一、Java中通用的方法和准则

1、不要在常量和变量中出现易混淆的字母

2、莫让常量蜕变成变量

3、三元操作符的类型务必一致

4、避免带有变长参数的方法重载

5、别让null值和空值威胁到变长方法

6、覆写变长方法也循规蹈矩

7、警惕自增的陷阱

8、不要让旧语法困扰你（goto的使用）

9、少用静态导入

10、不要在本类中覆盖静态导入的变量和方法

11、养成良好习惯，显示声明UID（使用显示序列号UID）

12、避免用序列化类在构造函数中为不变量赋值

13、避免为final变量复杂赋值（序列化的对象的final变量和反序列化变量的final变量有修改）

反序列化final变量在以下情况不会被重新赋值：

·通过构造函数为final变量赋值

·通过方法返回值为final变量赋值

·final修饰的属性不是基本类型

若是简单赋值，会对final变量进行重新赋值。

14、使用序列化类的私有方法巧妙解决部分属性持久化问题（没懂）

15、break万万不可忘

16、易变业务使用脚本编写（用Java代码调用JS代码，可以做数据爬虫）

17、慎用静态编译

18、避免instanceof非预期结果

19、断言绝非鸡肋

20、不要只替换一个类（常量类修改后未编译到其他类中）

二、基本类型

21、用偶判断，不用奇判读（如果是负数取余可能是负数，同时也不等于0）

22、用整数类型处理货币（浮点型可能会出现不准确的情况，使用BigDecimal或使用整型）

23、不要让类型默默转换（基本类型转换时，使用主动声明方式）

24、边界，计算和判断整型时需要判断边界范围

25、不要让四舍五入亏了一方（使用银行家舍入：四舍六入五考虑，五后非零就进一，五后为零看奇偶，五前为偶应舍去，五前为奇要进一。）

26、提防包装类型的null值（包装类型拆箱时需要null值校验）

27、谨慎使用包装类型的大小比较（基本类型直接比较大小，包装类型比较引用地址）

28、优先使用整型池（-128~127整型数直接从整型池中取出来）

29、优先选择基本类型（基本类型优先考虑）

30、不要随便设置随机种子（种子不同，长生不同随机数；种子相同，即使实例不同也产生相同的随机数。）

三、类、对象及方法

31、接口中不能存在实现代码

32、静态变量一定要先声明后赋值

33、不要覆写静态方法（静态方法不能覆写，但是可以隐藏）

34、构造函数尽量简化

35、避免在构造函数中初始化其他类

36、使用构造代码块精炼程序（编译器会把构造代码块插入到每个构造函数的最前端）

37、构造函数调用自身其他的构造函数时不重复插入构造代码块

38、使用静态内部类提高封装性

39、使用匿名类的构造函数（构造代码块就是匿名类的构造函数）

40、匿名类的构造函数很特殊（先调用父类的同参构造，然后在调用自己的构造代码块）

41、让多重继承成为现实（通过匿名内部类或者局部内部类继承其他父类来实现多继承）

42、让工具类不可实例化（私有化构造方法，并且final修饰类，保证没有子类能够实例化）

43、避免对象的浅拷贝（对象的属性拷贝规则：基本类型直接拷贝其值，对象拷贝地址应用，String属于特殊情况，默认其为基本类型）

44、推荐使用序列化实现对象的拷贝

45、覆写equals方法时不要识别不出自己

46、equals应考虑null值情景

47、在equals中使用getClass进行类型判断

48、覆写equals方法必须覆写hashCode方法

49、推荐覆写toString方法

50、使用package-info类为包服务（有三个好处）

51、不要主动进行垃圾回收（千万不要）

四、字符串

52、推荐使用String直接量赋值（String字符串与其他对象有区别）

53、注意方法中传递的参数要求（replaceAll方法第一个参数是正则表达式）

54、正确使用String,StringBuffer,StringBuilder（父类CharSequence）

55、注意字符串的位置（在‘+’表达式中，String字符串具有最高优先级）

56、自由选择字符串拼接方法（‘+’方法拼接字符串；concat方法拼接字符串;append方法拼接字符串）

57、推荐在复杂字符串操作中使用正则表达式（正则表达式是恶魔，威力巨大，但难以控制）

58、强烈建议使用UTF编码（UTF-8编码）

59、对字符串排序持一种宽容的心态（如果排序不是一个关键算法，使用Collator类即可）

五、数组和集合

60、性能考虑，数组是首选（对基本类型进行求和计算时，数组的效率是集合的10倍；性能要求较高的场景中使用数组替代集合）

61、若有必要，使用变长数组（可以使用Arrays.copyOf(datas,newLen)方法增加数组长度）

62、警惕数组的浅拷贝（copyOf方法是浅拷贝，基本类型拷贝其值，对象类型拷贝其引用地址）

63、在明确的场景下，为集合指定初始容量（ArrayList默认初始大小为10，每当长度不够时，扩展到1.5倍长度；指定初始容量，提高性能）

64、多种最值算法，适时选择（最值计算时使用集合最简单，使用数组性能最优）

65、避开基本类型数组转换列表陷阱（基本类型不能作为asList的输入参数，否则会引起程序逻辑混乱）

66、asList方法产生的List对象不可更改（返回的ArrayList只实现了size,toArray,get,set,contains方法，重新new ArrayList，asList返回对象作为构造函数入参）

67、不同的列表选择不同的遍历方法（ArrayList无序集合，采用普通for循环遍历效率较高，LinkedList是有序集合，采用增强for循环（foreach，底层采用迭代器）遍历效率更高；有关联的集合使用迭代器遍历效率高）

68、频繁插入和删除时使用LinkedList(LinkedList的插入效率比ArrayList高50倍以上；LinkedList的删除效率比ArrayList高40倍以上；ArrayList效率比LinkedList高；在最后的位置新增元素，效率差别不大。)

69、列表相等只需关心元素数据（在保证元素顺序的情况下，判断集合是否相等只需关注元素是否相等即可。）

70、子列表只是原列表的视图（subList产生的列表只是一个视图，所有的修改动作直接作用于原列表。）

71、推荐使用subList处理局部列表（list.subList(20,30).clear();）

72、生成子列表后不要在操作原列表（生成subList子列表后，保持原列表的只读状态）

73、使用Comparator进行排序（Comparable接口可以作为实现类的默认排序法，Comparator接口则是一个类扩展排序工具）  
74、不推荐使用binarySearch对列表进行搜索（从性能方面考虑，binarySearch是最好的选择）

75、集合中的元素必须要做到compareTo和equals同步（实现了compareTo方法，就应该覆写equals方法，确保两者同步）

76、集合运算是使用更优雅的方式（集合的并集、交集、差集）

77、使用shuffle打乱集合

78、减少HashMap中元素的数量（尽量让HashMap中的元素少量并简单，默认加载因子是0.75，达到总长度的0.75就会自动扩容）

79、集合中的哈希码不要重复（不太懂）

80、多线程使用Vector或HashTable（CopyOnWriteArrayList和ConcurrentHashMap的效果相同，效率更高）

81、非稳定排序推荐使用List（SortedSet中的元素被修改后可能会影响到其排序位置）

82、由点及面，一叶知秋—集合大家族（commons-collentions、google-collections是JDK之外的优秀数据集合工具包，使用拿来主义即可）

六、枚举和注解

83、推荐使用枚举定义常量（推荐使用枚举常量代替接口常量或类常量）

84、使用构造函数协助描述枚举项

85、小心switch带来的空值异常（优先判断输入参数是否为null）

86、在switch的default代码块中增加AssertionError错误

87、枚举类使用valueOf前必须进行校验

88、用枚举实现工厂方法模式更简洁（1、枚举非静态方法实现工厂方法模式2、通过抽象方法生成产品）

避免错误调用发生；性能好，使用便捷；降低类间耦合

89、枚举项的数量限制在64个以内（枚举项数量不要超过64，否则建议拆分）

90、