Java常用类【微信公众号：编码的艺术】https://github.com/muyutingfeng/smartMindMap

Object

是Java中其他所有类的祖先，没有Object类Java里向对象无从谈起

clone()的正确调用是需要实现Cloneable接口，如果没有实现Cloneable接口，并且子类直接调用Object类的clone()方法，则会抛出CloneNotSupportedException异常。

Cloneable接口仅是一个表示接口，接口本身不包含任何方法，用来指示Object.clone()可以合法的被子类引用所调用。

- 浅度克隆
  - 浅度克隆步骤
    - 1、实现java.lang.Cloneable接口
    - 2、重写java.lang.Object.clone()方法

    浅度克隆，对于其基本数据类型的属性，复制一份拷贝产生的对象，对于非基本数据类型的属性，仅仅复制一份引用指向原产生的对象，原新产生的对象和原始对象中的非基本数据类型的属性都指向的是同一个对象。
- 深度克隆
  - 深度克隆步骤
    - 重写clone()方法
    - 重写equals()方法
    - 重写hashCode()方法

    在浅度克隆的基础上，对于要克隆的对象中的非基本数据类型的属性也实现拷贝，也实现克隆，这样对于非基本数据类型的属性，复制的不是一份引用，原新产生的对象和原始对象中的非基本数据类型的属性指向的不是同一个对象

- equals(Object obj)

  ==与equals在Java中经常被使用，大家也都知道==与equals的区别：

  ==表示的是变量值完成相同（对于基础类型，地址中存储的是值，引用类型则存储指向实际对象的地址）

  equals()方法通常一个重写数据，表示该对象的唯一特征。

- hashCode()

  hashCode()具有如下约定：

  1).在Java应用程序等程序执行期间，对于同一对象多次调用hashCode()方法时，其返回的哈希码应相同的，前提是对对象进行equals比较时所用的信息没有被修改。在Java应用程序的一次执行期间外一次执行，同一对象的hashCode()返回的哈希码无须保持一致；

  2).如果两个对象相等（依据：调用equals()方法），那么这两个对象调用hashCode()返回的哈希码也必须相等；

  3).反之，两个对象调用hashCode()返回的哈希码相等，这两个对象不一定相等。

- toString()

  toString()方法通常会被子类所重写

- wait()/notify()/notifyAll()

  wait()：调用此方法将在当前线程等待，直到在其他线程上调用此方法的主调（某一对象）的notify()/notifyAll()方法。

  - wait(...) / notify() | notifyAll()
    - 唤醒在此对象监视器上等待的单个线程/所有线程。
    - wait(...) / notify() | notifyAll()

    notify()/notifyAll()：

Scanner

- Scanner类来获取用户的输入

  Scanner s = new Scanner(System.in);

  Scanner scan = new Scanner(System.in);
  // 从键盘接收数据

  // next方式接收字符串
  System.out.println("next方式接收：");
  // 判断是否还有输入
  if (scan.hasNext()) {
  String str1 = scan.next();
  System.out.println("输入的数据为：" + str1);
  }
  scan.close();

- next方法

  Scanner scan = new Scanner(System.in);
  // 从键盘接收数据

  // nextLine方式接收字符串
  System.out.println("nextLine方式接收：");
  // 判断是否还有输入
  if (scan.hasNextLine()) {
  String str2 = scan.nextLine();
  System.out.println("输入的数据为：" + str2);
  }
  scan.close();

- nextLine 方法

- next() 与 nextLine() 区别

  next():
  1、一定要读取到有效字符后才可以结束输入。
  2、对输入有效字符之前遇到的空白，next() 方法会自动将其去掉。
  3、只有输入有效字符后才将其后面输入的空白作为分隔符或者结束符。
  4、next() 不能得到带有空格的字符。

  nextLine():
  1、以enter为结束符,也就是说 nextLine()方法返回的是输入回车之前的所有字符。
  2、可以获得空白

Java常用包

- java.lang--语言包：包括Object类、Thread类、String、Math、System、Runtime、Class、Exception、Process
- java.util--实用工具包：Scanner、Date、Calendar、LinkedList、Hashtable、Stack、TreeSet等；
- java.net--网络功能包：URL、Socket、ServerSocket等
- java.sql--数据库连接包：实现JDBC的类库
- java.io--输入输出包：File FileInputStream是你与流相关的各种包
- 常用第三方lie
  - log4j：一个体常常用的log日志jar包。
  - apache commons：包含了大量程序，很多实用小工具。
  - maven：项目管理的；
  - gson：Google的Json解析器；
  - JUnit：java单元测试；
  - jsoup：html解析；
- Java常用接口：Comparable ,Collection,Set, List, Map, Runnable Iterable Iterator 等
- 面试题目
  - 1、clone方法
    保护方法，实现对象的浅复制，只有实现了Cloneable接口才可以调用该方法，否则抛出CloneNotSupportedException异常
    2、getClass方法：final方法，获得运行时类型。
    4、finalize方法：该方法用于释放资源，因为无法确定该方法什么时候被调用，很少使用。
    7、wait方法
    wait方法是使当前线程等待该对象的锁，当前线程必须是该对象的拥有者，也就是具有该对象的锁。wait()方法一直等待，直到获得锁或者被中断。wait(long timeout)设定一个超时间隔，如果在规定时间内没有获得锁就超时。
    调用该方法后当前线程进入阻塞状态，直到以下事件发生。
    （1）其他线程调用了该对象的notify方法。
    （2）其他线程调用了该对象的notifyAll方法。
    （3）其他线程调用了interrupt中断该线程。
    （4）时间间隔到了。
    此时该线程就可以被调度了，如果是被中断的话就抛出一个InterruptedException异常。
    8、notify方法：该方法唤醒在该对象上等待的某个线程。
    9、notifyAll方法：该方法唤醒在该对象上等待的所有线程。
  - 常用和重主要有哪些方法
  - 哪些有用途
    - String 字符串类
    - System 可调包括系统信息
    - StringBuilder 字符串工具类
    - Thread 线程类
    - Math 与数学有关的工具类
    - Arrays.sort 是常用的数组实现的集合
    - LinkedList 底层用链表实现的集合
    - HashMap 接口Map的一个实现类
    - HashSet 接口Set的一个实现类
    - Scanner 简单文本扫描器
    - Calendar 日期类
    - Date 日期类
    - File 目录或文件操作类
    - FileInputStream 输入流
    - FileOutputStream 输出流
    - PrintWriter
    - BufferedInputStream 具有缓冲的输入流
    - BufferedOutputStream 具有缓冲的输出流
    - BufferedReader 从字符输入流中读取文本，缓冲各个字符。
    - BufferedWriter 将文本写入字符输出流，缓冲各个字符

数组

- 声明数组
  - dataType[] arrayRefVar; // 首选的方法
  - double[] myList; // 首选的方法
- 创建数组
  - arrayRefVar = new dataType[arraySize];
  - 一、使用 dataType[arraySize] 创建了一个数组。
  - 二、把新创建的数组的引用赋值给变量 arrayRefVar。
- 处理数组
  - 数组的元素类型和数组的大小都是确定的，所以处理数组元素时候，我们通常使用基本循环或者 foreach 循环。
  - foreach 循环 ，JDK 1.5 引进了一种新的循环类型，被称为 foreach 循环或者加强型循环，它能在不使用下标的情况下遍历数组。
- Arrays 类
  - 提供的有用方法是静态的
  - 给数组赋值：通过 fill 方法。
    对数组排序：通过 sort 方法,按升序。
    比较数组：通过 equals 方法比较数组中元素值是否相等。
    查找数组元素：通过 binarySearch 方法能对排序好的数组进行二分查找法操作。

Number & Math 类

- 所有的包装类 Integer、Long、Byte、Double、Float、Short）都是抽象类 Number 的子类
  - Java 的 Math 包含了用于执行基本数学运算的属性和方法，很好等指数、对数、平方根和三角函数。
    Math 的方法都被定义为 static 形式，通过 Math 类可以在主函数中调用。
  - 序号 方法名描述
    1 xxxValue() 将 Number 对象转换为xxx数据类型的值并返回。
    2 compareTo() 将Number对象与参数比较。
    3 equals() 判断number对象是否与参数相等。
    4 valueOf() 返回一个 Number 对象指定的内置数据类型
    5 toString() 以字符串形式返回值。
    6 parseInt() 将字符串解析为int类型。
    7 abs() 返回参数的绝对值。
    8 ceil() 返回大于等于（>=）指定参数的最小整数。
    9 floor() 返回小于等于（<=）指定参数的最大整数。
    10 rint() 返回与参数最接近的整数。返回类型为double。
    11 round() 它表示四舍五入，算法为 Math.floor(x+0.5)，即将原来的数字加上 0.5 后再向下取整，所以，Math.round(11.5) 的结果为12，Math.round(-11.5) 的结果为-11。
    12 min() 返回两个参数中的最小值。
    13 max() 返回两个参数中的最大值。
    14 exp() 返回自然数底数e的参数次方。
    15 log() 返回参数的自然数底数的对数值。
    16 pow() 返回第一个参数的第二个参数次方。
    17 sqrt() 求参数的算术平方根。
    18 sin() 求指定double类型参数的正弦值。
    19 cos() 求指定double类型参数的余弦值。
    20 tan() 求指定double类型参数的正切值。
    21 asin() 求指定double类型参数的反正弦值。
    22 acos() 求指定double类型参数的反余弦值。
    23 atan() 求指定double类型参数的反正切值。
    24 atan2() 将笛卡尔坐标转为极坐标，并返回极坐标的角度值。
    25 toDegrees() 将参数转化为角度。
    26 toRadians() 将角度转换为弧度。
    27 random() 返回一个随机数。
  - 参数 Math.floor Math.round Math.ceil
    1.4 1 1 2
    1.5 1 2 2
    1.6 1 2 2
    -1.4 -2 -1 -1
    -1.5 -2 -1 -1
    -1.6 -2 -2 -1

Character 类

- Character 类用于对单个字符进行操作。
  Character 类在对象中包装一个基本类型 char 的值
  - 每一个char类型的参数传递给管理一个Character类型参数的方法时，那么编译器会自动将char类型参数转换为Character对象。这种特征称为装箱，反过来称为拆箱。
  - 转义序列 描述
    \b 在文中插入一个backspace
    \n 在文中换行插入一个空格键
    \s 在文中插入空格
    \t 在文中插入制表符
    \f 在文中换纸插入换页符
    \' 在文中插入单引号
    \" 在文中插入插入双引号
    \\ 在文中插入一个反斜杠
  - 下面是Character类的方法：
    序号 方法与描述
    1 isLetter() 是否是一个字母
    2 isDigit() 是否是一个数字字符
    3 isWhitespace() 是否是一个空格
    4 isUpperCase() 是否是大写字母
    5 isLowerCase() 是否是小写字母
    6 toUpperCase() 指定字母的大写形式
    7 toLowerCase() 指定字母的小写形式
    8 toString() 返回字符的字符串形式，字符串的长度仅为1

Date

- SimpleDateFormat 格式化日期
  - 序号 方法和描述
    1 boolean after(Date date) 若当调用此方法的Date对象在指定日期之后返回true,否则返回false。
    2 boolean before(Date date) 若当调用此方法的Date对象在指定日期之前返回true,否则返回false。
    3 Object clone() 返回此对象的副本。
    4 int compareTo(Date date) 比较当调用此方法的Date对象和指定日期。两者相等时候返回0。调用对象在指定日期之前则返回负数。调用对象在指定日期之后则返回正数。
    5 int compareTo(Object obj) 和date类date类型转换作用同于compareTo(Date) 。若则它触发ClassCastException。
    6 boolean equals(Object date) 当调用此方法的Date对象和指定日期相等时候返回true,否则返回false。
    7 long getTime() 返回自 1970 年 1 月 1 日 00:00:00 GMT 以来此 Date 对象表示的毫秒数。
    8 int hashCode() 返回此对象的哈希码值。
    9 void setTime(long time) 用自1970年1月1日00:00:00 GMT以后time毫秒数设置时间和日期。
    10 String toString() 把此 Date 对象转换为以下形式的 String： dow mon dd hh:mm:ss zzz yyyy 其中： dow 是一周中的某一天 (Sun, Mon, Tue, Wed, Thu, Fri, Sat)。
  - Date dNow = new Date();
    SimpleDateFormat ft = new SimpleDateFormat ("E yyyy.MM.dd 'at' hh:mm:ss a zzz");
    System.out.println("Current Date: " + ft.format(dNow));
  - y 西纪年份 2001
    M 月份 July or 07
    d 一月的日期 10
    h A.M./P.M. (1~12)格式小时 12
    H 一天中的小时 (0~23) 22
    m 分钟数 30
    s 秒数 55
    S 毫秒数 234
    E 星期几 Tuesday
- Calendar
  - 创建一个代表系统当前日期的Calendar对象
    - Calendar c = Calendar.getInstance();//默认是当前日期
  - 创建一个指定日期的Calendar对象
    - Calendar c1 = Calendar.getInstance();
      c1.set(2009, 6 - 1, 12);
  - 常量 描述
    Calendar.YEAR 年份
    Calendar.MONTH 月份
    Calendar.DATE 日期
    Calendar.DAY_OF_MONTH 日期，和上面的字段意义完全相同
    Calendar.HOUR 12小时制的小时
    Calendar.HOUR_OF_DAY 24小时制的小时
    Calendar.MINUTE 分钟
    Calendar.SECOND 秒
    Calendar.DAY_OF_WEEK 星期几
    - Calendar 类对象字段类型
  - 在具有默认语言环境的默认从地区
  - GregorianCalendar
- printf格式化日期
  - // 初始化 Date 对象
    Date date = new Date();
    //d的使用
    System.out.printf("全新日期时间信息：%tc%n",date);
    //Y的使用
    System.out.printf("年-月-日格式：%tY%n",date);
    //d的使用
    System.out.printf("月/日/年格式：%tD%n",date);
    //h的使用
    System.out.printf("时:HH:MM:SS PM格式（12制）：%tr%n",date);
    //H的使用
    System.out.printf("时:HH:MM:SS格式（24时制）：%tT%n",date);
    //R的使用
    System.out.printf("时:HH:MM格式（24时制）：%tR",date);

Java日期时间

StringBuffer 和 StringBuilder

- StringBuffer 和 StringBuilder 类的对象能够被多次的修改，并且不产生新的未使用对象
- StringBuilder 不是线程安全的（不能同步访问）,但较于 StringBuffer 有速度优势，多数情况下建议使用 StringBuilder 类
- StringBuffer线程安全，较慢，要求线程安全的情况下，则必须使用 StringBuffer 类
- 与String的不同
  - 序号 方法描述
    1 public StringBuffer append(String s) 将指定数字符串追加到此字符序列。
    2 public StringBuffer reverse() 将此字符序列用其反转形式取代。
    3 public delete(int start, int end) 移除此序列的子字符串中的字符。
    4 public insert(int offset, int i) 将 int 参数的字符表示形式插入此序列中。
    5 replace(int start, int end, String str) 使用给定 String 中的字符替换此序列的子字符串中的字符。
    - StringBuffer 方法

String 类

- String greeting = "菜鸟教程";
  - 创建字符串
- length()方法，它返回字符串对象包含的字符数。
  - 字符串长度
- stringC.concat(stringC2);
  - 连接字符串
- 我们知道输出格式化数字可以使用 printf() 和 format() 方法。
  String 类使用静态方法 format() 返回一个String 对象而不是 PrintStream 对象。
  String 类的静态方法 format() 能用来创建可复用的格式化字符串，而不仅仅是用于一次打印输出。
  - 创建格式化字符串
- 和序号 方法描述
  1 char charAt(int index)
  返回指定索引处的 char 值。
  2 int compareTo(Object o)
  把这个字符串和另一个对象比较。
  3 int compareTo(String anotherString)
  按字典顺序比较两个字符串。
  4 int compareToIgnoreCase(String str)
  按字典顺序比较两个字符串，不考虑大小写。
  5 String concat(String str)
  将指定字符串连接到此字符串的结尾。
  6 boolean contentEquals(StringBuffer sb)
  当且仅当字符串与指定的StringBuffer有相同顺序的字符时候返回真。
  7 static String copyValueOf(char[] data)
  返回指定数组中表示该字符序列的 String。
  8 static String copyValueOf(char[] data, int offset, int count)
  返回指定数组中表示该字符序列的 String。
  9 boolean endsWith(String suffix)
  测试此字符串是否以指定的后缀结束。
  10 boolean equals(Object anObject)
  将此字符串与指定的对象比较。
  11 boolean equalsIgnoreCase(String anotherString)
  将此 String 与另一个 String 比较，不考虑大小写。
  12 byte[] getBytes()
  使用平台的默认字符集将此 String 编码为 byte 序列，并将结果存储到一个新的 byte 数组中。
  13 byte[] getBytes(String charsetName)
  使用指定的字符集将此 String 编码为 byte 序列，并将结果存储到一个新的 byte 数组中。
  14 void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)
  将字符从此字符串复制到指定字符数组。
  15 int hashCode()
  返回此字符串的哈希码。
  16 int indexOf(int ch)
  返回指定字符在此字符串中第一次出现处的索引。
  17 int indexOf(int ch, int fromIndex)
  返回在此字符串中第一次出现指定字符处的索引，从指定的索引开始搜索。
  18 int indexOf(String str)
  返回指定字符串在此字符串中第一次出现处的索引。
  19 int indexOf(String str, int fromIndex)
  返回指定字符串在此字符串中第一次出现处的索引，从指定的索引开始。
  20 String intern()
  返回字符串对象的规范化表示形式。
  21 int lastIndexOf(int ch)
  返回指定字符在此字符串中最后一次出现处的索引。
  22 int lastIndexOf(int ch, int fromIndex)
  返回指定字符在此字符串中最后一次出现处的索引，从指定的索引处开始向后搜索。
  23 int lastIndexOf(String str)
  返回指定字符串在此字符串中最右边出现处的索引。
  24 int lastIndexOf(String str, int fromIndex)
  返回指定字符串在此字符串中最后一次出现处的索引，从指定的索引开始反向搜索。
  25 int length()
  返回此字符串的长度。
  26 boolean matches(String regex)
  告知此字符串是否匹配给定的正则表达式。
  27 boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)
  测试两个字符串区域是否相等。
  28 boolean regionMatches(int toffset, String other, int ooffset, int len)
  测试两个字符串区域是否相等。
  29 String replace(char oldChar, char newChar)
  返回一个新的字符串，它是通过用 newChar 替换此字符串中出现的所有 oldChar 得到的。
  30 String replaceAll(String regex, String replacement
  使用给定的 replacement 替换此字符串所有匹配给定的正则表达式的子字符串。
  31 String replaceFirst(String regex, String replacement)
  使用给定的 replacement 替换此字符串匹配给定的正则表达式的第一个子字符串。
  32 String[] split(String regex)
  根据给定正则表达式的匹配拆分此字符串。
  33 String[] split(String regex, int limit)
  根据匹配给定的正则表达式来拆分此字符串。
  34 boolean startsWith(String prefix)
  测试此字符串是否以指定的前缀开始。
  35 boolean startsWith(String prefix, int toffset)
  测试此字符串从指定索引开始的子字符串是否以指定前缀开始。
  36 CharSequence subSequence(int beginIndex, int endIndex)
  返回一个新的字符序列，它是此序列的一个子序列。
  37 String substring(int beginIndex)
  返回一个新的字符串，它是此字符串的一个子字符串。
  38 String substring(int beginIndex, int endIndex)
  返回一个新字符串，它是此字符串的一个子字符串。
  39 char[] toCharArray()
  将此字符串转换为一个新的字符数组。
  40 String toLowerCase()
  使用默认语言环境的规则将此 String 中的所有字符都转换为小写。
  41 String toLowerCase(Locale locale)
  使用给定 Locale 的规则将此 String 中的所有字符都转换为小写。
  42 String toString()
  返回此对象本身（它已经是一个字符串！）。
  43 String toUpperCase()
  使用默认语言环境的规则将此 String 中的所有字符都转换为大写。
  44 String toUpperCase(Locale locale)
  使用给定 Locale 的规则将此 String 中的所有字符都转换为大写。
  45 String trim()
  返回字符串的副本，忽略前导空白和尾部空白。
  46 static String valueOf(primitive data type x)
  返回相关data type类型x参数的字符串表示形式。
  - String 方法