常用预定义类

孙聪

网络与信息安全学院

2019-10-28

课程内容

- Java概述
- 面向对象程序设计概念
- Java语言基础
- Java面向对象特性
- Java高级特征
- 容器类
- 常用预定义类
- 异常处理
- 输入输出
- 线程

提要

提要

字符串操作

- String
- StringBuilder
- StringTokenizer
- Scanner

java.lang.String

- 用于操作字符串
- 典型构造方法
  - public String(): 构造一个空字符串
  - public String(char[] value):
    使用字符数组value中的字符以构造一个字符串
  - public String(String orig):
    使用原字符串orig的拷贝以构造一个新字符串

String的特性

- String对象是不可变的：String类中每个修改String值的方法，实际上都会创建一个新的String对象，以包含修改后的字符串的内容

```
public class Immutable{
    public static String lowerCase(String s){
        return s.toLowerCase();
    }
    public static void main(String[] args){
        String s=“Java”;
        System.out.println(s);
        String s2=lowerCase(s);
        System.out.println(s2);
        System.out.println(s);
    }
}
```

String的特性

- String类中存在一个专门的常量池，具有特殊的作用
- 由于String对象一经创建就不能被修改，因此String类型的字面常量会自动被加入常量池中。每当程序中出现字面常量时，搜索常量池中是否存在该字面常量字符串（用equals()方法判断），如果不存在，就将该字面常量加入常量池中；如果已存在，就直接返回常量池中的对象引用

```
public class StringImmediate{
    public static void main(String[] args){
        String s1=new String(“hello”);
        String s2=new String(“hello”);  //s1和s2会引用堆中的不同对象
        System.out.println(s1==s2);

        String s3=“hello”;  //搜索常量池，将“hello”加入常量池并将其引用返回给s3
        String s4=“hello”;  //再次搜索常量池，发现“hello”后将其引用直接返回给s4
                            //故s3和s4都引用到常量池中同一个对象
        System.out.println(s3==s4);
    }
}
```

java.lang.String主要接口方法

接口方法	用法
int length()	返回当前字符串的长度
char charAt(int index)	返回字符串中索引index位置上的字符
int compareTo(String anotherString) int compareTolgnoreCase(String str)	按字典顺序比较字符串的内容，比较结果为负数、0或正数（不忽略/忽略大小写）
boolean contains(CharSequence s)	当且仅当字符串包含参数指定的字符序列的内容时，返回true，否则返回false
boolean contentEquals(CharSequence cs) boolean contentEquals(StringBuffer sb)	如果当前字符串与参数的内容完全相同，则返回true；否则返回false
String concat(String str)	返回一个新的字符串，内容为原始字符串连接参数字符串
boolean endsWith(String suffix)	检测参数是否为当前字符串的后缀
boolean startsWith(String prefix) boolean startsWith(String prefix, int toffset)	检测当前字符串是否以参数起始
boolean equals(Object anObject) boolean equalslgnoreCase(String anotherString)	比较两个字符串的内容是否相同（不忽略/忽略大小写）

java.lang.String主要接口方法

接口方法	用法
int indexOf(...) int lastIndexOf(...)	返回参数在当前字符串中的起始索引；若当前字符串不包含参数所指定的字符（或字符串），则返回-1。搜索的起始位置可由参数指定。lastIndexOf() 方法的搜索从后向前进行
String replace(...)	将所有出现的旧字符（字符序列）都替换为新字符（字符序列），返回替换字符后的新字符串，若没有替换发生，则返回原始的字符串对象
CharSequence subSequence(...) String substring(...)	返回一个新的字符串（字符序列），该字符串（字符序列）是由参数索引指定的当前字符串的子字符串（字符序列）
String trim()	将当前字符串两端的空白字符删除后，返回新的字符串
String toLowerCase() String toUpperCase()	将当前字符串中的所有字符转换为小写（大写）。实际上创建了新的字符串
boolean isEmpty()	当且仅当length()方法返回0时，返回true
char[] toCharArray()	将当前字符串转化为一个字符数组

java.lang.String数据类型转换

- 各种基本数据类型与String类型之间可通过方法相互转换
- 基本类型值 ⇒ String类型字符串:
  使用String类的静态valueOf()方法
  - public static String valueOf(boolean b)
  - public static String valueOf(char c)
  - public static String valueOf(int i)
  - public static String valueOf(long l)
  - public static String valueOf(float f)
  - public static String valueOf(double d)
- String类型字符串 ⇒ 基本类型值:
  通过基本类型对应的Wrapper类的静态parse方法
- String类型字符串 ⇒ Wrapper类对象:
  通过基本类型对应的Wrapper类的静态valueOf方法
- 例: TestConversion.java

java.lang.StringBuilder

- 提供了一个字符串的可变序列，它对存储的字符序列可以任意修改，使用起来比String类灵活
- 在JavaSE 5中引入（此前使用StringBuffer），StringBuilder与StringBuffer兼容，但StringBuffer是同步的，StringBuilder非同步（效率高）
- 构造方法
  - StringBuilder()：构造一个空StringBuilder对象，初始容量为16个字符
  - StringBuilder(String str)：构造一个StringBuilder对象，初始内容为字符串str的拷贝

java.lang.StringBuilder主要接口方法

接口方法	用法
void setLength(int newLength)	设置字符序列的长度
void setCharAt(int index, char ch)	设置index位置的字符为ch
StringBuilder delete(int start, int end)	将(start, end-1)范围中的子串从字符序列中移除
StringBuilder deleteCharAt(int index)	移除当前字符序列中index所指位置的那个字符
StringBuilder replace(int start, int end, String str)	将(start, end-1)范围中的子串替换为str
StringBuilder reverse()	对当前的字符序列做逆序
String toString()	将当前字符序列转化为String类型的字符串
void trimToSize()	尝试减少用于存放字符序列的存储空间

- StringBuilder类的indexOf()、length()、charAt()、subSequence()、substring()等方法与String类中的方法类似
- StringBuilder类有两种特有的方法系列，即append()方法系列和insert()方法系列

java.lang.StringBuilder主要接口方法

- append()方法根据参数的数据类型在StringBuilder对象的末尾直接进行数据添加
  - public StringBuilder append (boolean b)
  - public StringBuilder append (char c)
  - public StringBuilder append (char[] str)
  - public StringBuilder append (char[] str, int offset, int len)
  - public StringBuilder append (double d)
  - public StringBuilder append (float f)
  - public StringBuilder append (int i)
  - public StringBuilder append (long l)
  - public StringBuilder append (Object obj)
  - public StringBuilder append (String str)
  - public StringBuilder append (StringBuffer sb)

java.lang.StringBuilder主要接口方法

- insert()方法系列根据参数的数据类型在StringBuilder对象的offset位置进行数据插入
  - public StringBuilder insert (int offset, boolean b)
  - public StringBuilder insert (int offset, char c)
  - public StringBuilder insert (int offset, char[] str)
  - public StringBuilder insert (int offset, float f)
  - public StringBuilder insert (int offset, int i)
  - public StringBuilder insert (int offset, long l)
  - public StringBuilder insert (int offset, Object obj)
  - public StringBuilder insert (int offset, String str)
- 例：StringBuilderDemo.java

java.util.StringTokenizer

- 是一个实现Enumeration接口的类，它使得应用程序可以将字符串分成多个记号，默认情况下以空格为分隔符，例如将字符串“this is a test”分成四个单词记号。用户也可以指定分隔符

java.util.StringTokenizer

- 是一个实现Enumeration接口的类，它使得应用程序可以将字符串分成多个记号，默认情况下以空格为分隔符，例如将字符串“this is a test”分成四个单词记号。用户也可以指定分隔符
- 主要构造方法
  - StringTokenizer(String str):
    以字符串str构建StringTokenizer对象
  - StringTokenizer(String str, String delim):
    使用delim分隔符，以字符串str构建StringTokenizer对象
  - StringTokenizer(String str, String delim, boolean returnDelims): returnDelims决定分隔符本身是否也作为结果返回

java.util.StringTokenizer主要接口方法

接口方法	用法
int countTokens()	返回识别的总记号数（nextToken()方法能够被调用的次数，超过这一次数则nextToken() 方法抛出异常）
boolean hasMoreTokens() boolean hasMoreElements()	测试是否还有识别的记号
String nextToken()	返回下一个识别的记号
String nextToken(String delim)	返回字符串delim分隔的下一个记号
Object nextElement()	与nextToken()方法返回相同的值，但返回值的声明类型为Object而非String

```
public class StringTokenizerDemo {
    public static void main(String args[]) {
        String str = “数学::英语::语文::化学”;
        StringTokenizer st = new StringTokenizer(str, “::”);
        System.out.println(“课程数: ” + st.countTokens());
        while (st.hasMoreTokens())
            System.out.print(st.nextToken()+“; ”);
        System.out.println();

        String[] result = str.split(“::”);
        for (int x=0; x<result.length; x++)
            System.out.print(result[x]+“; ”);
        System.out.println();

        Scanner scanner=new Scanner(str);
        scanner.useDelimiter(“::”);
        while(scanner.hasNext())
            System.out.print(scanner.next()+“; ”);
        scanner.close();
    }
}
```

java.util.Scanner

- 文本扫描器，能够使用正则表达式解析基本类型和字符串
- J2SE 5.0引入
- Scanner对其输入使用分隔符模式进行分割，分隔符模式默认为匹配空格
- 得到的记号可以通过各种next方法被转换为不同基本类型的值
- 输入：构造方法可接受多种输入对象 （File, InputStream, String, Readable, Path等），例：
  - `Scanner sc = new Scanner(System.in);`
  - `Scanner sc = new Scanner(new File("myNumbers"));`
  - `String input = "..."; Scanner s = new Scanner(input);`

java.util.Scanner

- 扫描结果通过各种next*()方法转化为不同类型的值，next*()仅在找到了一个完整的分隔符后才返回
  - next()：获取一个字符串
  - nextByte()：获取一个byte类型的整数
  - nextShort()：获取一个short类型的整数
  - nextInt()：获取一个int类型的整数
  - nextLong()：获取一个long类型的整数
  - nextFloat()：获取一个float类型的浮点数
  - nextDouble()：获取一个double类型的浮点数
  - nextLine()：获取一行文本（即以回车键为结束标志）
- 如果下一个记号与想要获得的类型不匹配，或者数值越界，则抛出InputMismatchException异常
- 可以通过hasNext*()方法判断下一个记号能否被解析为所需类型

java.util.Scanner

从控制台读取输入，计算个数不定的多个整数之和，以0表示输入结束

```
Scanner sc = new Scanner(System.in);
int sum=0, data;
while((data = sc.nextInt()) != 0)
    sum += data;
System.out.println(sum);
```

- 例：ScannerDemo.java

提要

java.lang.Math

- 包含基本数值运算方法，包括三角函数、指数、对数等
- 两个静态常量
  - E：自然对数的底数（e）
  - PI：圆周率π

java.lang.Math主要接口方法（三角函数）

接口方法	用法
static double sin(double a)	返回a的正弦值
static double cos(double a)	返回a的余弦值
static double tan(double a)	返回a的正切值
static double asin(double a)	返回a的反正弦值，返回值在-π/2到π/2之间
static double acos(double a)	返回a的反余弦值，返回值在0.0到π之间
static double atan(double a)	返回a的反正切值，返回值在-π/2到π/2之间
static double atan2(double y, double x)	根据直角坐标(x,y)转化为的极坐标(r,theta)，返回角度theta
static double sinh(double x)	返回x的双曲线正弦值
static double cosh(double x)	返回x的双曲线余弦值
static double tanh(double x)	返回x的双曲线正切值
static double toDegrees(double angrad)	将弧度转化为角度
static double toRadians(double angdeg)	将角度转化为弧度

java.lang.Math主要接口方法（指数/对数函数方法）

接口方法	用法
static double exp(double a)	返回e^a
static double expm1(double x)	返回$e^x - 1$
static double log(double a)	返回a的自然对数（以e为底）
static double log10(double a)	返回a的以10为底的对数
static double log1p(double x)	返回(1+x)的自然对数
static double pow(double a, double b)	返回a^b
static double sqrt(double a)	返回$\sqrt{a}$
static double cbrt(double a)	返回$\sqrt[3]{a}$
static int getExponent(T d)	类型T可为double或float，返回d的无偏差的指数
static double hypot(double x, double y)	返回$\sqrt{x^2+y^2}$的结果，无中间上溢或下溢
static T scalb(T d, int scaleFactor)	类型T可为double或float，返回$(d \times 2^{scaleFactor})$的值

java.lang.Math主要接口方法（浮点数操作方法）

接口方法	用法
static double ceil(double a)	返回大于等于a且等于一个整数的最小（最接近负无穷）的浮点值
static double floor(double a)	返回小于等于a且等于一个整数的最大（最接近正无穷）的浮点值
static double rint(double a)	返回最接近a且等于一个整数的浮点值
static long round(double a)	返回最接近double类型a的long值
static int round(float a)	返回最接近float类型a的int值
static T copySign(T magnitude, T sign)	类型T可为double或float，返回一个T类型的数，该数的绝对值由magnitude决定，符号由sign决定
static T nextAfter(T start, double direction)	类型T可为double或float，返回与start相邻的浮点数，方向由direction决定
static T nextUp(T d)	类型T可为double或float，返回与d相邻的浮点数，方向为正无穷大方向
static T nextDown(T d)	类型T可为double或float，返回与d相邻的浮点数，方向为负无穷大方向
static T signum(T d)	类型T可为double或float，符号函数，若d为0则返回0；若d>0则返回1.0；若d<0则返回-1.0
static double IEEEremainder(double f1, double f2)	根据IEEE 754标准，根据两个参数计算余数

java.lang.Math主要接口方法（其他方法）

接口方法	用法
static T abs(T a)	类型T可为double, float, int, long, 返回类型为T的a的绝对值
static T max(T a, T b)	类型T可为double, float, int, long, 返回a与b中较大的一个
static T min(T a, T b)	类型T可为double, float, int, long, 返回a与b中较小的一个
static double random()	返回[0.0, 1.0)之间的浮点数
*Exact()	*可为add, subtract, multiply, negate, increment, decrement等, 参数类型为int或long, 在运算结果溢出时抛出异常

- 例：MathDemo.java

java.util.Random

- Random类的实例可用来生成一系列的伪随机数，该类使用一个long类型的种子
- 如果使用同一个种子构造两个Random类的实例，生成的伪随机数序列相同

java.util.Random主要接口方法

接口方法	用法
protected int next(int bits)	返回下一个伪随机数
boolean nextBoolean()	返回下一个伪随机数，该数是取自随机数生成器序列的、服从均匀分布的boolean值
void nextBytes(byte[] bytes)	生成一系列随机的字节并将它们置于用户提供的字节数组中
double nextDouble()	返回下一个伪随机数，该数是取自随机数生成器序列的、服从0.0～1.0均匀分布的double值
float nextFloat()	返回下一个伪随机数，该数是取自随机数生成器序列的、服从0.0～1.0均匀分布的float 值
double nextGaussian()	返回下一个伪随机数，该数是取自随机数生成器序列的、服从高斯（正态）分布（平均值0.0，标准差1.0）的double值
int nextInt()	返回下一个伪随机数，该数是取自随机数生成器序列的、服从均匀分布的int类型值
int nextInt(int n)	返回下一个伪随机数，该数是取自随机数生成器序列的、服从[0，n)上均匀分布的int类型值
long nextLong()	返回下一个伪随机数，该数是取自随机数生成器序列的、服从均匀分布的long类型值
void setSeed(long seed)	使用一个long类型的参数为随机数生成器设置种子

提要

java.util.Arrays

- 包含很多实用的数组处理方法，包括对数组的排序、查找、比较和元素填充等，这些方法均为静态方法
- 可以看作Java的容器类体系的一部分

java.util.Arrays主要接口方法

接口方法	用法
asList(T... a)	使用数组参数a返回一个定长的List，对List的更改将会直接影响到a。本方法与Collection.toArray()一起，构成了基于数组的API与基于容器的API的桥梁
static int binarySearch(T[] a, T key) static int binarySearch(T[] a, int fromIndex, int toIndex, T key)	使用二分查找算法在指定的T[]类型数组a（或a的一部分）中查找指定的key值。数组必须预先按照升序排列，如果数组中不存在被查找的key，则返回(-插入点索引-1)。类型T可为byte, char, double, float, int, long, Object, short
static T[] copyOf(T[] original, int newLength)	将指定数组original复制到一个指定长度的新数组，同时进行必要的截断或用T类型的缺省值填充。类型T可为boolean, byte, char, double, float, int, long, short
static T[] copyOfRange(T[] original, int from, int to)	将指定数组original的一部分复制到一个新数组。类型T可为boolean, byte, char, double, float, int, long, short
static boolean deepEquals(Object[] a1, Object[] a2)	如果两个对象数组深层次地相等，则返回true；否则返回false（用于多维数组）

java.util.Arrays主要接口方法

接口方法	用法
static boolean equals(T[] a, T[] a2)	如果两个指定的T[]类型数组相等，则返回true，否则返回false。 类型T可为boolean, byte, char, double, float, int, long, Object, short
static void fill(T[] a, T val) static void fill(T[] a, int fromIndex, int toIndex, T val)	将指定val值（或对象引用）赋给数组a （或a中特定范围内）的每个元素。类型T可为boolean, byte, char, double, float, int, long, Object, short
static int hashCode(T[] a)	根据指定数组的内容生成哈希码。类型T可为boolean, byte, char, double, float, int, long, Object, short
static int deepHashCode(Object[] a)	使用指定对象数组的“深层内容”生成hash码并返回
static void sort(T[] a) static void sort(T[] a, int fromIndex, int toIndex)	将指定数组a（或a的指定范围内）的元素按照数值升序（或对象的自然顺序）排列。类型T可为byte, char, double, float, int, long, Object, short
static String toString(T[] a)	返回指定数组内容的字符串表示。类型T可为boolean, byte, char, double, float, int, long, Object, short
static String deepToString(Object[] a)	返回指定对象数组的“深层内容”对应的字符串表示

提要

Wrapper类

- Wrapper将基本类型表示成类，每个基本数据类型在java.lang包中都有一个对应的Wrapper类
- 每个Wrapper类对象都封装了基本类型的一个值

基本类型	Wrapper类
boolean	Boolean
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double

Wrapper类

- Wrapper类实例的构造：将基本数据类型值传递给Wrapper类的构造方法
- Wrapper类的常用方法和常量
  - 数值型Wrapper类中的MIN_VALUE, MAX_VALUE
  - byteValue()/shortValue()/longValue(), ... ：将当前Wrapper类型的值作为byte/short/long/...返回
  - valueOf()：将字符串转换为Wrapper类型的实例
  - toString()：将基本类型值转换为字符串
  - parseByte()/parseShort()/parseInt(), ... ：将字符串转换为byte/short/int/...类型值

Autoboxing / Autounboxing

- Autoboxing：在应该使用对象的地方使用基本类型的数据时，编译器自动将该数据包装为对应的Wrapper类对象
- Autounboxing：在应该使用基本类型数据的地方使用Wrapper类的对象时，编译器自动从Wrapper类对象中取出所包含的基本类型数据

```
public class Autoboxing{
    public static void main(String[] args){
        Integer a=25;
        Integer b=23;
        if(a.equals(b))
            System.out.println(a+ " equals to " +b);
        System.out.println( "The sum of " +a+ " and " +b+ " is " + (a+b));
    }
}
```

基本类型，Wrapper类对象，String之间的转换