Recursion – đệ quy, là một chủ đề rất hay. Nắm được đẹ quy – nắm được Recursion Algoritm, sẽ dễ dàng làm ngắn gọn code mà hiệu quả nữa. Chúng ta cùng nghiên cứu Recursion Algorithm qua các bài toán cụ thể, bắt đầu từ hôm nay 28/9.

Sách:

\_ Advanced Topics in Java, sách này có 1 chương nói về Recursion Algorithm

# 1. Bài toán giai thừa

## Code:

**public** **class** Test {

**int** fact(**int** n){

**if** (n < 0) {

**return** 0;

} **else** **if** (n==0) {

**return** 1;

} **else** {

**return** n\*fact(n-1);

}

}

**public** **static** **void** main(String[] args) {

Test test = **new** Test();

System.***out***.println(test.fact(-2));

System.***out***.println(test.fact(0));

System.***out***.println(test.fact(1));

System.***out***.println(test.fact(2));

System.***out***.println(test.fact(5));

}

}

## Result:



Mấu chốt là ở đoạn lệnh **return** n\*fact(n-1);

Nhìn vào điểm mấu chốt này ta biết sẽ trả về kết quả là: n\*(n-1)\*(n-2)\*…\*2\*1

Nhân đến khi nào bằng 0 sẽ dừng lại vì **else** **if** (n==0) {

**return** 1;

}

Như thế kết quả y chang định nghĩa về giai thừa => bài toán được giải quyết.

# 2. Bài toán giai thừa hiệu quả hơn

Bài toán ở trên chỉ để minh họa cho tính đệ quy trong chương trình, để bài toán hiệu quả hơn ta làm như sau:

## Code:

**package** demo.com;

**public** **class** Test {

**int** fact(**int** n){

**int** f = 1;

**while** (n > 0) {

f = f \* n;

n--;

}

**return** f;

}

**public** **static** **void** main(String[] args) {

Test test = **new** Test();

System.***out***.println(test.fact(-2));

System.***out***.println(test.fact(0));

System.***out***.println(test.fact(1));

System.***out***.println(test.fact(2));

System.***out***.println(test.fact(5));

}

}