

Theory: States of a thread

🕒 11 minutes 0 / 3 problems solved

Skip this topic

Start practicing

1175 users solved this topic. Latest completion was about 12 hours ago.

Throughout its life cycle, a thread's state changes. This is caused both by the actions of the programmer and internal events of the operating system. In this lesson, we will consider possible states of a thread and some features of the operating system (OS) level.

§1. The Thread.State enum

In Java, the state of a thread is presented by the `Thread.State` enum with six possible values:

- `NEW`: an instance of the class `Thread` has been created, but it has not yet started;
- `RUNNABLE`: a thread is executing in JVM but it may be waiting for OS resources such as processor;
- `BLOCKED`: a thread that is blocked waiting for a monitor lock (we will consider it later);
- `WAITING`: a thread is waiting for another thread indefinitely long to perform a task (e.g. `join` without timeout);
- `TIMED_WAITING`: a thread is waiting for another thread for a specified waiting time (e.g. `sleep`, `join` with timeout);
- `TERMINATED`: a thread is terminated when `run` method completely executes itself or an uncaught exception occurs. Once a thread terminates it never gets back to its runnable state.

To obtain the current state of a thread there is an instance method `getState()`.

Let's look at how these states are changed depending on the programmer's actions.

```
1 Thread worker = ... // new worker to make a difficult task
2 System.out.println(worker.getState()); // NEW
3
4 worker.start(); // start the worker
5 System.out.println(worker.getState()); // RUNNABLE
6
7 worker.join(); // waiting for completing the worker
8 System.out.println(worker.getState()); // TERMINATED
```

When a thread is created its state is `NEW`. When a thread is started its state is `RUNNABLE` (the method `run` may not be called yet). Finally, when a thread is completed the state is `TERMINATED`. At the same time, the `main` thread also went through the state `WAITING` (indefinitely long) after invoking the `thread.join()` method.

§2. The (almost) real lifecycle of a thread

The states discussed above are specific to Java's point of view. The real lifecycle of a thread is slightly different. As an example, the `RUNNABLE` state is actually more complicated than it might seem. In this state, a thread might actually be running or it might be ready to run.

Below you may see the simplified lifecycle of a thread related to the OS terminology. The diagram includes five states and events that cause the thread to jump from one state to another. Please, do not confuse these states with the corresponding states in Java (those are in UPPERCASE).

Current topic:

[States of a thread](#) ...

Topic depends on:

✗ [Exceptions in threads](#) ...

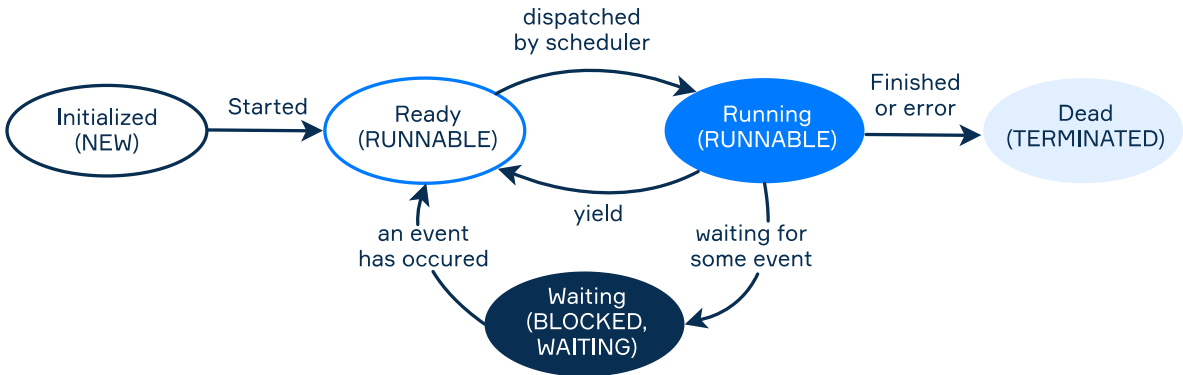
Table of contents:

[1 States of a thread](#)

[§1. The Thread.State enum](#)

[§2. The \(almost\) real lifecycle of a thread](#)

[Feedback & Comments](#)



The simplified lifecycle of a thread in the operating system

After initialization, the thread is ready to run (**Ready**). It is the responsibility of the **thread scheduler** to give some instants of time for a thread to run (**Running**) and then move it again to **Ready**. This is used to share the processor time between multiple threads concurrently. Otherwise, one thread could capture all the available processor time.

The **Waiting** state means that a thread is temporarily inactive (for example, a thread may be waiting for another thread or completing I/O). A thread in this state cannot continue its execution any further until it is moved to **Ready** state. Actually, this state is a set of different states that we will consider in the next topics.

Of course, the real life cycle of threads is even more complicated than we’ve discussed, for example, the **Waiting** state actually corresponds to the state at once. For now, it is enough to have a simplified view of how threads actually work.

 Report a typo

114 users liked this theory. 1 didn't like it. What about you?



Start practicing

[Comments \(0\)](#)

[Hints \(0\)](#)

[Useful links \(1\)](#)

[Show discussion](#)