面向对象与protected计算机等级考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/645/2021_2022__E9_9D_A2_E5_90_91_E5_AF_B9_E8_c97_645250.htm 戏言面向对象 说到protected这个词，我不可避免的就会想到一个概念面向对象。那么什么是面向对象呢？其实我个人认为面向对象这个概念是一直在发展变化的，到了今天，面向对象这个词也许让它叫做面向抽象更加贴切。在刚刚建立面向对象这个概念的时候，大概连创造者对于到底什么是面向对象都不是很清楚。要搞清楚面向对象（编程，或者设计）是什么，也许得看看过去的软件代码都是什么样的。 I.公元前 软件开发在最初的十几二十年里面，基本上就是面向过程的。面向过程的核心内容有两项，一个是控制流，另外一个就是数据流。在这一个时期里面，软件界最大的发展估计是数据结构与算法这两个“ 科目 ”了，这两者分别对应着两个“ 流 ”。在面向过程的软件代码里面，执行主体是过程或者函数。一个过程所代表的就是一个动作，动作的对象（这里还不是面向对象的对象）是一些数据，数据也许通过参数得到，也许通过全局变量得到，还有一些常量或者预定义值。如果我们仔细想一下，就会发现这是一个“ 动宾 ”结构的体系，比如说Basic里面比较著名的“ Line (x1, y1) -(x2, y2) ”，翻译成自然语言就是“ 画一条(x1,y1)到(x2,y2)的直线 ”。类似的例子还有很多，比如C语言里面的“ printf("%s\r\n", "Hello world!"). ”。 可是主语在哪里？ II.创世纪 面向过程的代码里面并没有突出一个主语，很多时候这个主语也并非不存在，就像上面的例子里面，主语就是一个屏幕。可是如果我们需要往打印机里面画

一条直线呢？（或者打印一个"Hello world"。）在面向过程的代码里面，我们就不得不自己写一个PrintLine的函数。（C语言往文件里面些东西就是fprintf。）如果我们要往远程设备上画一条直线，那还要写一个RemoteLine，如果……不需要我多说，您也会觉得麻烦。围绕着这样一个问题，人们就开始思考：是否能够把主语明确的给写出来？是否能够让我们少做一点重复性的工作？后来就有了面向对象这个东西，在面向对象是一个“ 主谓宾 ”结构的世界，绝大多数东西都有一个主语，比如我们所熟悉的“ g.DrawLine(pen, pt1, pt2). ”，由于我们有了“ 主语 ”，我们就可以让不同的东西，用相似的方法做相似的事情。如果光是把g换成h，仅仅解决了“ 在这个窗口画 ”与“ 在那个窗口画 ”的问题，如果我们希望他能够在其他类型的空间上画，我们还需要容许主语的类型可以不完全相同。但我们要解决的更多问题还是概念相同之处，例如打印机的g和屏幕的g都能够画线，因此有了诸如继承、封装等概念。这就是面向对象的一切了吗？ III.改革开放 随着面向对象概念的诞生，春风沐浴大地。正如上帝说要有光，于是有了光。上帝说要有毒蛇，于是有了毒蛇，上帝说要有苹果，于是有了苹果，结果亚当和夏娃吃了这个上帝创造出来的苹果受到了上帝的“ 惩罚 ”。真不明白，既然上帝不希望亚当和夏娃吃这个苹果，为什么还要创造这么一个东西？其实上帝创造这个苹果当然是不希望他们“ 吃 ”这个苹果，创造这个苹果实际上是为了产生浪漫的爱情以及其后千秋万代的动人故事。如果你把这个苹果仅仅看成是吃的，那么接下来你看到的就是痛苦的惩罚。如果你看到的是背后动人的故事，那么浪漫甜蜜等美好之辞就会充满你的大脑。 面向对

象也一样，他的核心意义并不在于你把东西封装成什么样了，不在于有什么东西被继承出来了，最重要的是他容许我们用抽象的方式来构建一个软件。比如当我们写代码写到：stream.Write(buff, 4, buff.Length - 4). 或者 hashbuff = hasher.ComputeHash(buff). 我们是否需要关心stream到底是什么，hasher用的又是什么算法呢？如果我们由始至终，在做相应的东西的操作都用相同的stream对象和hasher对象，任务是否都应当能够正确完成呢？应该是能够正确完成的，因为这正是我们的期待。如果让我们来设计某一个stream，是否应该从这个角度去考虑如何设计这一个类呢？如果我们定义这个stream变量，是否应该更抽象一点呢？考虑这么一个函数：void DoSomething(FileStream stream, MD5CryptoServiceProvider hasher, byte[] buff) {...} 如果写成如下形式将会更加灵活，也更加符合面向对象（面向抽象）的真实含义： void DoSomething(Stream stream, HashAlgorithm hasher, byte[] buff) {...} 换句话说，所有的封装、继承、接口等等，实际上是为了提供抽象能力而存在的。如果我们把protected当作保护“某些方法的存在”这个秘密的话，那就大错特错了。保护这些秘密严格说来应该是密码学的职责，而不是面向对象的职责。百考试题－全国最大教育类网站(100test.com) IV.回顾历史 面向对象的核心是面向抽象，但我们看到，实际发展的过程并非如此。我们在过去有着太多错误的概念了，比如说这个面向对象技术的面向对象，就太容易让我们认为，这项技术的核心就是面向对象。于是很多时候我们写一个“面向对象”的程序充斥的过度的对象，泛滥的继承，以及不知道为什么的封装。并且不少开发者，包括我在内，都曾经认为所

谓的面向对象就是把一些要素抽象成对象，进行封装，然后从某个基类派生出万物。好比有一个基类叫做物体，派生出活物与死物，活物派生出细菌病毒植物动物，动物里面有猴鸡狗猪和人，人里面有张三李四王二麻子（还有个娃）。 没错，面向对象当然得包括这些，但是这不是全部，更不是根本。根本就是在于我们写某些东西的时候，不需要关心具体的对象是什么，只需要知道至少它应该是一个什么。比如上一节当中的例子，DoSomething只需要知道stream是一个流，而hasher是一个哈希算法提供者就够了。至于具体提供的是什么样的流和哈希算法，则不应当是我们关心的，而是使用我们这段代码的用户所关心的。如此一来，我们就可以在设计这一段我们所关心的功能的时候，不需要考虑过多的、过于具体的、不断变化的问题。来源：考试大 仔细想想，我们是否真的已经明白了面向对象的核心所在呢？ V.封装保护的是什么 面向对象的封装并非保护你的秘密，而是防止被错误使用，是为了明确划分问题的界限。就“ 保护 ”这个词而言，更进一步的讲，它并非对使用该对象的用户（下面称为用户）做出使用某个成员的授权，而是对延展该类的设计人员（下面称为设计人员）做出延展问题领域的授权。现在让我们回过头来看一下wayfarer所写的例子： class Base { protected void Print() { Console.Write("This is protected method in Base Class!"). } } class Derived:Base { public new void Print() { base.Print(). } } class OtherClass 编辑特别推荐: 计算机等级考试二级JAVA考前密卷及参考答案 全国计算机等级考试二级JAVA全真模拟试题 全国计算机等级考试二级JAVA在线模拟考试 计算机等级考试二级Java初级练习(精选25题) JAVA技术学习法 Java学习如何走

出第一步 最值得学习的五种JAVA技术 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com