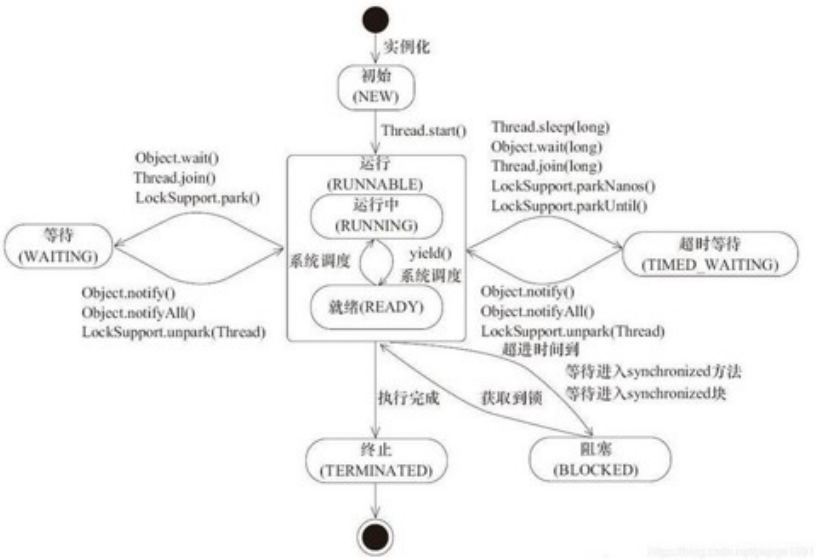
Tutorial on Java Multithreading

Yida Tao, Yao Zhao

Thread States

We've learned different thread states in the lecture. In this tutorial, we further provide a sample code `ThreadState.java`; you may execute this code to see how different thread states occur in action.



Producer-Consumer Problem

(The following descriptions are cited from https://www.baeldung.com/java-producer-consumer-problem) Producer and Consumer are two separate processes. Both processes share a common buffer or queue. The producer continuously produces certain data and pushes it onto the buffer, whereas the consumer consumes those data from the buffer. This problem has several complexities to deal with:

- Both producer and consumer may try to update the queue at the same time. This could lead to data loss or inconsistencies.
- Producers might be slower than consumers. In such cases, the consumer would process elements faster and wait.
- In some cases, the consumer can be slower than a producer. This situation leads to a queue overflow issue.
- In real scenarios, we may have multiple producers, multiple consumers, or both. This may cause the same message to be processed by different consumers.

Implementation

To implement the producer-consumer pattern, we need to solve the following problems:

- Synchronization on queue while adding and removing data
- On queue empty, the consumer has to wait until the producer adds new data to the queue
- When the queue is full, the producer has to wait until the consumer consumes data and the queue has some empty buffer

Approach 1

You may use `BlockingQueue` directly for this purpose. See lecture notes for sample code.

Approach 2

We may also use `Queue`, together with the Locking mechanism, to implement the exact behavior of `BlockQueue`. Read and execute `MyBlockingQueue.java`, and observe the results.

Try change the values of the following variables. What will happen?

```
int CAPACITY = 200;
int PRODUCER_WORK = 20;
int PRODUCER_CNT = 10;
int PRODUCER_OFF = 10;
int CONSUMER_WORK = 20;
int CONSUMER_CNT = 10;
int CONSUMER_OFF = 10;
```

Example:

- `PRODUCER_CNT = 10->100`: Products are produced too fast. Many products are left unconsumed (blocking).
- `CONSUMER_CNT = 10->100`: Products are consumed too fast. Consumers keep waiting but no more product can be produced (blocking).
- `PRODUCER_OFF = 10->1000`: Products are produced slowly, but all products will finally be consumed.
- `PRODUCER_WORK = 20->21`: The queue is left with `210-200=10` products.