

RECURSION (ÖZYİNELEME)

Recursion (Özyineleme) nedir?

- Problemleri daha basit alt problemlere bölerek çözme yöntemi.
- Alt problemler de kendi içlerinde başka alt problemlere bölünebilir.
- Alt problemler çözülebilecek kadar küçülünce bölme işlemi durur.
- Özyinelemeli bir algoritma bir problemi çözmek için problemi iki veya daha fazla alt probleme bölen bir yöntemdir.

Faktöryel Hesabı

```
faktoryel(0) = 1;  
faktoryel(n) = n*faktoryel(n-1);
```

faktoryel(3)

Faktöryel Hesabı

$$\text{faktoryel}(3) = 3 * \text{faktoryel}(2)$$

```
faktoryel(0) = 1;  
faktoryel(n) = n*faktoryel(n-1);
```

Faktöryel Hesabı

$$\begin{aligned}\text{faktoryel}(3) &= 3 * \text{faktoryel}(2) \\ &= 3 * (2 * \text{faktoryel}(1))\end{aligned}$$

```
faktoryel(0) = 1;  
faktoryel(n) = n*faktoryel(n-1);
```

Faktöryel Hesabı

$$\begin{aligned}\text{faktoryel}(3) &= 3 * \text{faktoryel}(2) \\ &= 3 * (2 * \text{faktoryel}(1)) \\ &= 3 * (2 * (1 * \text{faktoryel}(0)))\end{aligned}$$

```
faktoryel(0) = 1;  
faktoryel(n) = n*faktoryel(n-1);
```

Faktöryel Hesabı

$$\begin{aligned}\text{faktoryel}(3) &= 3 * \text{faktoryel}(2) \\ &= 3 * (2 * \text{faktoryel}(1)) \\ &= 3 * (2 * (1 * \text{faktoryel}(0))) \\ &= 3 * (2 * (1 * 1))\end{aligned}$$

```
faktoryel(0) = 1;  
faktoryel(n) = n*faktoryel(n-1);
```

Faktöryel Hesabı

$$\begin{aligned}\text{faktoryel}(3) &= 3 * \text{faktoryel}(2) \\ &= 3 * (2 * \text{faktoryel}(1)) \\ &= 3 * (2 * (1 * \text{faktoryel}(0))) \\ &= 3 * (2 * (1 * 1)) \\ &= 3 * (2 * 1)\end{aligned}$$

```
faktoryel(0) = 1;  
faktoryel(n) = n*faktoryel(n-1);
```


Faktöryel Hesabı

$$\text{faktoryel}(3) = 3 * \text{faktoryel}(2)$$

$$= 3 * (2 * \text{faktoryel}(1))$$

$$= 3 * (2 * (1 * \text{faktoryel}(0)))$$

$$= 3 * (2 * (1 * 1))$$

$$= 3 * (2 * 1)$$

$$= 3 * 2$$

$$\text{faktoryel}(0) = 1;$$

$$\text{faktoryel}(n) = n * \text{faktoryel}(n-1);$$

Faktöryel Hesabı

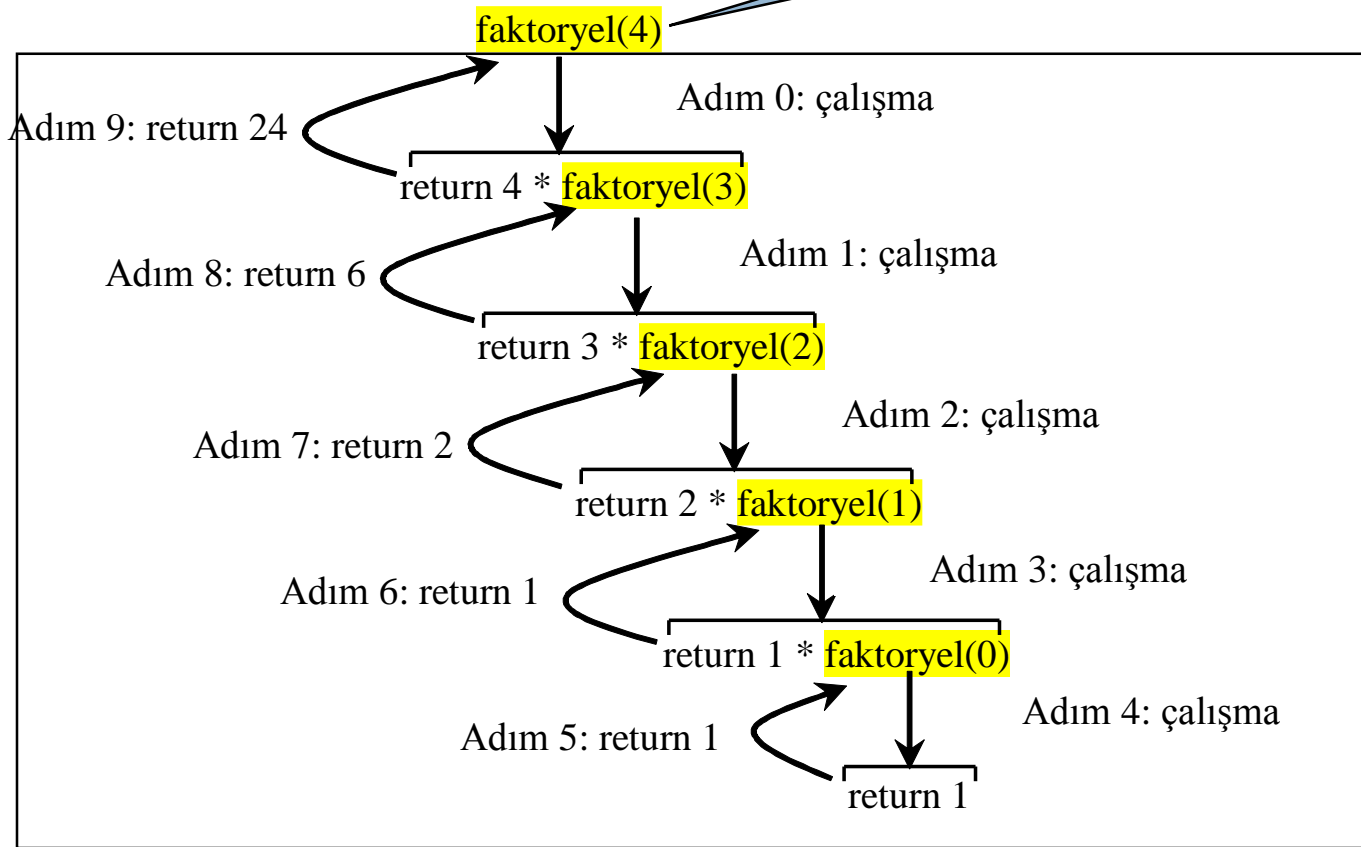
$$\begin{aligned}\text{faktoryel}(3) &= 3 * \text{faktoryel}(2) \\ &= 3 * (2 * \text{faktoryel}(1)) \\ &= 3 * (2 * (1 * \text{faktoryel}(0))) \\ &= 3 * (2 * (1 * 1)) \\ &= 3 * (2 * 1) \\ &= 3 * 2 \\ &= 6\end{aligned}$$

`faktoryel(0) = 1;`

`faktoryel(n) = n*faktoryel(n-1);`

Trace Recursive faktoryel

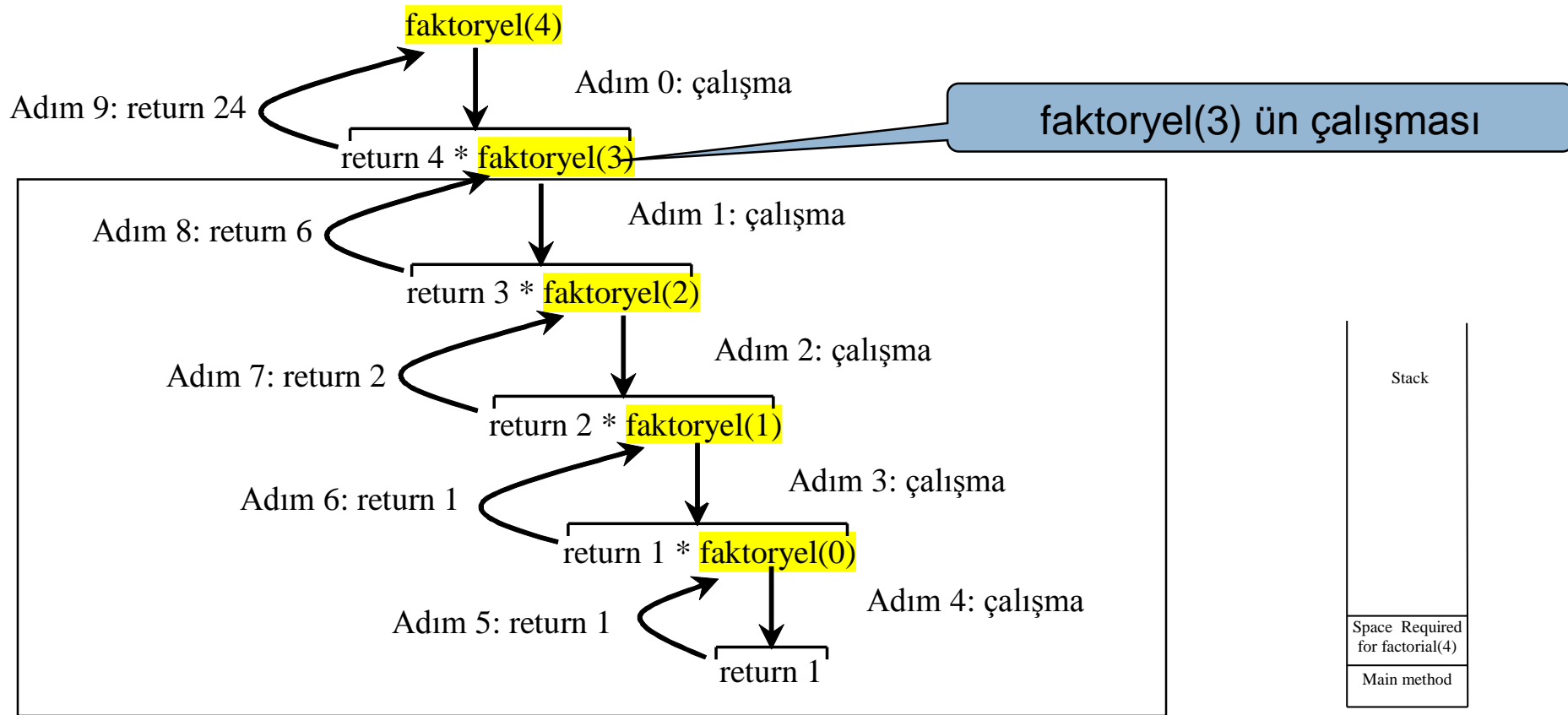
faktoryel(4) ün çalışması



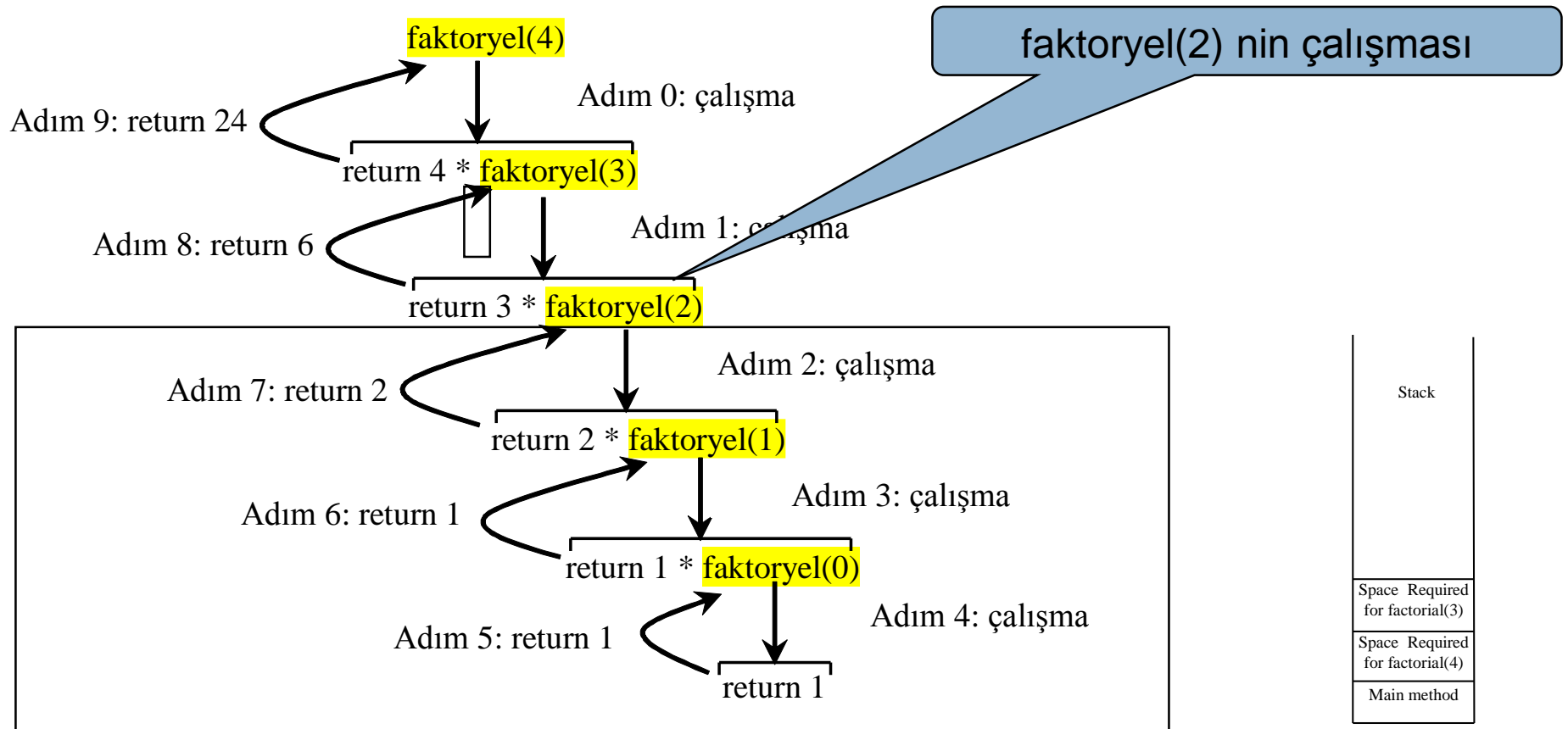
Stack

Main method

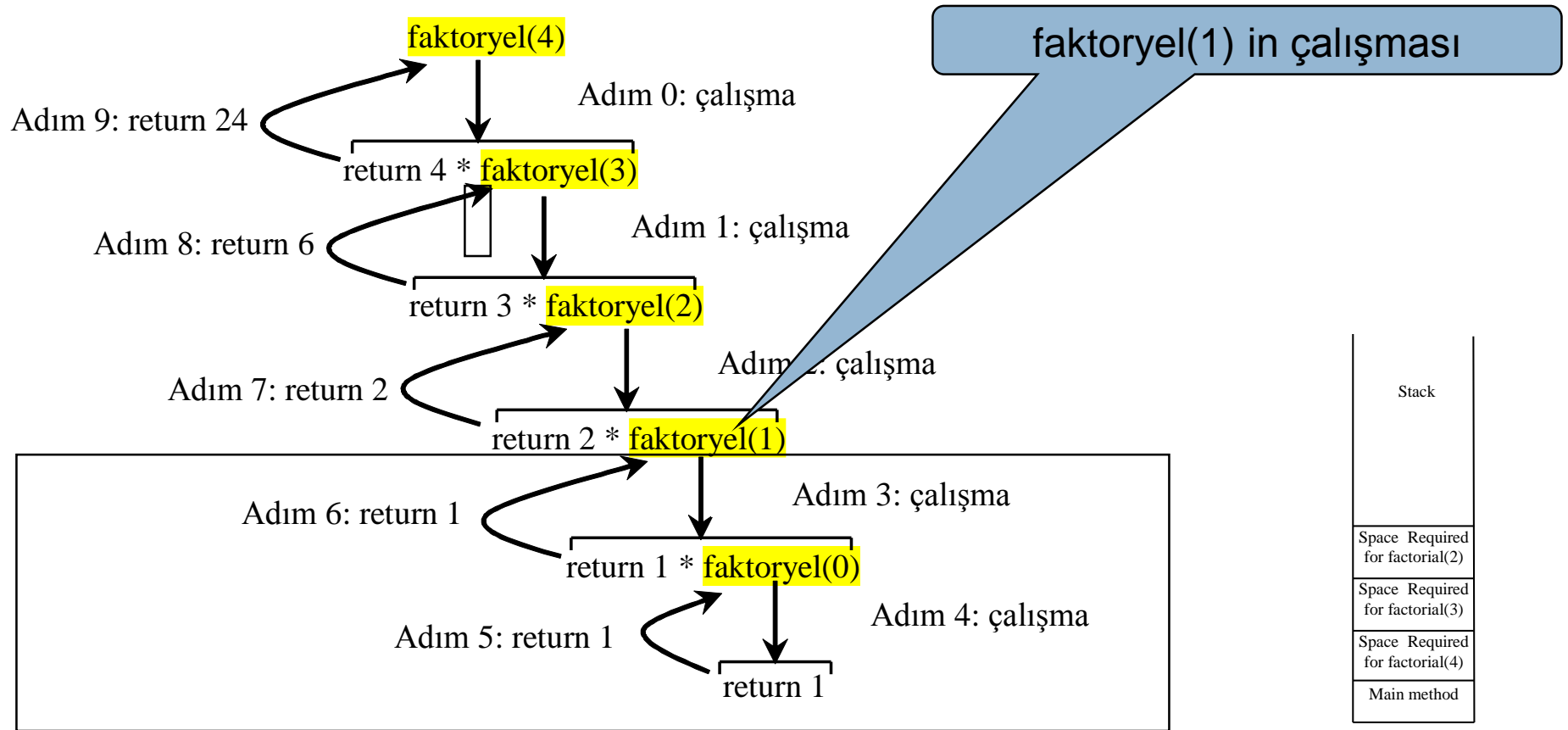
Rekursif faktoryel



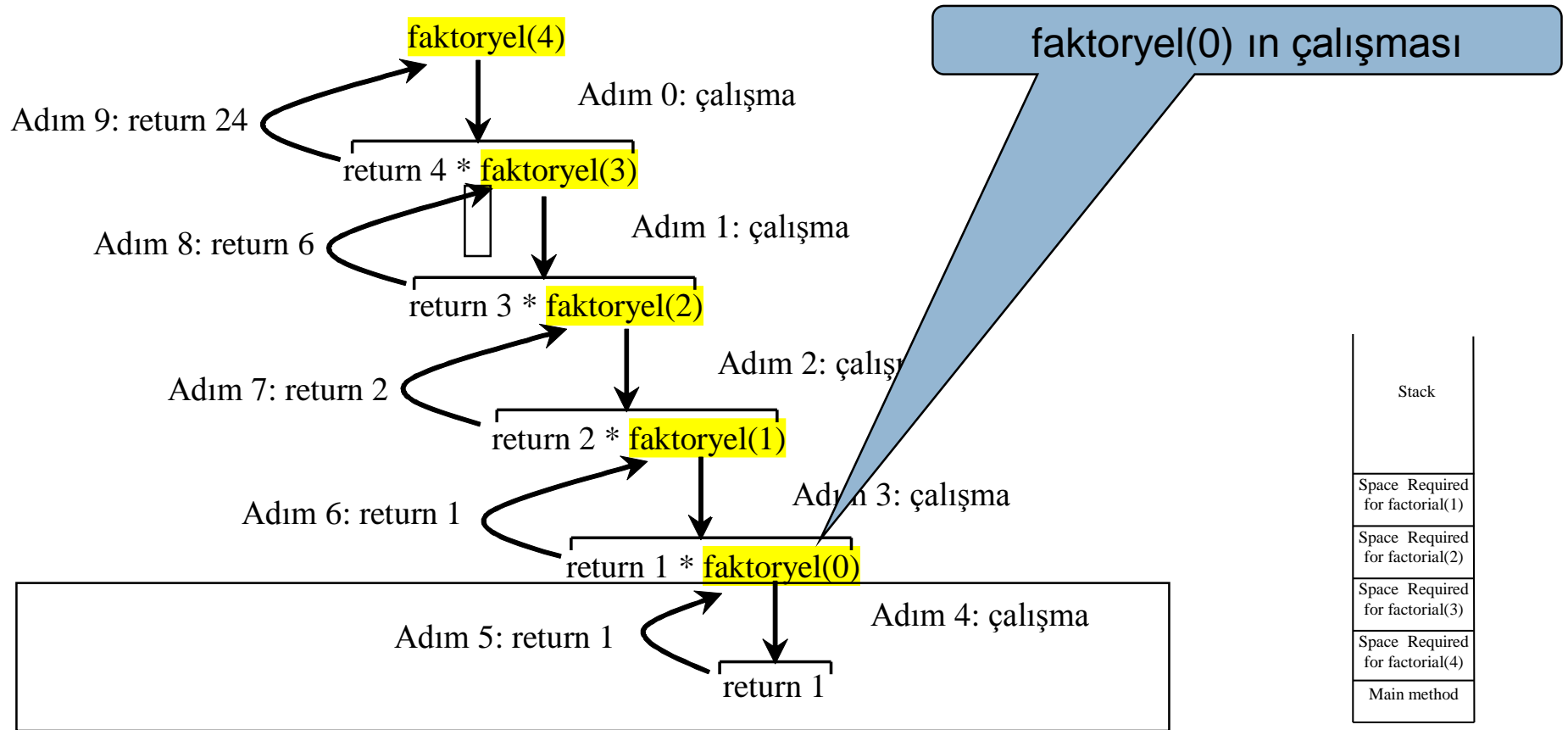
Recursive faktoryel



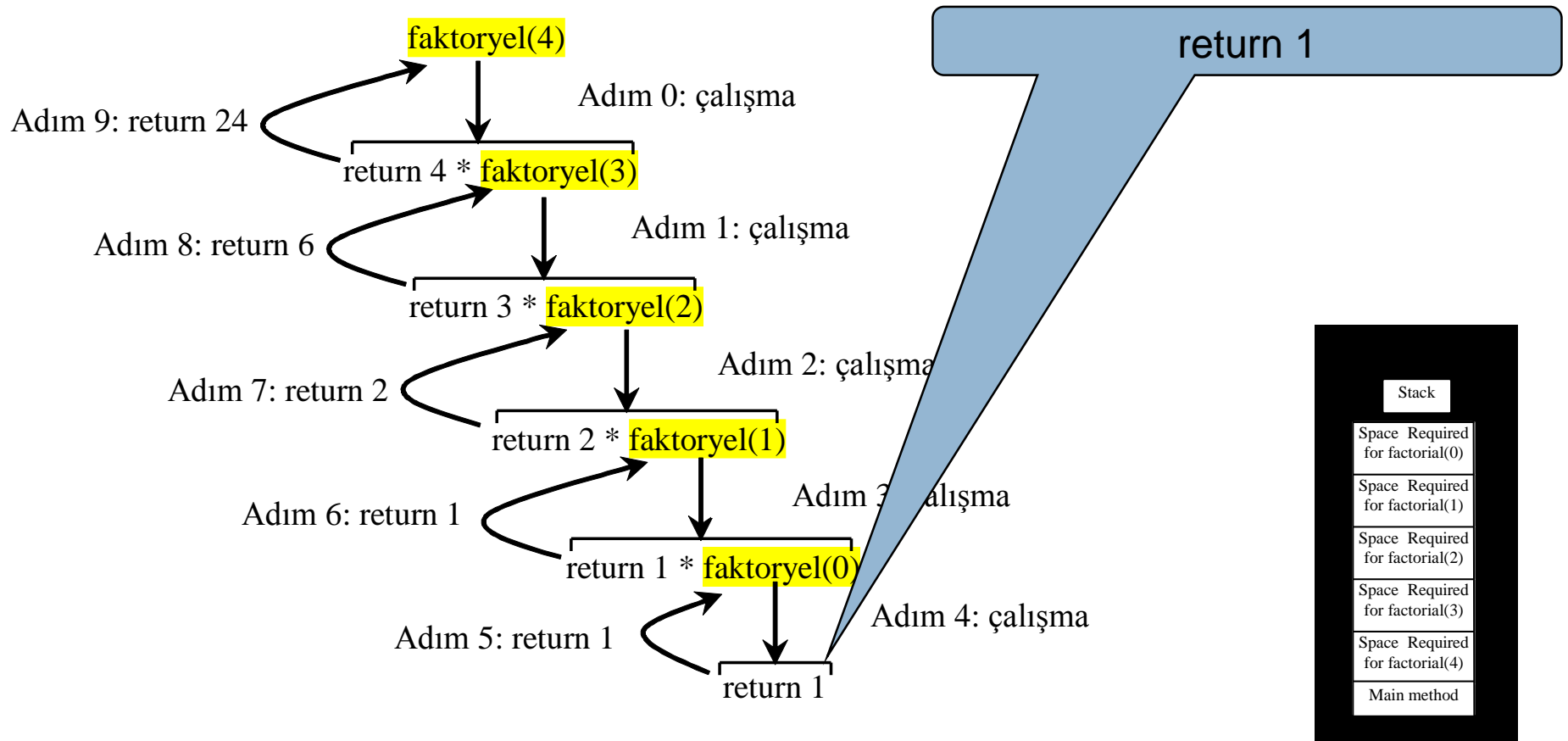
Recursive faktoryel



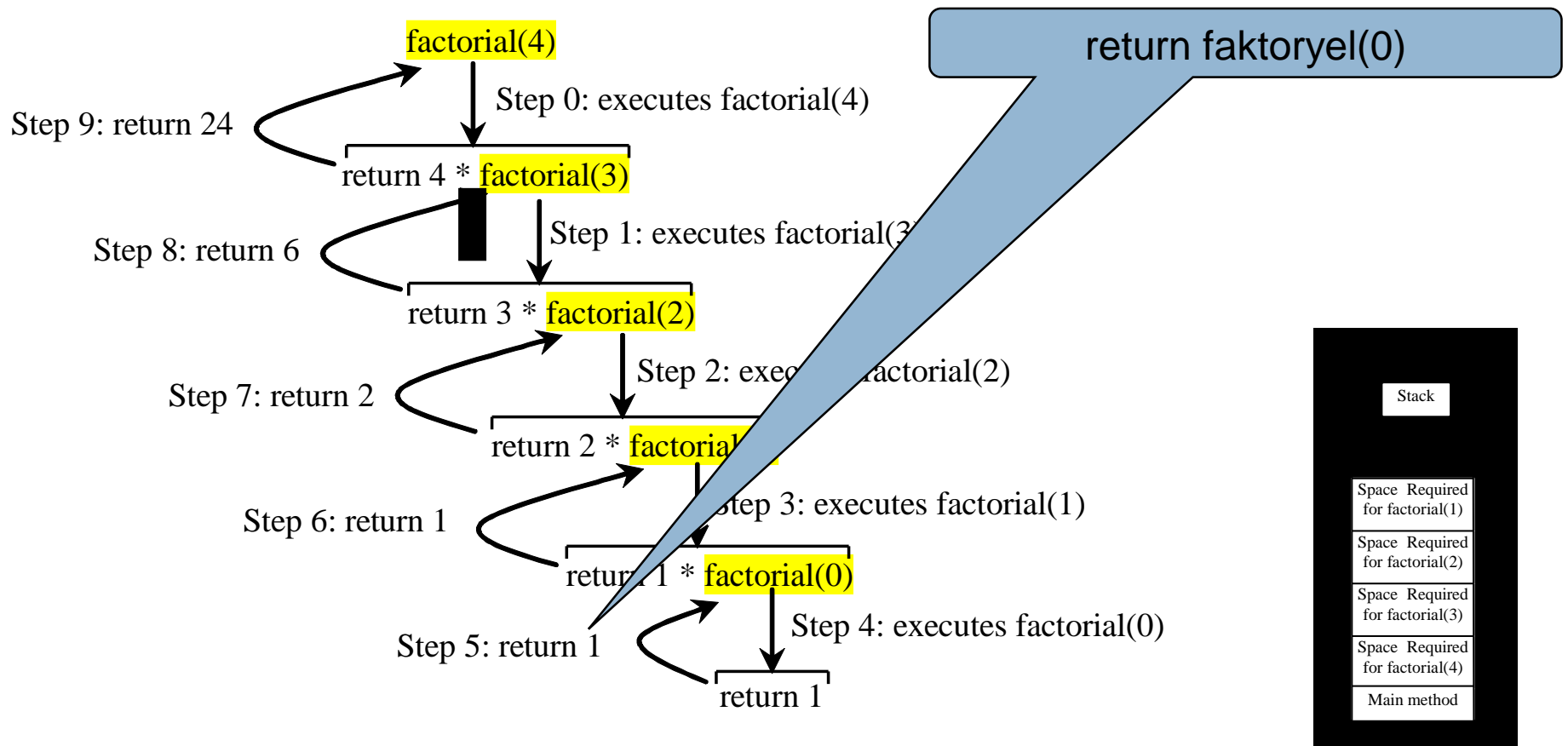
Recursive faktoryel



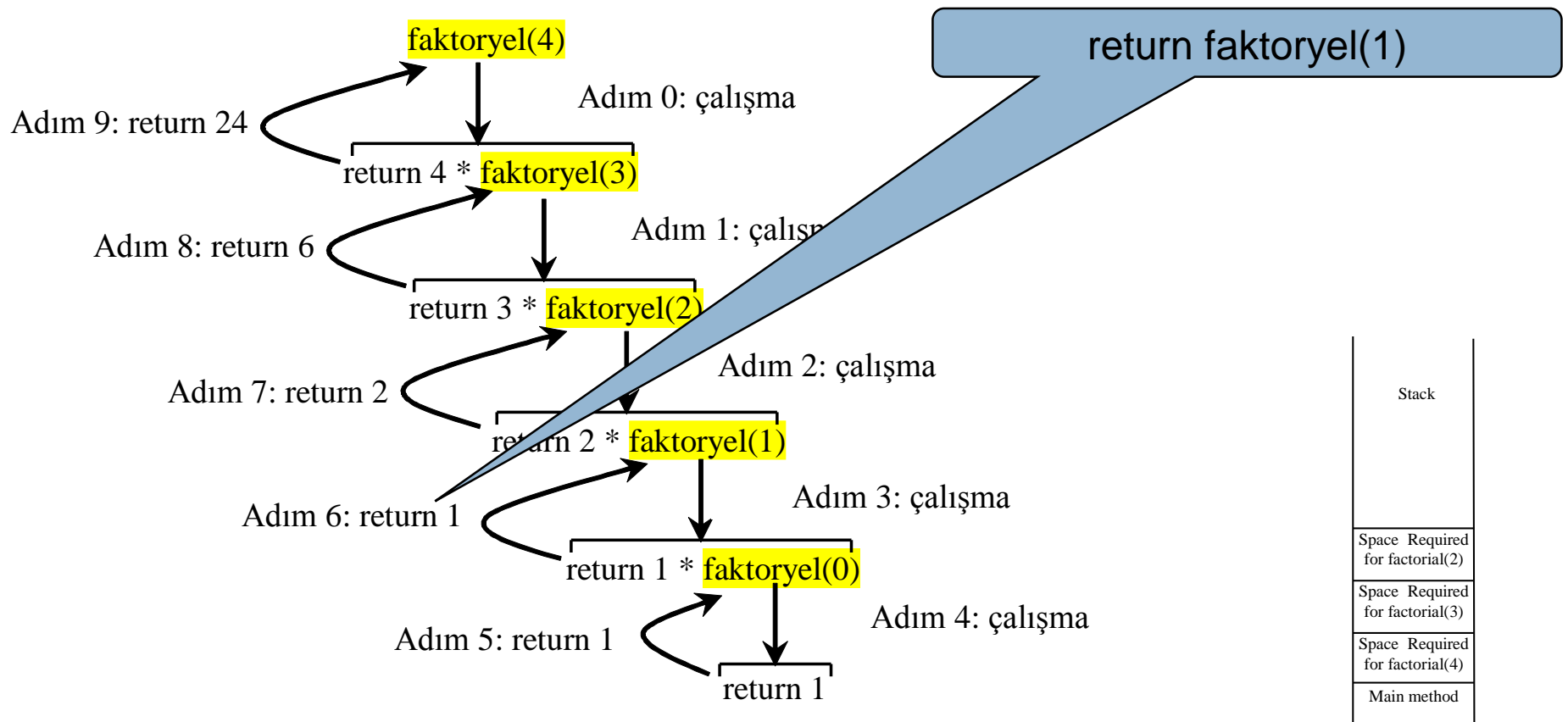
Recursive faktoryel



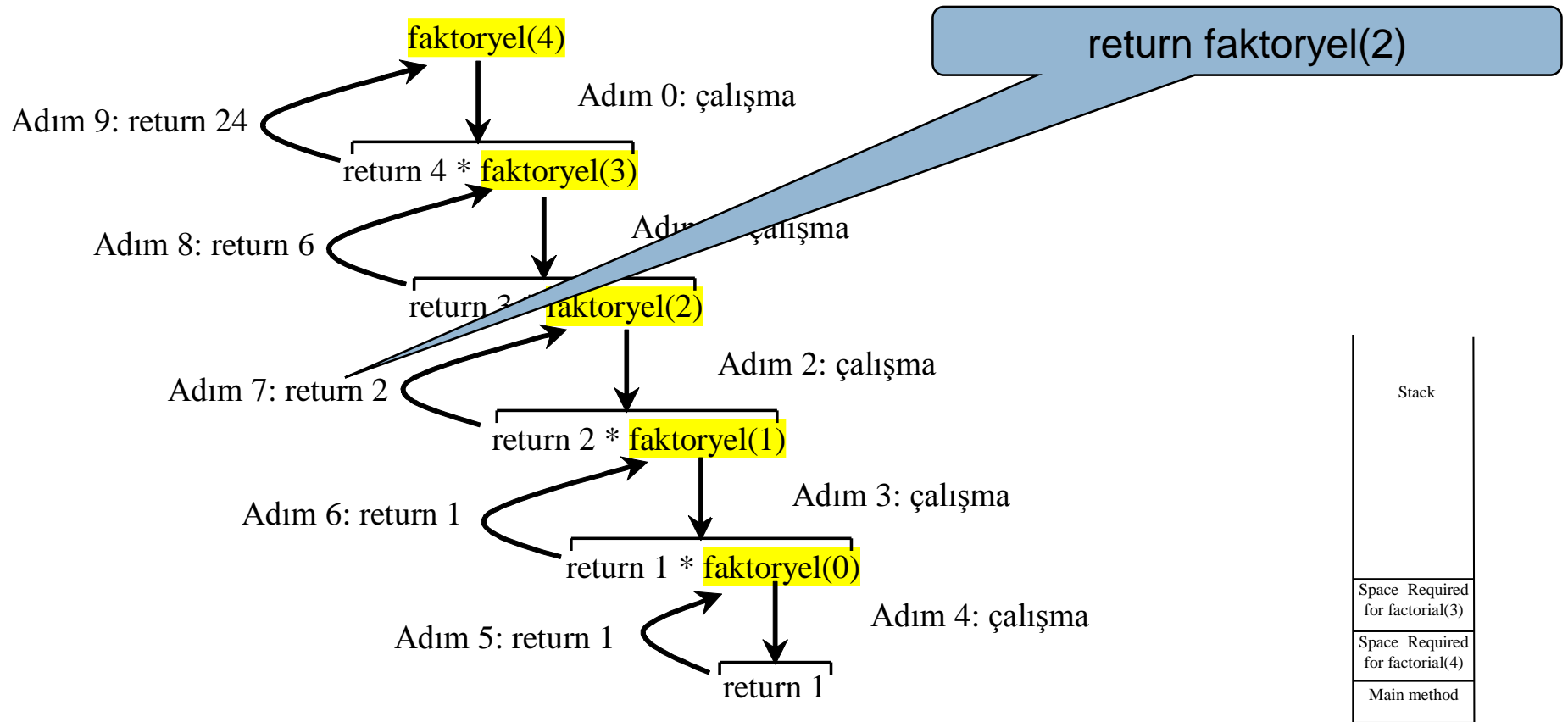
Recursive faktoryel



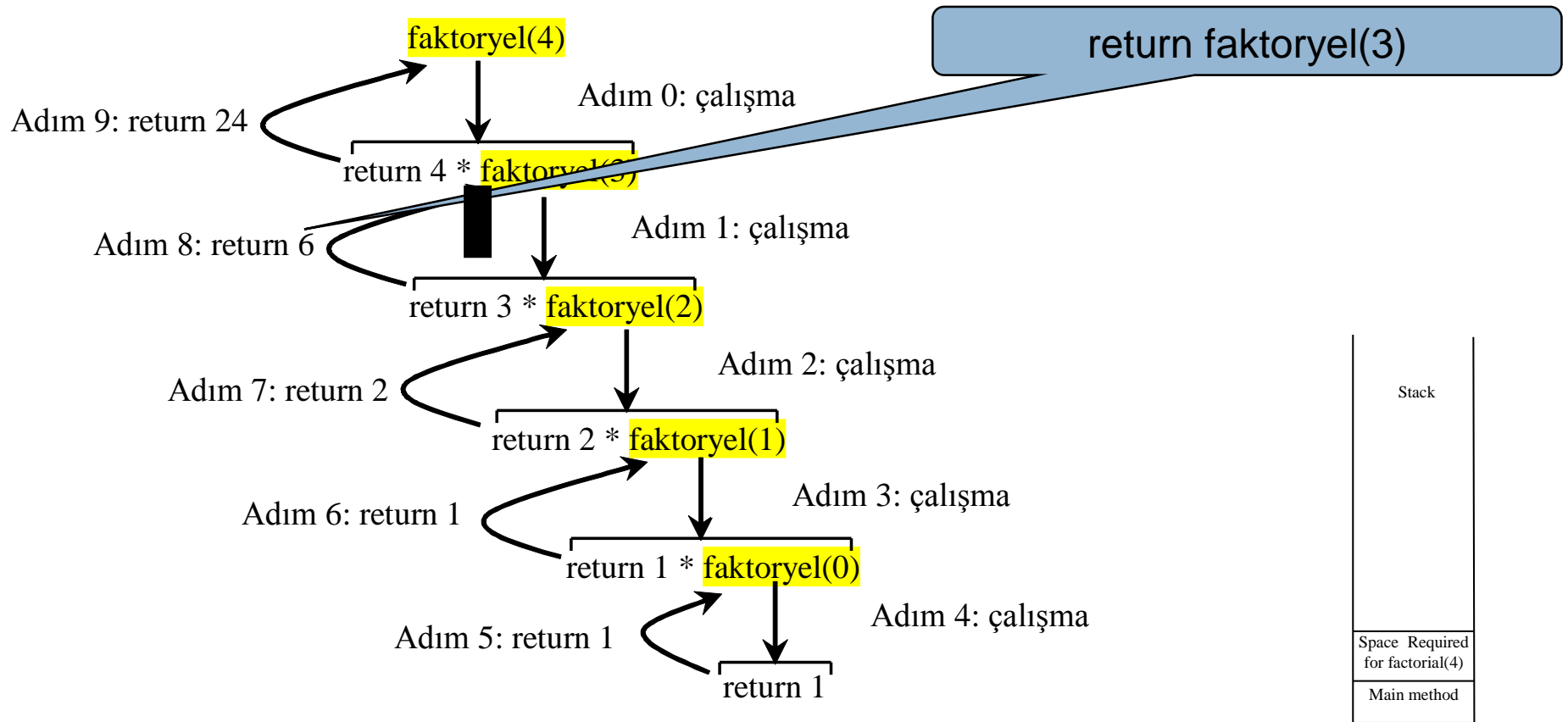
Recursive faktoryel



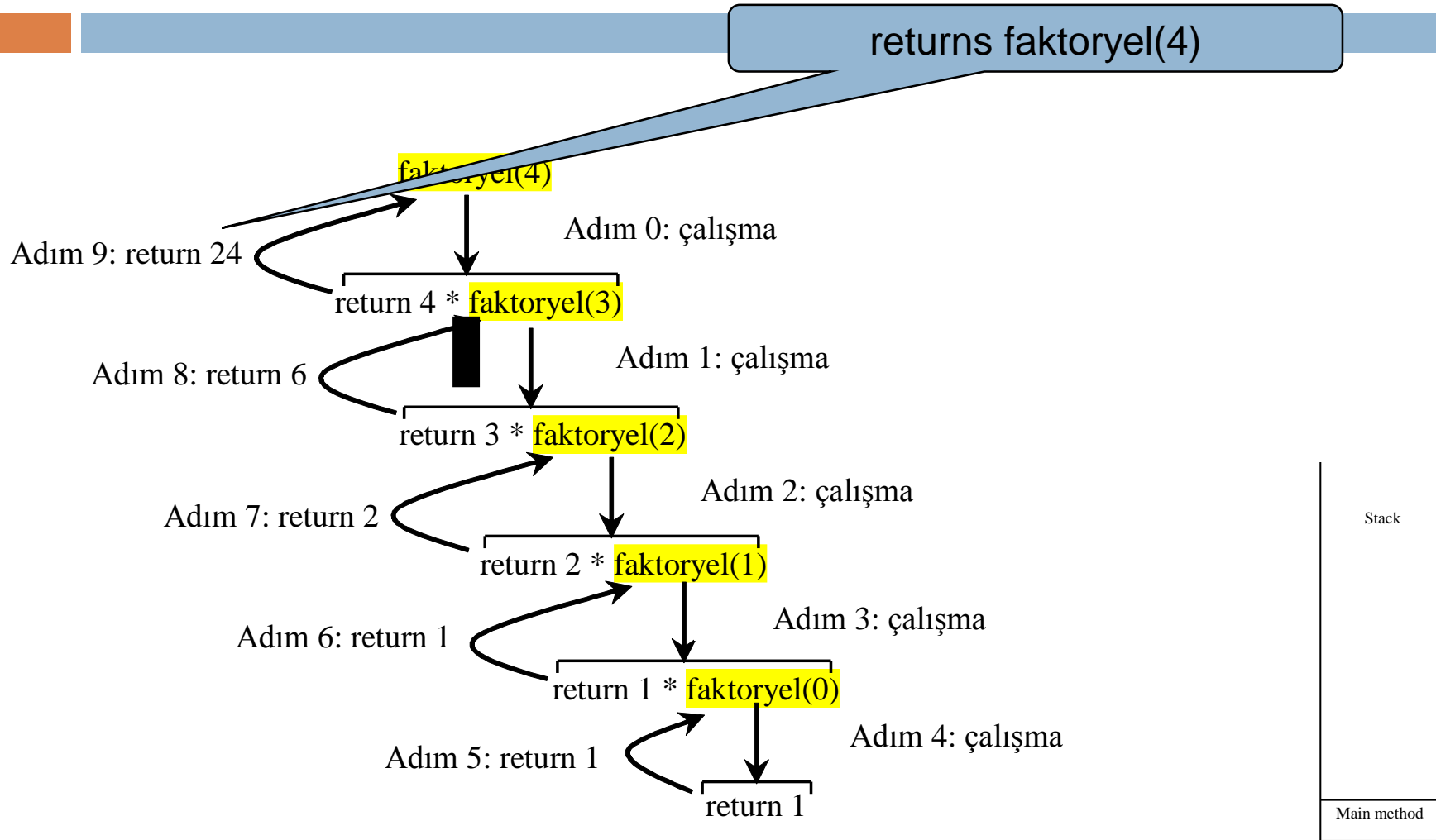
Recursive faktoryel



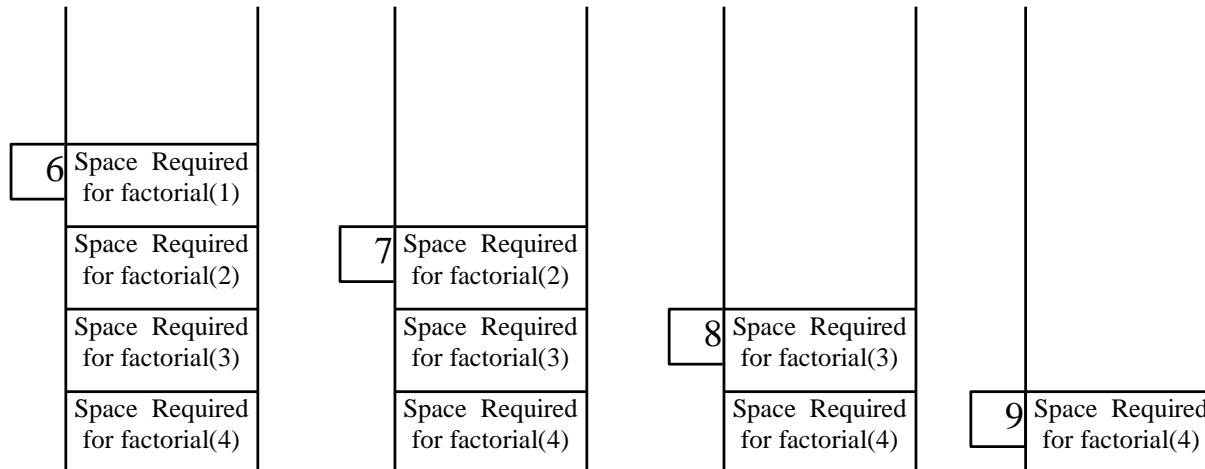
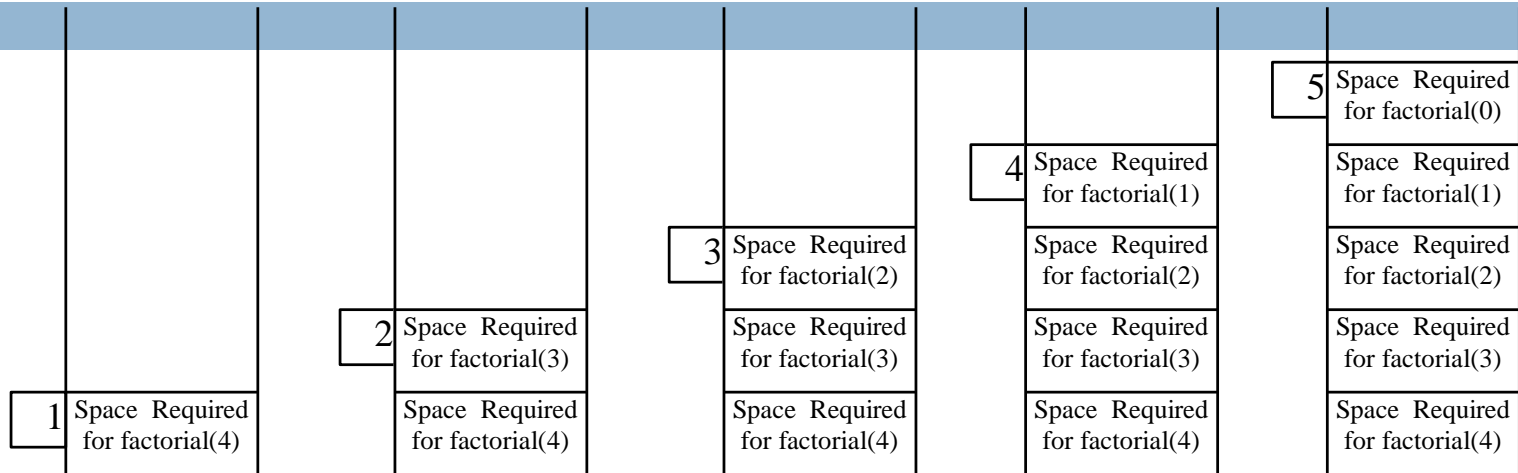
Recursive faktoryel



Recursive faktoryel



faktoryel(4) Stack



Basamak Sayısı

- Ozyinelemeli Tanim

$\text{basamak}(n) = 1$

-> if ($-9 \leq n \leq 9$)

$1 + \text{basamak}(n/10)$

-> degilse

- Ornek

$\text{basamak}(321) =$

$1 + \text{basamak}(321/10) = 1 + \text{basamak}(32) =$

$1 + [1 + \text{basamak}(32/10)] =$

$1 + [1 + \text{basamak}(3)] =$

$1 + [1 + (1)] =$

3

Basamak Sayısı

```
int basamakSayisi(int n) {  
    if ((-10 < n) && (n < 10))  
        return 1  
    else  
        return 1 +  
            basamakSayisi (n/10);  
}
```


Özyineleme

- $f(x)$ problemini çözmek istiyorsak fakat direkt olarak çözemiyorsak
- Varsayalım y 'nin x 'den küçük herhangi bir değeri için $f(y)$ 'yi çözebiliyoruz
- $f(x)$ 'i çözmek için $f(y)$ 'yi kullanırız
- Bu yöntemin çalışabilmesi için $f(x)$ 'in direkt olarak hesaplanabildiği en az bir x değerinin olması gerekir. (e.g. *taban durumu*)

Bir Sayının kuvvetini hesaplama

```
static int power(int k, int n) {  
    // k`n`ın n. üssü  
    if (n == 0)  
        return 1;  
    else  
        return k * power(k, n - 1);  
}
```

Toplam Hesaplama

```
public int sum(int num) {  
    int result;  
    if (num == 1) {  
        result = 1;  
    } else {  
        result = num + sum(num - 1);  
    }  
    return result;  
}
```

Faktoriyel

□ $5! = 5*4*3*2*1$

$$4! = 4*3*2*1$$

$$3! = 3*2*1$$

$$2! = 2*1$$

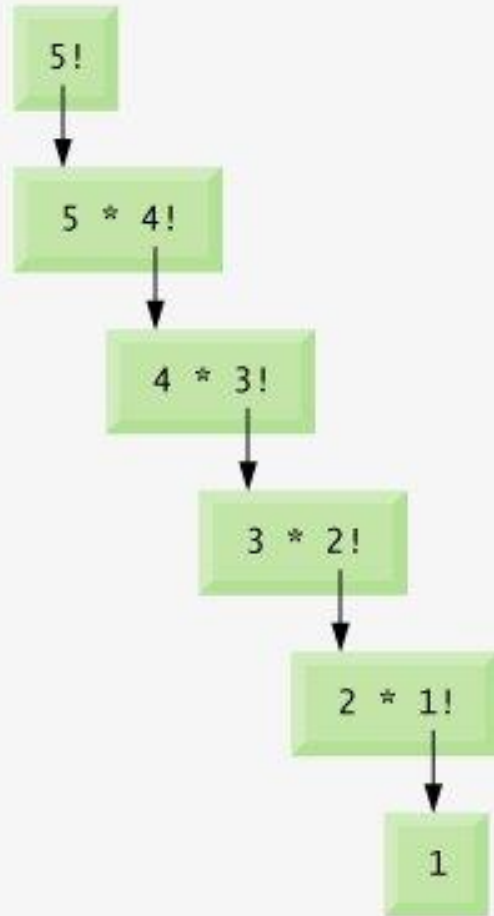
$$1! = 1$$

```
int faktoriyel = 1;  
for ( int sayac = 5; sayac >= 1; sayac-- )  
    faktoriyel = faktoriyel * sayac;
```

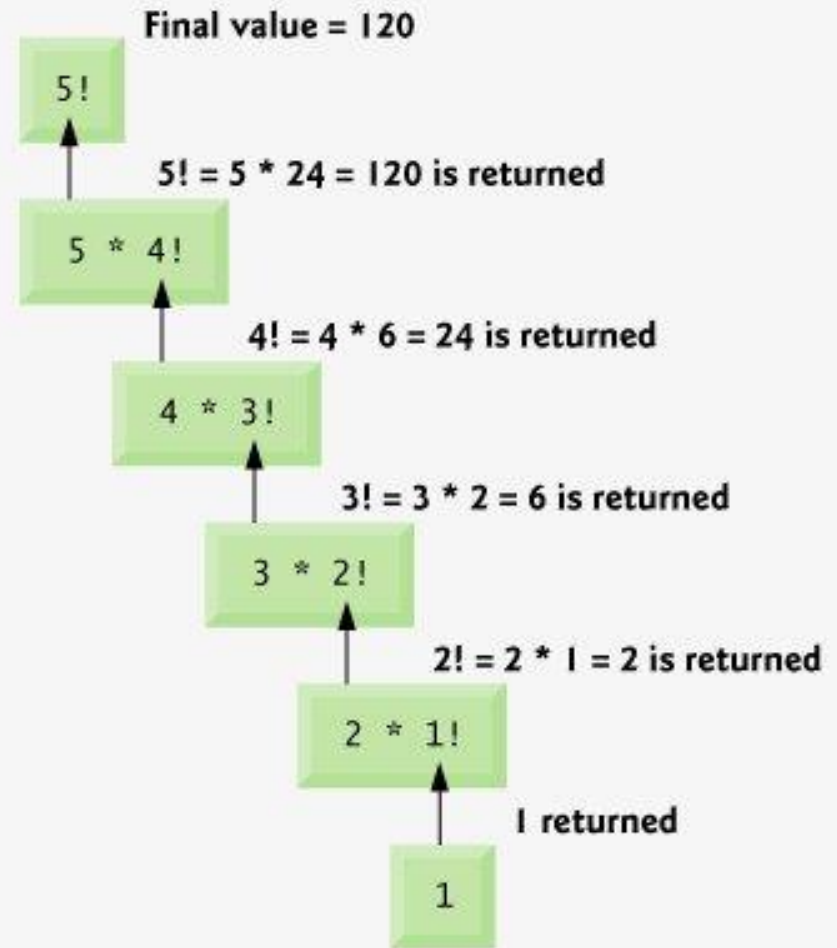
Özyinelemeli Faktöriyel

```
public int faktoriyel(int N) {  
    if (N == 0) return 1;  
    return N*faktoriyel(N-1);  
}
```

Recursive faktoryel



(a) Sequence of recursive calls.



(b) Values returned from each recursive call.

```
public double faktoriyelGoster(int n, int sayac) {  
    double fakt;  
    if (n <= 0) {  
        System.out.println("Taban Duruma ulasti");  
        fakt = 1;  
    } else {  
        sayac++;  
        System.out.println(sayac + ". program  
cagiriyor ");  
        fakt = n * faktoriyelGoster(n - 1, sayac);  
        //System.out.println("Faktoriyel = " + fakt);  
    }  
    System.out.println(sayac + ". programdan  
Cikiyor ");  
    sayac--;  
    return fakt;  
}
```

Faktoriyel programında ozyinelemeli metodların çağırılması

1. program cagiriyor
 2. program cagiriyor
 3. program cagiriyor
 4. program cagiriyor
 5. program cagiriyor
- Taban Duruma ulasti
5. programdan Cikiyor
 5. programdan Cikiyor
 4. programdan Cikiyor
 3. programdan Cikiyor
 2. programdan Cikiyor
 1. programdan Cikiyor

Fibonacci Sayıları

$$F(n) := \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F(n-1) + F(n-2) & \text{if } n > 1. \end{cases}$$

$$\text{fibonacci}(0) = 0$$

$$\text{fibonacci}(1) = 1$$

$$\text{fibonacci}(n) = \text{fibonacci}(n-1) + \text{fibonacci}(n-2)$$

Fibonacci Java

```
public int fibonacci(int sayi) {  
    if ( ( sayi == 0 ) || ( sayi == 1 )  
        return sayi;  
    else  
        return fibonacci( sayi - 1 ) + fibonacci( sayi - 2 );  
}
```

Fibonacci Metodunun Çalışması

