

Introduction to Java

Recursive Methods

متدهای بازگشتی

بازگشتی بودن فرآیندی است که در آن یک تابع به طور مکرر خودش را فراخوانی می‌کند تا آنکه به شرط خاصی برسد. استفاده از این روش جهت انجام محاسبات تکراری که در آن منطقی وجود دارد مناسب است. از نظر ریاضی، یک تابع بازگشتی یا خودگردی تابعی است که برحسب خودش تعریف می‌شود. به عنوان مثال فاکتوریل عدد صحیح و مثبت m ، که آن را با $m!$ نمایش می‌دهند، به شکل متداول به صورت زیر تعریف می‌گردد.

$$m! = m(m-1)(m-2) \dots 3. 2. 1$$

در ضمن فاکتوریل صفر برابر 1 است.

راه دیگر برای تعریف فاکتوریل که روش بازگشتی است این است که اگر $m = 1$ ، خروج؛ در غیر این صورت $m(m-1)!$.

مثال متد زیر مجموع اعداد طبیعی 1 تا n را به شکل بازگشتی محاسبه می کند.

```
Public static int sum (int n)
{
    if (n<=1)
        return (n);
    else
        return (n+sum (n-1));
}
```

مقدار بازگشتی	فراخوانی تابع
1	sum(1)
2+1 2+sum(1)	sum(2)
3+2+1 3+sum(2)	sum(3)
4+3+2+1 4+sum(3)	sum(4)

نحوه عملکرد این تابع در بعضی مقادیر n
در جدول زیر تجزیه تحلیل و نمایش داده
شده است.

مثال

محاسبه فاکتوریل n .

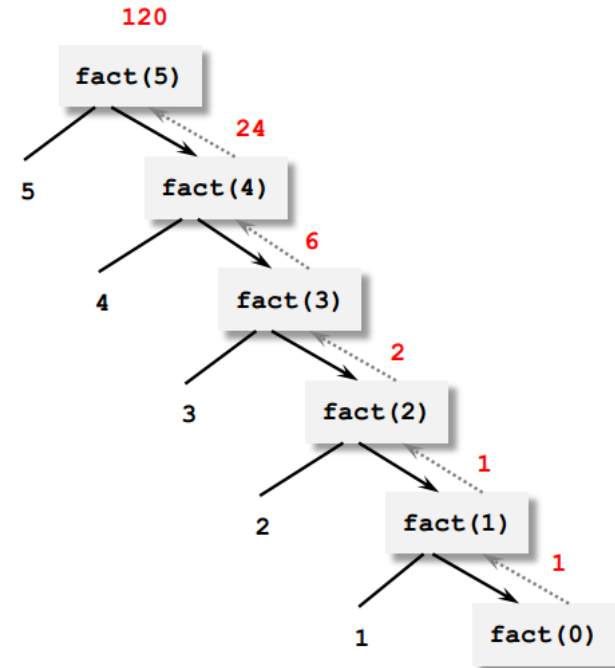
$$N! = 1 * 2 * \dots * (N - 1) * N$$

$$N! = \begin{cases} 1 & N = 0 \\ N \times (N - 1)! & N \geq 1 \end{cases}$$

```
public static long fact(int N)
{
    if (N == 0) return 1;
    else return N * fact(N - 1);
}
```

فتم بازگشتی

فراخوانی بازگشتی




مثال: بزرگ ترین مقسوم علیه مشترک

الگوریتم اقلیدس. [۳۰۰ سال قبل از میلاد]

$$\gcd(p, q) = \begin{cases} p & q = 0 \\ \gcd(q, p \% q) & \text{otherwise} \end{cases}$$

$$\begin{aligned} \gcd(4032, 1272) &= \gcd(1272, 216) \\ &= \gcd(216, 192) \\ &= \gcd(192, 24) \\ &= \gcd(24, 0) \\ &= 24 \end{aligned}$$

$$4032 = 3 * 1272 + 216$$


مثال: بزرگ ترین مقسوم علیه مشترک

الگوریتم اقلیدس. [۳۰۰ سال قبل از میلاد]

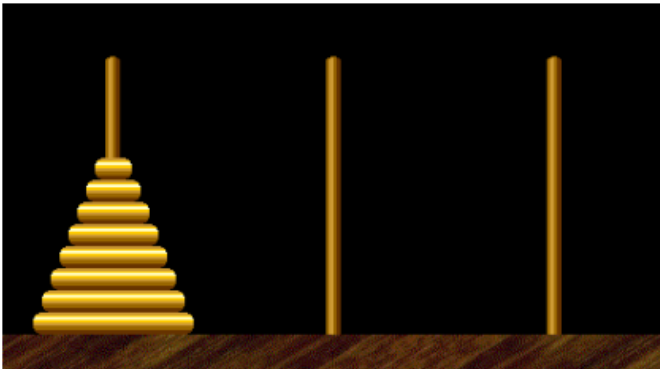
$$\text{gcd}(p, q) = \begin{cases} p & q = 0 \\ \text{gcd}(q, p \% q) & \text{otherwise} \end{cases}$$

```
public static int gcd(int p, int q)
{
    if (q == 0) return p;
    else return gcd(q, p % q);
}
```

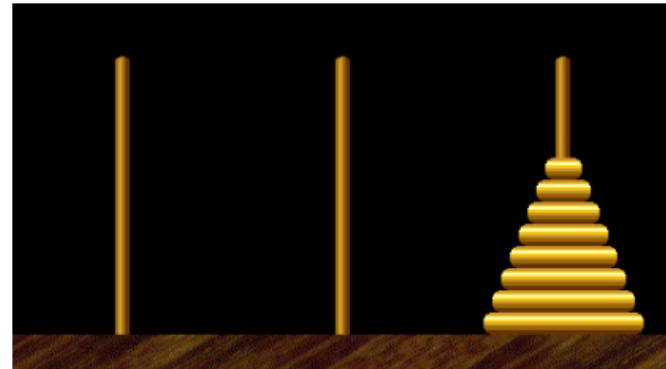
مثال: برج های هانوی

- هدف.** انتقال دیسک ها از میله سمت چپ به میله سمت راست با حفظ ترتیب.
- در هر حرکت تنها مجاز به انتقال یک دیسک هستیم.
 - هیچ گاه مجاز نیستیم یک دیسک بزرگ تر را بر روی یک دیسک کوچک تر قرار دهیم.

شروع



پایان



Java Method Overloading

With **method overloading**, multiple methods can have the same name with different parameters:

```
static int plusMethodInt(int x, int y) {  
    return x + y;  
}  
  
static double plusMethodDouble(double x, double y) {  
    return x + y;  
}  
  
public static void main(String[] args) {  
    int myNum1 = plusMethodInt(8, 5);  
    double myNum2 = plusMethodDouble(4.3, 6.26);  
    System.out.println("int: " + myNum1);  
    System.out.println("double: " + myNum2);  
}
```

Overloading Methods

- Java allows for method overloading.
- A Method is overloaded when the class provides several implementations of the same method, but with different parameters
 - The methods have the same name
 - The methods have differing numbers of parameters or different types of parameters
 - The return type MUST be the same

```
public float calculateInterestForMonth()
{
    return lowBalanceForMonth * (defaultRate/12.0);
}

public float calculateInterestForMonth(float rate)
{
    return lowBalanceForMonth * (rate/12.0);
}
```

```
static int plusMethod(int x, int y) {  
    return x + y;  
}
```

```
static double plusMethod(double x, double y) {  
    return x + y;  
}
```

```
public static void main(String[] args) {  
    int myNum1 = plusMethod(8, 5);  
    double myNum2 = plusMethod(4.3, 6.26);  
    System.out.println("int: " + myNum1);  
    System.out.println("double: " + myNum2);  
}
```

مثال

برنامه ای بنویسید که عدد x و pow را از ورودی دریافت کرده x را به توان

برساند. pow .