# Java技巧

## 类的设计技巧

1. 一定要保证数据私有
2. 一定要对数据初始化
3. 不要在类中使用过多的基本类型
4. 不是所有的域都需要独立的域访问器和域更改器
5. 将职责过多的类进行分解
6. 类名和方法名要能够体现它们的职责

命名类名的良好习惯是采用一个**名词**、**前面有形容词修饰的名词**、或**动名词修饰名词**。

对于方法来说，习惯是访问器方法用**小写的get开头**，更改器方法**用小写set开头**。

## 继承设计的技巧

1. 将公共操作放在超类
2. 不要使用受瓮中保护的域
3. 使用继承实现“is-a”关系
4. 除非所有继承的方法都有意义，否则不要使用继承
5. 在覆盖方法时，不要改变预期的行为
6. 使用多态，而非类型信息
7. 不要过多地使用反射

## 使用异常机制的技巧

1. 异常处理不能代替简单的测试。

与执行简单的测试相比，捕获异常所花费的时间大大超过了前者，因此使用异常的基本规则是：只是在异常情况下使用异常机制。

1. 不要过分地细分化异常。
2. 利用异常层次结构。

不要只抛出RuntimeException异常。应该寻找更加适当的子类或创建自己的异常类。不要只捕获Thowable异常，否则，会使程序代码更难读、更难维护。考虑已检查异常与未检查异常的区别。已检查异常本来就很庞大，不要为逻辑错误抛出这些异常。（例如，反射库的做法就不正确。调用都却经常需要捕获那些早已知道不可能发生的异常）将一种异常转换成另一种更加适合的异常时不要犹豫。例如，在解析某个文件中的一个整数是，捕获NumberFomatException异常，然后将它转换成IOException或MySubsystemException的子类。

1. 不要压制异常。

在Java中，往往强烈地倾向关闭异常。如果编写了一个调用另一个方法的方法，而这个方法有可能100年才抛出一个异常，那么，编译器会因为没有将这个异常列在throws表中产生抱怨。面没有半这个异常列在throws表中主要出于编译器将会对所有调用这个方法的方法进行异常处理的考虑。因此，应该将这个异常关闭。

1. 在检查异常时，“苛刻”要比放任更好。

当检测到错误的时候，有些程序员担心抛出异常。在用无效的参数调用一个方法时，返回一个虚拟的数值，还是抛出一个异常？我们认为，在出错的地方抛出一个EmptyStackException异常要比在后面抛出一个NullPointerException异常更好。

1. 不要羞于传递异常

很多程序员都感觉应该捕获抛出的全部异常。如果调用了一个抛出异常的方法。例如，FileInputStream构造器或readLine方法，这些方法就会本能地捕获这些可能产生的异常。其实，传递异常要比捕获这些异常更好。

# 附A 数据类型表格

## 表1.1 java类型

|  |  |  |
| --- | --- | --- |
| **类型** | **存储需求** | **取值范围** |
| Int | 4字节 | -2 147 483 648 ~ 2 147 483 647（正好超过20亿） |
| Short | 2字节 | -32 768 ~ 32 768 |
| Long | 8字节 | -9 223 372 036 854 775 808 ~ 9 223 372 036 854 775 807 |
| Byte | 1字节 | -128 ~ 127 |
| Float | 4字节 | 大约 ±3.402 823 47E + 38E（有效位数6~7位） |
| Double | 8字节 | 大约 ±1.797 693 134 862 315 70E +308（有效位数15位） |

## 表1.2 特殊字符的转义序列表

|  |  |  |
| --- | --- | --- |
| **转义序列** | **名称** | **Unicode值** |
| \b | 退格 | \u0008 |
| \t | 制表 | \u0009 |
| \n | 换行 | \u000a |
| \r | 回车 | \u000d |
| \” | 双引号 | \u0022 |
| \’ | 单引号 | \u0027 |
| \\ | 反斜杠 | \u005c |

## 表1.3 运算符优先级

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序列号** | **符号** | **名称** | **结合性(与操作数)** | **目数** | **说明** |
| 1 | . | 点 | 从左到右 | 双目 |  |
| ( ) | 圆括号 | 从左到右 |  |  |
| [ ] | 方括号 | 从左到右 |  |  |
| 2 | + | 正号 | 从右到左 | 单目 |  |
| - | 负号 | 从右到左 | 单目 |  |
| ++ | 自增 | 从右到左 | 单目 | 前缀增，后缀增 |
| - - | 自减 | 从右到左 | 前缀减，后缀减 |
| ~ | 按位非/取补运算 | 从右到左 | 单目 |  |
| ！ | 逻辑非 | 从右到左 | 单目 | “!”不可以与“=”联用 |
| 3 | \* | 乘 | 从左到右 | 双目 |  |
| / | 除 | 从左到右 | 双目 | 整数除法：取商的整数部分，小数部分去掉，不四舍五入 |
| % | 取余 | 从左到右 | 双目 |  |
| 4 | + | 加 | 从左到右 | 双目 |  |
| - | 减 | 从左到右 | 双目 |  |
| 5 | << | 左移位运算符 | 从左到右 | 双目 |  |
| >> | 带符号右移位运算符 | 从左到右 | 双目 |  |
| >>> | 无符号右移 | 从左到右 | 双目 |  |
| 6 | < | 小于 | 从左到右 | 双目 | 关系运算符“大于”说明 |
| <= | 小于或等于 | 从左到右 | 双目 |  |
| > | 大于 | 从左到右 | 双目 |  |
| >= | 大于或等于 | 从左到右 | 双目 |  |
| instanceof | 确定某对象是否属于指定的类 | 从左到右 | 双目 |  |
| 7 | == | 等于 | 从左到右 | 双目 | 关系运算符“==”说明 |
| != | 不等于 | 从左到右 | 双目 |  |
| 8 | & | 按位与 | 从左到右 | 双目 |  |
| 9 | | | 按位或 | 从左到右 | 双目 |  |
| 10 | ^ | 按位异或 | 从左到右 | 双目 |  |
| 11 | && | 短路与 | 从左到右 | 双目 |  |
| 12 | || | 短路或 | 从左到右 | 双目 |  |
| 13 | ? : | 条件运算符 | 从右到左 | 三目 |  |
| 14 | = | 赋值运算符 | 从右到左 | 双目 |  |
| += | 混合赋值运算符 |  |
| -= |  |
| \*= |  |
| /= |  |
| %= |  |
| &= |  |
| |= |  |
| ^= |  |
| <<= |  |
| >>= |  |
| >>>= |  |

## 表1.4 用于printf的转换符

|  |  |  |
| --- | --- | --- |
| **转换符** | **类型** | **举例** |
| d | 十进制整数 | 159 |
| X | 十六进制整数 | 9f |
| o | 八进制整数 | 237 |
| f | 定点浮点数 | 15.9 |
| e | 指数浮点数 | 1.59e+01 |
| g | 通用浮点数 | -- |
| a | 十六进制浮点数 | 0x1.fccdp3 |
| s | 字符串 | Hello |
| c | 字符 | h |
| b | 布尔 | True |
| h | 散列码 | 42625b2 |
| tx | 日期时间 | **见表1.6** |
| % | 百分号 | % |
| n | 与平台有关的行分隔符 | --- |

## 表1.5 用于printf的标志

|  |  |  |
| --- | --- | --- |
| **标志** | **目的** | **举例** |
| + | 打印正数和负数的符号 | +12346.12 |
| 空格 | 在正数之前添加空格 | \_132645 |
| 0 | 数字前面补0 | 003321.22 |
| - | 左对齐 | 1321.54\_ |
| ( | 将负数括在括号内 | (654.31) |
| , | 添加数组分隔符 | 1,4564 |
| #（对于f格式） | 包含小数点 | 1,455.25 |
| #（对于x或0格式） | 添加前缀0x或0 | 0xcafe |
| $ | 给定被格式化的参数索引。例如，%1$d,%1$x将以十进制和十六进制格式打印第一个参数 | 159 9F |
| < | 格式化前面说明的数值。例如，%d%<x以十进制和十六进制打印同一数值 | 159 9F |

## 表1.6 日期与时间的转换符

|  |  |  |
| --- | --- | --- |
| **转换符** | **类型** | **举例** |
| c | 完整的日期和时间 | Mon Feb 09 18:05:18 PST 2016 |
| F | ISO 8601 日期 | 2017-02-26 |
| D | 美国格式的日期（月/日/年） | 02/25/2017 |
| T | 24小时时间 | 19:21:22 |
| r | 12小时时间 | 06:05:22 pm |
| R | 24小时时间没有秒 | 19:09 |
| Y | 4位数字的年（前面补0） | 2017 |
| y | 年的后两位数字（前面补0） | 17 04 |
| C | 年的前两位数字（前面补0） | 20 |
| B | 月的完整拼写 | February |
| b或h | 月的缩写 | Feb |
| m | 两位数字的月（前面补0） | 02 |
| d | 两位数字的日（前面补0） | 02 |
| e | 两位数字的月（前面不补0） | 2 |
| A | 星期几的完整拼写 | Monday |
| a | 星期几的缩写 | Mon |
| j | 三位数中的年中的日子（前面补0），在001-366之间 | 062 |
| H | 两位数字的小时（前面补0），在0-23之间 | 15 |
| k | 两位数字的小时（前面不补0）在0-23之间 | 5 |
| I | 两位数字的小时（前面补0），在0-12之间 | 06 |
| l（L的小写） | 两位数字的小时（前面不补0），在0-12之间 | 6 |
| M | 两位数字的分钟（前面补0） | 05 |
| S | 两位数字的秒钟（前面被0） | 52 |
| L | 三位数字的毫秒（前面补0） | 015 |
| N | 九位数字毫微秒（前面补0） | 047000000 |
| P | 上午或下午的大写标志 | AM PM |
| p | 上午或下午的小写标志 | am pm |
| z | 从GMT起，RFC822数字位移 | -0800 |
| Z | 时区 | PST |
| s | 从格林威治时间 1970-01-01 00:00:00起的秒数 | 107884319 |
| Q | 从格林威治时间 1970-01-01 00:00:00起的毫秒数 | 107884319047 |

如：System.out.printf("%1$s %2$tB %2$te,%2$tY %2$tl","Due date",new Date());

打印：Due date 二月 27,2017 6



**图1.1 printf格式说明符语法**

## 表1.7 Java库中的具体集合

|  |  |
| --- | --- |
| ArrayList | 一种可以动态增长和缩减的索引序列 |
| LinkedList | 一种可以在任何位置进行高效地插入和删除操作的有序序列 |
| ArrayDeque | 一种用循环数组实现的双端队列 |
| HashSet | 一种没有重复元素的无序集合 |
| TreeSet | 一种有序集 |
| EnumSet | 一种包含枚举类型值的集 |
| LinkedHashSet | 一种可以记住元素插入次序的集 |
| PriorityQueue | 一种允许高效删除最小元素的集合 |
| HashMap | 一种存储键/值关联的数据结构 |
| TreeMap | 一种键值有序排列的映射表 |
| EnumMap | 一种键值属于枚举类型的映射表 |
| LinkedHashMap | 一种可以记住键/值项添加次序的映射表 |
| WeakHashMap | 一种其值无用武之地后可以被垃圾回收器回收的映射表 |
| IdentityHashMap | 一种用==，而不是equals比较键值的映射表 |