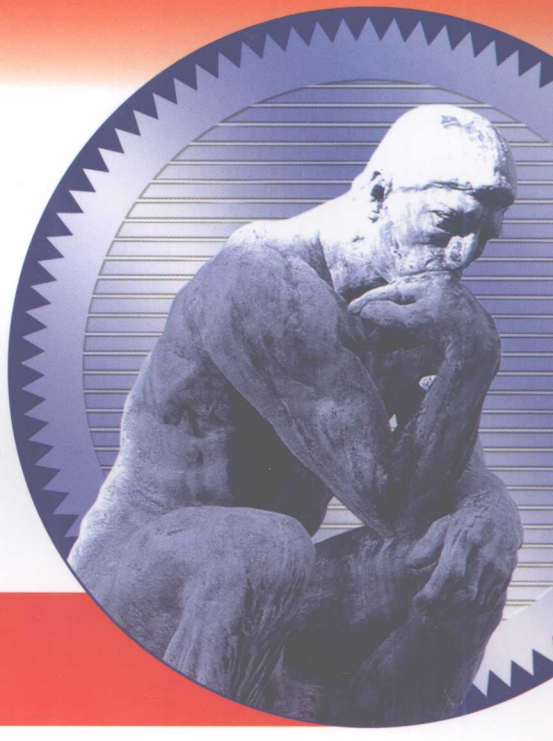
Beijing Hope Electronic Press
希望
www.bhp.com.cn

（SCJP Sun Certified Java5 Programmer）
Java5（Java V1.5）考试号310-055

Java5
国际认证

SCJP
试题精解



北京希望电子出版社　总策划
施　铮　编　著

- 完全承袭与涵盖SCJP 5.0（考试号310–055）
- 精辟解析直接命中Java 5.0的要害
- 750道超仿真模拟试题

Sun Certified 100%

科学出版社
www.sciencep.com

希望 Hope Electronic Press www.bhp.com.cn

(SCJP, Sun Certified Java5 Programmer)
Java5（Java v1.5）考试号310-055

Java5 国际认证

SCJP 试题精解

北京希望电子出版社 总策划
施 铮 编 著

- 完全承袭与涵盖SCJP 5.0（考试号310-055）
- 精辟解析直接命中Java 5.0的要害
- 750道超仿真模拟试题

Sun Certified 100%

科学出版社
www.sciencep.com

内 容 简 介

这是一本 SUN 国际认证（SCJP, Sun Certified Java5 Programmer）——Java5（Java v1.5）典型试题精解图书，针对 SCJP 应试的所有知识点，选取典型案例进行详尽探讨，便于有效地提高考生应试成功率。

全书共分 9 章，分别介绍了 Java 语言基础、Java 运算符、修饰符、类型转换及造型、程序流程控制、Java 面向对象编程、线程控制、输入/输出流以及部分 Java 基础包。内容详尽，选例精彩，是广大 SCJP 应试者必备教材。

本书既可作为 SCJP 学习练习，亦可配合 SCJP 试题使用，同时还可以作为学习 Java 语言和程序设计参考书。

图书在版编目（CIP）数据

Java5 国际认证（SCJP）试题精解 / 施铮编著. —北京：科学出版社，2007.

Java 开发专家

ISBN 978-7-03-018520-4

Ⅰ. J… Ⅱ. 施… Ⅲ. Java 语言—程序设计—工程技术人员—资格考试—试题 Ⅳ. TP312－44

中国版本图书馆 CIP 数据核字（2007）第 051424 号

责任编辑：但明天 / 责任校对：张视明

责任印刷：双 青 / 封面设计：刘孝琼

科学出版社 出版

北京东黄城根北街16号

邮政编码:100717

http://www.sciencep.com

双青印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2007 年 6 月第 一 版 开本：787×1092 1/16

2007 年 6 月第一次印刷 印张：37 3/5

印数：1—4000 字数：856 595

定价：52.00 元

前 言

技术是讲究实力的。近年来，认证在就业市场上已获得肯定与追捧。根据知名就业网站的调查，与信息技术相关的工作机会中竟有62%是以提出认证作为录用参考，其中又以需要Java认证资格者居多。这几年来，Java大红大紫不仅被微软视为劲敌，而各大企业竞相争取掌握Java技术的人才，也使Java专业认证成为IT人员最想取得的一张证书。特别是随着互联网全面走向应用的今天，全球已经掀起一股学习Java语言开发技术的热潮，而且已成为一条非专业人员变成编程高手的快车道。Java语言自然是转行IT的入门首选。

Sun Java认证是业界惟一经Sun授权的Java国际认证（SCJP, Sun Certified Java Programmer）。SCJP认证是Sun诸多认证中首要的Java程序设计认证，它在全球范围内得到了赞许和业界认可。认证杂志的Ed Tittel说过：Sun公司最为人所知的认证就是Java 2平台的Sun认证程序员考试，不仅仅因为它是其他多数Java认证的先决条件，而且因为其他的认证主办者也认可或需要这项认证。

SCJP由Sun公司出题，考试合格者将获得由美国Sun公司直接签发的英文SCJP证书，该证书可作为各企事业单位Java程序设计、开发、管理人员上岗的依据。目前拥有SCJP认证的人已成为各IT公司及相关企业争相竞聘的高层次Java开发热门人才。并且，获得此项认证将成为获取Sun公司其他高级Java技术认证的先决条件。

SCJP考试是Sun公司Java技术认证考试中的基础性认证。通过此项认证即能清楚表明此程序员理解了Java程序设计语言的基本语法和结构，并能使用J2SE 5.0创建服务器和桌面系统上均能运行的Java技术应用程序。

SCJP认证考试内容涉及所有Java相关知识、编程概念及开发技巧。SCJP考试被认为是IT业内最难的考试之一。根据以往经验，可以断言，很多没有过关的考生并没有做好考前准备。这里所指的考生是那些非常熟悉Java并且有着多年Java程序设计经验的人！作为程序员，我们倾向于仅仅掌握、完成当前项目所需的知识，但SCJP考试却要求你全面掌握Java语言，不仅仅是你在工作中已熟悉的那部分内容。因此，仅凭经验很难通过考试，因为即使你以为自己所了解的内容也可能与你想象的有一定出入。光凭经验无法使你的代码正确地运行。对有Java编程资深的人尚且如此，更不用说初学者和新手了。一定要深入领会核心基础知识，全面掌握在使用该语言过程中可能出现的任何内容。只有通过大量实际的有针对性的训练，才能获取该难能可贵的证书。

目前SCJP认证的相关图书多限于基本概念的讲述，而讲解、分析应试试题的图书却不多。并且，更没有和Sun公司最新推出对应Java5中文版SCJP5认证考试配套的，这和国内对SCJP认证需求的飞速发展趋势十分相背。本书正是为了解决上述问题而推出的。

本书完全承袭与涵盖了Sun公司SCJP 5.0（对应考试号310-055）的考试内容及范围，除了具有完整而且精辟的试题与解析直接命中Java 5.0的要害外，更多地包含了大量超仿真模拟试题，让你在考场上犹如做本书的模拟试题一样自在，并以一种轻松愉快的心情来取得SCJP 5.0认证，提高应试成功率，这正是本书的目的。

本书将SCJP5认证的大量知识精练为一个个有针对性、指导性的规则，便于记忆和考前

准备。并通过对大量具有针对性的模拟试题的分析和讲解，在加强对理论的理解和记忆的同时，提高读者的应试技巧和实际解题能力。题目精解部分依据认证考试大纲内容进行安排，便于读者有计划、有步骤地进行学习。每章的示例均按照由易至难的思路编排，每个示例对每一个选项都给予详尽的解释和分析，指出其错误迷惑之处。本书文字叙述浅显易懂，循序渐进，即使是没有 Java 经验的新手，通过本书的强化集中练习，也可以较快地掌握 SCJP 认证的内容，为顺利过关作好准备。

考试目标

第 1 节：生命、初始化与范围

编写代码，声明类（包括抽象类和各种形式的嵌套类）、接口和枚举，正确使用 package 和 import 语句，包括 including static imports。

编写代码，声明接口。编写代码，实现或扩展一或多个接口。编写代码，声明抽象类。开发代码，扩展抽象类。

开发代码，用静态、实例和局部变量方式声明、初始化并使用基本类型、数组和对象。并使用正确的变量名标识符。

编写代码，声明静态和非静态方法，如果可以的话，用符合 JavaBean 命名标准的方法名字。并编写代码，声明和使用变量长度参数列表。

给定一个代码例子，确定某个方法是否正确地重载了另外一个方法，并识别正确的方法返回值，包括共变返回。

给定一组类和父类，为一个或多个类编写构造函数。给定一个类声明，确定是否要创建一个默认的构造函数，如果是这样，确定构造函数的行为。给定一个嵌套的或非嵌套的类列表，编写代码将类实例化。

第 2 节：流控制

编写代码，实现 if 或 switch 语句，认识这些语句的正确的参数类型。

编写代码，实现所有各种形式的循环和迭代，包括使用 for、增强循环(for-each), do, while, labels, break 和 continue；并讲解在循环执行过程中和循环执行完成后，循环变量的取值情况。

编写代码，使用断言，区分断言的正确和错误使用。

编写代码，使用异常和异常处理语句(try, catch, finally)，声明有抛出异常的方法和重载方法。

了解在代码段中某个点抛出异常的结果。注意异常可能是运行时异常、检查异常或错误。

了解可能导致下列异常发生的情况。ArrayIndexOutOfBoundsException,ClassCastException, IllegalArgumentException, IllegalStateException, NullPointerException, NumberFormatException, AssertionError, ExceptionInInitializerError, StackOverflowError 或 NoClassDefFoundError。理解虚拟机会抛出哪个异常，了解程序抛出这些异常的情形。

第 3 节：API 内容

编写代码，使用基本封包类（如 Boolean, Character, Double, Integer 等），和自动装箱、拆箱。了解 String, StringBuilder 和 StringBuffer 等几个类之间的区别。

给定一个情形，涉及到浏览文件系统、读写文件，编写正确的解决方案，使用到 java.io 中的类（有时候要组合使用）：BufferedReader,BufferedWriter, File, FileReader, FileWriter 和 PrintWriter。

编写代码实现对象的序列化和反序列化，使用到 java.io 中的下列 API：DataInputStream, DataOutputStream, FileInputStream, FileOutputStream, ObjectInputStream, ObjectOutputStream 和 Serializable。

使用 java.text 包中的标准 J2SE API，正确地对某个地区中的日期、数字和货币值进行格式化或解析；并且给定一个情景，如果要用默认地区或某种特定的地区，确定适当的所需使用的方法。了解 java.util.Locale 类的目的和使用方式。

编写代码，使用 java.util.regex 包中的标准 J2SE API，对字符串或流进行格式化或解析。为字符串编写代码，使用 Pattern 和 Matcher 类以及 String.split 方法。认识和使用正则表达式的匹配（限于(dot), * (star), + (plus), ?, \d, \s, \w, [], ()）。*, +和?的使用局限于贪婪方法，括号只用于组机制，而不是匹配过程中的捕获内容。为流编写代码，使用 Formatter 和 Scanner 类以及 PrintWriter.format/printf 方法。认识和使用格式化字符串中的格式化参数（限于%b, %c, %d, %f, %s）。

第 4 节：并发

编写代码，用 java.lang.Thread 和 java.lang.Ruannable 定义、实例化和启动新线程。

认识线程能退出的状态，了解线程状态从一个转到另一个的途径。

给定一个情景，编写代码，适当使用对象锁定方式，保护静态或实例变量不发生并发访问问题。

给定一个情景，编写代码，正确使用 wait, notify 或 notifyAll。

第 5 节：面向对象概念

编写代码实现紧封装、松耦合和类间高聚合，讲解其优点。

给定一个情景，编写代码，展示多态的使用。此外，确定何时有必要使用类型转换，并了解与对象引用类型转换有关的编译器 和运行时错误。

讲解修饰符在构造函数、实例或静态变量，以及实例或静态方法方面对继承机制的影响。

给定一个情景，编写代码声明或引发重载方法，并编写代码声明或引发父类、重载运算符。

编写代码，实现 is-a 和 has-a 关系。

第 6 节：集合／泛化

给定一个设计情景，确定要使用哪个集合类或接口，能适当地实现这一设计目的，包括使用 Comparable 接口。

区分 hashCode 和 equals 方法正确和错误的重载方式，并讲解==和 equal 方法之间的区别。

编写代码，使用 Collections API 的泛化版，特别是 Set, List 和 Map 接口以及实现类。了解非泛化 Collections API 的局限性，以及如何将代码重构以使用泛化版本。

编写代码，正确地在类或接口声明、实例变量、方法参数和返回类型中使用类型参数。编写泛化方法或者使用了 wildcard 类型的方法，理解这两种方法之间的相似与不同。

使用 java.util 包中的功能编写代码，对列表进行排序操作、二进制搜索或者将列表转换为数组。使用 java.util 包中的功能编写代码，对数组进行排序操作、二进制搜索或者将数组

转换为列表使用 java.util.Comparator 和 java.lang.Comparable 接口影响列表和数组的排序。此外，认识基本封包类“自然排序”的作用，以及 java.lang.String 的排序。

第 7 节：基础知识

给定一段示例代码和情景，编写代码，使用正确的访问修饰符、包声明和 import 语句，与示例代码进行交互（通过访问或继承）。

给定一个示例类和一个命令，确定期望的运行时行为。

如果将对象引用和基本数据值传给方法，在方法之中对参数执行赋值或其它修改操作，确定其结果。

给定一个示例类，了解在什么时候某个对象会进入垃圾收集，确定垃圾收集系统能确保什么、不能确保什么。了解 System.gc 和结束的行为。

给定 JAR 文件内或外部署的完全符合条件的类名，为这个类创建适当的目录结构。给定代码示例和类路径，确定这个类路径是否能够让代码成功地编译。

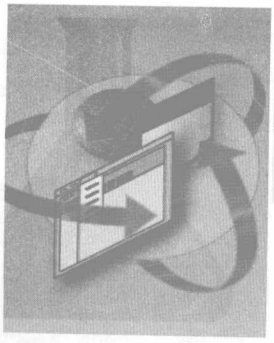
编写代码，正确使用各种运算符，包括赋值运算符（限于=, +=, -=）、算术运算符（限于+, -, *, /, %, ++, --）、关系运算符（限于<, <=, >, >=, ==, !=）、instanceof 运算符、逻辑运算符（限于&, |, ^, !, &&, ||）以及条件运算符?:），以产生期望的结果。编写代码确定两个对象或两个基本类型变量是否相等。

本书由具有丰富的 SCJP 考试经验的专家编写，是 SCJP 应试人员的必备教材，同时也是一本 Java5 的优秀参考书，可作为 Java 开发人员的速查手册。既可作为 Java5 的辅导材料和备考教材，也可用于培训学校教材。

由于作者水平有限，时间紧任务重，难免存在不妥之处，敬请读者指正。作者联系电子邮箱 xuanxuan_boys@126.com。

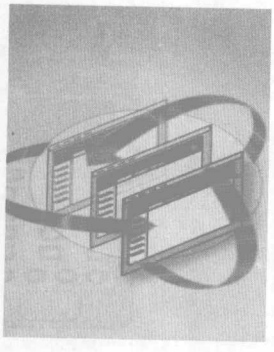
施　铮

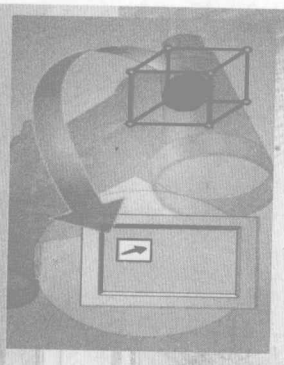
目　录

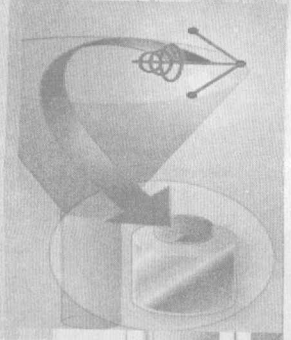


Chapter 1

语言基础







Chapter 1

语言基础

1.1 Java 源文件

1.2 关键字和标识符

1.3 应用程序入口

1.4 编译、运行程序

1.5 基本数据类型

1.6 文字值

1.7 数组

1.8 可变参数方法

1.9 变量初始化

1.10 参数传递

1.11 垃圾回收器

1.12 异常解析

1.1　Java 源文件

规则 01： 源文件是一个包含 Java 代码的文本文件，以.java 作为其文件后缀名。源文件通过编译产生可执行的字节代码文件，即类文件，类文件以.class 作为文件后缀名。该源文件中的字符必须满足 Java 语言的语法规定，否则在编译时会产生编译错误。

规则 02： 源文件名必须满足 Java 语言标识符的规定。

规则 03： 一个源文件可以不包含任何代码定义，即它是一个空白文件，编译时自然不会产生编译错误。

规则 04： 一个源文件最多只能包含一个顶层（相对于内部类而言）的 public 类定义。如果在一个源文件中存在两个或两个以上的顶层 public 类定义，则会产生编译错误。但可以允许没有顶层的 public 类定义，并且一个源文件可以包含不限数量的多个非 public 顶层类定义。

规则 05： 在一个源文件中，如果存在一个顶层的 public 类定义，则该源文件名必须同定义的顶层 public 类名一致。如果源文件中没有一个顶层的 public 类定义，则该源文件名可以随意命名，不需要与源文件中定义的任何类名一致，只要符合 Java 语言标识符规定即可。

规则 06： 一个源文件共有 3 种顶层元素，分别为 package（包声明）、import（输入语句）和 class（类声明）。这 3 种顶层元素并不是必须同时出现，如果同时存在，则必须按 package、import、class 的顺序出现，即任何 import 出现在所有类定义之前。如果使用 package，则包声明必须出现在类和输入语句之前。

规则 07： 包是用来组织类的。在源文件中，包声明通过关键字 package 后跟包名来实现。包名由一系列由句点作为分隔符的路径名构成，由于包名的构成元素映射的是系统路径名，所以命名包名的字符必须采用满足路径名的要求，例如不能包含/、\、:、*、?等字符。值得注意的是，一个源文件最多只能包含一个包声明语句。如果源文件中没有包声明语句，则类被放置在默认的路径中。

规则 08： import 语句用于引入其他包中的类，以供源文件中的代码使用。既可以引入一个明确指定的类，也可以通过通配符*引入整个包中的所有类。当然，也可以不要 import 语句引用其他包中的类，但在使用类时，需要写全其所在的包名，例如，java.util.Date now = new java.util.Date()，这正是应用 import 语句解决写全包名的烦琐问题所在。要注意，import 语句不同与包声明语句，在一个源文件中，import 语句只能有一条，而包声明邮局则可以有多条，并用分号隔开。

规则 09： 为了简化代码，Java5 提供了静态导入机制。静态导入实现无需在使用其他类的静态成员时前缀其类名，这使得代码更为简洁。要使用静态成员（方法和变量），必须给出提供这个方法的类。使用静态导入，可以使被导入类的所有静态变量和方法在当前类中直接可见，使用这些静态成员也无需再给出它们的类名。静态导入的语法形式必须为 import static，而不能为 static import。

规则 10： 如果在通过通配符*引入的两个不同的包中存在同名的类，例如 java.util 和 java.sql 包中都有一个名为 Date 的类，当代码中不加包名而直接使用 Date 类时，则会产

生编译错误，因为编译器无法确定哪个包中的同名类被使用。解决此问题有两种方式，一是明确地引入要使用包中的类，二是在使用类时写全包名来明确调用。

规则 11：java.lang 包由系统自动引入，不需要明确引入就能使用其中的类。

规则 12：声明一个类的语法为 class classname {}。class 为类声明关键字，classname 为自定义的类名。

规则 13：注释可以出现在 Java 源代码文件中任意一行的开始和结尾处。注释既可以单独占一行，也可以放在一行语句的最后面，并且还可以出现在源文件中 3 种顶层元素 package、import 和 class 之前。

规则 14：注释存在两种形式，一种是单行注释，以//开头；另一种是多行注释，以/*开头，并以*/结尾。

1.2 关键字和标识符

规则 01：Java 语言中共有 53 个关键字和保留字。

abstract	assert				
boolean	break	byte			
case	catch	char	class	const	continue
default	do	double			
else	enum	extends			
false	final	finally	float	for	
goto					
if	implements	import	instanceof	int	interface
long					
native	new	null			
package	private	protected	public		
return					
short	static	strictfp	super	switch	synchronized
this	throw	throws	transient	true	try
void	volatile				
while					

规则 02：goto 和 const 是保留字，尽管在 Java 语言中未被使用，但程序员不能使用其作为标识符使用。

规则 03：Java 语言中规定，标识符不能使用规定的关键字和保留字。

规则 04：标识符必须只能以字母、$或_开头，不能以数字作为标识符的开头。随后的字符只能是字母、$字符、下划线和数字的任意组合。

规则 05：标识符长度没有限制。

规则 06：标识符是区分大小写的。

1.3 应用程序入口

规则 01：main()方法是一个 Java 应用程序的入口。为了创建一个可运行的应用程序，必须

在其类中定义一个 main()方法。main()方法的完整定义语法为 public static void main(String[] args)。

规则 02：main()方法只作为 Java 应用程序的入口。当 Java 虚拟机调用 main()方法时，应用程序被启动。

规则 03：由于数组声明根据方括号的位置不同，存在两种方式，所以 main()方法也相应地存在两种表示语法。分别为 public static void main(String[] args){...}和 public static void main(String args[]){...}。

规则 04：main()方法前的修饰符 public 不是必须的，声明为 public 是为了 main()方法可以从任意一个 Java 运行环境中调用。如果应用程序的主方法 main()没有被 public 修饰，那么尽管不会产生编译错误，但是该应用程序会因为 JVM 找不到其执行入口而无法被运行。static 修饰符是必须的，这样可以在不需要构造类实例的前提下，直接执行应用程序。

规则 05：main()方法中定义了一个一维字符串数组，用来从命令行接收用户的参数。采用命令行方式执行 Java 应用程序的命令行语句，由四部分构成。第一部分为命令名 java；第二部分为命令参数，为可选项；第三部分为应用程序的名称，即源文件中的公共类名；第四部分为用户参数，多个参数之间用空格隔开。从类名后的第一个字符串开始为参数，第一个参数存储在第一个数组元素 args[0]中，第二个参数存储在第二个数组元素 args[1]中，依此类推。

规则 06：main()方法中接收用户参数的数组名称 Java 语言没有限定，可以不为 args，但要符合标识符的命名规定。

规则 07：main()方法中接收用户参数的数组大小，由系统根据用户实际键入的参数个数自动决定。因此，当无用户键入参数时，可以创建一个零长度的数组。为了获得传入参数的数量，可通过数组本身提供的 length 属性来获得。

规则 08：Java 中规定 main()方法没有返回值，因此在 main()方法的返回值声明必须为 void，而不能没有定义或定义为其他类型。

1.4 编译、运行程序

规则 01：采用 JDK 命令行方式编译一个源文件采用 javac 命令。javac 命令的完整语法形式为 javac [options] [source files]。

规则 02：默认情况下，编译器将 javac 命令生成的 class 类文件放置在对应的.java 源文件相同的目录下。为了更好地控制版本、测试和部属类文件，可以使用-d 选项来指定存放 class 类文件的目录。需要注意的是，如果指定的目标目录不存在，则会获得一个编译器错误。

规则 03：采用 JDK 命令行方式运行一个类文件采用 java 命令。java 命令的完整语法形式为 java [options] class [args]。

规则 04：Java5 提供一个名为 java.util.Properties 的类，该类用于访问系统的持久信息，并且可以检索自己的属性。在 java 命令中，使用-D（注意是大写）选项来设置系统属性。系统属性由名称=值对组成，这些值对必须直接追加到选项-D 的后面，中间不能有空格。如果属性值包含了空格，那么整个属性值必须像字符串一样全部

放置在双引号内。

规则 05：使用 System.getProperties()方法获得 Properties 类实例，使用 Properties 类上定义的 getProperty(String key)方法获取指定属性值，使用 setProperty(String key, String value)方法设置属性值。

规则 06：java 命令中的 args 为可选项，用于从命令行向 Java 应用程序传递参数。这些参数之间用空格分隔，被传入到应用程序入口 main()方法的 args 字符串数组中。

规则 07：javac 和 java 命令均使用-classpath 或-cp 选项来指定目标位置的路径，路径包含大量由分隔符分开的目录位置。对于 UNIX 系统，正斜杠/用于分隔组成路径的目录，路径间的分隔符使用冒号。对于 Windows 的系统，反斜杠\用于分隔组成路径的目录，路径间的分隔符使用分号。

规则 08：默认情况下，javac 和 java 命令都不会搜索当前目录。因此在指定类路径中，要将当前目录指定为一个搜索位置，应使用句点来代表当前路径。

规则 09：类路径从左至右进行搜索，一旦找到指定目标类，搜索就终止了，因此搜索路径位置间的顺序排列很重要。

规则 10：当把类放置到包中时，必须使用它的完全限定名称，即包名加类名。

规则 11：类路径可以包含绝对路径或相对路径。绝对路径以斜杠开头，代表从系统根目录开始，与当前目录无关，因此绝对目录是惟一的。相对路径是不以斜杠开头的路径，而是以当前目录为根目录的路径。

规则 12：JAR 文件代表 Java 存档，用于压缩数据和存档数据。JAR 文件不仅包含类文件，而且还维持整个文件间的目录结构。

规则 13：在 javac 和 java 命令的类路径中指定要使用的目标 JAR 文件。在类路径中包含 JAR 文件时，不仅需要指定 JAR 文件所在目录路径，而且还要包括 JAR 文件的名称。需要注意的是，JAR 文件不被强制放置在系统默认的 JAR 文件放置../jre/lib/ext 目录中。

1.5 基本数据类型

规则 01：Java 语言中共有 8 种基本数据类型，分别为布尔型（boolean）、字符型（char）、字节型（byte）、短整型（short）、整型（int）、长整型（long）、单精度型（float）和双精度型（double）。

规则 02：这 8 种基本数据类型的有效字节表示位数。

类型	有效字节数	类型	有效字节数
boolean	1	int	32
byte	8	float	32
char	16	long	64
short	16	double	64

规则 03：Java 语言中的布尔型数据的取值只能为 true 或 false，不像其他语言，例如 C++。可以允许用整型值代替 true 或 false。

规则 04：Java 语言中共有 4 种带符号的整数型数据，分别为 byte、short、int 和 long。

规则 05：4 种带符号的整数型数据的取值范围也有规定。

类型	有效字节数	最小值	最大值
byte	8	-27	2^7-1
short	16	-2^{15}	2^{15}-1
int	32	-2^{31}	2^{31}-1
long	64	-2^{63}	2^{63}-1

注意：取值范围计算公式为 $2^{(number\ of\ bits-1)}$到 $2^{(number\ of\ bits-1)}$–1。最小值和最大值中底数 2 的指数均比对应类型的有效字节数小 1。

规则 06：Java 语言中的 char 数据类型属于整数型，它是一个无符号的整数型，其取值范围为 0～2^{16}-1。

规则 07：Java 语言中的字符型是基于 16 位的统一字符集，不是基于 7 位的 ASCII 字符集。为了和 ASCII 字符集兼容，统一字符集包括了 ASCII 字符集，即用高 9 位为 0 的字符映射为 ASCII 字符集中对应编码的字符。也就是说，高 9 位为 0，低 7 位和 ASCII 字符集中对应字符的编码一致。例如 65 在 ASCII 字符集和统一字符集中均代表大写字母 A。

规则 08：Java 语言中共有两种浮点型基本数据类型，分别为 float 和 double，它们对应的封装类 Float 和 Double 都定义有 NaN、Positive_Infinity、Negative_Infinity 常量，用于表示非正常运算结果。NaN（Not a Number）常量类属性代表运算结果是一个非数值；Positive_Infinity 常量类属性代表正无限大；Negative_Infinity 常量类属性代表负无限大。值得注意的是，尽管两个 Infinity 常量代表无限值，但其只是一个特殊数值，而非非数值，这一点不同于 NaN 常量类属性。

1.6 文字值

规则 01：文字值指在源文件中使用的值，不是程序运行时的实际值。Java 语言中有两种文字值，分别为基本数据类型文字值和字符串。要注意，文字值不能作为变量，也就是说不能给文字值赋值，即文字值不能出现在赋值表达式的左边。

规则 02：布尔型文字值只能采用 true 或 false 两种形式。true 或 false 带双引号时不是布尔型文字值，而是字符串。

规则 03：字符型文字值是用单引号括起的一个字符。由于此种形式的字符型文字值只能表示由键盘输入的各类字符型值，对于无法由键盘输入的其他统一字符集中的字符，则无法表示。因此 Java 语言定义了另一种形式的字符型文字值表示方式，即以\u 作为前缀，后跟 4 位 16 进制的整数，例如\u1234。

规则 04：Java 语言中共定义了 8 个特殊的字符。

字符	意义	字符	意义
'\n'	代表一个新行	'\r'	代表回车
'\t'	代表制表位	'\b'	代表退格
'\f'	代表进纸	'\''	代表单引号
'\"'	代表双引号	'\\'	代表反斜线

规则 05：整数型文字值有 3 种进制表示，默认为十进制，以 0 作为前缀代表的是 8 进制，

以 0x 或 0X 作为前缀代表的是 16 进制。请注意，16 进制中的字母数，既可以是大写，也可以是小写。

规则 06：整数型文字值默认的数据类型是 32 位的整型。如果需要声明为 64 位的长整型文字值，则需要加后缀 l 或 L 来表示。

规则 07：浮点型文字值有两种表示方式，一种是小数，例如 1.4444。另一种是科学计数法，例如 1.4E+21。请注意，科学计数法中的大小写为 E、e 均可，但 E、e 前后必须均有数值。

规则 08：浮点型文字值默认的数据类型是 64 位的双精度型。如果需要声明为 32 位的单精度型文字值，则需要加后缀 f 或 F 来表示。请注意，尽管浮点型文字值默认的数据类型是双精度型，但也可以通过加后缀 D 或 d 来明确声明。

规则 09：字符串文字值是指用双引号括起的一个字符序列，例如"smith"。请注意，含单个字符的字符串不是字符值，例如"a"不同于'a'。在 Java 语言中，字符串是对象类型，字符则是数据类型。

1.7 数组

规则 01：数组是一个具有相同类型元素的有序集合。数组中每个元素的类型相同，且与数组声明的类型一致。在 Java 语言中，数组是由基本数据类型（或其他引用类型）构成，代表一个对象。这意味着数组不是简单作为存储基本数据类型的存储器，而是一个可以具有方法和属性的对象。

规则 02：为了使用一个数组必须采取 3 个步骤——声明数组、创建数组、初始化数组。

规则 03：声明一个数组就是通知编译器数组的名称和数组元素的类型，请注意，声明数组实际上并未给数组分配内存空间。Java 语言中规定声明一个数组可以采用两种形式：datatype[] arrayName;或者 datatype arrayName[]。这两种形式在性能上没有区别，只是方括号的位置放置不同。一般而言，第二种形式可读性强，第一种方式常用于一个把数组作为返回值的方法声明中，例如 double[] MethodName()。

规则 04：数组元素的类型可分为 3 种，分别为基本数据类型、对象引用型和数组对象。实际上可以合并为两种，因为 Java 中的数组也是一种特殊的对象，所以第二种和第三种均为对象引用型。

规则 05：构造一个数组就是根据数组的大小为数组分配存储空间。Java 语言中规定，构造一个指定长度的数组语法为 new datatype[size]。请注意，数组长度只在构造时指定，在声明时不能指定。

规则 06：指定一个数组的大小有两种方式，一是用一个变量，二是用一个明确的数。由于数组长度直到运行期才被使用，所以第一种方式优于第二种方式。其优点是在编写程序时，不需要知道数组的具体大小，直到运行时根据实际需求再决定。

规则 07：Java 语言中规定，用来指定数组长度的数值类型只能是字节型、短整型或整型，而不能是长整型，更不能是浮点型。

规则 08：声明数组和构造数组可以在一条语句中一次性完成，语法为 datatype arrayName[] = new datatype[size]。

规则 09：Java 中的数组是一个特殊的对象，其实例和一般的对象既相似又有区别。数组对