《JAVA程序设计语言》面试重点总结

春招来临，想着复习一下Java基础内容，便拿着学校发给我们的《Java语言程序设计》(第2版）清华大学出版社这本书看了一遍，自己总结了里面可能春招会考到的技术点，希望对大家有帮助。

该笔记的pdf地址：https://github.com/Java567/InterviewSummary

一、JAVA语言简述

1.1 面向对象程序设计的特性

- 封装性：把数据和对数据的操作封装在一个类里面，其中对象是类的实例化对象。
- 继承性：是指A类通过继承B类，从而可以调用B类的属性和方法。A类可以称为子类（派生类），B类可以称为父类（超类、基类）
- 多态性：实现多态的方式可以分为重写（又称覆盖）、重载这两种。

  重写（覆盖）：是指方法名相同、参数列表相同、方法返回值相同，只是里面实现的方法不同。

  不在同一个类中、在继承或者实现的类中。且子类方法的访问修饰符要大于父类

  的修饰符。

  重载：是指方法名相同，但参数不同，可以在同一个类中也可以在继承的类中。

- 抽象性：一体现在子类和父类中，父类是子类的抽象表述。二体现在类和对象中，类是一个抽象的，而对象是具体的。

二、Java数据类型与表达式

2.1 Java符号

2.2.1 标识符

> 标识符是以字母、汉字、数字、$、下划线(_)组成。
>
> 定义标识符需要在意的点：
>
> - Java的关键字不能作为标识符，如if、int...
> - Java是一个大小写敏感的语言。
> - 要见名知意

2.2 数据结构与变量

2.2.1数据类型

Java数据类型可分为基本类型（简单数据类型）和引用类型（复合数据类型）。

> 基本类型（简单数据类型）：
>
> - 整数类型
> - 浮点类型
> - 字符类型

> - 布尔类型

> 引用类型（复合数据类型）：
>
> - 类（class）
> - 接口（interface）
> - 数组

关键字	数据类型	所占字节	默认值
byte	字节型	1	0
short	短整形	2	0
int	整形	4	0
long	长整型	8	0
float	单精度浮点型	4	0.0F
double	双精度浮点型	8	0.0D
char	字符型	2	0
boolean	布尔型	1	false

2.2.2 常量

Java的字符编码采用了Unicode码，一个字符用16位无符号数据表示。'A'的编码为65。

2.2.3 变量(Java中变量必须先声明后使用)

> 1、变量的定义与赋值：
>
> 类型 变量名 = 值;
>
> 2、强制类型转化
>
> - 系统自动转换：数据表示范围小的“短数据类型”转化为数据表示范围大的“长数据类型”（小-->大）
> - 强制转换：（大-->小）
>
> 3、自动数据类型自动转换的递增顺序：
>
> - byte->short->char->int->long->float->double

2.3 表达式与运算符

2.3.1 位运算符

位运算符：是对操作数以二进制比特（bit）位为单位进行的操作运算。

运算符	用法	操作
&	a&b	a和b都是true，结果才为true
|	a|b	a和b有一个为true，结果就为true
^	a^b	a和b是不同值，结果就为true

三、数组和方法

4.1 数组

4.1.1 一维数组

> 1、声明数组
>
> - 数组元素类型 数组名[];　　例如：int sum[];
> - 数组元素类型[] 数组名;　　例如：int[] sum;
>
> 2、创建数组空间
>
> - 数组名 = new 数组元素类型[数组];　　例如(已经声明了数组)：sum = new int[100];
> - 类型 数组名[] = {初值表};　　例如：int sum[] = {1,2,3,4,5}

利用Arrays类中的 toString()方法可以将数组转换为串的形式。

利用Arrays类中的 sort()方法可以将数组排序。

4.1.2 多维数组

> 声明数组跟一维差不多，例如 int a[][; int[[[a; int a[[= new int[2[3;
>
> - 获取数组名的行数：数组名.length
> - 获取数组名的列数：数组名[].length

4.2 方法

4.2.1 参数传递

> 在Java中，参数传递是以传值的方式进行的，将实参的存储值传递给形参。
>
> - 对于基本类型，在方法内对形参数据进行如何修改，实参都不会变。
> - 对于引用类型（如对象、数组），在方法内对形参数据进行如何修改，实参还改变。

四、类与对象

4.1 类的定义

```
[修饰符] class 类名 [extends 父类名] [implements 接口列表]{

}
```

- 修饰符有访问控制修饰（如public）和类型修饰（如abstract、final）

4.2 对象的创建与引用

4.2.1 对象的初始化和构造方法

> 在创建对象时，要给对象的属性方法成员分配内存空间，并进行初始化。如果没有设定初值，则系统默认按照规则设定初值。
>
> - 特别的，引用类型的成员变量的默认初值为null

> 构造方法是给对象设置初值的规范方式，构造方法是根据方法参数给对象属性赋不同的初值。

> - 如果类未指定构造方法，会默认自动提供一个构造方法。
>
> 构造方法：1-方法名与类名同名、2-没有返回类型、3-一个类可提供多个构造方法

4.3 类变量和静态方法

4.3.1 类变量

用static 修饰符修饰的属性->静态属性，相应的成员变量称为类变量或者静态变量。

4.3.1.1类变量的访问形式

- 通过类名做前缀来访问的
- 通过对象做前缀来访问的
- 甚至通过变量名直接访问（要有条件的，static）

静态变量在存储上归属类空间，不依赖于任何对象，通过对象去访问静态变量，实质上还是访问类空间的那个变量。

4.3.2 静态方法

用static修饰符的方法称为静态方法（类方法）。

4.3.2.1静态方法的访问形式

- 类名做前缀
- 通过对象来调用
- 甚至通过方法名直接访问（要有条件的，static）

4.4 理解this

> this 出现在类的实例方法或构造方法中，用来代表使用该方法的对象。用this作前缀可访问当前对象的实例变量或成员方法。具体形式为：
>
> - this.实例变量
> - this.成员方法(参数)
>
> this的用途：
>
> - 在实例变量和局部变量名称相同时，用this作前缀来特指访问实例变量
> - 把当前对象的引用作为参数传递给另一个方法
> - 在一个构造方法中调用同类的另一个构造方法，形式为this(参数)。但要注意，用this调用构造方法，必须是方法体中的第一个语句。

4.5 使用包组织类

> 三类：
>
> - java.*：核心包，有applt、awt、beans、io、lang、math、net、sql、text、util
> - javax.*：扩展包，有swings、security、rmi
> - org.*：组织扩展包，主要用于CORBA和XML处理等。

五、继承和多态

5.1 继承

5.1.1 Java继承的实现

子类对象除了可以访问子类中直接定义的成员外，也可访问父类的所有非私有成员。

5.1.2 构造方法在类继承中的作用

> 构造方法不能继承。由于子类对象要对继承自父类的成员进行初始化，因此在创建子类对象时除了执行子类的构造方法外，还需要调用父类的构造方法。
>
> - 子类在自己的构造方法中使用关键字super来调用父类的构造方法，但super必须是子类构造方法中第一个可执行语句。
> - 子类在自己定义的构造方法中如果没有用super明确调用父类的构造方法，则在创建对象时，首先自动执行父类的无参构造方法，然后再执行自己定义的构造方法。

5.2 几个特殊类

5.2.1 Object类

Object类是所有Java类的最终祖先，如果类的定义时不包括关键字extends，则编译将自动认为该类直接继承Oject类。

5.2.2 Class类

Java运行环境中提供了反射机制，这种机制允许在程序中动态的获取类的信息以及动态的调用对象的方法。

5.3 访问控制修饰符

5.3.1 公共访问控制符(public)

5.3.2 默认访问控制符

5.3.3 私有访问控制符(private)

private用来声明类的私有成员，他提供了最高的保护级别。用private修饰的域或方法只能被该类自身所访问和修改，而不能在任何其他类（包括该类的子类）中访问。

5.3.4 保护访问控制符

控制等级	同一类中	同一包中	不同包的子类中	其他
private	可直接访问			
默认	可直接访问	可直接访问		
protected	可直接访问	可直接访问	可直接访问	
public	可直接访问	可直接访问	可直接访问	可直接访问

5.4 final修饰符的使用

5.4.1 final作为类修饰符

> 被final修饰符所修饰的类称为最终类。最终类的特点是不允许继承。

5.4.2 用final修饰方法

> 用final修饰符的方法，是功能和内部语句不能被更改的最终方法，在子类中不能再对父类的final方法重新定义。

> 以下是默认是final的：
>
> - 所有已被private修饰符限定为私有的方法
> - 所有包含在final类的方法

六、常用数据类型处理类

6.1 字符串的处理

6.1.1 String类

字符串常量也是以对象形式存储的，Java编译时将自动对每个字符串常量创建一个对象，因此，当将字符串常量传递给构造方法时，将自动将常量对应的对象传递给方法参数。

> 方法：
>
> - toCharArray()：可得到字符串对应的字符数组
> - concat(String str)：用于字符串的拼接
> - compareTo(String str)：当前串大，则返回值>0
>
> 当前串小，则返回值<0
>
> 两串相等，则返回值=0S

6.1.2 StringBuffer类

> String类不能改变串对象中的内容。
>
> StringBuffer类可实现字符串内容的添加、删除和修改。

StringBuffer类没有直接定义equals()方法，所以它将继承Object类的equal()方法

6.2 日期和时间

6.2.1 Date类

> 在Java中日期用代表毫微数的一个长整数进行存储表示
>
> 日期的构造方法如下：
>
> - Date()：创建一个代表当前时间的日期对象。
> - Date(long date)：根据毫微数创建日期对象。
>
> 其他方法：
>
> - toString()：按星期、月、日、小时、分、秒、年的默认顺序输出
> - int compareTo(Date antherDate)：将当前日期与某个日期比较
> - getTime()：得到日期对象对应的毫微秒值

6.2.2 Calender类

> Calender类主要用于与年、月、日等整数值的转换，Calender是一个抽象类，不能直接创建对象，但可以使用静态方法getInstance()获得当前日期的日历对象。
>
> - Calender ringtNow = Calender.getInstance();
>
> 从日历获取有关的年份、月份、星期、小时等信息，可以通过 int get(int field)方法得到。其中field值是由Calender类的静态常量决定
>
> - YEAR：年
> - MONTH：月

> - DAY_OF_WEEK：星期几
> - HOUR：小时
> - MINUTE：分
> - SECOND：秒
>
> 其他方法：
>
> - long getTimeInMillis()：返回当前日历对应的毫微秒值。
> - Date getTime()：得到当前日历对应的日期对象

七、抽象类和接口

7.1 抽象类和抽象方法

7.1.1 抽象类的定义

> 抽象类代表着一种优化了的概念组织方式。抽象类用来描述事物的一般状态和行为，然后在其子类中再去实现这些状态和行为，以适应对象的多样性。

```
abstract class 类名称{
    成员常量定义;
    方法(){};  //定义具体方法
    astract 方法();     //定义抽象方法
}
```

注意：抽象类表示的是一个抽象概念，不能被实例化为对象。

问：abstract 和 final 修饰符可以同时修饰一个类吗?

答：不行，final的类不能被重写和继承，而abstract的类是抽象类，本身没有实现，只能通过子类来实现，也就是必须被继承，所以说它们是没法同时做修饰符的。

7.2 接口

7.2.1 接口定义

> 接口由常量和抽象方法组成，由关键字interface引导接口

```
[public] interface 接口名[extneds 父接口名]{
    [public] [static] [final] 域类型 域名 = 常量值;
    [public] [abstract] 返回值 方法名(参数列表)[throw 异常列表];
}
```

> 注意：
>
> - 声明接口可给出控制符，用public修饰的是公共接口。
> - 接口具有继承性，一个接口还可以继承多个父接口，父接口间用逗号隔开。
> - 接口中所有属性的修饰默认是public static final，也就是均为静态变量
> - 接口中所有方法的修饰默认是public abstract

7.2.2 接口的实现

- 一个类可以实现多个接口。

7.3 内嵌类

内嵌类是指嵌套在一个类中的类，因此有时也称为嵌套类或内部类。

内嵌类：一般是指用来实现一些没有通用意义的功能逻辑。

> 静态内嵌类：
>
> - 它可不需要通过外层类的对象来访问

系统定义的异常	异常的解释
ClassNotFoundExecption	未找到要装载的类
ArrayIndexOutOfBoundExecption	数组访问越界
FileNotFoundException	文件找不到
IOException	输入、输出错误
NullPointerException	空指针访问
ArithemeticException	算术运算错误，如除数为0
NumberFormatException	数字格式错误，如除数为0
InterruptedException	中断异常，线程在进行暂时处理时（如睡眠）被调度打断将引发该异常