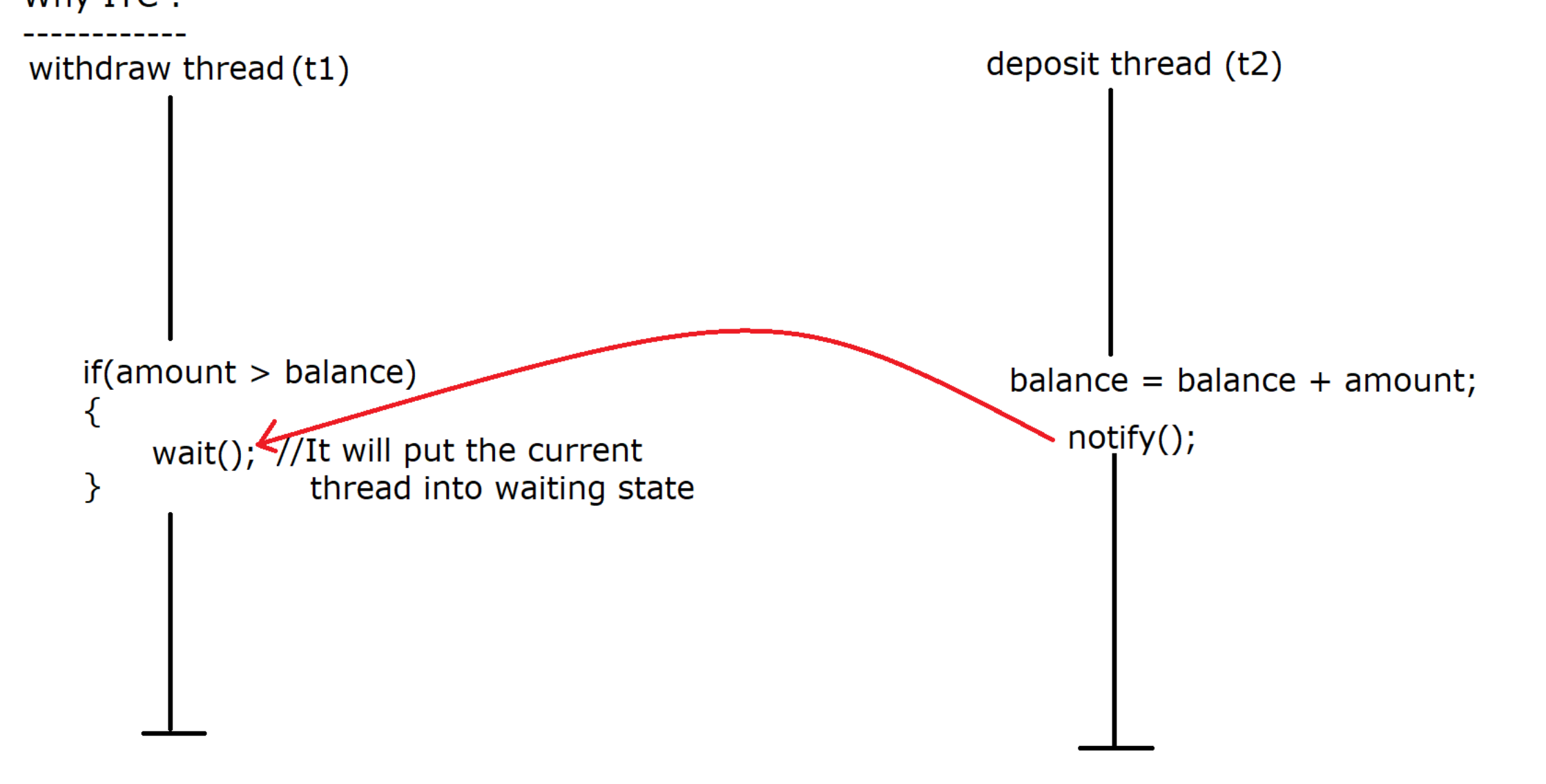
**** Inter-Thread Communication (ITC)
----
* It is a communication OR co-ordination between two OR more than two **synchronized threads** to **complete a particular task** on the **same object.**

* In order to achieve Inter-Thread communication (ITC) we should use the following methods provided by **Object class.**

Note : All these methods must be used from **synchronized area only** otherwise we will get a runtime exception i.e. java.lang.IllegalMonitorStateException (Current thread is not the owner)

a) public final void wait() throws InterruptedException
b) public final native void notify()
c) public final native void notifyAll()

Why ITC :



*** Why wait(), notify() and notifyAll() methods are defined in Object class but not in Thread class ?

Ans : All these methods must be used from **synchronized area only** that means the current thread must be the owner of the lock and lock is available with Object class so all these methods are defined in Object class but not in Thread class.

*** What is the difference between sleep(long ms) and wait() :
----

sleep(long ms)	wait()
1) Method of Thread class.	1) Method of Object class.
2) It is a static method.	2) It is non static method.
3) It will never release the object lock.	**3) It will release the Object lock.**
4) Synchrinized area is not required.	4) Synchrinized area is **required**.
5) Will wake up after completion of time period.	5) Will wake up after getting the notification from another thread on the same object.