# 二、面向对象编程

[二、面向对象编程](#header-n0)  
 [1.面向对象基础](#header-n3)  
 [1.1 方法](#header-n4)  
 [1.2 构造方法](#header-n8)  
 [1.3 方法重载](#header-n11)  
 [1.4 继承](#header-n13)  
 [1.5 多态](#header-n24)  
 [1.6 抽象类](#header-n39)  
 [1.7 接口](#header-n42)  
 [1.8 静态字段和静态方法](#header-n50)  
 [1.9 包](#header-n59)  
 [1.10 作用域](#header-n73)  
 [1.11 classpath和jar](#header-n76)  
 [1.12 模块](#header-n82)  
 [2.Java核心类](#header-n83)  
 [2.1 字符串和编码](#header-n84)  
 [2.1.1 字符串比较](#header-n85)  
 [2.1.2 去除首尾空白字符](#header-n97)  
 [2.1.3 替换子串](#header-n103)  
 [2.1.4 分割字符串](#header-n105)  
 [2.1.5 拼接字符串](#header-n109)  
 [2.1.6 类型转换](#header-n113)  
 [2.2 StringBuilder](#header-n87)  
 [2.3 StringJoiner](#header-n88)  
 [2.4 包装类型](#header-n89)  
 [2.5 JavaBean](#header-n90)  
 [2.6 枚举类](#header-n91)  
 [2.7 BigInteger](#header-n92)  
 [2.8 BigDecimal](#header-n93)  
 [2.9 常用工具类](#header-n94)

## 1.面向对象基础

### 1.1 方法

定义方法：

修饰符 方法返回类型 方法名(方法参数列表){  
 若干方法语句;  
 return 方法返回值;  
}

### 1.2 构造方法

任何类都有构造方法，如果我们没有编写构造方法，编译器会自动为我们生成一个默认的构造方法。

一个构造方法可以调用其他构造方法，调用其他构造方法的语法是this(...)

### 1.3 方法重载

方法名相同，但各自的参数不同，称为方法重载。

### 1.4 继承

继承是面向对象编程中非常强大的一种机制，它首先可以复用代码。

Java只允许一个class继承自一个类。

继承时，子类无法访问父类的private字段或private方法。为了让子类可以访问父类的字段，我们需要把private改为protected。

protected关键字可以把字段和方法的访问权限控制在继承树内部。

Java使用extends关键字来实现继承。

super关键字表示父类(超类)。

如果父类没有默认的构造方法，子类就必须显式调用super()并给出参数以便让编译器定位到父类的一个合适的构造方法。

子类不会继承任何父类的构造方法，子类默认的构造方法是编译器自动生成的。

向上转型：把一个子类类型安全地变为父类类型的赋值。

向下转型：和向上转型相反。

### 1.5 多态

在继承关系中，子类如果定义了一个与父类方法签名完全相同的方法，被称为覆写(Override)。

如果方法签名不同就是Overload，Overload方法是一个新方法。

Java的实例方法调用的是基于运行时的实际类型的动态调用，而非变量的声明类型。

多态具有一个非常强大的功能，就是允许添加更多类型的子类实现功能扩展，却不需要修改基于父类的代码。

**覆写Object方法：**

所有的class最终都继承自Object，而Object定义了几个重要的方法：

toString()：把instance输出为String；

equals()：判断两个instance是否逻辑相等；

hashCode()：计算一个instance的哈希值。

在必要的情况下，我们可以覆写Object的这几个方法。

**调用super：**

在子类的覆写方法中，如果要调用父类的被覆写的方法，可以通过super来调用。

**final：**

继承可以允许子类覆写父类的方法。如果一个父类不允许子类对它的某个方法进行覆写，可以把该方法标记为final。用final修饰的方法不能被Override。

### 1.6 抽象类

如果父类的方法本身不需要实现任何功能，仅仅是为了定义方法签名，目的是让子类去覆写它，那么可以父类的方法声明为抽象方法。把一个方法声明为abstract，表示它是一个抽象方法，本身没有任何方法语句。

如果一个class定义了方法，但没有具体的执行代码，这个方法就是抽象方法，抽象方法用abstract修饰。因为无法执行抽象方法，因此这个类也必须申明为抽象类。我们无法实例化一个抽象类。

### 1.7 接口

如果一个抽象类没有字段，所有方法全部都是抽象方法，就可以把该抽象类定义为接口：interface。

当一个具体的class去实现一个interface时，需要使用implements关键字。

在Java中一个类只能继承自另一个类，不能从多个类继承，但是一个类可以实现多个interface。

**接口继承**

一个interface可以继承自另一个interface，interface继承自interface使用extends。

**default方法**

default方法的目的是当新增一个方法时，会涉及到修改全部子类，如果新增的是default方法，那么子类就不必全部修改，只需要在需要覆写的地方去覆写新增方法。

### 1.8 静态字段和静态方法

**静态字段**

在class中定义的字段称为实例字段，每个实例都有独立的字段，各个实例的同名字段互不影响。用static修饰的字段称为静态字段，静态字段只有一个共享空间，所有的实例都会共享该字段。

Java中推荐用类名来访问静态字段，类名.静态字段

**静态方法**

用static修饰的方法称为静态方法，调用静态方法直接通过类名就可以调用，静态方法类似其他编程语言的函数。

静态方法内部无法访问this变量，也无法访问实例字段，它只能访问静态字段。

**接口的静态字段**

interface不能定义实例字段，但是interface是可以有静态字段的，并且静态字段必须为final类型。

### 1.9 包

在Java中我们使用package来解决名字冲突。

**包作用域**

位于同一个包的类，可以访问包作用域的字段和方法，不用public、protected、private修饰的字段和方法就是包作用域。

**import**

在一个class中引用其他的class的方法：

* 直接写出完整类名
* 使用import语句
* 使用import static语句导入一个类的静态字段和静态方法

### 1.10 作用域

在Java中public、protected、private可以用来限定访问作用域。

一个.java文件只能包含一个public类，但可以包含多个非public类。

### 1.11 classpath和jar

classpath是JVM用到的环境变量，它用来只是JVM如何搜索class。

强烈不推荐在系统环境变量中设置classpath，那样会污染整个系统环境，应该在启动JVM时设置classpath。实际上就是给java传入-classpath或-cp参数。

**jar包**

如果有很多.class文件散落在各层目录中不便于管理，将目录打一个包，变成一个文件就方便多了。jar包就是zip包，将文件压缩为zip格式后更改文件后缀为jar就完成了。

jar包含一个特殊的MANIFEST.MF文件可以提供jar包的信息，这样可以直接运行jar包。

### 1.12 模块

## 2.Java核心类

### 2.1 字符串和编码

#### 2.1.1 字符串比较

字符串比较时必须使用equals()方法而不能用==，要忽略大小写比较使用equalsIgnoreCase()方法。

#### 2.1.2 去除首尾空白字符

使用trim()方法可以移除字符串首尾空白字符，空白字符包括空格、\t、\r、\n；

strip()方法也可以移除字符串首尾空白字符，类似中文的空格字符\u3000也会被移除。

#### 2.1.3 替换子串

#### 2.1.4 分割字符串

使用split()方法，并且传入的也是正则表达式。

#### 2.1.5 拼接字符串

拼接字符串使用静态方法join()，它用指定的字符串连接字符串数组。

#### 2.1.6 类型转换

要把任意基本类型或引用类型转换为字符串，可以使用静态方法valueof()

### 2.2 StringBuilder

### 2.3 StringJoiner

### 2.4 包装类型

### 2.5 JavaBean

### 2.6 枚举类

### 2.7 BigInteger

### 2.8 BigDecimal

### 2.9 常用工具类