第六章 常用类

本章内容

- 字符串相关类（String、 StringBuffer 、 StringBuilder）
- 基本数据类型包装类
- Math类（不重要）
- File类
- 枚举类

String 类

- java.lang.String 类代表**不可变**的字符序列。
- “xxxxx”为该类的一个对象。
- String类的常见构造方法：
  - String(String original)
    - 创建一个String对象为original的拷贝。
  - String(char[] value)
    - 用一个字符数组创建一个String对象
  - String(char[] value,int offset,int count)
    - 用一个字符数组从offset项开始的count个字符序列创建一个String对象

String 类举例(1)

```
public class Test {
  public static void main(String[] args) {
    String s1 = "hello"; String s2 = "world";
    String s3 = "hello";
    System.out.println(s1 == s3); //true

    s1 = new String ("hello");
    s2 = new String("hello");
    System.out.println(s1 == s2); //false
    System.out.println(s1.equals(s2)); //true

    char c[]= {'s','u','n',' ','j','a','v','a'};
    String s4 = new String(c);
    String s5 = new String(c,4,4);
    System.out.println(s4); //sun java
    System.out.println(s5); //java
  }
}
```

String 类常用方法(1)

```
public char charAt(int index)
返回字符串中第index个字符。
public int length()
返回字符串的长度。
public int indexOf(String str)
返回字符串中出现str的第一个位置
public int indexOf(String str,int fromIndex)
返回字符串中从fromIndex开始出现str的第一个位置
public boolean equalsIgnoreCase(String another)
比较字符串与another是否一样（忽略大小写）
public String replace(char oldChar,char newChar)
在字符串中用newChar字符替换oldChar字符
```

String 类举例(2)

```
public class Test {
  public static void main(String[] args) {
    String s1 = "sun java",s2 = "Sun Java";
    System.out.println(s1.charAt(1));//u
    System.out.println(s2.length());//8
    System.out.println(s1.indexOf("java"));//4
    System.out.println(s1.indexOf("Java"));//-1
    System.out.println(s1.equals(s2));//false
    System.out.println(s1.equalsIgnoreCase(s2));
    //true

    String s = "我是程序员，我在学java";
    String sr = s.replace('我','你');
    System.out.println(sr);
    //你是程序员，你在学java
  }
}
```

String 类常用方法(2)

```
public boolean startsWith(String prefix)
判断字符串是否以prefix字符串开头
public boolean endsWith(String suffix)
判断字符串是否以prefix字符串结尾
public String toUpperCase()
返回一个字符串为该字符串的大写形式
public String toLowerCase()
返回一个字符串为该字符串的小写形式
public String substring(int beginIndex)
返回该字符串从beginIndex开始到结尾的子字符串
public String substring(int beginIndex,int endIndex)
返回该字符串从beginIndex开始到endIndex结尾的子字符串
public String trim()
返回将该字符串去掉开头和结尾空格后的字符串
```

String 类举例(3)

```
public class Test {
  public static void main(String[] args) {
    String s = "Welcome to Java World!";
    String s1 = "  sun java   " ;
    System.out.println(s.startsWith("Welcome"));
    //true
    System.out.println(s.endsWith("World"));
    //false
    String sL = s.toLowerCase();
    String sU = s.toUpperCase();
    System.out.println(sL);
    //welcome to java world!
    System.out.println(sU);
    //WELCOME TO JAVA WORLD!
    String subS = s.substring(11);
    System.out.println(subS);//Java World!
    String sp = s1.trim();
    System.out.println(sp);//sun java
  }
}
```

String 类常用方法(3)

- 静态重载方法
  - public static String valueOf(…)可以将基本类型数据转换为字符串；例如：
    - public static String valueOf(double d)
    - public static String valueOf(int i)
    - …… …
    - b + “”;
- 方法 public String[] split(String regex)可以将一个字符串按照指定的分隔符分隔，返回分隔后的字符串数组。

String 类举例(4)

```
public class Test {
    public static void main(String[] args) {
      int j = 1234567;
      String sNumber = String.valueOf(j); //j+""
      System.out.println
          ("j 是"+sNumber.length()+"位数。");
      String s = "Mary,F,1976";
      String[] sPlit = s.split(",");
      for(int i=0;i<sPlit.length;i++) {
          System.out.println(sPlit[i]);
      }
    }
}
```

- 输出结果：

```
j 是7位数。
Mary
F
1976
```

课 堂 练 习

课堂练习

1: 编写一个程序，输出一个字符串中的大写英文字母数，小写英文字母数以及非英文字母数。

2: 编写一个方法，输出在一个字符串中，指定字符串出现的次数。

TestString.java

StringBuffer 类

- java.lang.StringBuffer 代表可变的字符序列。
- StringBuffer和String类似，但StringBuffer可以对其字符串进行改变。
- StringBuffer类的常见构造方法：
  - StringBuffer()
    - 创建一个不包含字符序列的“空”的StringBuffer对象。
  - StringBuffer(String str)
    - 创建一个StringBuffer对象，包含与String对象str相同的字符序列。

StringBuffer常用方法（1）

- 重载方法 public StringBuffer append(…) 可以为该StringBuffer 对象添加字符序列，返回添加后的该StringBuffer对象引用，例如：

public StringBuffer append(String str)

public StringBuffer append(StringBuffer sbuf)

public StringBuffer append(char[] str)

public StringBuffer append

(char[] str,int offset,int len)

public StringBuffer append(double d)

public StringBuffer append(Object obj)

…… …… ……

StringBuffer常用方法(2)

- 重载方法 public StringBuffer insert(…)可以为该StringBuffer 对象在指定位置插入字符序列，返回修改后的该StringBuffer对象引用，例如：

public StringBuffer insert

(int offset,String str)

public StringBuffer insert

(int offset,double d)

…… ……

- 方法 public StringBuffer delete(int start,int end) 可以删除从start开始到end-1为止的一段字符序列，返回修改后的该StringBuffer对象引用。

StringBuffer常用方法(3)

- 和 String 类含义类似的方法：

  public int indexOf(String str)

  public int indexOf(String str,int fromIndex)

  public String substring(int start)

  public String substring(int start,int end)

  public int length()

- 方法 public StringBuffer reverse()用于将字符序列逆序，返回修改后的该StringBuffer对象引用。

StringBuffer类举例

```
public class Test {
    public static void main(String[] args) {
    String s = "Mircosoft";
    char[] a = {'a','b','c'};
    StringBuffer sb1 = new StringBuffer(s);
    sb1.append('/').append("IBM")
       .append('/').append("Sun");
    System.out.println(sb1);
    StringBuffer sb2 = new StringBuffer("数字");
    for(int i = 0;i<=9;i++){sb2.append(i);}
    System.out.println(sb2);
    sb2.delete(8,sb2.length()).insert(0,a);
    System.out.println(sb2);
    System.out.println(sb2.reverse());
  }
}
```

- 输出结果：

```
Mircosoft/IBM/Sun
数字0123456789
abc数字012345
543210字数cba
```

基本数据类型包装类

- 包装类（如：Integer，Double等）这些类封装了一个相应的基本数据类型数值，并为其提供了一系列操作。

- 以java.lang.Integer类为例；构造方法：
  - Integer(int value)
  - Integer(String s)

包装类常见方法

- 以下方法以java.lang.Integer为例

  public static final int MAX_VALUE 最大的int型数（2^{31}-1）

  public static final int MIN_VALUE 最小的int型数（-231）

  public long longValue() 返回封装数据的long型值

  public double doubleValue() 返回封装数据的double型值

  public int intValue() 返回封装数据的int型值

  public static int parseInt(String s)

  throws NumberFormatException

  将字符串解析成int型数据，返回该数据

  public static Integer valueOf(String s)

  throws NumberFormatException

  返回Integer对象，其中封装的整型数据为字符串s所表示。

TestPrimitive.java

课 堂 练 习

课堂练习

编写一个方法，返回一个double型二维数组，数组中的元素通过解析字符串参数获得。

如字符串参数:

“1,2;3,4,5;6,7,8”

对应的数组为：

d[0,0]=1.0 d[0,1]=2.0

d[1,0]=3.0 d[1,1]=4.0 d[1,2]=5.0

d[2,0]=6.0 d[2,1]=7.0 d[2,2]=8.0

ArrayParser.java

Math类

- java.lang.Math提供了一系列静态方法用于科学计算；其方法的参数和返回值类型一般为double型。

  abs 绝对值

  acos,asin,atan,cos,sin,tan

  sqrt 平方根

  pow(double a, double b) a的b次幂

  log 自然对数

  exp e为底指数

  max(double a, double b)

  min(double a, double b)

  random() 返回 0.0 到 1.0 的随机数

  long round(double a) double型的数据a转换为long型（四舍五入）

  toDegrees(double angrad) 弧度->角度

  toRadians(double angdeg) 角度->弧度

Math类举例

```
public class Test {
  public static void main(String[] args) {
    double a = Math.random();
    double b = Math.random();
    System.out.println(Math.sqrt(a*a+b*b));
    System.out.println(Math.pow(a,8));
    System.out.println(Math.round(b));
    System.out.println(Math.log(Math.pow(Math.E,15)));
    double d = 60.0, r = Math.PI/4;
    System.out.println(Math.toRadians(d));
    System.out.println(Math.toDegrees(r));
  }
}
```

- 输出结果：

```
0.22724854767821204
3.0369119934905976E-10
0
15.0
1.0471975511965976
45.0
```

File 类

- java.io.File类代表系统文件名（路径或文件名）。
- File类的常见构造方法：
  - public File(String pathname)
  - 以pathname为路径创建File对象，如果pathname是相对路径，则默认的当前路径在系统属性user.dir中存储。
- File的静态属性String separator存储了当前系统的路径分隔符。

- 通过File对象可以访问文件的属性。

```
public boolean canRead()          public boolean canWrite()
public boolean exists()           public boolean isDirectory()
public boolean isFile()           public boolean isHidden()
public long lastModified()        public long length()
public String getName()           public String getPath()
```

TestFile.java

- 通过File对象创建空文件或目录（在该对象所指的文件或目录不存在的情况下）。

```
public boolean createNewFile()throws IOException
public boolean delete()
public boolean mkdir()
```

ListFile.java

课 堂 练 习

课堂练习

编写一个程序，在命令行中以树状结构展现特定的文件夹及其子文件(夹)

总结

- String
  - 正则表达式
- 基础类型包装类
- Math
- File
  - 递归
- 枚举类型
- Apache Commons