

알고리즘 4

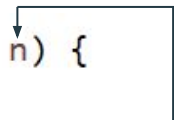
교재 참고 예제로 구성되어 있습니다
본 문서는 학습 참고용으로만 사용하시고 웹이나 타인에게 배포를 금합니다

재귀 함수란?

재귀 함수란 자기 자신을 다시 호출하는 함수를 의미

재귀를 효과적으로 사용하면 프로그램을 간결하고 효율성 좋게 작성할 수 있음

```
static void recur(int n) {  
    if(n>0) {  
        recur(n-1);  
        System.out.println(n);  
    }  
}
```



재귀 단계

모든 재귀 함수는 **기본 단계(Base Case)**와 **재귀 단계(Recursive Case)** 두 부분으로 나뉜다

재귀함수는 실수로 무한반복 함수로 만들기 쉬우므로 주의해야 한다

재귀단계: 함수가 자기 자신을 호출하는 부분

기본단계: 함수가 스스로를 다시 호출하지 않도록 하는 부분

```
def countdown(i):
```

```
    print i
```

```
    if i <= 1:
```

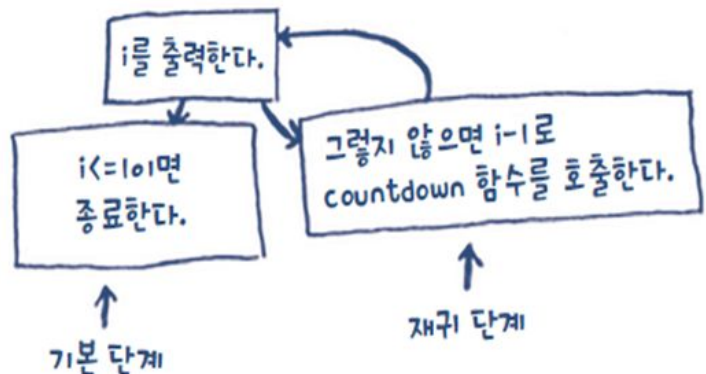
```
        return
```

```
    else:
```

```
        countdown(i-1)
```

기본 단계

재귀 단계



기본단계

재귀호출이 정상 동작하려면 '**종료 조건**'이 필요하다.

특정 조건이 되면 더는 자신을 호출하지 않고 멈추도록 설계되어야 한다.

(ex. 러시아 인형. 맨 마지막에는 사탕이 나옴) 재귀호출 함수가 계산 결과를 돌려줄 때는 **return** 명령을 사용해서 종료조건의 결과값부터 돌려준다.

```

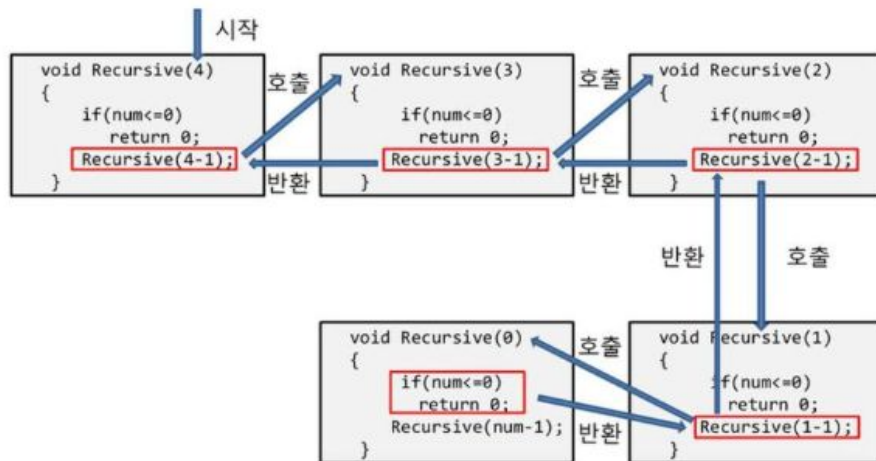
2
3 public class Recursive {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         recursive(4);
8     }
9
10    private static void recursive(int num) {
11        if(num<=0)
12            return;
13        System.out.printf("recursive() call %d\n", num);
14        recursive(num-1);
15    }
16 }

```

```

recursive() call 4
recursive() call 3
recursive() call 2
recursive() call 1

```



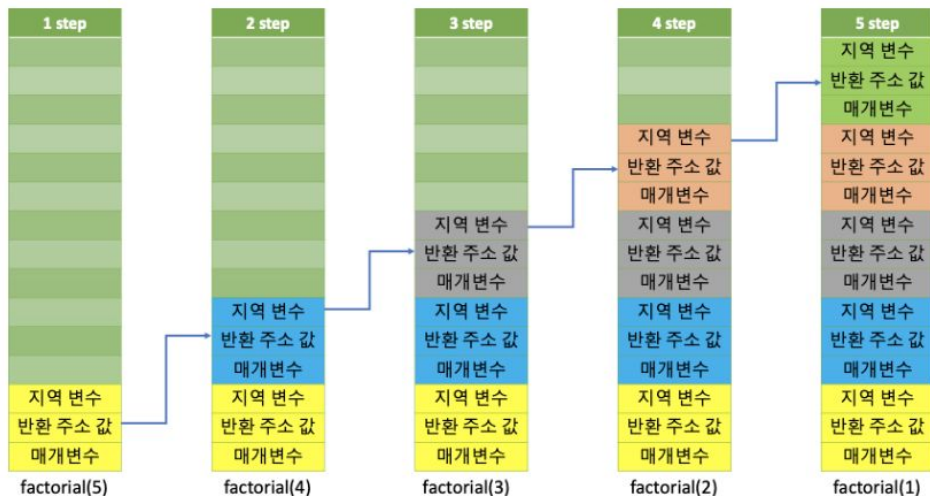
스택 프레임

함수가 호출되면 스택에 함수의 매개변수,
호출이 끝난 뒤 돌아갈 복귀 주소값,
함수의 지역변수 등이 저장된다.

이렇게 스택에 차례대로 저장되는 함수의 호출 정보를

스택 프레임이라고 한다.

재귀 함수도 함수 이므로 호출 정보가 스택 프레임에 저장된다



10진수를 2진수로 바꾸기

- 재귀함수를 이용해 10진수 정수(양수)를 2진수로 바꿔 출력하는 프로그램을 작성하세요
- 재귀함수로 구현한 뒤 스택프레임을 그려보고 동작원리를 생각해보세요

[1] 10진수를 2로 나누는 작업을 몫이 0이 나올때 까지 반복한다

[2] 10진수 정수를 2로 나눈 나머지값을 출력한다

- 구현 되었다면 2진수 뿐 아니라 8진수, 16진수로도 변환하는 함수도 같은 원리로 구현해보세요

Factorial

- 1~N까지의 곱을 구하는 팩토리얼을 재귀함수를 이용해 구현해보세요

```
n!  
1! : 1  
2! : (1)x 2  
3! : (1x2)x3  
4! : (1x2x3)x4  
5! : (1x2x3x4)x5  
n! : (n-1)!xn
```

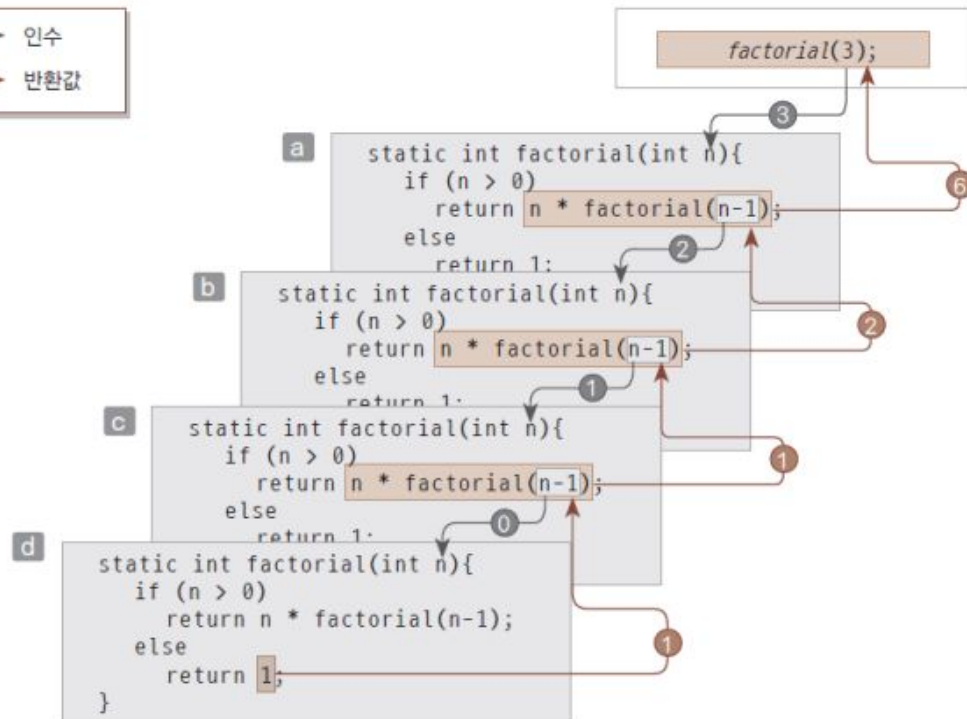
여기서 $1!=1$ 그리고 $n!=n \times (n-1)!$ 라는 팩토리얼 성질을 이용해서 프로그램을 만들어보자

<terminated> Factorial (1) [Java]

정수를 입력하세요:

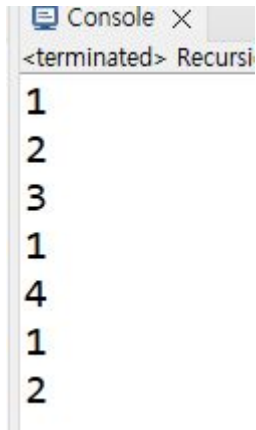
5

120



Recursion

```
static void recur(int n) {  
    if(n>0) {  
        recur(n-1);  
        System.out.println(n);  
        recur(n-2);  
    }  
}  
|  
public static void main(String[] args) {  
    recur(4);  
}
```

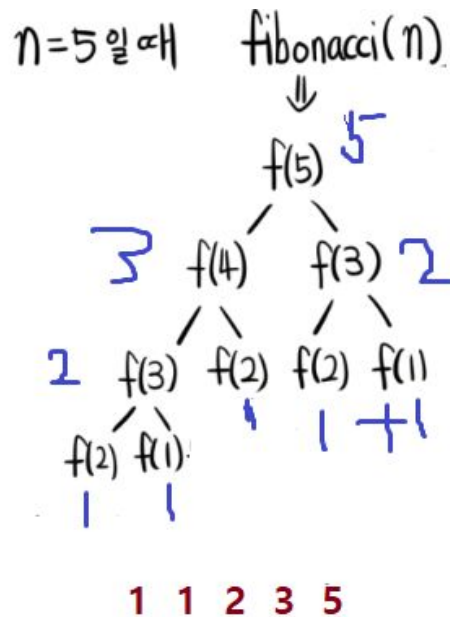


```
Console X  
<terminated> Recursi  
1  
2  
3  
1  
4  
1  
2
```


Fibonacci

피보나치 수열은 반복문을 이용해서도 구현
가능하지만 재귀함수를 통해 구현해보자.

```
Console X
<terminated> FibonacciRecur (2) [Java Application] C:\Java\jdk-14\bin#
1, 1, 2, 3, 5, 8, 13, 21, 34, 55,
```



Pallindrome

배열 앞과 맨 뒤의 문자가 같을 경우
배열을 잘라내어 다시 같은지 비교한다.
다르면 **false**를 반환하지만 같으면 또
다시 배열을 잘라낸다

```
Console X  
<terminated> PalindromeRecur [Java A  
[m, a, d, a, m]  
[a, d, a]  
[d]  
true
```

회문문자열을 판별하는 프로그램을 재귀함수를 이용해서 구현해보자.

배열을 범위를 지정해 copy해주는 **Arrays.copyOfRange(배열, start, end)** 를 활용해보자

=> 배열의 start 인덱스부터 end-1까지 카피하여 새로운 배열을 반환한다

[1] 배열의 크기가 1이하인 경우 **true** 를 리턴한다

[2] 두 문자열이 다른 경우에는 **false** 를 리턴한다

[3] 배열의 맨 앞과 맨뒤를 비교후 같은 경우 다음 문자를 비교하기 위해 재귀함수를 사용하며 배열을 잘라낸다