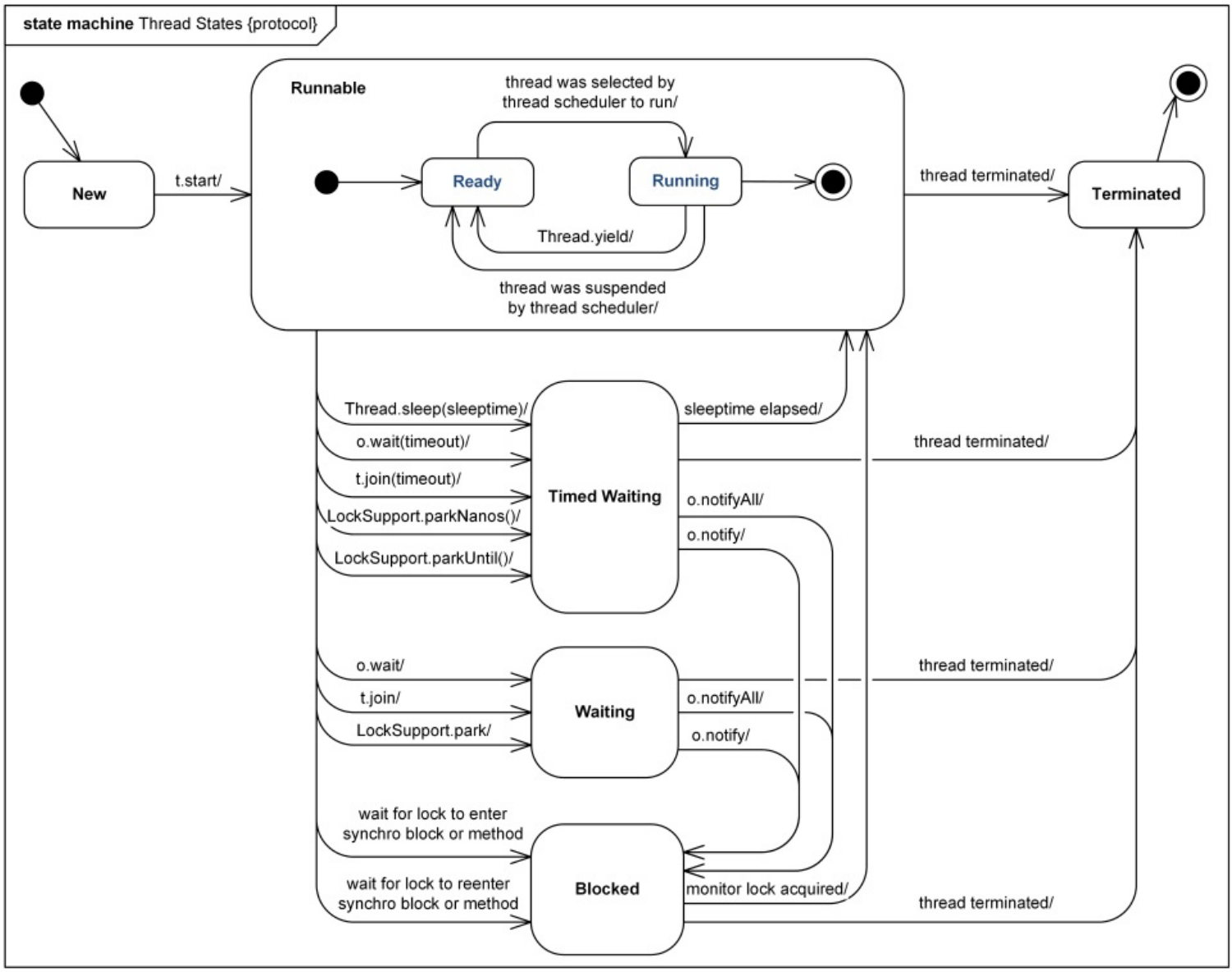
Java Thread States and Life Cycle

UML Protocol State Machine Diagram Example

This is an example of UML **protocol state machine** diagram showing **Thread states** and **Thread life cycle** for the Thread class in **Java™**. A thread is a lightweight process, the smallest unit of scheduled execution. Thread state is Java Virtual Machine (JVM) state reported by JVM to a Java program, it does not reflect any operating system (OS) thread state. At any given point in time an instance of the Thread class in **Java 5** to **Java 9** could be in one of the following Thread states:

- **New**,
- **Runnable**,
- **Timed waiting**,
- **Waiting**,
- **Blocked**,
- **Terminated**.

Now obsolete **Java 2** had different set of states: **Ready**, **Running**, **Suspended**, **Sleeping**, **Blocked**, **Monitor states**. The UML diagram below shows two of these legacy states - **Ready** and **Running** - as part of **Runnable** state.



Protocol state machine example - Thread states and life cycle in Java.

New is the thread state for a thread which was created but has not yet started.

A thread in the **Runnable** state is executing from the JVM's point of view but in fact it may be waiting for some resources from the OS such as processor. This state could be considered as a composite state with two substates. When a thread transitions to the Runnable state, the thread first goes into the **Ready** substate. Thread scheduling decides when the thread could actually start **Running**. Thread.yield() is

explicit recommendation to thread scheduler to pause the currently executing thread to allow some other thread to execute. Thread is **alive** if it has been started and has not yet died.

Timed waiting is a thread state for a thread waiting with a specified waiting time. A thread moves to the timed waiting state due to calling one of the following methods with a specified (positive) waiting time:

- Thread.sleep(sleeptime)
- Object.wait(timeout)
- Thread.join(timeout)
- LockSupport.parkNanos(timeout)
- LockSupport.parkUntil(timeout)

A thread moves to the **Waiting** state after calling one of the following methods without timeout:

- Object.wait()
- Thread.join()
- LockSupport.park()

Note, that a thread in the waiting state is waiting indefinitely for another thread to provide something which is needed by the waiting thread. Once it is done, that other thread could call Object.notify() or Object.notifyAll() on the object the threads share. A thread that has called Thread.join() is waiting for a specified thread to terminate. It means that waiting state could be made a composite state with states corresponding to these specific conditions.

Thread is in the **Blocked** state while waiting for the monitor lock to enter a synchronized block or method or to reenter a synchronized block or method after calling Object.wait().

A synchronized statement or method acquires a mutual-exclusion lock on behalf of the executing thread, executes a block or method, and then releases the lock. While the executing thread owns the lock, no other thread may acquire the lock and is blocked waiting for the lock.

After thread has completed execution of run() method, it is moved into **Terminated** state.