https://vi.wikipedia.org/wiki/Ph%C3%A9p\_to%C3%A1n\_thao\_t%C3%A1c\_bit

Trong Java, tất cả các giá trị mang kiểu số nguyên đều có dấu, và các toán tử  << và >> thực hiện các phép dịch chuyển số học. Java còn thêm vào toán tử >>> để thực hiện phép dịch chuyển luận lý sang phải, nhưng bởi vì phép dịch chuyển sang trái số học và luận lý là như nhau, nên không có toán tử <<< trong Java.

Một vài chi tiết về các toán tử dịch chuyển trong Java:

* Thao tác  << (dịch trái), >> (dịch phải có dấu), và >>> (dịch phải không dấu) được gọi là các toán tử dịch chuyển.
* Kiểu giá trị mà phép dịch bit biểu thị là dạng cao cấp của toán hạng bên trái. Ví dụ, aByte >>> 2 thì tương đương với ((int) aByte) >>> 2.
* Nếu như kiểu giá trị cao cấp của toán hạng bên trái là int, thì chỉ có năm bit thấp nhất theo thứ tự của toán hạng bên phải được sử dụng như là khoảng cách dịch chuyển. Điều này giống như là toán hạng bên phải được sử dụng cho một toán tử luận lý thao tác bit AND & với giá trị che đậy 0x1f (0b11111). Khoảng cách dịch chuyển thực ra luôn nằm trong khoảng từ 0 tới 31, một cách bao quát.
* Nếu như kiểu giá trị cao cấp của toán hạng bên trái là long, thì chỉ có sáu bit thấp nhất theo tứ tự của toán hạng bên phải được sử dụng như là khoảng cách dịch chuyển. Điều đó giống như là toán hạng bên phải được sử dụng cho một toán tử luận lý thao tác bit AND & với giá trị che đậy 0x3f (0b111111). Khoảng cách dịch chuyển thực ra luôn nằm trong khoảng từ 0 tới 63, một cách bao quát.
* Kết quả của n >> s là n bị dịch chuyển sang phải s bit và đem 0 vào bên trái tương ứng.
* Trong toán tử nói chung và phép dịch bit nói riêng, kiểu dữ liệu  byte được hàm ý chuyển thành int. Nếu giá trị byte đó là âm, và bit bậc cao nhất là một, thì các số một sẽ được điền vào để lấp đầy các bytes được thêm vào ở kiểu int. Do đó byte b1=-5; int i = b1 | 0x0200; sẽ cho kết quả i == -5.