一、选择题

1．软件测试的目的是（ B ）。

A．试验性运行软件

B．发现软件错误

C．证明软件正确

D．找出软件中全部错误

2．软件测试中白盒法是通过分析程序的（B ）来设计测试用例的。

A．应用范围

B．内部逻辑

C．功能

D．输入数据

3．黑盒测试是根据程序的（C ）来设计测试用例的。

A．应用范围

B．内部逻辑

C．功能

D．输入数据

4．为了提高软件测试的效率，应该（D  ）。

A．随机地选取测试数据

B．取一切可能的输入数据作为测试数据

C．在完成编码以后制定软件的测试计划

D．选择发现错误可能性最大的数据作为测试用例

5．与设计测试用例无关的文档是（A ）。

A．项目开发计划

B．需求规格说明书

C．设计说明书

D．源程序

6．测试的关键问题是（B ）。

A．如何组织软件评审

B．如何选择测试用例

C．如何验证程序的正确性

D．如何采用综合策略

7．软件测试用例主要由输入数据和（C ）两部分组成。

A．测试计划

B．测试规则

C．预期输出结果

D．以往测试记录分析

8．成功的测试是指运行测试用例后（B）。

A．未发现程序错误

B．发现了程序错误

C．证明程序正确性

D．改正了程序错误

9．下列几种逻辑覆盖标准中，查错能力最强的是（D ）。

A．语句覆盖

B．判定覆盖

C．条件覆盖

D．条件组合覆盖

10．在黑盒测试中，着重检查输入条件组合的方法是（D ）。

A．等价类划分法

B．边界值分析法

C．错误推测法

D．因果图法

11．单元测试主要针对模块的几个基本特征进行测试，该阶段不能完成的测试是（A ）。

A．系统功能

B．局部数据结构

C．重要的执行路径

D．错误处理

12．软件测试过程中的集成测试主要是为了发现（B ）阶段的错误。

A．需求分析

B．概要设计

C．详细设计

D．编码

13.不属于白盒测试的技术是（D）。

A．路径覆盖

B．判定覆盖

C．循环覆盖

D．边界值分析

14.一个完整的测试部门，一般不包含以下角色（D ）

A．测试主管

B．测试工程师

C．测试设计人员

D．培训师

15.软件测试的目的是（ B ）

A．避免软件开发中出现的错误

B．发现软件开发中出现的错误

C．尽可能发现并排除软件中潜藏的错误，提高软件的可靠性

D．修改软件中出现的错误

16.为了提高测试的效率，正确的做法是（ A ）。

A．选择发现错误可能性大的数据作为测试用例

B．在完成程序的编码之后再制定软件的测试计划

C．随机选取测试用例

D．使用测试用例测试是为了检查程序是否做了应该做的事

17.以下那一种选项不属于软件缺陷（ D ）。

A．软件没有实现产品规格说明所要求的功能

B．软件中出现了产品规格说明不应该出现的功能

C．软件实现了产品规格没有提到的功能

D．软件实现了产品规格说明所要求的功能但因受性能限制而未考虑可移植性问题

18.单元测试中设计测试用例的依据是（ D ）。

A．概要设计规格说明书

B．用户需求规格说明书

C．项目计划说明书

D．详细设计规格说明书

19.下面有关软件缺陷的说法中错误的是（ C）。

A．缺陷就是软件产品在开发中存在的错误

B．缺陷就是软件维护过程中存在的错误、毛病等各种问题

C．缺陷就是导致系统程序崩溃的错误

D．缺陷就是系统所需要实现某种功能的实效和违背

20.下面有关测试原则的说法正确的是（ A ）。

A．测试用例应由测试的输入数据和预期的输出结果组成

B．测试用例只需选取合理的输入数据

C．程序最好由编写该程序的程序员自己来测试

D．使用测试用例进行测试是为了检查程序是否做了它该做的事

21.以程序的内部结构为基础的测试用例技术属于（ D ）。

A．灰盒测试

B．数据测试

C．黑盒测试

D．白盒测试

22.导致软件缺陷的最大原因是：\_\_\_\_\_\_\_\_。   （ A）

A．软件需求说明书

B．设计方案

C．编码

D．维护

23. 测试用例是为达到最佳的测试效果或高效的揭露隐藏的错误而精心设计的少量测试数据，至少应该包括\_\_\_\_\_\_\_\_。                （A）

A．测试输入、执行条件和预期的结果。

B．测试目标、测试工具

C．测试环境

D．测试配置

24.在软件测试阶段，测试步骤按次序可以划分为以下几步： ( A )

A.单元测试、集成测试、系统测试、验收测试

B.验收测试、单元测试、系统测试、集成测试

C.单元测试、集成测试、验收测试、系统测试

D.系统测试、单元测试、集成测试、验收测试

25.软件测试是软件质量保证的重要手段，下述哪种测试是软件测试的最基础环节？  （  B   ）

A．功能测试

B．单元测试

C．结构测试

D．验收测试

26.必须要求用户参与的测试阶段是\_\_\_\_\_\_\_\_。   （ D ）

A. 单元测试

B. 集成测试

C. 确认测试

D. 验收测试

27. 在用白盒测试中的逻辑覆盖法设计测试用例时，有语句覆盖、分支覆盖、条件覆盖、判定-条件覆盖、条件组合覆盖和路径覆盖等，在下列覆盖中，其中\_\_\_\_\_\_\_\_是最弱的覆盖准则。    （ A ）

A．语句覆盖

B．条件覆盖

C．判定-条件覆盖

D．路径覆盖

28. 下列可以作为软件测试对象的是（ D）。

A．需求规格说明书

B．软件设计规格说明

C．源程序

D．以上全部

29. 调试是\_\_\_\_\_\_\_\_。   （ D ）

A．发现与预先定义的规格和标准不符合的问题

B．发现软件错误征兆的过程

C．有计划的、可重复的过程

D．消除软件错误的过程

30. 下列\_\_\_\_\_\_\_\_\_不属于软件缺陷         （ A）

A．测试人员主观认为不合理的地方

B．软件未达到产品说明书标明的功能

C．软件出现了产品说明书指明不会出现的错误

D．软件功能超出产品说明书指明范围

31. 下列软件属性中，软件产品首要满足的应该是（A）

A．功能需求

B．性能需求

C．可扩展性和灵活性

D．容错纠错能力

32. 为了提高测试的效率，正确的做法是 （A）。

A 选择发现错误可能性大的数据作为测试用例

B 在完成程序的编码之后再制定软件的测试计划

C 随机选取测试用例

D 使用测试用例测试是为了检查程序是否做了应该做的事

33. 产品发布后修复软件缺陷比项目开发早期这样做的费用要高  （ D ）

A．1～2倍

B．10～20倍

C．50倍

D．100倍或更高

34. 白盒测试是根据程序的\_\_\_\_\_\_\_\_\_来设计测试用例,黑盒测试是根据软件的规格说明来设计测试用例。     （ C ）

A．功能

B．性能

C．内部逻辑

D．内部数据

35. 条件覆盖的目的是\_\_\_\_\_\_\_\_。         （  C ）

A．使每个判定的所有可能的条件取值组合至少执行一次（条件组合覆盖）

B．使程序中的每个判定至少都获得一次“真”值和“假”值。

C．使程序中的每个判定中每个条件的可能值至少满足一次。

D．使程序中的每个可执行语句至少执行一次。

36. 如果一个判定中的复合条件表达式为（A > 1）or（B <= 3），则为了达到100%的条件覆盖率，至少需要设计多少个测试用例（ B ）。

A. 1

B. 2

C. 3

D. 4

37. 根据软件需求规格说明书，在开发环境下对已经集成的软件系统进行的测试是（ A ）。

A.系统测试

B.单元测试

C.集成测试

D.验收测试

38. 下列哪一项不是白盒测试？（C）

A.单元测试

B.集成测试

C.系统测试

D.回归测试

39．在Assert类中断言对象为NULL是\_\_\_\_\_。（C）

 A. assertEquals

B. assertTrue

C. assertNull

D. fail

40．在Assert类中断言两个对象相等是\_\_\_\_\_。（A）

A. assertEquals

B. assertTrue

C. assertSame

D. fail

41．有一组测试用例使得每一个被测试用例的分支覆盖至少被执行一次，它满足的覆盖标准\_\_\_\_\_\_\_\_\_\_\_。（B）

A.语句覆盖

B.判定覆盖

C.条件覆盖

D.路径覆盖

42. 在Assert类中断言测试失败是\_\_\_\_\_。（D）

A. assertEquals

B. assertTrue

C. assertSame

D. fail

43.软件测试的目的是\_\_\_\_\_\_\_\_\_\_\_。（C）

A．表明软件的正确性

B．评价软件质量

C．尽可能发现软件中的错误

D．判定软件是否合格

44.在Junit中， testXXX()方法就是一个测试用例，测试方法是\_\_\_\_\_\_。（B）

A． private  void  testXXX()

B． public  void  testXXX()

C． public  float testXXX()

D． public   int  testXXX()

45.在下面所列举中的逻辑测试覆盖中，测试覆盖最强的是\_\_\_\_\_\_\_\_\_\_。（B）

A．条件覆盖

B．条件组合覆盖

C．语句覆盖

D．判定覆盖

46.在下面所列举中的逻辑测试覆盖中，测试覆盖最弱的是\_\_\_\_\_\_\_\_\_\_。（C）

A．条件覆盖

B．条件组合覆盖

C．语句覆盖

D．判定覆盖

47.软件测试是软件质量保证的重要手段，下述哪种测试是软件测试的最基础环节？（B）

A．集成测试

B．单元测试

C．目的测试

D．确认测试

48.Junit的TestCase类提供和tearDown（）方法，分别完成对测试环境的建立和拆除。（A）

A． setUp()

B． set()

C． setap()

D． setDown()

49. Junit测试在单元测试阶段测试，主要用于\_\_\_\_\_\_\_\_。（A）

Ａ．白盒测试

Ｂ．灰盒测试

Ｃ．黑盒测试

Ｄ．确认测试

50.Junit适用于java开发人员在\_\_\_\_\_\_阶段，进行单个方法实现功能或者类本身的测试，主要用于白盒测试。（C）

Ａ．集成测试

Ｂ．验收测试

Ｃ．单元测试

Ｄ．确认测试

51.软件测试是按照特定的规程，\_\_\_\_\_\_\_\_\_\_\_的过程。（A）

Ａ．发现软件错误

Ｂ．说明程序正确

Ｃ．证明程序没有错误

Ｄ．设计并运行测试用例

52.下列模型哪个是软件测试过程模型 \_\_\_\_\_\_\_\_\_\_\_。（A）

Ａ． W模型

Ｂ．漠布模型

Ｃ． L模型

Ｄ． G模型

53.Junit有两个包： \_\_\_\_\_\_\_\_\_\_和Junit.extensions。（B）

Ａ．Junit.frametest

Ｂ．Junit.framework

Ｃ．Junit.amework

Ｄ．Junit.assert

54.单元测试一般以\_\_\_\_\_\_\_\_\_\_为主。（A）

Ａ．白盒测试

Ｂ．黑盒测试

Ｃ．系统测试

Ｄ．分析测试

55.编码阶段产生的错误由\_\_\_\_\_\_\_\_\_\_检查出来的。（A）

Ａ．单元测试

Ｂ．集成测试

Ｃ．系统测试

Ｄ．有效性测试

56．单元测试范围包括（A）

A．方法间集成

B．平行类测试

C．类簇测试

D．以上都是

57．关于JUNIT，描述错误的是（B ）

A．JUNIT是Java语言的单元测试框架　 　B．JUNIT只能测试公共函数

C．JUNIT推荐先测试后实现的方法　　 D．setUp、tearDown函数只执行一次

58、单元测试的测试用例主要根据（ B  ）的结果来设计。

A．需求分析

B．源程序

C．概要设计

D．详细设计

59、单元测试的测试目的是（A  ）

A．保证每个模块能正常工作

B．保证每个组件能正常工作

C．确保缺陷得到解决

D．使程序正常运行

60、Junit是（  C ）语言的单元测试框架。

A．C

B．.Net

C．Java

D．C++

61. 单元测试的依据是（ ）

A． 模块功能规格说明 B． 系统模块结构图

C． 系统需求规格说明 D． 详细设计说明书

62. 已下对单元测试不正确的说法是（ ）

A．单元测试的主要目的是针对编码过程中可能存在的各种错误

B．单元测试一般是由程序开发人员完成的

C．单元测试是一种不需要关注程序结构的测试

D．单元测试属于白盒测试的一种

63．单元测试将根据在（）阶段中产生的规格说明进行。

A．可行性研究与计划 B．需求分析

C．概要设计 D．详细设计

64．在进行单元测试时，常用的方法是（ ）

A． 采用黑盒测试，辅之以白盒测试

B． 采用白盒测试，辅之以黑盒测试

C． 只使用黑盒测试

D． 只使用白盒测试

65．测试驱动开发的含义（） B

A．先写程序后写测试的开发方法

B．先写测试后写程序，即“测试先行”

C．用单元测试的方法写测试

D．不需要测试的开发

三、填空题

1. 单元测试是以\_\_详细 设计\_说明书为指导，测试源程序代码。

  详细设计

 2. Junit中的所有的Assert方法全部放在 Assert 类，用于对比\_期望值和实际值是否相同。

Assert，期望值

3. Junit是一个开放源代码的\_\_java\_测试框架，用于编写和运行可重复的测试。

java

4.在Junit中3.8中testXXX()测试方法必须满足几个条件：\_public\_\_， \_\_void\_和无方法参数。

  public ,void

5. JUnit 将测试失败的情况分为两种：failure 和 error。

6.在一个完整的测试过程中，一个 Mock 对象将会经历两个状态：Record 状态和 Replay 状态。

三、判断题

1．软件测试的目的是尽可能多的找出软件的缺陷。（Y）

2．Beta 测试是验收测试的一种。（Y）

3．验收测试是由最终用户来实施的。（N）

4．项目立项前测试人员不需要提交任何工件。（N）

5．单元测试能发现约80%的软件缺陷。（Y）

7．自底向上集成需要测试员编写驱动程序。（Y）

9．测试人员要坚持原则，缺陷未修复完坚决不予通过。（N）

11．我们可以人为的使得软件不存在配置问题。（N）

13、好的测试员不懈追求完美。 ( Y)

14、测试程序仅仅按预期方式运行就行了。 (N)

16、软件测试员可以对产品说明书进行白盒测试。 ( N)

17、静态白盒测试可以找出遗漏之处和问题。 ( Y)

18、理论上白盒测试可以发现软件所有的缺陷。 (N)

19、可以发布具有配置缺陷的软件产品。 (Y)

20、软件必须进行某种程度的兼容性测试。 ( Y)

21、所有软件都有一个用户界面，因此必须测试易用性。(N)

22．软件测试的目的是尽可能多的找出软件的缺陷。（Y）

23．Beta测试是验收测试的一种。（Y）

24．验收测试是由最终用户来实施的。（N）

25．项目立项前测试人员不需要提交任何工件。（Y）

28．测试人员要坚持原则，缺陷未修复完坚决不予通过。（ N ）

30．我们可以人为的使得软件不存在配置问题。（N）

31. 在JUnit3，所有的test case的方法名都要以"test"为前缀。（ Y）

32. 在JUnit4，所有的test case的方法前需要加上@Test。（ Y ）

33.一个 Mock 对象将会经历两个状态：Record 状态和 Replay 状态。（N ）

三、名词解释

1.软件缺陷：（符合下列规则的叫软件缺陷)：

　1）．软件未达到产品说明书的功能

　2）．软件出现了产品说明书指明不会出现的错误

　3)．软件功能超出产品说明书指明范围

　4)．软件未达到产品说明书虽未指出但应达到的目标

　5)．软件测试员认为难以理解、不易使用、运行速度缓慢、或者最终用户认为不好

2.单元测试：

单元测试是对软件设计的最小单元——模块进行正确性检验的测试工作，主要测试模块在语法、格式和逻辑上的错误。

3．白盒测试：

白盒测试也称为结构测试或逻辑驱动测试，前提是知道产品内部工作过程，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的机构测试程序，检测程序中的每条通路是否都能够按预定要求正确工作，而不管产品的功能，主要用于软件验证。

4.回归测试

回归测试指软件系统被修改或扩充（如系统功能增强或升级）后重新进行的测试，是为了保证对软件所做的修改没有引入新的错误而重复进行的测试。

5．验收测试

  验收测试是软件产品完成了功能测试和系统测试之后，在产品发布之前所进行的软件测试活动。

6．失败测试

 失败测试纯粹为了破坏软件而设计和执行的测试案例，被称为失败测试。

7．驱动模块

驱动模块用来代替被测单元的上层模块的。驱动模块能接收测试数据，调用被测单元，也就是将数据传递给被测单元，最后打印测试的执行结果。可将驱动模块理解为被测单元的主程序。

8. 桩模块

桩模块又称为存根模块，它用来代替被测单元的子模块。设计桩模块的目的是模拟实现被测单元的接口。桩模块不需要包括子模块的全部功能，但应做少量的数据操作，并打印接口处的信息。

9．白盒测试

白盒测试也称为结构化测试、基于代码的测试，是一种测试用例设计方法，它从程序的控制结构导出测试用例。

10.软件测试：

在软件投入运行之前，对软件需求分析、设计规格说明书和编码的 最终复审，是软件质量保证的关键步骤。

11.软件测试的目的：

为了检验软件系统是否满足需求

发现软件中的缺陷，包括需求、设计方面的缺陷和程序中包含的BUG。这里缺陷是一种泛称，它可以指软件功能的错误，也可以指性能低下，易用性差以及其他软件工作产品中的缺陷等等。

 12.α测试：

系统开发接近完成时对软件进行的测试，开发者在测试现场

13.β测试：

系统开发根本完成时对软件进行的测试，开发者不在测试现场

14. 测试用例（Test Case）：

测试用例是为了高效率地发现软件缺陷而精心设计的少量测试数据。

15. 单元测试的目的：

1. 验证代码能否达到详细设计的预期要求。
2. 发现代码中不符合编码规范的地方。
3. 准确定位发现的错误，以便排除错误。

四、简答题（3×5＝15分）

1.计算环路复杂度方法有哪三种？

（1）V(G) = 判定节点数+ 1

（2）V(G) = E-N+2

（3）V(G) =区域数+ 1

2.白盒测试有几种方法？

白盒测试方法分为两大类：静态测试方法和动态测试方法。

静态测试方法：检查软件的表示和描述是否一致,没有冲突或者没有歧义。

动态测试方法：语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、路径覆盖。

3．在单元测试中，所谓单元是如何划分的？

单元测试是对软件基本组成单元的测试。所谓“单元”是指：

* 具有明确的功能
* 具有明确的规格定义（详细设计规格说明书）
* 有与其他部分明确的接口定义
* 能够与程序的其他部分清晰的进行区分
* 例子：
  + 1. 在传统的结构化编程语言如C语言中，单元一般是模块，也就是函数或子过程；
    2. 在C++和JAVA中， 单元是类或类的方法；

4．简述JUnit单元测试步骤

① 计划单元测试

② 设计单元测试

③ 实现单元测试

④ 执行单元测试

⑤ 单元测试结果分析并提交测试报告

5．Junit4.x和JUnit3.x有哪些区别

解：junit4和junit3有一定的区别，在junit3中，测试方法必须以test开始，即方法为testXXX(),在junit4中没有这个限制。因为在junit4中可以利用注解（annotation）@Test来指定测试方法。但是，在junit4中不用继承TestCase父类，如果使用继承，则junit4的注解将会失效。 junit4不继承TestCase，自然不能直接使用assertEqual等方法，如果要使用断言必须直接静态引用，即 Assert.assertEqual(int ,int  ) 这种方式。

6．分析单元测试和代码调试的区别

7．简述单元测试的主要任务

解：单元测试的主要任务是：模块接口测试；局部数据结构测试；路径测试；错误处理测试；边界测试。

8．如果开发时间紧迫，是否可以跳过单元测试而直接进行集成测试？为什么？

解：不可以。因为没有经过单元测试的模块会遗留大量的缺陷到集成测试阶段，而在集成测试阶段对这些缺陷定位困难，导致后续工作展开困难，修复缺陷成本成指数级增长。

9.为以下程序段设计一组测试用例，要求分别满足语句覆盖、判定覆盖、条件覆盖。

 int test(int A,int B)

{

    if((A>1) AND (B<10))     then

           X=A-B;

    if((A=2) OR (B>20))     then

           X=A+B;

    return x;

}

答：

语句覆盖测试用例：A=2，B=0；

判定覆盖测试用例：A=3，B=0；A=2，B=20；

条件覆盖测试用例：A=2，B=0；A=0，B=21；

五．编程题

根据给出的类，写出测试代码，将空白处填上代码，使测试通过。

1. 计算器类

**package** edu.sziit.junit4;

**public** **class** Calculator

{

**public** **int** add(**int** a, **int** b)

{

**return** a + b;

}

**public** **int** subtract(**int** a, **int** b)

{

**return** a - b;

}

**public** **int** multiply(**int** a, **int** b)

{

**return** a \* b;

}

**public** **int** divide(**int** a, **int** b)

{

**if** (b == 0) {

**throw** **new** ArithmeticException("除数不能为0");

}

**return** a / b;

}

}

**计算类的测试类**

package edu.sziit.junit4;

import static org.junit.Assert.\*;

import org.junit.After;

import org.junit.AfterClass;

import org.junit.Assert;

import org.junit.Before;

import org.junit.BeforeClass;

import org.junit.Ignore;

import org.junit.Test;

public class CalculatorTest {

private Calculator cal = null;

@BeforeClass

public static void beforeClass() {

System.out.println("===在所有测试方法执行前执行--beforeClass=====\n");

}

/\*\*

\* 在测试方法执行完前执行

\*/

@Before

public void beforeMethod() {

System.out.println("在每个测试方法执行前执行--beforeMethod...");

cal = new Calculator();

}

// 若超时1s，将报error错误

@Test(timeout = 1000)

// 若期待有异常抛出：expected = Exception.class

public void testSubtract() // throws Exception

{

System.out.println("测试方法testSubstrub被执行...");

assertEquals(-1, cal.subtract(1, 2));

// Assert.assertEquals(-1, cal.divide(1, 0));

}

@Test(expected=Exception.class)

public void testDivide() throws Exception{

cal.divide(1, 0);

}

@Test

@Ignore("该testMultiply测试方法由于XX原因需要被忽略掉")

public void testMultiply() {

System.out.println("测试方法testMultiply被执行...");

assertEquals(2, cal.multiply(1, 2));

}

@Test

public void testadd() {

int result = cal.add(1, 5);

Assert.assertEquals(6, result);

}

@Test

public void testSubtract2() {

int result = cal.subtract(1, 6);

Assert.assertEquals(-5, result);

}

@Test

public void testMultiply2() {

int result = cal.multiply(2, 3);

Assert.assertEquals(6, result);

}

/\*\*

\* 在测试方法执行完后执行

\*/

@After

public void afterMethod() {

// cal = null;// 在每个测试方法执行后主动销毁对象

System.out.println("在每个测试方法执行后执行--afterMethod...\n");

}

@AfterClass

public static void afterClass() {

System.out.println("=====在所有测试方法执行后执行--afterClass=====");

}

}

2. 求数据最大值类，Largest类

**package** edu.sziit.junit4;

**public** **class** Largest

{

**public** **int** getLargest(**int**[] arr) **throws** Exception

{

**if** (**null** == arr || 0 == arr.length)

{

**throw** **new** Exception("数组不能为空");

}

**int** result = arr[0];

**for** (**int** i = 0; i < arr.length; i++)

{

**if** (result < arr[i])

{

result = arr[i];

}

}

**return** result;

}

}

**Largest类的测试类**

package edu.sziit.junit4;

import static org.junit.Assert.\*;

import org.junit.After;

import org.junit.AfterClass;

import org.junit.Assert;//不要将3.8和4.x的API混用

import org.junit.Before;

import org.junit.BeforeClass;

import org.junit.Ignore;

import org.junit.Test;

public class LargestTest {

private Largest lar;

@BeforeClass

public static void setUpBeforeClass() throws Exception {

}

@AfterClass

public static void tearDownAfterClass() throws Exception {

}

@Before

public void setUp() throws Exception {

lar =new Largest();

}

@After

public void tearDown() throws Exception {

}

@Test

public void testGetLargest() {

int[] arr = { 1, 10, 54, 36, 65 };

int result = 0;

try

{

result = lar.getLargest(arr);

}

catch (Exception e)

{

e.print

Assert.fail("测试失败");

}

Assert.assertEquals(65, result);

}

@Test(expected = Exception.class)

@Ignore(value = "This case is ignored")

public void testGetLargest2() throws Exception

{

lar.getLargest(null);

}

@Test(expected = Exception.class)

public void testGetLargest3() throws Exception

{

lar.getLargest(new int[] {});

}

}

3.测试套件

如果CalculatorTest是Calculator的测试类和LargestTest是Largest的测试类，则在 JUnit4.x中利用测试套件同时测试Calculator和Largest的代码如下所示：

package edu.sziit.junit4;

import org.junit.runner.RunWith;

import org.junit.runners.Suite;

@RunWith(Suite.class)//代表以Suite来运行这个测试类

@Suite.SuiteClasses({CalculatorTest.class, LargestTest.class}) //表示要测试的类。

public class TestAll {

}

知识点：

@Before；

对于每一个测试方法都要执行一次（注意与BeforeClass区别，后者是对于所有方法执行一次）  
@After：释放资源

对于每一个测试方法都要执行一次（注意与AfterClass区别，后者是对于所有方法执行一次）  
@Test：测试方法，在这里可以测试期望异常和超时时间   
@Test(expected=ArithmeticException.class)检查被测方法是否抛出ArithmeticException异常   
@Ignore：忽略的测试方法   
@BeforeClass：针对所有测试，只执行一次，且必须为static void   
@AfterClass：针对所有测试，只执行一次，且必须为static void   
一个JUnit4的单元测试用例执行顺序为：   
@BeforeClass -> @Before -> @Test -> @After -> @AfterClass;   
每一个测试方法的调用顺序为： @Before -> @Test -> @After;

**JUnit的各种断言**

JUnit提供了一些辅助函数，用于帮助你确定某个被测试函数是否工作正常。通常而言，我们把所有这些函数统称为断言。断言是单元测试最基本的组成部分。

1. assertEquals([String message], expected,actual)

比较两个基本类型或对象是否相等（expected和actual是原始类型数值(primitive value)或者必须为实现比较而具有equal方法）；

2.assertFalse([String message], boolean condition)

对布尔值求值，看它是否为“真“；

3.assertTrue([String message], boolean condition)

对布尔值求值，看它是否为“假“；

4.assertNull([String message], java.lang.Object object)

检查对象是否为“空“；

5.assertNotNull([String message],java.lang.Object object)

检查对象是否不为“空”；

6.assertSame([String message],expected,actual)

检查两个对象是否为同一实例；

7.assertNotSame([String message],expected,actual)

检查两个对象是否不为同一实例；

8. fail( String message )

使测试立即失败，其中 message 参数是可选的。这种断言通常被用于标记某个不应该到达的分支（例如，在一个预期发生的异常之后） 。