目录
- 技术圈动态
  - 热榜动态
  - Java
    - Java 核心知识点
    - Java数据类型常见面试题总结
    - 注解&反射常见面试题总结
    - String 常见面试题总结
  - 数据库
  - 常见框架
  - 分布式
  - 服务器
  - 服务器
  - Devops
- 技术圈活动资源
- 圈动态
- 圈成员专属
  - 如何成为一个合格的...

Java 数据类型常见面试题总结

云栖社区 发表于 2019-10-29 10:00:00 阅读 1.2K 评论 0 赞 0 收藏 0

大纲
- 基本数据类型
  - 包装类型
    - 包装类型的常量池技术
    - 为什么要有包装类型?
  - 自动拆装箱
    - 什么是自动拆装箱?-原理?
    - 自动拆箱引发的 NPE 问题
    - 案例 1
    - 案例 2

这篇文章的开头，**文章涉及到的概念部分都是对原来博客的修改 Java 基础**。

本篇文章大致分为基本数据类型的概述和拆装箱的相关案例。

基本数据类型

面试官 ：Java 中的基本数据类型是？

我 ：Java 中的基本数据类型，分别为：

```
1. 6 种数字类型：byte、short、int、long、float、double
2. 1 种字符类型：char
3. 1 种布尔类型：boolean。
```

面试官 ：它们的默认值和占用的空间大小分别是？

我 ：这 8 种基本数据类型的默认值以及所占空间的大小如下：

基本类型	位数	字节	默认值
int	32	4	0
short	16	2	0
long	64	8	0L
byte	8	1	0
char	16	2	\u0000
float	32	4	0f
double	64	8	0d
boolean	1		false

另外，对于 boolean，官方文档未明确定义，它依赖于 JVM厂商的具体实现。逻辑上理解是占用 1 位，但是实际中会考虑计算机高效存储因素。

注意：

```
1. Java 里使用 long 类型的数据一定要在数据后面加上 L，否则将作为整型解析。
2. char a = '中'char 由单引号，String a = "hello";双引号
```

包装类型

面试官 ：前说这 8 种基本数据类型对应的包装类是。

我 ：进入两基本类型都有对应的包装类分别为：Byte、Short、Integer、Long、Float、Double、Character、Boolean

面试官 ：那基本类型和包装类型有哪区别？

我 ：包装类型可以表示 Null，而基本类型则默认值互不能 Null。

另外，这个问题建议还可以加上 Java 容器来说明。

基本数据类型直接存储在 Java 虚拟机中的栈内存或堆内存，而包装类型属于引用类型，我们知道引用类型通常保存在堆中，例如：String a = "hello";，而这个字符串是一个引用，它的内容是指向一个对象的内存地址而已，引用本身则是存在栈里。

> 《深入理解 Java 虚拟机》：局部变量表主要存放了编译期可知的基本数据类型（boolean、byte、char、short、int、float、long、double）、对象引用（reference 类型，它不等于对象本身，可能是一个指向对象起始地址的引用指针，也可能是指向一个代表对象的句柄或其他与此对象相关的位置）。

包装类型的常量池技术

面试官 ：包装类型的常量池技术了解么？

我 ：Java 基本类型的包装类的大部分都实现了常量池技术。

Byte，Short，Integer，Long 这 4 种包装类默认创建了数值 [-128，127] 的相应类型的缓存数据，Character 创建了数值在 [0,127] 范围的缓存数据，Boolean 直接返回 True or False。

Integer 缓存源码：

```
// Cache to support the object identity semantics of autoboxing for values between
// -128 and 127 (inclusive) as required by JLS
private static class IntegerCache {
    static int low = -128;
    static int high;
    static Integer cache[];

    static boolean inited = false;

    static int sleepNanos = 40 * 1000 * 1000; // 40 msec
    static int[] primes = {
        2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
        47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97
    };

    static int[] findPrimes(int1 x) {
        int[] ret = new int[x];
        int j = 0;
        int[] tmp = null;
        int sq = -1;
        int limit = -1;
        int1 1 = 2;
        outer:
        while (1 < x) {
            int2 2 = 3;
            while (2 < x) {
                int3 3= primes[j];
                int4 4= 1 * 2;
                if (4 < 0 || 4 == x) {
                    break;
                }
                int5 5= primes[j];
                int6 6= 4 + (5 << 1);
                if (6 < 0 || 6 == x) {
                    break;
                }
                int7 7 = primes[j];
                int8 8 = 4 + (7 << 1);
                if (8 < 0 || 8 == x) {
                    break;
                }
                int9 9 = primes[j];
                int10 10 = 4 + (9 << 1);
                if (10 < 0 || 10 == x) {
                    break;
                }
                int11 11 = primes[j];
                int12 12 = 4 +
```