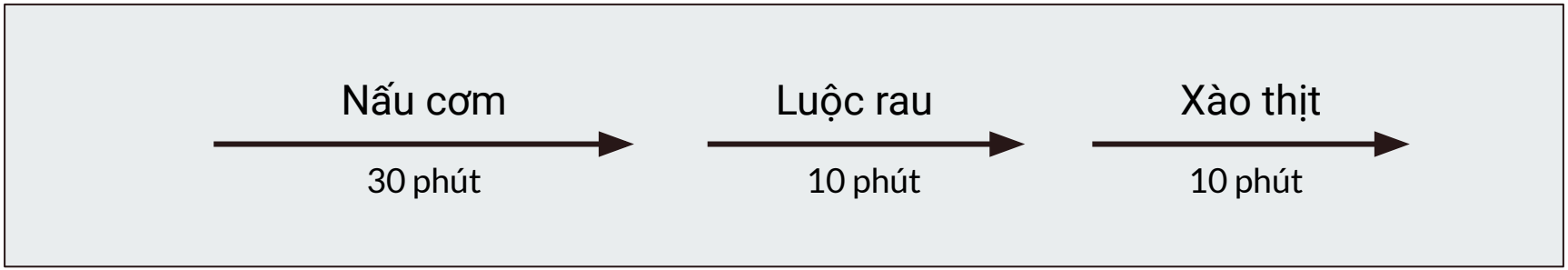
Java Multi-threading

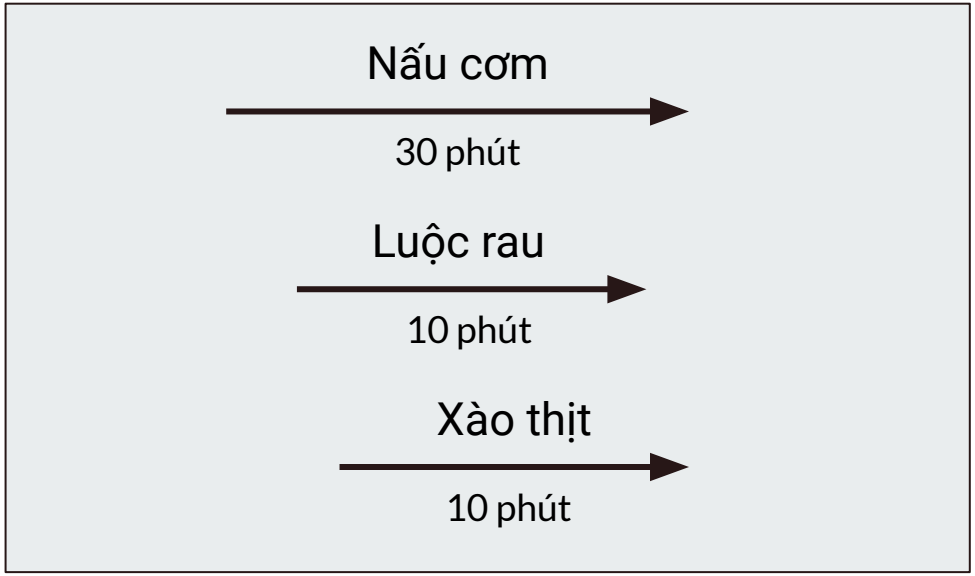
Ngọc Lục

Thread và Multi-thread

Thread về cơ bản là một tiến trình con (sub-process). Một đơn vị xử lý nhỏ nhất của máy tính có thể thực hiện một công việc riêng biệt. Trong Java, các luồng được quản lý bởi JVM

Multi-thread là một tiến trình thực hiện nhiều luồng đồng thời. Một ứng dụng Java, ngoài luồng chính có thể có các luồng khác thực thi đồng thời làm ứng dụng chạy nhanh và hiệu quả hơn


Nấu cơm
30 phút
Luộc rau
10 phút
Xào thịt
10 phút


Nấu cơm
30 phút
Luộc rau
10 phút
Xào thịt
10 phút

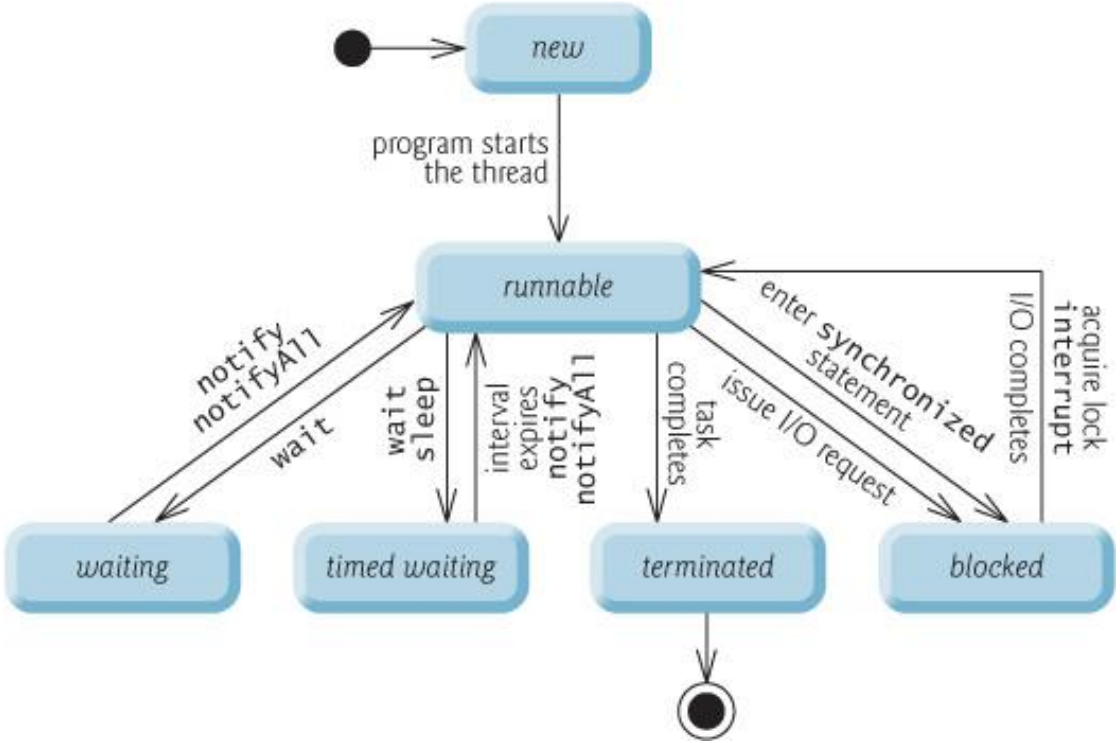
Ưu điểm

- Nó không chặn người sử dụng vì các luồng là độc lập và có thể thực hiện nhiều công việc cùng một lúc.
- Có thể thực hiện nhiều hoạt động với nhau để tiết kiệm thời gian.
- Luồng là độc lập vì vậy nó không ảnh hưởng đến luồng khác nếu ngoại lệ xảy ra trong một luồng duy nhất

Nhược điểm

- Càng nhiều luồng thì càng xử lý phức tạp
- Xử lý vấn đề về tranh chấp bộ nhớ
- Cần phát hiện tránh các luồng chết, luồng chạy mà không làm gì trong ứng dụng cả

Vòng đời của Thread



	Process	Thread
Khái niệm	Một chương trình đang chạy được gọi là tiến tình	Một chương trình có thể có nhiều Thread
Không gian địa chỉ	Có một không gian địa chỉ riêng biệt	Chia sẻ không gian địa chỉ với nhau
Đa nhiệm	Máy tính cho phép chạy từ 2 chương trình đồng thời	Một chương trình có thể chạy từ 2 luồng đồng thời
Giao tiếp	Tốn kém và bị giới hạn	Ít tốn kém hơn so với process
Thành phần	Không gian địa chỉ, biến global, xử lý tín hiệu, sub-process, thông tin tính toán	Thanh ghi, trạng thái, stack, bộ đếm chương trình
Điều khiển	Không thuộc quyền kiểm soát của Java	Phụ thuộc quyền kiểm soát của Java

Tạo Thread

Có 2 cách để tạo Thread:

- Kế thừa lớp Thread
- Triển khai interface Runnable

Tạo Thread bằng cách kế thừa class Thread

Với cách này ta cần khai báo một class mới và để class đó kế thừa class Thread. Override lại method **run()** của lớp Thread.

```
public class ThreadDemo extends Thread{
    @Override
    public void run() {
        System.out.println("Thread is running...");
    }
}
```

Tạo một đối tượng của lớp vừa khai báo và gọi tới method **start()**

```
public class Main {
    public static void main(String[] args) {
        ThreadDemo threadDemo = new ThreadDemo();
        threadDemo.start();
    }
}
```

Tạo Thread bằng cách triển khai interface Runnable

Với cách này ta cần khai báo một class mới và để class đó implement interface Runnable. Override lại method **run()** của lớp Thread.

```
public class ThreadDemo implements Runnable{
    @Override
    public void run() {
        System.out.println("Thread is running...");
    }
}
```

Tạo một đối tượng của lớp vừa khai báo. Tạo instance của lớp Thread. Gọi method **start()**

```
public class Main {
    public static void main(String[] args) {
        ThreadDemo threadDemo = new ThreadDemo();
        Thread thread = new Thread(threadDemo);
        thread.start();
    }
}
```

Dừng thực thi với phương thức Thread.sleep()

Phương thức **Thread.sleep()** khiến cho luồng hiện thời tạm ngừng thực thi trong một khoảng thời gian xác định.

```
sleep(long millis) throws InterruptedException
```

```
sleep(long millis, int nanos) throws InterruptedException
```

Join các Thread

Phương thức **join()** được sử dụng để đảm bảo cho quá trình thực thi của Thread đang chạy không bị gián đoạn bởi các Thread khác.

```
join() throws InterruptedException
```

```
join(final long millis)
```

```
join(long millis, int nanos)
```