**练习题**

1. 软件调试的目的是？ A

**A.** 找出错误所在并改正之

**B.** 排除存在错误的可能性

**C.** 对错误性质进行分类

**D.** 统计出错的次数

1. 下列叙述中,哪一项是正确的 ？ D
2. 用黑盒法测试时，测试用例是根据程序内部逻辑设计的;
3. 测试是为了验证该软件已正确地实现了用户的要求;
4. 对面向对象程序来说,单元测试的最小单元是每条程序语句,即以分号结尾的程序;
5. 发现错误多的程序模块，残留在模块中的错误也多。
6. 创建一个基于JUNIT的单元测试类，该类必须扩展? C

**A．**TestSuite  **B.** Assert **C.** TestCase **D.** JFCTestCase

1. 以下对单元测试，不正确的说法是? C

**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。

1. 测试驱动开发的含义是? 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

1. 用JUNIT断言一个方法输出的是指定字符串，应当用的断言方法是? C

**A．**assertNotNull( ) **B.** assertSame()

**C.** assertEquals() **D.** assertNotEquals()

1. TestCase是junit.framework中的一个? C

**A．**方法  **B.** 接口  **C.** 类 **D.** 抽象类

1. TestSuite是JUNIT中用来? A

**A．**集成多个测试用例 **B.** 做系统测试用的  **C.** 做自动化测试用的 **D.** 方法断言

1. 对于测试程序的一些命名规则，以下说法正确的一项是? C

**A．**测试类的命名只要符合Java类的命名规则就可以了；

**B**．测试类的命名一般要求以Test打头，后接类名称，如：TestPerson；

**C**．测试类的命名一般要求以Test结尾，前接类名称，如：PersonTest；

**D**．测试类中的方法都是以testXxx（）形式出现。

1. 通常，初始化一个被测试对象，会在测试类的 ? 中进行。 B

**A．**tearDown()  **B.** setUp() **C.** 构造方法 **D.** 任意位置

1. 以下不属于单元测试优点的一项是? D

**A．**它是一种验证行为 **B.** 它是一种设计行为

**C．**它是一种编写文档的行为  **D.** 它是一种评估行为

1. 从技术角度分，不是一类的测试是? C

**A．**黑盒测试  **B.** 白盒测试 **C.** 单元测试  **D.** 灰盒测试

1. 数据驱动测试也称? C

**A．**单元测试  **B.** 白盒测试 **C.** 黑盒测试  **D.** 确认测试

1. 逻辑驱动测试也称? C

**A．**单元测试  **B.** 灰盒测试  **C.** 白盒测试 **D.** 用户测试

1. 以下不属于白盒测试的优点是? B

**A．**增大代码的覆盖率 **B.** 与软件的内部实现无关

**C．**提高代码的质量 **D.** 发现代码中隐藏的问题

1. 组装测试又称为? A

**A．**集成测试  **B.** 系统测试  **C.** 回归测试  **D.** 确认测试

1. 对于单元测试框架，除了用于Java的JUnit还有CppUnit、NUnit等，它们是? A

**A．**C++单元测试框架、.NET单元测试框架 **B.** C语言单元测试框架、通用单元测试框架

**C．**C++单元测试框架、自动化单元测试框架 **D.** 自动化单元测试框架、.NET单元测试框架

1. 对于JFCUnit，以下说法不正确的是? D

**A**. 它是JAVA GUI的测试框架

**B.** 它是JUnit的扩展，用于GUI的测试

**C**． 编写JFCUnit的测试用例需要扩展JFCTestCase

**D.** JFCUnit是Java GUI测试框架和JUnit没有任何关系

1. 以下哪一项不属于WEB测试的范畴? F

**A．**功能  **B.** 性能  **C.** 可用性

**D.** 安全性 **E.** 客户端兼容性  **F.** 可扩展性

1. 连接速度测试属于? C

**A．**功能测试  **B.** 客户端兼容性测试  **C.** 性能测试 **D.** 安全测试

1. LoadRunner是? A

**A．**工业标准级负载测试工具  **B.** 一款下载软件

**C.** 单元测试框架  **D.** 集成开发工具

1. JUnit的特征，不正确的一项是? D

**A．**用于测试期望结果的断言

**B.** 用于共享共同测试数据的测试工具

**C．**易于集成到测试人员的构建过程中,Jnuit和Ant的结合可以实施增量开发

**D．**JNuit是收费的，不能做二次开发

1. JUnit的两个模式：集成模式和? A

**A．**命令模式  **B.** 适配器模式  **C.** 单例模式  **D.** 接口模式

1. 测试6的阶乘，断言方法是? B

**A**．Assert.assertSame(720,jc.jieChen(6))

**B.** Assert.assertEquals(720,jc.jieChen(6))

**C．**Assert.assertNull(720,jc.jieChen(6))

**D.** Assert.assertTrue(720,jc.jieChen(6))

1. 什么是软件测试？

答案1：软件测试是为了发现错误而执行程序的过程。答案2：软件测试是根据软件开发各阶段的规格说明和程序的内部结构而精心设计的一批测试用例（即输入数据及其预期的输出结果），并利用这些测试用例运行程序，以及发现错误的过程。

1. 软件测试的分类？

**1.从技术角度分：**

黑盒测试 白盒测试 灰盒测试 ALAC测试

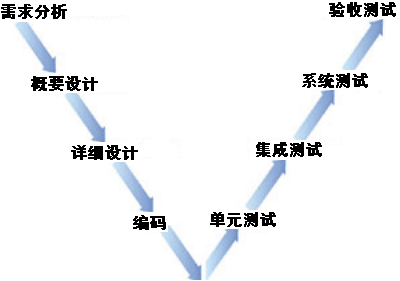
**2.从执行阶段角度分：**

单元测试 集成测试 确认测试 系统测试

1. TDD的中文全称是什么？它的思想是什么？

TDD中文是测试驱动开发，它是一种以“测试先行”为原则的开发方法，开发人员在编写产品代码前，通常先写出对应的测试程序，后在编写产品代码，然后再进行测试。

1. 请解释下图的含义



这是测试模型中的“V”模型描述了一些不同的测试级别，并说明了这些级别所对应的生命周期中不同的阶段。如模型图中所示，左边下降的是开发过程各阶段，与此相对应的是右边上升的部分，即各测试过程的各个阶段。

1. 单元测试的概念是什么?

答案1：单元测试是开发者编写的一小段代码，用于检验被测代码的一个很小的、很明确的功能是否正确。

答案2：单元测试是由程序员自己来完成，最终受益的也是程序员自己。可以这么说，程序员有责任编写功能代码，同时也就有责任为自己的代码编写单元测试。执行单元测试，就是为了证明这段代码的行为和我们期望的一致。

1. 编写一个测试输出为“你好！南工院”的测试案例

**要求：以show()作为字符串的打印方法。**

**要点提示：**测试程序中要用断言assertEquals(“你好！南工院”,XXX.show())或Assert. assertEquals(“你好！南工院”,XXX.show()).

**注意：**测试类命名和测试方法命名（类命名应以Test结尾，测试方法命名应以test打头）

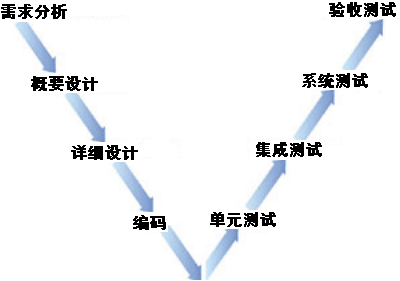
1. WEB测试主要围绕哪几个方面？

功能 性能 可用性 客户端兼容性 安全性

1. 什么是压力测试？

进行压力测试是指实际破坏一个Web应用系统，测试系统的反映。压力测试是测试系统的限制和故障恢复能力，也就是测试Web应用系统会不会崩溃，在什么情况下会崩溃。黑客常常提供错误的数据负载，直到Web应用系统崩溃，接着当系统重新启动时获得存取权。

1. 请画出测试模型中“V”模型的结构图？

****

1. 怎么编写测试用例？

1.模拟正常的使用过程

2.分支覆盖，路径覆盖

3.真实数据，边界数据

4.语句数据简单，容易理解

5.避免依赖

6.记录日志

**上机练习：**

1. 编写一个N的阶乘程序，并对其进行测试。

要求：1.正确、完整的程序代码和测试代码

2.基于JUNIT单元测试框架

3.代码结构良好、可读性强。

**N阶乘程序：**

**import java.math.BigInteger;**

**import java.util.Scanner;**

**/\*\***

**@author daiyongming**

**\***

**\*/**

**public class JieChen {**

**long result = 1;**

**long doJieChen(int a){**

**for(int i = 2; i <= a; i++){**

**result\*=i;**

**}**

**return result;**

**}**

**public static void main(String args []){**

**JieChen jc = new JieChen();**

**Scanner c = new Scanner(System.in);**

**System.out.print(jc.doJieChen(c.nextInt()));**

**}**

**}**

**测试N阶乘程序：**

**import junit.framework.TestCase;**

**/\*\***

**\* @author daiyongming**

**\***

**\*/**

**public class JieChenTest extends TestCase {**

**/\* (non-Javadoc)**

**\* @see junit.framework.TestCase#setUp()**

**\*/**

**JieChen j;**

**protected void setUp() throws Exception {**

**super.setUp();**

**j = new JieChen();**

**}**

**public void testDoJieChen(){**

**assertEquals("这是测试阶乘的值:",24,j.doJieChen(4));**

**}**

**}**

1. 实现一个将字符串全部变成小写的转换器，并对其进行测试

要求：1.正确、完整的程序代码和测试代码

2.基于JUNIT单元测试框架

3.代码结构良好。

**转换器程序：**

package demo;

public class StringDemo {

public StringDemo() {

}

public String smallString(String str)// 字符串变小写

{

String temp = "error";

if (str.equals("") || str == null) {

return temp;

}

String s = str.toLowerCase();

return s;

}

}

**测试程序：**

package demo.test;

import demo.StringDemo;

import junit.framework.TestCase;

public class testStringDemo extends TestCase {

private StringDemo str;

protected void setUp(){

str = new StringDemo();

}

public void testSmallString() {

assertEquals("一个字母变小写",str.smallString("A"),"a");

assertEquals("字符串全是大写",str.smallString("ABC"), "abc");

assertEquals("含有小写的字符串",str.smallString("aBc"), "abc");

assertEquals("含有数字的字符串",str.smallString("A1C"), "a1c");

assertEquals("全是数字的字符串",str.smallString("123"), "123");

assertEquals("含有特殊字符的处理",str.smallString(",Adc"), ",adc");

assertEquals("异常处理",str.smallString(""),"error");

}

}

1. 翻译

UML is an object-oriented modeling language, object-oriented rather than a method of modeling, it is only a model for the elements and symbols and that their definition of semantics, do not involve how to System modeling, the process itself did not provide the concept, which means that users in the use of UML modeling, you can use any appropriate method and process. Selection process with the model / software development process of different factors, such as software developed by the type (such as real-time systems, information systems and desktop products) and the development of the objective. Users will be selected according to different needs of different process. However, the use of UML modeling still has a largely unified framework for the process. The framework includes UML modeling in the process of common elements, but also for users with their choice of projects for the development of modeling technology to provide a great degree of freedom. From the ability of view, UML is very strong, from its participation in the Joint object-oriented method of drawing a lot of expansion of the concept and plans. But precisely because of this, UML also the complexity of the various far beyond the previous object-oriented approach, as well as its use of modeling elements to achieve more than 100, the development of a system to draw the plans of as many as nine.

UML是一种面向对象的建模语言，而不是一种面向对象的建模方法，它只是给出一套用于建模的元素及表示符号并定义了它们的语义，而不涉及如何进行系统建模，它本身没有提供过程的概念，这意味着用户在使用UML进行建模时，可以选用任何适合的方法和过程。过程的选用与模型/软件开发过程的不同因素有关，诸如所开发软件的种类(如实时系统、信息系统和桌面产品)和开发的目的等。用户将根据不同的需要选用不同的过程。然而，使用UML建模仍然有着大致统一的过程框架。该框架包含了UML建模过程中的共同要素，同时又为用户选用与其所开发的工程相适合的建模技术提供了很大的自由度。从表达能力来看，UML无疑是很强的，它从参加联合的各种面向对象方法中吸取了许多扩充的概念和图性。但恰恰因为这一点，UML的复杂性也远远超出了以往各种面向对象方法，以至它所使用的建模元素达到一百多个，开发一个系统要画的图达9种之多。