参数传递

调用函数

- 如果函数有参数，调用函数时必须传递给它数量、类型正确的值
- 可以传递给函数的值是表达式的结果，这包括：
  - 字面量
  - 变量
  - 函数的返回值
  - 计算的结果

```
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);
```

类型不匹配?

- 当函数期望的参数类型比调用函数时给的值的类型宽的时候，编译器能悄悄替你把类型转换好
  - char—>int—>double
- 当函数期望的参数类型比调用函数时给的值的类型窄的时候，需要你写强制类型转换
  - (int)5.0
- 当函数期望的参数类型与调用函数时给的值的类型之间无法转换的时候—>不行!

传过去的是什么?

```
public static void swap(int a, int b)
{
  int t;
  t = a;
  a = b;
  b = t;
}

public static void main(String[] args) {
  int a = 5;
  int b = 6;
  swap(a,b);
}
```

Java语言在调用函数时，永远只能传值给函数

- 这样的代码能交换a和b的值吗?

传值

- 每个函数有自己的变量空间，参数也位于这个独立的空间中，和其他函数没有关系
- 过去，对于函数参数表中的参数，叫做“形式参数”，调用函数时给的值，
- 由于容易让初学者误会实际参数就是实际在函数中进行计算的参数，误会调用函数的时候把变量而不是值传进去了，所以我们不建议继续用这种古老的方式来称呼它们
- 我们认为，它们是参数和值的关系

当参数类型是数组或对象时?

参

```
public static void swap(int a, int b)
{
  int t;
  t = a;
  a = b;
  b = t;
}

public static void main(String[] args) {
  int a = 5;
  int b = 6;
  swap(a,b);
}
```

值

本地变量

本地变量

- 函数的每次运行，就产生了一个独立的变量空间，在这个空间中的变量，是函数的这次运行所独有的，称作本地变量
- 定义在函数内部的变量就是本地变量
- 参数也是本地变量

变量的生存期和作用域

- 生存期：什么时候这个变量开始出现了，到什么时候它消亡了
- 作用域：在（代码的）什么范围内可以访问这个变量（这个变量可以起作用）
- 对于本地变量，这两个问题的答案是统一的：大括号内——块

本地变量的规则

- 本地变量是定义在块内的
  - 它可以是定义在函数的块内
  - 也可以定义在语句的块内
  - 甚至可以随便拉一对大括号来定义变量
- 程序运行进入这个块之前，其中的变量不存在，离开这个块，其中的变量就消失了
- 块外面定义的变量在里面仍然有效
- 不能在一个块内定义同名的变量，也不能定义块外面定义过的变量
- 本地变量不会被默认初始化
- 参数在进入函数的时候被初始化了

```c
int main()
{
    int scp1; // scp1 出现了
    {   // scp1 还是存在的
        int scp2;
        // scp2 出现了
        {   // scp1 与 scp2 都还是存在的
            int scp3; // scp1、scp2 和 scp3 都存在
        }
        // <-- scp3 不存在了
        // scp1 与 scp2 还存在
    }
    // <-- scp2 不存在了
    // scp1 还存在
}
```

```
void f(int k)
{
  int i=10;
  if ( k>i ) {
     int k = 12;  //不行
     int j = i+k;
  }
  int m = j;   //不行
}
```