面向对象的核心思想:“找合适的对象，做适合的事情”。JavaSE中已经描述了很多常用的类，可以使用这些类创建对象。API就是应用程序接口，Application Program Interface。

这些常用的类一般位于java.lang包下，不需要导包。

# 1 Object类

Object类所有类的基类，因此任何类的对象都可以使用Object中提供的方法，也可以根据需要其中的重写方法。API文档中对Object的解释：

*Class Object is the root of the class hierarchy. Every class has Object as a superclass. All objects, including arrays, implement the methods of this class.*

Object的构造是无参构造。下面讲解Object中常用的方法。

## 1.1 equals()方法

原型是：public boolean equals(Object obj)

这个方法用于比较两个对象是否相等，默认相当于“==”的功能：

对于值类型，比较他们的值是否相等；对于引用类型，比较他们的内存地址是否相等。

但是我们可以在自定义类中重写equals方法来比较两个对象是否相等。例如，我们规定，只要学生的学号相等，就是同一个对象，那么学生类需要重写equals方法：

|  |
| --- |
| **package** com.zhang.test;  **public class** Student {  **private** String **stuNo**;  **private** String **name**;  **private int age**;  **public** Student(String stuNo) {  **this**.**stuNo** = stuNo;  }  *// 重写equals方法* **public boolean** equals(Object obj) {  **if**(!(obj **instanceof** Student)) {  **return false**; *// 若不是Student类型，直接不是相等的对象* }  **if**(obj == **this**) {  **return true**; *// 若比较的对象直接==此对象，肯定是相等的对象* }  *// 由于用了instanceof，这时可以放心强制转换* Student student = (Student)obj;  **if**(student.getStuNo() == **this**.getStuNo()) {  **return true**;  } **else** {  **return false**;  }  }   **public** String getStuNo() {  **return stuNo**;  }   **public void** setStuNo(String stuNo) {  **this**.**stuNo** = stuNo;  }   **public** String getName() {  **return name**;  }   **public void** setName(String name) {  **this**.**name** = name;  }   **public int** getAge() {  **return age**;  }   **public void** setAge(**int** age) {  **this**.**age** = age;  } } |

Demo类：

|  |
| --- |
| **package** com.zhang.test;  **public class** Demo **extends** Object {  **public static void** main(String[] args) {  Student student1 = **new** Student(**"1001"**);  Student student2 = **new** Student(**"1001"**);  Student student3 = **new** Student(**"1002"**);  System.***out***.println(student1 == student2); *// 使用==还是不等的* System.***out***.println(student1.equals(student2)); *// 使用equals相等* System.***out***.println(student1.equals(student3)); *// 不等* } } |

对于String类，Java 已经为我们重写好了equals方法，以后比较字符串是否相等时，就用equals方法。

## 1.2 hashCode()方法

方法原型：public int hashCode()；

作用：返回该对象的哈希码值，这个值默认是由对象的内存地址计算出来的，每次运行程序返回结果可能不一样。

提供此方法是为了提高哈希表的性能，以后会学到。一般就是重写equals方法的同时重写hashCode方法，让“相同”的对象返回相同的hash码，让不相同的对象尽量返回不同的哈希码，这样有利于提高哈希表性能。具体的做法是将用于判断相同对象的属性相加，返回这个字符串的对应的hashCode。

## 1.3 getClass()方法

方法原型：public final Class<?> getClass()；

方法返回的是当前对象运行的字节码对象Class类对象，通过Class对象可获得该对象的完整类名。即obj.getClass().getName()得到的是对象的类名。该方法是final修饰的，不可重写。例如：

|  |
| --- |
| **package** com.zhang.test;  **public class** Demo **extends** Object {  **public static void** main(String[] args) {  Class c = **new** Demo().getClass();  System.***out***.println(c.getName());  *// 输出结果是com.zhang.test.Demo* } } |

## 1.4 toString()方法

方法的原型：public String toString()；

作用：返回该对象的字符串表示。自定义类可重写此方法以方便利用toString()直接输出对象的一些信息。查看JDK源码，发现toString()默认返回的是：

|  |
| --- |
| **public** String toString() {  **return** getClass().getName() + **"@"** + Integer.*toHexString*(hashCode()); } |

就是：“类名+@+hashCode值的十六进制形式”。

当用System.out.println(对象)输出对象时，显示的就是toString()返回的值。String类已经重写了toString()方法，所以直接输出看到的是字符串。

## 1.5 finalize()和clone()方法

方法原型分别是：

protected Object clone() throws CloneNotSupportedException和

protected void finalize() throws Throwable

clone方法的作用是创建并返回这个对象的一个副本，该副本的各属性值都和之前的对象一样，但是内存地址不一样，相当于拷贝一份。finalize方法是用于垃圾回收的，知道即可。

这两个方法都是protected修饰的，因此自定义对象不能直接访问这两个方法，必须在自定义对象中重写这两个方法，然后对象才能访问。

并且，如果类重写了clone()方法，则这个类必须要实现“Cloneable”标记接口。

例子：

|  |
| --- |
| **package** com.zhang.test;  *// 需要实现Cloneable接口* **public class** Student **implements** Cloneable {  **private** String **name**;  **private int age**;  **public** Student(String name, **int** age) {  **this**.**name** = name;  **this**.**age** = age;  }   **public** String toString() {  **return this**.getName() + **" "** + **this**.getAge();  }   *// 重写clone和finalize* **protected** Object clone() **throws** CloneNotSupportedException {  **return super**.clone(); *// 一定是super.clone* }   **protected void** finalize() **throws** Throwable {  **super**.finalize();  }   **public** String getName() {  **return name**;  }   **public void** setName(String name) {  **this**.**name** = name;  }   **public int** getAge() {  **return age**;  }   **public void** setAge(**int** age) {  **this**.**age** = age;  } } |

Demo使用类：

|  |
| --- |
| **package** com.zhang.test;  **public class** Demo **extends** Object {  **public static void** main(String[] args) **throws** Throwable {  Student stu1 = **new** Student(**"张三"**, 12);  Student stu2 = (Student) stu1.clone(); *// 需要强转* System.***out***.println(stu1 == stu2); *//两者不等* System.***out***.println(stu1.toString());  System.***out***.println(stu2.toString()); *//但属性相同* stu1.finalize();  } } |

# 2 Scanner类

之前用过Scanner sc = new Scanner(System.in)；为什么能这样用，因为Scanner有这样的构造器：public Scanner(InputStream source)，而System.in的类型就是InputStream。所以这样就能把数据显示在控制台。

主要用到的方法：

hasNextXxx(),判断下一个输入的数据是不是某种类型；Xxx就是一些基本数据类型；

nextXxx(),获取下一个输入的数据。如果是字符串类型，那么就不要Xxx。

# 3 String类

## 3.1 构造方法

String提供了多个构造。

（1）public String()：初始化一个空字符序列。

（2）public String(String otiginal)：由字符串参数创建一个新的该字符串副本。

（3）public String(char[] value)：用字符数组来创建字符串

（4）public String(char[] value, int offset, int count)：把字符数组的一部分转换成字符串。

（5）public String(byte[] bytes)：把字节数组按照ASCII码转成字符串

（6）public String(byte[] bytes, int offset, int length)：把字节数组一部分按照ASCII码转成字符串。

其他的构造方法就不介绍了。

## 3.2 字符串的特性

先看例子：

|  |
| --- |
| **package** com.zhang.test; **public class** Demo {  **public static void** main(String[] args) {  String str1 = **"hello"**;  String str2 = **"hello"**;  String str3 = **new** String(**"hello"**);  System.***out***.println(str1 == str2); *// true* System.***out***.println(str1 == str3); *// false* } } |

为什么出现这样的情况？

Java中有字符串常量池，当把字符串常量直接赋值给String变量时（即String str = “”的形式），Java会先在字符串池中找这个常量，如果找到，直接引用这个字符串；如果找不到，就在池中创建该字符串并引用。然后，当再有相同的字符串常量赋值给String变量时，就能直接引用常量池中内容了。因此，上面的str1和str2都是同一个对象。

但是通过new创建就不一样了，这样会每次新创建字符串来赋值给String变量，因此不相等。如果单单比较字符串值是否相等，应该使用equals方法。

例2：

|  |
| --- |
| **package** com.zhang.test; **public class** Demo{  **public static void** main(String[] args) {  String str = **"hello world"**;  System.***out***.println(str == **"hello "** + **"world"**);  *//true。因为编译器先将字符串优化成"hello world"，这样，字符串池中就先有了该字符串。* } } |

并且字符串有不可变性。当String变量的值改变时，只会指向另一个字符串，而不会改变原有字符串的值。

下面讲String中的方法。

## 3.3 String中常用的方法

### 3.3.1 获取方法

（1）public int length()：返回此字符串的长度；

（2）public char charAt(int index)：获取指定索引处的字符；

（3）public int indexOf(String str)和public int indexOf(int ch)：获取第一次出现指定字符串/字符处的索引号。int ch参数传递的是int类型，实际上也就是支持传递char类型。

（4）public int lastIndexOf(String str)和public int lastIndexOf(int ch)：获取最后一次出现指定字符串/字符处的索引号。

（5）public int indexOf(String str,int fromIndex)和public int indexOf(int ch, int fromIndex):返回指定字符串/字符在此字符串中从指定位置后第一次出现处的索引。

（6）String substring(int start):从指定位置开始截取字符串,默认到末尾。

（7）String substring(int start,int end):从指定位置开始到指定位置结束截取字符串。

（8）String concat(String str):把字符串拼接。返回新字符串。

### 3.3.2 判断方法

（1）boolean equalsIgnoreCase(String str)：比较字符串内容相同时忽略大小写；

（2）boolean contains(String str)：判断是否包含指定字符串；

（3）boolean startsWith(String str)：判断字符串是否以指定的字符串开头；

（4）boolean endsWith(String str)：判断字符串是否以指定的字符串结尾；

（5）boolean isEmpty()：判断字符串是否为空。注意，字符串为空指的是空字符串””，而不是null。因为null是空指针，null不能调用方法，否则出现空指针异常NullPointerException。

### 3.3.3 转换方法

（1）byte[] getBytes()：把字符串转换为字节数组。

（2）char[] toCharArray()：把字符串转换为字符数组。

（3）static String valueOf(char[] chs)：把字符数组转成字符串。

（4）static String valueOf(int i)：把int类型的数据转成字符串。

注意：String类的valueOf静态方法可以把任意类型的数据转成字符串。

（5）String toLowerCase()：把字符串转成小写。

（6）String toUpperCase()：把字符串转成大写。

### 3.3.4 其他方法

（1）String replace(char old,char new)和String replace(String old,String new)：将字符串中原有的指定字符串替换成新的字符串/字符。

（2）String trim()：去除字符串的两端空格。

（3）int compareTo(String str)

（4）int compareToIgnoreCase(String str)

上面两个的含义：按字典顺序比较两个字符串（得到的结果是整数，0表示相等，数值是根据ASCII值差值计算出来的）。

# 4 StringBuilder和StringBuffer类

由于String是不可变的，所以在频繁改变字符串对象的应用中，需要使用可变的字符串缓冲区类，解决了大量字符串拼接耗费的内存和时间。

StringBuffer类是线程安全的可变字符序列。而StringBuilder类是线程不安全的。在使用上都是一样的。

## 4.1 什么是线程安全

线程安全，即数据同步，访问时数据会加锁，保证数据的一致性。因此在多线程应用时多使用线程安全的类。

线程不安全，数据不同步，不会在访问时加锁，效率高一些。安全和效率是矛盾的。

## 4.2 StringBuilder / StringBuffer的构造方法

以StringBuilder为例，StringBuffer也是一样的。

（1）public StringBuilder()：无参构造

（2）public StringBilder(int capacity)：创建指定初始容量大小的可变字符序列对象

（3）public StringBuffer(String str)：创建指定字符串内容的StringBuilder对象

可以通过capacity()和length()方法来获得对象的当前容量和长度。容量会根据需要增大。

## 4.3 给字符串序列添加内容

主要使用append和insert方法：

（1）public StringBuilder append(String str)：有其他重载，可把任意类型数据（有其他重载）添加进对象，并且也返回对象本身。

（2）public StringBuilder insert(int offset,String str):在指定位置把任意类型的数据（有其他重载）插入，并返回对象本身。

因此，可以链式编程：

sb.append("hello").append(true).append(12).append(34.56);

## 4.4 其他常用方法

（1）public StringBuffer deleteCharAt(int index)：删除指定位置的字符，并返回本身。

（2）public StringBuffer delete(int start, int end):从指定位置开始删除内容，到指定位置结束，并返回本身。注意Java中一般含左不含右。

清空对象的内容可用：sb.delete(0, sb.length());

（3）public StringBuffer replace(int start,int end,String str):从start开始到end用str替换

（4）StringBuffer的截取功能：注意返回值类型变为String了：

public String substring(int start)

public String substring(int start,int end)

（5）反转功能：public StringBuffer reverse()，就是将原有内容倒序再返回。

## 4.5 和String的联系

### 4.5.1 String类型和可变字符类型的相互转换

有时需要进行String和StringBuffer/StringBuilder的转换。不能直接把字符串值赋给StringBuffer。

（1）String转换成StringBuilder / StringBuffer：

方法一：用StringBuffer的构造方法：StringBuffer sb = new StringBuffer(str);

方法二：通过append方法将字符串加进StringBuffer对象里。

（2）将StringBuilder / StringBuffer转换成String：

方法一：String的构造方法：String s = new String(StringBuffer sb);

方法二：调用StringBuffer的toString()方法。

### 4.5.2 作为参数传递

String作为参数传递：不会影响传进来的实参，但如果是StringBuffer/StringBuilder，就会影响实参了。

# 5 Arrays类

Arrays类包含用来操作数组（比如排序和搜索）的各种方法。全部都是静态方法。

常用的有：

（1）public static String toString(int[] a)：把数组转成字符串

（2）public static void sort(int[] a) 对数组排序，升序

（3）public static int binarySearch(int[] a,int key) 二分查找数组中元素返回索引，前提是数组是升序的。如果找不到，返回的索引小于0。

# 6 基本类型包装类

为了对基本数据类型进行更多的操作，Java针对每一种基本数据类型提供了对应的类类型，就是包装类类型。

下面是基本数据类型所对应的包装类：

byte -> Byte；short -> Short；int –> Integer；long -> Long；float -> Float；double -> double；char -> Character；boolean - > Boolean

这些类中主要都是静态成员。常用的是，包装类中提供了把字符串转换成特定的数据类型的方法，即parseXxx方法，比如：Interger.parseInt(str)。

## 6.1 Integer

### 6.1.1 构造方法

（1）public Integer(int value)：由int类型创建Integer对象

（2）public Integer(String s)：可将字符串转换成Integer类型，字符串必须符合要求，否则不能转换会出现NumberFormatException异常。

### 6.1.2 将十进制转换为其他进制的字符串形式

常用的有将十进制转换为二进制、八进制和十六进制的字符串。分别用以下方法：

（1）public static String toBinaryString(int i)：转换为二进制

（2）public static String toOctalString(int i)：八进制

（3）public static String toHexString(int i)：十六进制。

例子：String s = Integer.toHexString(100);

还能用toString()方法将十进制转为任意进制的字符串，注意该方法的原型，是静态方法：

public static String toString(int t, int radix)，参数1是十进制数，参数2是要转换成的进制数。

比如String fif = Integer.toString(100, 15)，就是把100转为15进制。

参数2的取值范围是[2, 36]，因为数字加上字母能表示的符号只有36个。如果不在这个范围，就不进行转换，返回的还是十进制的字符串形式。

既然能把十进制转为任意进制，那怎么把其他进制字符串转为十进制？还是用Integer.parseInt(String s, int radix)这个重载。参数2就表示参数1的进制。

比如：Integer.parseInt(“7a”, 16)，就表示把16进制的7a转为十进制返回int类型。

需要注意不能写该进制没有的符号，比如：Integer.parseInt(“123”,2)；这样会出现异常，因为2进制中没有2和3。当然，不写参数2，就是直接将字符串当做十进制转换成十进制的数字。

### 6.1.3 int类型的最值

Integer中提供了两个常量表示int类型的最大值和最小值，分别是：

public static final int MAX\_VALUE；

public static final int MIN\_VALUE。

外界访问使用Integer.MAX\_VALUE。

### 6.1.4 int和String相互转换

（1）int转换成String

使用String.valueOf()方法：String str = String.valueOf(100)；

使用Integer.toString()方法：String str = Integer.toString(100);

（2）String转换成int

使用Integer.parseInt()方法：int i = Integer.parseInt(“100”);

先把String转成Integer，再用Integer.intValue方法：

Integer integer = new Integer(“100”);

int i = integer.intValue();

### 6.1.5 装箱和拆箱

基本数据类型转成对应的包装类型叫做装箱，常用方法valueOf()，例如：Integer i = new Integer.valueOf(num)；

包装类型转成基本数据类型叫做拆箱，常用方法xxxValue，例如：int num = i.intValue()。

JDK5后，自动帮助我们完成了装箱和拆箱过程，也就是基本数据类型和包装类型可以直接相互赋值，例如：Integer i = 100；int num = new Integer(100)。

### 6.1.6 Integer常量池

Integer中有常量池，值的范围是[-128, 127]，如果你直接赋值给Integer的变量值在此范围，那么会直接从常量池中得到。例子：

|  |
| --- |
| **package** com.zhang.test;  **public class** Demo {  **public static void** main(String[] args) {  Integer i1 = 127;  Integer i2 = 127;  System.***out***.println(i1 == i2); *// true。因为两者直接从常量池中拿到相同的对象。不会新创建对象* Integer j1 = 300;  Integer j2 = 300;  System.***out***.println(j1 == j2); *// false，因为会创建不同对象* Integer k1 = **new** Integer(127);  Integer k2 = **new** Integer(127);  System.***out***.println(k1 == k2); *// false,因为new还是会创建对象* } } |

## 6.2 Character

Character 类的对象中包装一个 char类型的值。该类提供了判断字符类别和转换大小写的方法。详细如下：

构造方法：public Character(char ch)；

常用方法（静态）：

public static boolean isUpperCase(char ch)：判断字符是否是大写字符

public static boolean isLowerCase(char ch)：判断字符是否是小写字符

public static boolean isDigit(char ch)：判断字符是否是数字字符

public static char toUpperCase(char ch)：把字符转换为大写字符

public static char toLowerCase(char ch)：把字符转换为小写字符