**Process**: là đại diện cho 1 chương trình hoặc lệnh đang chạy độc lập. Mỗi process có không gian nhớ riêng biệt.

Process không trực tiếp chia sẻ được dữ liệu cho nhau.

**Thread**: là đơn vị nhỏ hơn của process. Mỗi process có thể chứa nhiều thread cùng chạy song song.

Các cách tạo thread:

Lớp kế thừa Thread:

Không có khả năng tái sử dụng

Không linh hoạt: vì class chỉ có thể kế thừa Thread mà không thể kế thừa lớp khác.

Lớp kế thừa Runnable:

Dễ dàng tái sử dụng vì có thể truyền vào nhiều đối tượng Thread.

Có thể kế thừa các lớp khác vì Runnable là 1 interface

Sử dụng new Thread và biểu thức lambda:

Dễ dàng tạo 1 thread

Sử dụng ExecutorService:

Quản lý các thread tốt hơn.

Có thể quản lý số lượng thread.

Số task vượt quá số lượng thread được phép chạy đồng thời nó sẽ cho vào hàng đợi và phải chờ cho đến khi có 1 thread hoàn thành.

**Multi thread:**

Là việc cho phép nhiều thread chạy đồng thời, cải thiện hiệu suất.

Ưu điểm:

Tăng năng suất làm việc.

Tận dụng tối đa tài nguyên hệ thống.

Nhược điểm:

Khó quản lý lỗi

Có thể dẫn đến xung đột dữ liệu

Để biết 1 thread hoàn thành hay chưa

Sử dụng phương thức isAlive(): true là đang chạy còn false là đã kết thúc.

Sử dụng phương thức join() để ngăn chặn việc xử lý các lệnh của luồng main chạy trước khi thread thực hiện xong.

Việc giới hạn số lượng thread là điều vô cùng quan trọng.

Việc giới hạn số lượng thread cho 1 tác vụ sẽ hạn chế việc nó chiếm hết tài nguyên của tác vụ khác.

Có thể giới hạn thread trong ExecutorService