

函数

对象的操作

- `String s = "hello";`
- `int i = s.length();`
- `System.out.println(s+ "bye");`
- 这些都是对象在执行函数

素数求和

```
int m = in.nextInt();
int n = in.nextInt();
if ( m==1 ) m=2;
int cnt=0;
int sum=0;
for ( int i=m; i<=n; i++ )
{
    boolean isPrime = true;
    for ( int k=2; k<i; k++ )
    {
        if ( i % k == 0 )
        {
            isPrime = false;
            break;
        }
    }
    if ( isPrime )
    {
        cnt++;
        sum+=i;
    }
}
System.out.println("在"+m+"和"+n+"之间有"+cnt+"个素数，总和为"+sum);
```

```
public static boolean isPrime(int i)
{
    boolean isPrime = true;
    for ( int k=2; k<i; k++ )
    {
        if ( i % k == 0 )
        {
            isPrime = false;
            break;
        }
    }
    return isPrime;
}
```

```
int m = in.nextInt();
int n = in.nextInt();
if ( m==1 ) m=2;
int cnt=0;
int sum=0;
for ( int i=m; i<=n; i++ )
{
    if ( isPrime(i) )
    {
        cnt++;
        sum+=i;
    }
}
System.out.println("在"+m+"和"+n+"之间有"+cnt+"个素数");
```

求和

- 求出1到10、20到30和35到45的三个和

求和

```
int i;  
int sum;  
sum=0;  
for ( i=1; i<=10; i++ )  
{  
    sum += i;  
}  
System.out.println(1+"到"+10+"的和是"+sum);
```

```
sum=0;  
for ( i=20; i<=30; i++ )  
{  
    sum += i;  
}  
System.out.println(20+"到"+30+"的和是"+sum);
```

```
sum=0;  
for ( i=35; i<=45; i++ )  
{  
    sum += i;  
}  
System.out.println(35+"到"+45+"的和是"+sum);
```

“代码复制”是程序质量不良的表现

- 三段几乎一模一样的代码！

求和函数

```
int i;
int sum;
sum=0;
for ( i=1; i<=10; i++ )
{
    sum += i;
}
System.out.println(1+"到"+10+"的和是"+sum);
sum=0;
for ( i=20; i<=30; i++ )
{
    sum += i;
}
System.out.println(20+"到"+30+"的和是"+sum);
sum=0;
for ( i=35; i<=45; i++ )
{
    sum += i;
}
System.out.println(35+"到"+45+"的和是"+sum);
```

```
public static void sum(int a, int b)
{
    int i;
    int sum=0;
    for ( i=a; i<=b; i++ )
    {
        sum += i;
    }
    System.out.println(a+"到"+b+"的和是"+sum);
}

public static void main(String[] args) {
    sum(1,10);
    sum(20,30);
    sum(35,45);
}
```

什么是函数？

- 函数是一块代码，接收零个或多个参数，做一件事情，并返回零个或一个值
- 可以先想像成数学中的函数：
 - $y = f(x)$

函数定义

返

函

参

函

函

```
public static void sum(int a, int b)
```

```
{
```

```
    int i;
```

```
    int sum=0;
```

```
    for ( i=a; i<=b; i++ )
```

```
    {
```

```
        sum += i;
```

```
    }
```

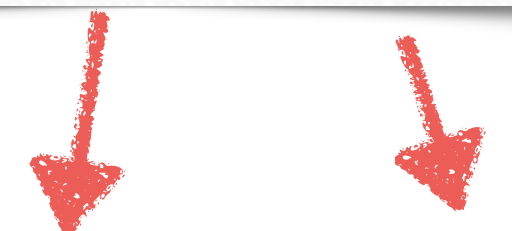
```
    System.out.println(a+"到"+b+"的和是"+sum);
```

```
}
```


调用函数

- 函数名(参数值);
- ()起到了表示函数调用的重要作用
- 即使没有参数也需要()
- 如果有参数，则需要给出正确的数量和顺序

```
sum(1,10);  
sum(20,30);  
sum(35,45);
```



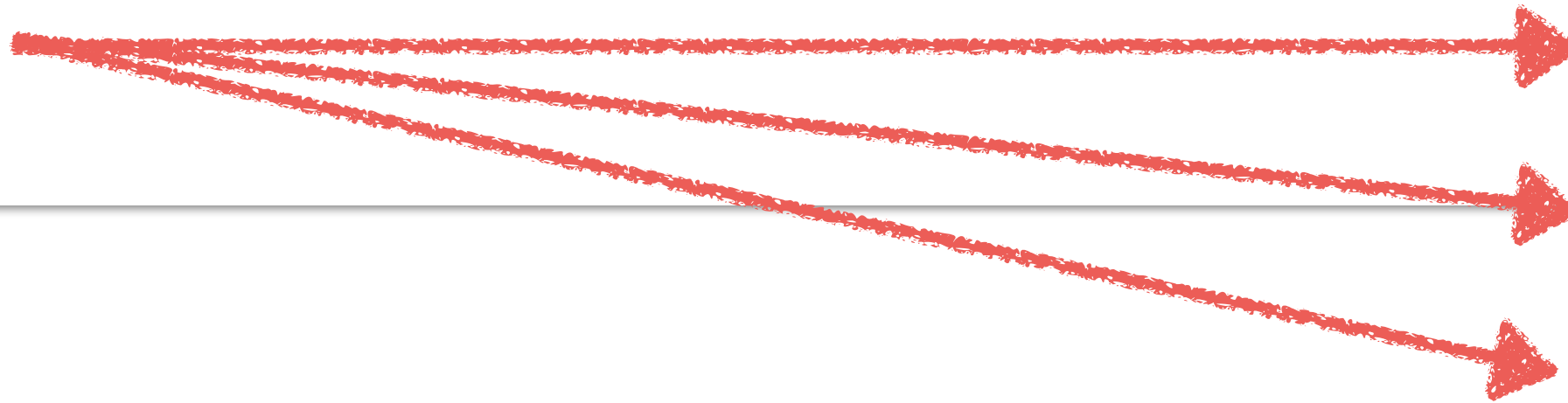
```
public static void sum(int a, int b)  
{  
    int i;  
    int sum=0;  
    for ( i=a; i<=b; i++ )  
    {  
        sum += i;  
    }  
    System.out.println(a+"到"+b+"的和是"+sum);  
}
```

函数返回

- 函数知道每一次是哪里调用它，函数结束的时候会返回到正确的地方

```
public static void sum(int a, int b)
{
    int i;
    int sum=0;
    for ( i=a; i<=b; i++ )
    {
        sum += i;
    }
    System.out.println(a+"到"+b+"的和")
}
```

```
sum(1,10);
sum(20,30);
sum(35,45);
```

Three red arrows originate from the closing curly brace of the 'sum' function in the code block above. They point to the three lines of function calls in the block below: 'sum(1,10);', 'sum(20,30);', and 'sum(35,45);'. This illustrates how the function knows where to return after each call.

从函数中返回值

```
public static int max(int a, int b)
{
    int ret;
    if ( a>b )
    {
        ret = a;
    }
    else
    {
        ret = b;
    }
    return ret;
}
```

- **return** 停止函数的执行，并送回一个值
- **return**;
- **return** 表达式;
- 一个函数里可以出现多个**return** 语句，但是保持单一出口是好的做法

从函数中返回值

```
public static int max(int a, int b)
{
    int ret;
    if ( a>b )
    {
        ret = a;
    }
    else
    {
        ret = b;
    }
    return ret;
}
```

```
int a =5;
int b =6;
int c;
c = max(10,12);
c = max(a,b);
c = max(c, 23);
c = max(max(c,a), 5);
System.out.println(max(a,b));
max(12,13);
```

- 可以赋值给变量
- 可以再传递给函数
- 甚至可以丢弃
- 有的时候要的是副作用

没有返回值的函数

- void 函数名(参数表)
- 不能使用带值的return
- 可以没有return
- 调用的时候不能做返回值赋值

如果函数有返回值，则必须使用带值的return

```
public static void sum(int a, int b)
{
    int i;
    int sum=0;
    for ( i=a; i<=b; i++ )
    {
        sum += i;
    }
    System.out.println(a+"到"+b+"的和是"+sum);
}
```