# **Thread**

# **1. Các cách tạo thread?**

Có hai cách chính để tạo thread trong Java:

* Cách 1: Kế thừa từ lớp Thread và ghi đè phương thức run()
* Cách 2: Triển khai interface Runnable

# **2. Synchronous vs Asynchronous**

### **1. Synchronous (Đồng bộ)**

Trong lập trình đồng bộ, các tác vụ được thực hiện theo một thứ tự tuần tự. Khi một tác vụ được gọi, chương trình sẽ chờ tác vụ đó hoàn thành trước khi tiếp tục với tác vụ kế tiếp. Điều này có nghĩa là luồng thực thi sẽ bị chặn (blocked) cho đến khi tác vụ hoàn tất.

#### **Đặc điểm chính của Synchronous:**

* **Blocking (Chặn)**: Luồng thực thi bị chặn cho đến khi tác vụ hoàn thành.
* **Thứ tự tuần tự**: Các tác vụ phải được thực hiện theo thứ tự, nghĩa là không thể thực thi song song.
* **Đơn giản**: Dễ hiểu và dễ theo dõi trong các tình huống đơn luồng (single-threaded).
* **Hiệu năng thấp** trong các ứng dụng yêu cầu độ phản hồi cao hoặc có tác vụ tốn thời gian (như I/O hoặc mạng), vì luồng bị chặn trong suốt thời gian chờ.

### **2. Asynchronous (Bất đồng bộ)**

Ngược lại với lập trình đồng bộ, lập trình bất đồng bộ cho phép các tác vụ được thực hiện song song. Khi một tác vụ bất đồng bộ được gọi, chương trình không chờ nó hoàn thành mà tiếp tục thực thi tác vụ tiếp theo ngay lập tức. Tác vụ bất đồng bộ sẽ hoàn thành ở một thời điểm không xác định trong tương lai, thường thông qua callback, Future, hoặc CompletableFuture.

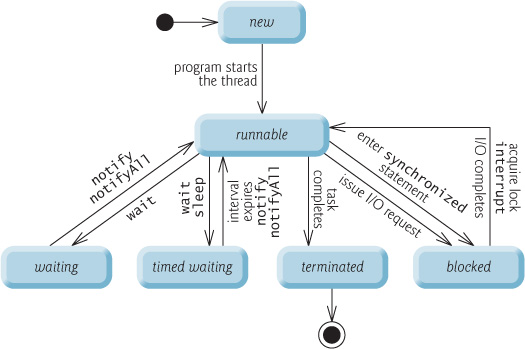
#### **Đặc điểm chính của Asynchronous:**

* **Non-blocking (Không chặn)**: Luồng thực thi không bị chặn, chương trình tiếp tục ngay lập tức.
* **Song song**: Các tác vụ có thể được thực thi song song với nhau mà không phải chờ đợi.
* **Hiệu năng cao** trong các ứng dụng yêu cầu nhiều tác vụ đồng thời (như ứng dụng mạng, xử lý I/O).
* **Phức tạp hơn** để xử lý do sự khó khăn trong việc quản lý trạng thái và luồng logic khi các tác vụ hoàn thành ở thời điểm khác nhau.

| Synchronous | Asynchronous |
| --- | --- |
| Luồng bị chặn cho đến khi tác vụ hoàn thành | Luồng không bị chặn, tiếp tục thực thi ngay lập tức |
| Thứ tự thực thi là tuần tự | Các tác vụ được thực hiện song song |
| Phù hợp cho các tác vụ ngắn và đơn giản | Phù hợp cho các tác vụ tốn thời gian hoặc nhiều tác vụ đồng thời |
| Dễ hiểu và triển khai | Khó quản lý hơn, yêu cầu xử lý đồng bộ hóa kết quả |
| Hiệu năng thấp trong các ứng dụng đòi hỏi phản hồi nhanh | Hiệu năng cao, tận dụng tài nguyên tốt hơn |

**3. Phân biệt trường hợp sử dụng, ưu nhược điểm của async và sync**

# **4. Vòng đời của thread**



* **NEW** : Đây là trạng thái khi luồng vừa được khởi tạo bằng phương thức khởi tạo của lớp Thread nhưng chưa được start(). Ở trạng thái này, luồng được tạo ra nhưng chưa được cấp phát tài nguyên và cũng chưa chạy. Nếu luồng đang ở trạng thái này mà ta gọi các phương thức ép buộc stop,resume,suspend … sẽ là nguyên nhân sảy ra ngoại lệ IllegalThreadStateException .
* **RUNNABLE** : Sau khi gọi phương thức start() thì luồng test đã được cấp phát tài nguyên và các lịch điều phối CPU cho luồng test cũng bắt đầu có hiệu lực. Ở đây, chúng ta dùng trạng thái là Runnable chứ không phải Running, vì luồng không thực sự luôn chạy mà tùy vào hệ thống mà có sự điều phối CPU khác nhau.
* **WAITING** : Thread chờ không giới hạn cho đến khi một luồng khác đánh thức nó.
* **TIMED\_WAITING** : Thread chờ trong một thời gian nhất định, hoặc là có một luồng khác đánh thức nó.
* **BLOCKED**: Đây là 1 dạng của trạng thái “Not Runnable”, là trạng thái khi Thread vẫn còn sống, nhưng hiện tại không được chọn để chạy. Thread chờ một monitor để unlock một đối tượng mà nó cần.
* **TERMINATED** : Một thread ở trong trạng thái terminated hoặc dead khi phương thức run() của nó bị thoát.

# **5. Một số thông tin liên quan đến luồng**

### **Định danh của luồng (ThreadId)**

ThreadId là định danh của luồng, nó dùng để phân biệt với các luồng khác cùng tiến trình hoặc cùng tập luồng. Đây là thông số mà máy ảo java tự tạo ra khi ta tạo luồng nên ta không thể sửa đổi cũng như áp đặt thông số này khi tạo luồng. Nhưng ta có thể lấy được nó thông qua phương thức getId() của lớp Thread

### **Tên của luồng (ThreadName)**

ThreadName là tên của luồng, đây là thuộc tính mà ta có thể đặt hoặc không đặt cho luồng. Nếu ta không đặt cho luồng thì máy ảo java sẽ tự đặt với quy tắc sau: “Thread-” + Thứ tự luồng được tạo ra, bắt đầu từ 0.

### **Độ ưu tiên của luồng (Priority)**

Như đã nói ở phần trước, mỗi luồng có 1 độ ưu tiên nhất định. Đây sẽ là thông số quyết định mức ưu tiên khi cấp phát CPU cho các luồng.

Trong java, đế đặt độ ưu tiên cho 1 luồng ta dùng phương thức: **void setPriority(int newPriority)**

* int newPriority : Là giá trị từ 1 đến 10.

Java có định nghĩa sẵn 3 mức ưu tiên chuẩn như sau:

* Thread.**MIN\_PRIORITY** (giá trị **01**)
* hread.**NORM\_PRIORITY** (giá trị **05**)
* Thread.**MAX\_PRIORITY** (giá trị **10**)

Để lấy độ ưu tiên của 1 luồng, ta dùng phương thức: **int getPriority()**