第4章 数学函数、字符和字符串

目 录

contents

4.1 常用数学函数

Math是final类：在java.lang.Math中，所有数学函数都是静态方法

- Math类中定义了常用的数学常量，如
  - PI : 3.14159265358979323846
  - E : 2.7182818284590452354（自然对数的底）
- 方法:注意都是静态函数
  - 三角函数
    sin, cos, tan, asin, acos, atan,toRadians,toDigrees
  - 指数
    exp, log, log10，pow, sqrt
  - 取整
    ceil, floor, round
  - 其它
    min, max, abs, random（[0.0,1.0)）

4.1 常用数学函数

Math的三角函数

Math 类包含表 4-1 中所示的三角函数方法。

表 4-1 Math 类中的三角函数方法

方法	描述
sin(radians)	返回以弧度为单位的角度的三角正弦函数值
cos(radians)	返回以弧度为单位的角度的三角余弦函数值
tan(radians)	返回以弧度为单位的角度的三角正切函数值
toRadians(degree)	将以度为单位的角度值转换为以弧度表示
toDegrees(radians)	将以弧度为单位的角度值转换为以度表示
asin(a)	返回以弧度为单位的角度的反三角正弦函数值
acos(a)	返回以弧度为单位的角度的反三角余弦函数值
atan(a)	返回以弧度为单位的角度的反三角正切函数值

sin、cos 和 tan 的参数都是以弧度为单位的角度。asin 和 atan 的返回值是 $-\pi/2 \sim \pi/2$ 的一个弧度值，acos 的返回值在 0 到 π 之间。1° 相当于 $\pi/180$ 弧度，90° 相当于 $\pi/2$ 弧度，而 30° 相当于 $\pi/6$ 弧度。

4.1 常用数学函数

Math的指数函数

Math 类中有 5 个与指数函数有关的方法，如表 4-2 所示。

例如，

```
Math.exp(1) 返回 2.71828
Math.log(Math.E) 返回 1.0
Math.log10(10) 返回 1.0
Math.pow(2, 3) 返回 8.0
Math.pow(3, 2) 返回 9.0
Math.pow(4.5, 2.5) 返回 22.91765
Math.sqrt(4) 返回 2.0
Math.sqrt(10.5) 返回 4.24
```

表 4-2 Math 类中的指数函数方法

方法	描述
exp(x)	返回 e 的 x 次方
log(x)	返回 x 的自然底数
log10(x)	返回 x 的以 10 为底的对数
pow(a, b)	返回 a 的 b 次方
sqrt(x)	对于 x ⩾ 0 的数字，返回 x 的平方根

4.1 常用数学函数

Math的取整函数

Math 类包括五个取整方法，如表 4-3 所示。

表 4-3 Math 类中的取整方法

方法	描述
ceil(x)	x 向上取整为它最接近的整数。该整数作为一个双精度值返回
floor(x)	x 向下取整为它最接近的整数。该整数作为一个双精度值返回
rint(x)	x 取整为它最接近的整数。如果 x 与两个整数的距离相等，偶数的整数作为一个双精度值返回
round(x)	如果 x 是单精度数，返回（int）Math.floor(x+0.5)；如果 x 是双精度数，返回（long）Math.floor(x+0.5)

4.1 常用数学函数

例 生成随机字符

◆Math.random方法生成[0.0,1.0)之间的double类型的随机数，可以用它写出简单的表达式来生成任意范围的随机数，如：

```
(int)(Math.random()*10);                    //[0,10)
50+(int)(Math.random()*50);                 //[50,100)
```

一般地

```
a+(int)(Math.random()*b)                    返回[a, a+b)
a+(int)(Math.random()*(b+1))                返回[a, a+b]
```

4.1 常用数学函数

例 生成随机字符

◆编写生成随机字符的方法。

Java中每个字符对应一个Unicode编码从0000到FFFF。要生成一个随机字符，就是产生一个从0到65535之间的随机数。所以, 计算表达式为：

(int)(Math.random( ) * (65535 + 1)) 。

◆英文大、小写字母的Unicode是一串连续的整数，如

‘a’ 的编码是: int x = ‘a’ //x = 97，char可以自动转int

由于char类型可自动地被转换为int类型，所以我们可以使用字符字面量代表整数值：

‘a’ (97), ‘b’ (98)，…, ‘z’ (122)

4.1 常用数学函数

例 生成随机字符

◆因此，随机生成从‘a’-‘z’之间的字符就等于生成‘a’-‘z’之间的随机数，又因为可以使用字符字面量代表对应整数值，因此可用下面表达式

(char)(‘a’+(int)(Math.Random()*(‘z’-‘a’+1)))

随机产生从‘a’-‘z’之间的字符

◆将上面讨论一般化，按如下表达式，可以生成任意2个字符ch1和ch2（ch1<ch2）之间的随机字符

(char)(ch1+(int)(Math.rabdom()*(ch2-ch1+1)))

4.1 常用数学函数

例 生成随机字符

```
public class RandomCharacter {
        /** Generate a random character between ch1 and ch2 */
        public static char getRandomCharacter(char ch1, char ch2) {
                return (char) (ch1 + (int)(Math.random( ) * (ch2 - ch1 + 1)));
        }

        /** Generate a random lowercase letter */
        public static char getRandomLowerCaseLetter( ) {
                return getRandomCharacter('a', 'z');
        }

        /** Generate a random digit character */
        public static char getRandomDigitCharacter( ) {
                return getRandomCharacter('0', '9');
        }

}
```

4.2 字符数据类型和操作

Unicode和ASCII码

◆Java对字符采用16位Unicode编码，因此char类型的大小为二个字节

◆16位的Unicode用以\u开头的4位16进制数表示，范围从’\u0000’到’\uffff’,不能少写位数

◆Unicode包括ASCII码，从’\u0000’到’\u007f’对应128个ASCII字符

◆JAVA中的ASCII字符也可以用Unicode表示，例如

```
char letter = ‘A’ ;
char letter = ‘\u0041’ ; //等价，\u后面必须写满4位16进制数
```

◆++和--运算符也可以用在char类型数据上（因为char自动转整数），运算结果为该字符之后或之前的字符，例如下面的语句显示字符b

```
char ch = ‘a’;
System.out.println(++ch); //显示b
```

4.2 字符数据类型和操作

特殊字符的转义

◆和C++一样，采用反斜杠(\)后面加上一个字符或者一些数字位组成转义序列，一个转义序列被当做一个字符

如\n \t \b \r \f \\ \' \"

◆如果想打印带”“ 的信息 He said “Java is fun “

System.out.println(“He said \”Java is fun \”“);

4.2 字符数据类型和操作

字符型数据和数值类型数据之间的转换

◆char类型数据可以转换成任意一种数值类型，反之亦然。将整数转换成char类型数据时，只用到该数据的低16位，其余被忽略。例如

```
char ch = （char）0xAB0041；//0xAB0041是int字面量，要赋值给char，必须强制类型转换
System.out.println(ch);          //显示A
```

◆要将浮点数转成char时，先把浮点数转成int型，然后将整数转换成char

```
char ch = （char）65.25；
System.out.println(ch);          //显示A
```

◆当一个char型转换成数值型时，这个字符的Unicode码就被转换成某种特定数据类型

```
int i = ‘A’ ；//不用强制类型转换
System.out.println(i);           //显示65
```

4.2 字符数据类型和操作

字符型数据和数值类型数据之间的转换

◆如果转换结果适用于目标变量（不会有精度损失），可以采用隐式转换；否则必须强制类型转换

```
int i = ‘A’ ;
byte b = （byte） ‘\uFFF4’ ; //取低8位二进制数F4赋值给b
```

◆所有数值运算符都可以用在char型操作数上，如果另一个操作数是数值，那么char型操作数就自动转换为数值；如果另外一个操作数是字符串，那么char型操作数会自动转换成字符串再和另外一个操作数字符串相连

```
int i = ‘2’ + ‘3’ ; //0x32和0x33
System.out.println（i）； // i为50+51=101
int j = 2 + ‘a’ ; //j = 2 + 97 = 99
System.out.println(j + “ is the Unicode of ” + (char)j);//99 is the Unicode of c
```

4.2 字符数据类型和操作

字符的比较和测试，Character类

◆两个字符可以通过关系运算符进行比较，如同比较二个数值：通过字符的Unicode值进行比较

◆Java为每个基本类型实现了对应的包装类，char类型的包装类是Character类。注意包装类对象为引用类型，不是值类型

◆Character类的作用

- ➢将char类型的数据封装成对象
- ➢包含处理字符的方法和常量

◆方法：下面方法都是静态方法

isDigit方法判断一个字符是否是数字

isLetter方法判断一个字符是否是字母

isLetterOrDigit方法判断一个字符是否是字母或数字

isLowerCase方法判断一个字符是否是小写

isUpperCase方法判断一个字符是否是大写

toLowerCase方法将一个字符转换成小写

toUpperCase方法将一个字符转换成大写

4.2 字符数据类型和操作

字符的比较和测试，Character类

```
package hust.cs.javacourse.ch3;

public class CharacterTest {
    public static void main(String[] args){
        System.out.println("Character.isDigit('1') is:" + Character.isDigit('1'));
        System.out.println("Character.isDigit('a') is:" + Character.isLetter('a'));
        System.out.println("Character.isLetterOrDigit('+') is:" + Character.isLetterOrDigit('+'));
        System.out.println("Character.isUpperCase('A') is:" + Character.isUpperCase('A'));
    }
}
```

```
Character.isDigit('1') is:true
Character.isDigit('a') is:true
Character.isLetterOrDigit('+') is:false
Character.isUpperCase('A') is:true
```

4.3 字符串类型

String类：是一个final类，不能被继承

- java.lang.String表示一个固定长度的字符序列，实例化后其内容不能改。
  - 构造函数
  - 长度(length)
  - 获取字符(charAt)
  - 连接(concat)
  - 截取(substring)
  - 比较(equals, equalsIngnoeCase, compareTo, startWith，endWith, regionMatch)
  - 转换(toLowerCase, toUpperCase, trim, replace)
  - 查找(indexOf, lastIndexOf)
  - 字符串和数组间转换(getChars, toCharArray),
  - 字符串和数字间转换(valueOf)

4.3 字符串类型

String类对象的构造

◆从字面值创建字符串

String newString = new String(stringLiteral);

例如：

String message = new String("Welcome to Java");

◆由于字符串经常使用，java提供了创建字符串的简写形式。

String newString = stringLiteral;

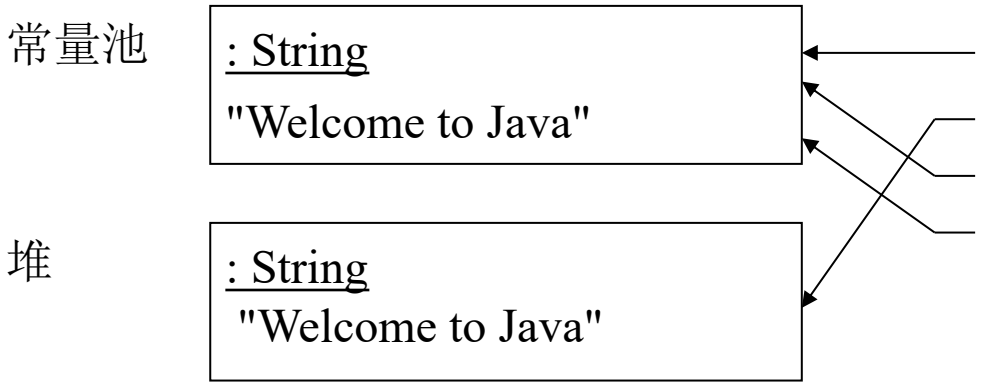
例如：

```
String m1 = “Welcome”；          //字符串的内容都是不可修改的
String m2 = “Welcome”；      //m1和m2通过内存优化引用了同一常量对象：m1==m2
String m3 = "Wel" +"come";                  //m1==m2==m3
String m4 = "Wel" +new String("come");  //m1!=m4
```

4.3 字符串类型

规范字符串和常量池

- 由于字符串是不可变的，为了提高效率和节省内存，Java中的字符串字面值维护在字符串常量池中）。这样的字符串称为规范字符串(canonical string)。
- 可以使用字符串对象（假设内容为Welcome to Java）的intern方法返回规范化字符串。intern方法会在字符串常量池中找是否已存在”Welcome to Java”,如果有返回其地址。如果没有，在池中添加“Welcome to java”再返回地址。即intern方法一定返回一个指向常量池里的字符串对象引用。




```
String s = "Welcome to Java";
String s1 = new String("Welcome to Java");
String s2 = s1.intern();
String s3 = "Welcome to Java";
System.out.println(s1 == s); //false
System.out.println(s2 == s); //true
System.out.println(s3 == s); //true
```

直接用字符串字面量构造的字符串在常量池里，如s。用new String方法构造的字符串在堆里，如s1。只有字面量在常量池里,例如：”Wel” + “come”， 而”Wel”+new String(“come”)不在常量池里，在堆里。

4.3 字符串类型

String类对象是不可变的

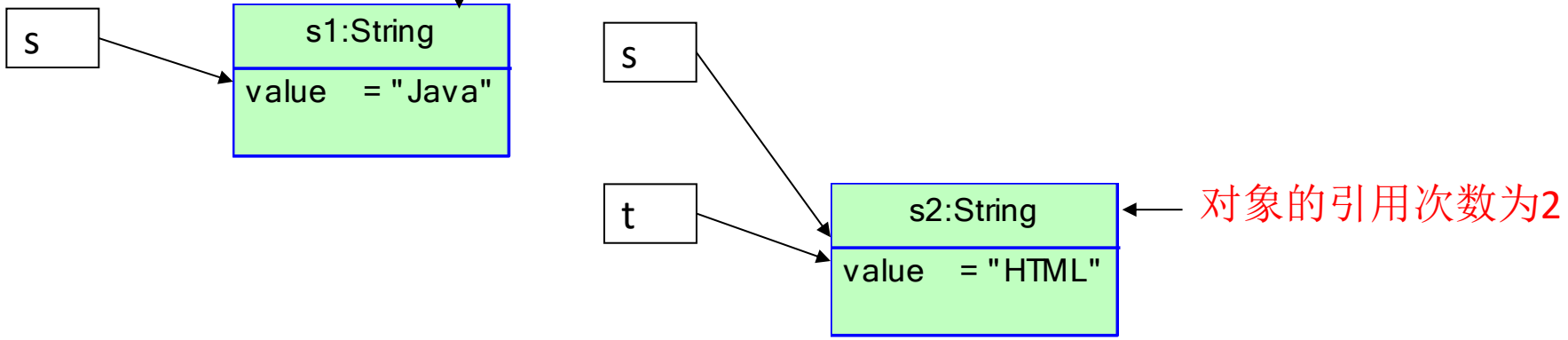
◆字符串对象创建之后，其内容是不可修改的。

String s = “java”;

s = “HTML”;

String t =s;

对象的引用次数为0，意味着可以自动进行垃圾回收了



4.3 字符串类型

字符串的比较

◆equals方法用于比较两个字符串是否包含相同的内容（字符序列）：

- ➢两个字符串内容相同，返回true
- ➢两个字符串内容不同，返回false
- ➢比较字符串内容不能直接比较二个引用变量，比较二个引用变量只是判断这二个引用变量是否指向同一个对象（如s1 == s2）

◆equalsIgnoreCase忽略大小写比较内容是否相同

◆regionMatch比较部分内容是否相同

◆startsWith判断是否以某个字符串开始

◆endsWith判断是否以某个字符串结束

◆compareTo方法用于比较两个字符串的大小，即第一个不同字符的差值（字典序）。

s1.compareTo(s2)的返回值:

- ➢当两个字符串相同时，返回 0
- ➢当s1按字典排序在s2之前，返回小于 0 的值
- ➢当s1按字典排序在s2之后，返回大于 0 的值

4.3 字符串类型

字符串的比较

```
String s0 = "Java";
String s1 = "Welcome to " + s0;
String s2 = "Welcome to Java";
String s3 = "welcome to java";
String s6 = "Welcome to Java";

// equals用于比较两个字符串的内容是否相同
System.out.println("s1.equals(s2) is " + s1.equals(s2)); //true

// equalsIgnoreCase忽略大小写
System.out.println("s2.equals(s3) is " + s2.equals(s3)); //false
System.out.println("s2.equalsIgnoreCase(s3) is " + s2.equalsIgnoreCase(s3)); //true

// regionMatches比较部分字符串: 给定两个串的起始位置和长度
System.out.println("s2.regionMatches(11, s0, 0, 4) is " + s2.regionMatches(11, s0, 0, 4) ); //true
System.out.println("s3.regionMatches(11, s0, 0, 4) is " + s3.regionMatches(11, s0, 0, 4));//false
System.out.println("s3.regionMatches(true, 11, s0, 0, 4) is " + s3.regionMatches(true, 11, s0, 0, 4));//true,忽略大小写
```

4.3 字符串类型

字符串的比较

```
String s0 = "Java";
String s1 = "Welcome to " + s0;
String s2 = "Welcome to Java";
String s3 = "welcome to java";
String s6 = "Welcome to Java";

// startsWith判断是否以某个字符串开始
// endsWith判断是否以某个字符串结束
System.out.println("s2.startsWith(s0) is " + s2.startsWith(s0) );//false
System.out.println("s2.endsWith(s0) is " + s2.endsWith(s0) );  //true

// compareTo根据字典排序比较两个字符串
String s4 = "abc";
String s5 = "abe";
System.out.println("s4.compareTo(s5) is " + s4.compareTo(s5) );//-2
```

4.3 字符串类型

字符串长度和获取单个字符

◆调用length( )方法可以获取字符串的长度（不是字节数）。

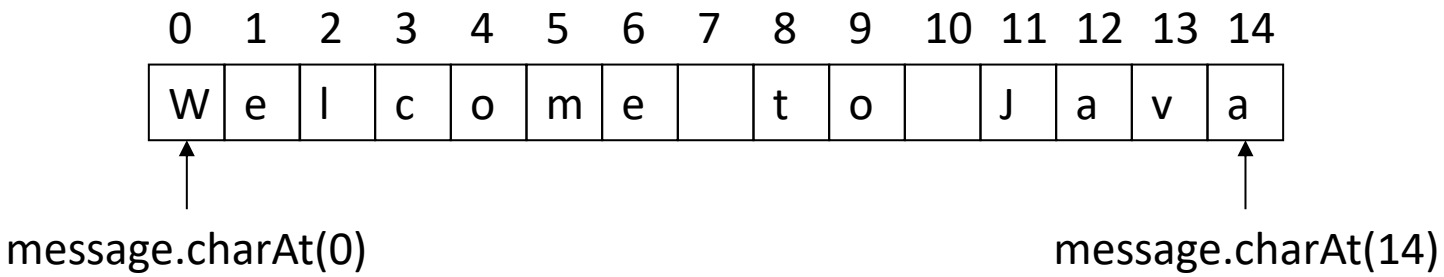
例如：

message.length( )返回15

◆charAt(index)方法可以获取指定位置的字符。index必须在0到s.length()-1之间。

例如：

message.charAt(0)返回字符’ W’



4.3 字符串类型

连接字符串

◆concat方法用于连接两个字符串。例如：

```
String s3 = s1.concat(s2);
```

◆使用加号(+)连接两个字符串。例如：

```
String s3 = s1 + s2;
```

s1 + s2 + s3 等价于s1.concat(s2).concat(s3)

◆连接操作返回一个新的字符串：因为String类型的实例内容不可修改。

4.3 字符串类型

截取子串

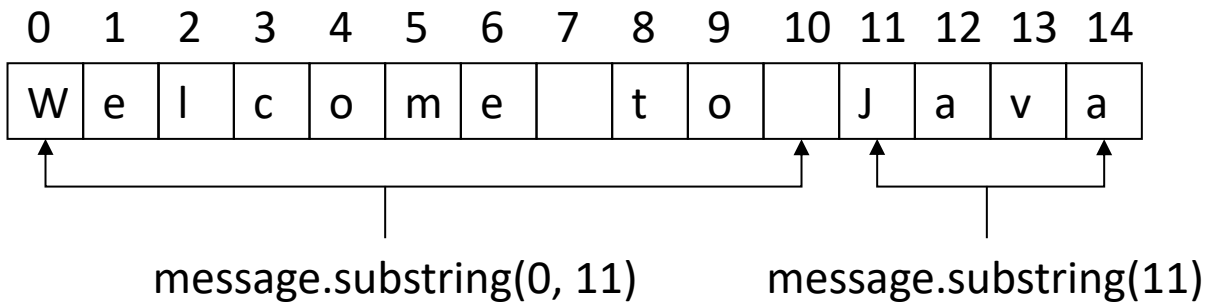
◆substring用于截取字符串的一部分，返回新字符串。

➢public String substring(int beginIndex, int endIndex)

返回字符串的子串。子串从beginIndex开始，直到endIndex-1

➢public String substring(int beginIndex)

返回字符串的子串。子串从beginIndex开始，直到字符串的结尾。



4.3 字符串类型

字符串转换

- toLowerCase将字符串转换成小写形式，得到新串
- toUpperCase将字符串转换成大写形式，得到新串
- trim删除两端的空格，得到新串
- replace字符替换，得到新串

4.3 字符串类型

字符串转换

```
String s0 = "Java";
String s1 = " Welcome to Java ";

// toLowerCase将字符串转换成小写形式
System.out.println("s1.toLowerCase() is " + s1.toLowerCase() );

// toUpperCase将字符串转换成大写形式
System.out.println("s1.toUpperCase() is " + s1.toUpperCase() );

// trim删除两端的空格
System.out.println("s1.trim() is " + s1.trim( ) );

// replace字符替换
System.out.println( “s1.replace(s0, \” HTML\ “) is ” + s1.replace(s0, “HTML” ) ); //Welcome to HTML
```

4.3 字符串类型

查找字符或字符串

◆indexOf返回字符串中字符或字符串匹配的位置，返回-1表示未找到。

"Welcome to Java".indexOf('W') returns 0.

"Welcome to Java".indexOf('x') returns -1.

"Welcome to Java".indexOf('o ‘,5) returns 9.

"Welcome to Java".indexOf("come") returns 3.

"Welcome to Java".indexOf("Java", 5) returns 11.

"Welcome to Java".indexOf("java", 5) returns -1.

"Welcome to Java".lastIndexOf('a') returns 14.

4.3 字符串类型

字符数组和字符串间的转换

◆toCharArray将字符串转换成字符数组

```
String s = “Java”;
char[ ] charArray = s.toCharArray( );// charArray.length=4
```

◆将字符数组转换成字符串

➢使用String的构造函数，可同时初始化

```
new String(new char[ ] { ‘J’ , ‘a’ , ‘v’ , ‘a’ } );
```

➢使用valueOf方法：静态方法

```
String.valueOf( new char[ ] { ‘J’ , ‘a’ , ‘v’ , ‘a’ } );
String.valueOf(2.34);  //2.34转“2.34”
```

4.3 字符串类型

基本数据类型和字符串间的转换

- valueOf方法将基本数据类型转换为字符串。例如

```
String s1 = String.valueOf(1.0);  // “1.0”
String s2 = String.valueOf(true); // “true”
```

- 字符串转换为基本类型：利用包装类

```
Double.parseDouble(str)
Integer.parseInt(str)
Boolean.parseBoolean(str)
```

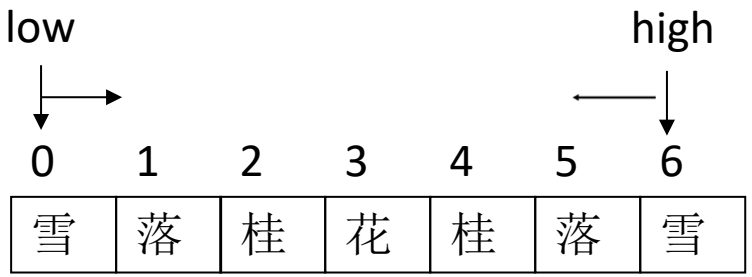
4.3 字符串类型

例子 判断回文

◆ 回文是指顺读和倒读都一样的词语。例如“mom”， “dad”，” noon”都是回文。

编写程序，判断一个字符串是否是回文。

```
public class CheckPalindrome {
    public static boolean isPalindrome(String s) {
            // The index of the first character in the string
            int low = 0;
            // The index of the last character in the string
            int high = s.length( ) - 1;
            while (low < high) {
                    if (s.charAt(low) != s.charAt(high)) return false; // Not a palindrome
                    low++;
                    high--;
            }
            return true; // The string is a palindrome
    }
}
```



4.3 字符串类型

例子 判断回文

```
public class CheckPalindrome {
	/** Main method */
	public static void main(String[] args) {
		// Prompt the user to enter a string
		String s = JOptionPane.showInputDialog("Enter a string:");
		String output = "";
		if (isPalindrome(s))
			output = s + " is a palindrome";
		else
			output = s + " is not a palindrome";
		// Display the result
		JOptionPane.showMessageDialog(null, output);
	}
}
```

4.3 字符串类型-StringBuilder与StringBuffer

◆String类一旦初始化完成，字符串就是不可修改的。

◆StringBuilder与StringBuffer(final类）初始化后还可以修改字符串。

◆StringBuffer修改缓冲区的方法是同步（synchronized）的，更适合多线程环境。

◆StringBuilder线程不安全，与StringBuffer工作机制类似。

◆由于可修改字符串， StringBuilder 与StringBuffer 增加了String类没有的一些函数，例如：append、insert、delete、replace、reverse、setCharAt等。

◆仅以StringBuilder为例：

```
StringBuilder  stringMy=new StringBuilder( );
StringMy.append( “Welcome to” );
StringMy.append( “ Java” );
```

4.3 字符串类型-StringBuilder与StringBuffer

StringBuffer

◆StringBuffer用于处理可变内容的字符串。

- append方法在字符串的结尾追加数据
- insert方法在指定位置上插入数据
- reverse方法翻转字符串
- replace方法替换字符
- toString方法返回String对象
- capacity方法返回缓冲区的容量
- length方法返回缓冲区中字符的个数
- setLength方法设置缓冲区的长度
- charAt方法返回指定位置的字符
- setCharAt方法设置指定位置的字符

4.3 字符串类型-StringBuilder与StringBuffer

修改StringBuffer中的字符串

◆追加

```
StringBuffer bf = new StringBuffer();
bf.append(“Welcome”);
bf.append(‘ ‘);
bf.append(“to ”);
bf.append(“Java”);
System.out.println(bf.toString()); //Welcome to Java
```

◆插入

```
bf.insert(11,”HTML and ”) //Welcome to HTML and
JAVA
```

◆删除：Welcome to Java

```
bf.delete(8,11); //Welcome Java,不含11
bf.deleteCharAt(8);//Welcome o Java
bf.reverse(); //avaJ ot emocleW
bf.replace（11， 15， “HTML”）;//Welcome to HTML，不含15
bf.setCharAt(0,’w’);//welcome to Java
```

所有对StringBuffer对象内容进行修改的方法，都返回指向相同StringBuffer对象的引用

```
StringBuffer bf = new StringBuffer();
StringBuffer bf1 = bf.append("Welcome ");
StringBuffer bf2 = bf.append("to ");
StringBuffer bf3 = bf.append("Java");
assert bf==bf1 && bf==bf2 && bf == bf3;
```

因此以上语句可以直接写成：

```
bf.append("Welcome ").append("to ").append("Java");
```

4.3 字符串类型-StringBuilder与StringBuffer

StringBuffer

- ◆toString(): 从缓冲区返回字符串
- ◆capacity()：返回缓冲区容量。length <= capacity
  当字符串长度超过缓冲区容量，capacity会自动增加
- ◆length()：返回缓冲区中字符数量
- ◆setLength(newLength)：设置缓冲区长度
- ◆charAt(index)：返回下标为index的字符

4.3 字符串类型-StringBuilder与StringBuffer

StringBuffer例子

- 编写程序，检查回文。

```
public static boolean isPalindrome(String s) {

// Create a new string that is the reversal of s
        String s2 = reverse(s);
// Compare if the reversal is the same as the original string
        return s2.equals(s);
}

public static String reverse(String s) {
        StringBuffer strBuf = new StringBuffer(s);
        strBuf.reverse();
        return strBuf.toString();
}
```

4.4 格式化控制台输出

◆JDK1.5开始提供了格式化控制台输出方法

System.out.printf(format, item1, item2, ...); //格式化字符串，后面跟要打印的变量列表

◆格式化字符串

String.format(format, item1, item2, ...); //返回一个格式化好了的String

◆格式描述符：%[argument_index$][flags][width][.precision]conversion，其中

◆可选的 argument_index 是一个十进制整数，用于表明参数在参数列表中的位置。第一个参数由 "1$" 引用，第二个参数由 "2$" 引用

◆conversion：应该如何格式化参数的字符

%b 布尔值	%c 字符	%d 十进制整数
%f 浮点数	%e 科学计数法	%s 字符串

String s = String.format(“格式$：%1$d,%2$s”, 99,“abc”); //结果”格式$：99，abc“

4.4 格式化控制台输出

```
public class TestPrintf {
    public static void main(String[] args) {
        System.out.printf("boolean : %6b\n", false);
        System.out.printf("boolean : %6b\n", true);
        System.out.printf("character : %4c\n", 'a');
        System.out.printf("integer : %6d, %6d\n", 100, 200);
        System.out.printf("double : %7.2f\n", 12.345);
        System.out.printf("String : %7s\n", "hello");
    }
}
```