\r 表示接受键盘输入，相当于按下了回车键；

\n 表示换行；

\t 表示制表符，相当于 Table 键；

\b 表示退格键，相当于 Back Space；

\'表示单引号；

\"表示双引号；

\\表示一个斜杠 "\" 。

比如上面的"Welcome \n XXX",它的运行结果是：

Welcome

XXX

2.2.1 变量的概念

```
int x=0,y;
y=x+3;
```

第一句代码分配了两块内存用于存储整数，分别用 x，y 作为这两块内存的变量名，并将 x 标识的内存中的数据置为 0，y 标识的内存中的数据为其原始状态，可以认为是一个未知数。第二句代码的执行过程，程序首先取出 x 代表的那块内存单元的数，加上 3，然后把结果放到 y 所在的那块内存单元，这样就完成了 y=x+3 的运算。

2.2.2 Java 的变量类型

在 Java 中内建有 8 种基本变量类型来存储整数、浮点数、字符和布尔值。

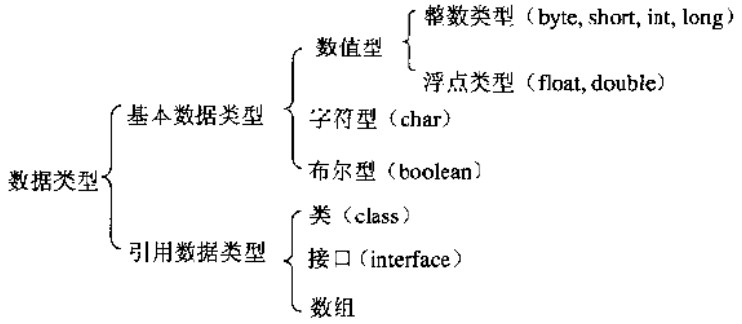


表 2.1

类 型 名	大小/位	取值范围
byte	8	-128~127
short	16	-32768~32767
int	32	-2147483648~2147483647
long	64	-9223372036854775808~9223372036854775807

变量的作用域

```
int x = 12;
        {
            int q = 96;  // x和q都可用
            System.out.println("x is "+x);
            System.out.println("q is "+q);
        }
    q = x;              /* 错误的行，只有x可用， q 超出了作用域范围 */
      System.out.println("x is "+x);
}
}
```

q 作为在里层的代码块中定义的一个变量，只有在那个代码块中位于这个变量定义之后的语句，才可使用这个变量，q=x 语句已经超过了 q 的作用域，所以编译无法通过。记住这样的一个道理，在定义变量的语句所属的那层大括号之间，就是这个变量的有效作用范围，但不能违背变量先定义后使用的原则。

2.2.7 局部变量的初始化

在一个函数或函数里面的代码块中定义的变量称为局部变量，局部变量在函数或代码块被执行时创建，在函数或代码块结束时被销毁。局部变量在进行取值操作前必须被初始化或进行过赋值操作，否则会出现编译错误，如下面的代码：

传参

对上面定义的“参数类型 形式参数 1，参数类型 形式参数 2，…”部分，称之为参数列表。在程序中调用某个函数、执行其中的程序代码时，有时候需要给函数传递一些参数，譬如，有一个实现了两个数相加的函数被调用时，就必须给这个函数传递那两个要相加的数。函数要接收调用程序传递进来的参数，必须为每个传递进来的参数定义一个变量，这些变量就在函数名后面的一对小括号中进行定义，各个变量之间以逗号（，）隔开，在小括号中定义的这些变量就叫参数列表。有的函数并不需要接收任何参数，即使这样，在定

```
public static int getArea(int x,int y)
{
    if(x<=0|y<=0)
    return -1;
    return x*y;
}
```

确定传参的 合法性

2.3.2 函数的参数传递过程

下面来看一下函数的参数传递过程，如图 2.4 所示。

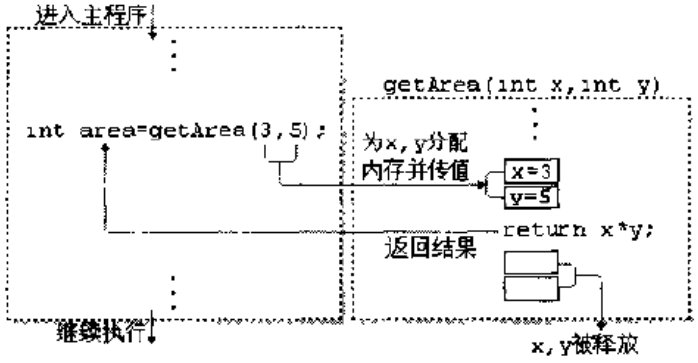


图 2.4

形式参数 x 和 y 就相当于函数 getArea 中定义的两个局部变量，在函数被调用时创建，并以传入的实参作为初始值，函数调用结束时也就被释放了，不会影响到主程序中其他的 x 和 y（如果有的话），因为它们属于不同作用域中的变量，它们是互不相干的变量。

2.3.3 函数的重载

函数的重载就是在同一个类中允许同时存在一个以上的同名函数，只要它们的参数个

数或类型不同即可。在这种情况下，该函数就叫被重载（overloaded）了，这个过程称为函数的重载（method overloading）。

数。重载函数的参数列表必须不同，要么是参数的个数不同，要么是参数的类型不同。重载函数的返回值类型可以相同，也可以不同。

算法

思考题 2：假设你要让 x 的值在 0 至 9 之间循环变化，请写出相应的程序代码。

答案：

```
int x=0;
while(true)
{
x = (x+1)%10;
}
```

这样，x 的取值总在 0 与 9 之间循环。

思考题 1：某个培训中心要为新到的学员安排房间，假设共有 x 个学员，每个房间可以住 6 人，让你用一个公式来计算他们要住的房间数（千万不要像我以前的有些学员开玩笑，说男生和女生是不能分在一块的，我们就不在这考虑了）。

答案：（x+5）/6。这种算法还可用在查看留言板的分页显示上，其中 x 是总共的留言数，6 是每页显示的留言数，结果就是总共有多少页。

赋值运算符

注意：在 Java 里可以把赋值语句连在一起，如：

x = y = z = 5；

在这个语句中，所有三个变量都得到同样的值 5。

“+=”是将变量与所赋的值相加后的结果再赋给该变量，如 x+=3 等价于 x=x+3。

异或

XOR 运算符叫做异或，只有当“^”连接的两个布尔表达式的值不相同时，该组合才返回 true 值。如果两个都是 true 或都是 false，该组合将返回 false 值。

优先级

表 2.7

高	. [] () ; ,
↑	++ -- ~ ! (数据类型)
	* / %
	+ -
	<< >> >>>
	< > <= >=
优	== !=
先	&
级	^
	|
	&&
	||
↓	?:
低	= *= /= %= += -= <<= >>= >>>= &= ^= |=

If 语句的简写

对于 if…else…语句，还有一种更简洁的写法：

变量 = 布尔表达式? 语句 1:语句 2;

下面的代码：

```
if(x>0)
y=x;
else
y=-x;
```

可以简写成：

```
y = x>0?x:-x;
```

这是一个求绝对值的语句，如果 x>0，就把 x 赋值给变量 y，如果 x 不大于 0，就把-x 赋值给前面的 y。也就是：如果问号“?”前的表达式为真，则计算问号和冒号中间的表达式，并把结果赋值给变量 y，否则将计算冒号后面的表达式，并把结果赋值给变量 y，这种写法的好处在于代码简洁，并且有一个返回值。

Swith 语句

2.5.3 switch 选择语句

switch 语句用于将一个表达式的值同许多其他值比较，并按比较结果选择下面该执行哪些语句，switch 语句的使用格式如下：

```
switch(表达式)
{
case 取值1:
    语句块1
    Break;
…
case 取值n:
    语句块n
    break;

default:
    语句块n+1
    break;
}
```

```
int x=2;
switch(x)
{
    case 1:
       System.out.println("Monday");
      break;
    case 2:
      System.out.println("Tuesday");
      break;
    case 3:
      System.out.println("Wednesday");
      break;
    default:
      System.out.println("Sorry,I don't Know");
}
```

switch 语句判断条件可以接受 int，byte，char，short 型，

swith 接受的数据类型

2.5.7 break 与 continue 语句

2．continue 语句

continue 语句只能出现在循环语句（while，do，for）的子语句块中，无标号的 continue 语句的作用是跳过当前循环的剩余语句块，接着执行下一次循环。

请看下面打印 1 到 10 之间的所有奇数的例子，当 i 是偶数时就跳过本次循环后的代码，直接执行 for 语句中的第三部分，然后进入下一次循环的比较，是奇数就打印 i。

程序清单：PrintOddNum.java

```
public class PrintOddNum
{
    public static void main(String [] args)
    {
      for(int i=0;i<10;i++)
      {
          if(i%2==0)
            continue;
          System.out.println(i);
        }
    }
}
```

多学两招:

为了充分和深入了解数组，必须向大家讲解有关内存分配的一些背后知识。Java 把内存划分成两种：一种是栈内存，另一种是堆内存。

在函数中定义的一些基本类型的变量和对象的引用变量都是在函数的栈内存中分配，当在一段代码块（也就是一对花括号{}之间）定义一个变量时，Java 就在栈中为这个变量分配内存空间，当超过变量的作用域后，Java 会自动释放掉为该变量所分配的内存空间，该内存空间可以立即被另作他用，先前讲到的知识都属于栈中分配的变量。

堆内存用来存放由 new 创建的对象和数组，在堆中分配的内存，由 Java 虚拟机的自动垃圾回收器来管理（关于自动垃圾回收器请参看第 1 章中的垃圾回收器介绍）。在堆中产生了一个数组或对象后，还可以在栈中定义一个特殊的变量，让栈中的这个变量的取值等于数组或对象在堆内存中的首地址，栈中的这个变量就成了数组或对象的引用变量，以后就可以在程序中使用栈中的引用变量来访问堆中的数组或对象，引用变量就相当于是为数组或对象起的一个名称（叫代号也行）。引用变量是普通的变量，定义时在栈中分配，引用变量在程序运行到其作用域之外后被释放，而数组和对象本身在堆中分配，即使程序运行到使用 new 产生数组和对象的语句所在的代码块之外，数组和对象本身占据的内存不会被释放，数组和对象在没有引用变量指向它时，才会变为垃圾，不能再被使用，但仍然占据内存空间不放，在随后一个不确定的时间被垃圾回收器收走（释放掉）。这也是 Java 比较占内存的原因。

```
int x[];          //定义了一个数组 x，这条语句执行完后的内存状态如图 2.13 所示
x=new int[100]; //数组初始化，这条语句执行完后的内存状态如图 2.14 所示
```

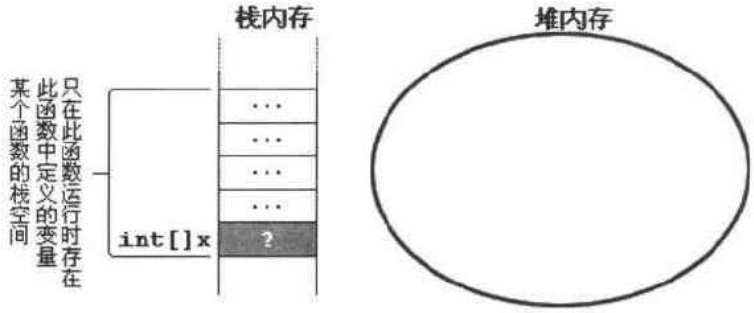
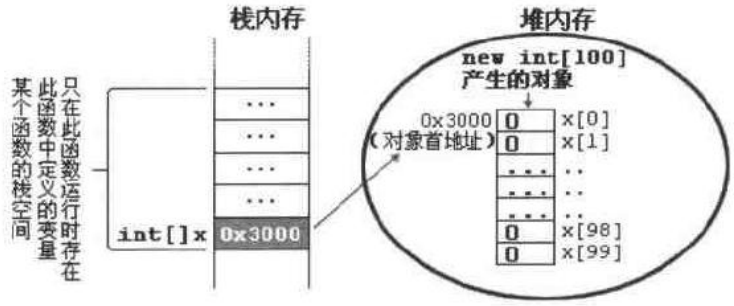


图 2.13



1. 使用 System.arraycopy()函数复制数组

例如，System.arraycopy(source, 0, dest, 0, x); 语句的意思就是：复制源数组中从下标 0 开始的 x 个元素到目的数组，从目标数组的下标 0 所对应的位置开始存储。下面是使用

关于 Arrays.sort 方法的帮助信息，读者可以自己动手查阅 JDK 文档解决了吧！