java认证:Java中的assert关键字Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/561/2021_2022_java_E8_AE_A4_E8_AF_81_c104_561596.htm J2SE 1.4在语言上提供了一个新特性，就是assertion(断言)功能，它是该版本在Java语言方面最大的革新。在软件开发中，assertion是一种经典的调试、测试方式。 assertion(断言)在软件开发中是一种常用的调试方式，很多开发语言中都支持这种机制，如C，C 和Eiffel等，但是支持的形式不尽相同，有的是通过语言本身、有的是通过库函数等。另外，从理论上来说，通过assertion方式可以证明程序的正确性，但是这是一项相当复杂的工作，目前还没有太多的实践意义。 在实现中，assertion就是在程序中的一条语句，它对一个boolean表达式进行检查，一个正确程序必须保证这个boolean表达式的值为true；如果该值为false，说明程序已经处于不正确的状态下，系统将给出警告或退出。一般来说，assertion用于保证程序最基本、关键的正确性。assertion检查通常在开发和测试时开启。为了提高性能，在软件发布后，assertion检查通常是关闭的。下面简单介绍一下Java中assertion的实现。 1．1) 语法表示 在语法上，为了支持assertion，Java增加了一个关键字assert。它包括两种表达式，分别如下： assert expression1. assert expression1: expression2. 在两种表达式中，expression1表示一个boolean表达式，expression2表示一个基本类型或者是一个对象(Object)，基本类型包括boolean,char,double,float,int和long。由于所有类都为Object的子类，因此这个参数可以用于所有对象。 1、assert gt. 如果gt.为true，则程序继续执行。 如果为false，则程序抛

出AssertionError，并终止执行。 2、assert gt. : gt. 如果gt.为true，则程序继续执行。 如果为false，则程序抛出java.lang.AssertionError，并输出gt.。 1．2) 语义含义 在运行时，如果关闭了assertion功能，这些语句将不起任何作用。如果打开了assertion功能，那么expression1的值将被计算，如果它的值为false，该语句强抛出一个AssertionError对象。如果assertion语句包括expression2参数，程序将计算出expression2的结果，然后将这个结果作为AssertionError的构造函数的参数，来创建AssertionError对象，并抛出该对象；如果expression1值为true，expression2将不被计算。 一种特殊情况是，如果在计算表达式时，表达式本身抛出Exception，那么assert将停止运行，而抛出这个Exception。 1．3) 一些assertion例子 下面是一些Assert的例子。 assert 0 lt. value:"value=" value. assert ref != null:"ref doesnt equal null". assert isBalanced(). 1．4) 编译 由于assert是一个新关键字，使用老版本的JDK是无法编译带有assert的源程序。因此，我们必须使用JDK1.4(或者更新)的Java 编译器，在使用Javac命令时，我们必须加上-source 1.4作为参数。-source 1.4表示使用JDK 1.4版本的方式来编译源代码，否则编译就不能通过，因为缺省的Javac编译器使用JDK1.3的语法规则。 一个简单的例子如下：javac -source 1.4 test.java 1．5) 运行 由于带有assert语句的程序运行时，使用了新的ClassLoader和Class类，因此，这种程序必须在JDK1.4(或者更高版本)的JRE下运行，而不能在老版本的JRE下运行。 由于我们可以选择开启assertion功能，或者不开启，另外我们还可以开启一部分类或包的assertion功能，所以运行选项变得有些复杂。通过这些选项，我们可以过滤

所有我们不关心的类，只选择我们关心的类或包来观察。下面介绍两类参数： 参数 -esa和 -dsa： 它们含义为开启(关闭)系统类的assertion功能。由于新版本的Java的系统类中，也使了 assertion语句，因此如果用户需要观察它们的运行情况，就需要打开系统类的assertion功能 ，我们可使用-esa参数打开，使用 -dsa参数关闭。 -esa和-dsa的全名为-enablesystemassertions和-disenablesystemassertions，全名和缩写名有同样的功能。 参数 -ea和 -ea： 它们含义为开启(关闭)用户类的assertion功能：通过这个参数，用户可以打开某些类或包的assertion功能，同样用户也可以关闭某些类和包的assertion功能。打开assertion功能参数为-ea；如果不带任何参数，表示打开所有用户类；如果带有包名称或者类名称，表示打开这些类或包；如果包名称后面跟有三个点，代表这个包及其子包；如果只有三个点，代表无名包。关闭 assertion功能参数为-da，使用方法与-ea类似。 -ea和-da的全名为-enableassertions和-disenableassertions，全名和缩写名有同样的功能。 下面表格表示了参数及其含义，并有例子说明如何使用。 参数 例子 说明 -ea java -ea 打开所有用户类的assertion -da java -da 关闭所有用户类的assertion -ea:gt. java -ea:MyClass1 打开MyClass1的assertion -da:gt. java -da: MyClass1 关闭MyClass1的assertion -ea:gt. java -ea:pkg1 打开pkg1包的assertion -da:gt. java -da:pkg1 关闭pkg1包的assertion -ea:... java -ea:... 打开缺省包(无名包)的assertion -da:... java -da:... 关闭缺省包(无名包)的assertion -ea:gt.... java -ea:pkg1... 打开pkg1包和其子包的assertion -da:gt.... java -da:pkg1... 关闭pkg1包和其子包的assertion -esa java -esa 打开系统类的assertion -dsa java -dsa 关闭系统类的assertion 更多