**Lập trình đa tiến trình (Multithreading) trong Java**

**1. Đa tiến trình trong ngôn ngữ lập trình Java**

Trong Java, Multithreading được gọi là đa luồng, còn Multitasking được gọi là đa tiến trình. Đa luồng và đa tiến trình được sử dụng để tạo ra hệ thống đa nhiệm (*multitasking*). Một chương trình được gọi là đa luồng khi chương trình đó có 2 luồng trở lên chạy song song với nhau và một luồng (*thread*) là đơn vị nhỏ nhất của tiến trình (*process*). Luồng là đơn vị nhỏ nhất trong chương trình có thể thực hiện được một công việc riêng biệt và các luồng này được quản lý bởi máy ảo Java. Một luồng gồm có 4 thành phần chính đó là: định dang, một bộ đếm chương trình, một tập thanh ghi và ngăn xếp. Một ứng dụng Java ngoài luồng chính có thể có các luồng khác thực thi đồng thời. Đa luồng trong Java giúp công việc được hoàn thành một cách nhanh chóng.

Tóm lại, đa luồng trong Java có thể hiểu đơn giản là xử lý nhiều luồng dữ liệu song song với nhau để thực hiện các nhiệm vụ khác nhau cùng một lúc. Tuy nói là cùng một lúc nhưng thời gian để chuyển qua lại giữa các luồng đó vẫn có dộ trễ nhưng rất ngắn (*chỉ tính bằng đơn vị nano giây*). Để có thể hiểu về đa luồng, tôi có một ví dụ đơn giản như sau: Ta có 2 tiến trình 1 và 2, khi chạy thì tiến trình 1 sẽ được chạy trong một khoảng thời gian nhất định rồi tạm dừng rồi chuyển sang chạy tiến trình 2 và tiến trình 2 cũng chạy trong một khoảng thời gian nhất định và chuyển về tiến trình 1. Quá trình này được thực hiện liên tục đến khi nào 1 trong 2 tiến trình kết thúc. Lưu ý là quá trình chuyển đổi qua lại giữa 2 tiến trình này rất nhanh, thời gian để thực thi 2 tiến trình này cũng rất nhanh nên chúng ta sẽ có cảm giác hai tiến trình 1 và 2 chạy song song với nhau nhưng thực tế nó vẫn có 1 khoảng chênh lệch thời gian nhất định.

**2. Các cách tạo luồng trong Java**

Java cung cấp cho chúng ta 1 lớp có tên là Thread. Thread là 1 lớp có thể tạo ra 1 lớp chạy đa tiến trình được. Trong Java, chúng ta có hai cách chính để tạo luồng đó là tạo 1 đối tượng của lớp kế thừa lớp Thread và cách thứ hai đó là implements từ 1 Interface có tên là Runnable. Trong phạm vi của bài ngày hôm nay, tôi chỉ nên khái quát về hai cách này, còn về chi tiết sang bài tiếp theo tôi sẽ trình bày kỹ hơn và đưa ra ví dụ cụ thể.

Hai cách trên có những điểm giống nhau nhưng cũng có những điểm khác nhau như sau:

* Giống nhau: đều cùng được dùng để tạo luồng.
* Khác nhau: Nếu chúng ta tạo luồng bằng cách tạo 1 lớp kế thừa từ lớp Thread thì chúng ta sẽ không thể kế thừa thêm 1 lớp nào khác vì ngôn ngữ lập trình Java không hỗ trợ tính đa kế thừa. Còn nền chúng ta tạo luồng bằng cách implements Interface Runnable thì chúng ta có thể kế thừa một lớp khác ngoài lớp Thread.