Java中的语句、分支和路径覆盖测试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/213/2021_2022_Java_E4_B8_AD_E7_9A_84_c104_213757.htm 简介 代码覆盖是一种用来度量已执行的软件测试水平的方法。收集覆盖度量数据的过程很简单：监测您的代码，并对所监测的版本运行测试。这样就可以生成相关数据，展示已执行哪些代码，或者更重要的是，未执行哪些代码。覆盖测试是对单元测试的完美补充：单元测试可以显示出是否代码按预期执行，而代码覆盖可以表明还需要对哪些代码进行测试。 大多数开发人员都能理解这一过程，也赞同其价值主张，他们通常追求100%的覆盖率。尽管100%的覆盖率是个极好的目标，但类型不当的100%覆盖率依然会留下未知的问题。典型的软件开发是根据要测试的语句或分支的数量来度量覆盖率的。即便有着100%的语句或分支覆盖率，代码逻辑依然可能存在严重的逻辑bug，只能为开发人员和管理员带来虚假的安全感。 为何100%的覆盖率还不够？这是因为语句和分支覆盖率无法表明代码中的逻辑是否已执行。语句和分支覆盖对于未执行代码块中出现的明显问题来说很有用，但它们经常会错过与决策结构和决策交互相关的bug。而另一方面，路径覆盖则是一种有助于及早发现缺陷的更为健壮和全面的技术。 在了解路径覆盖之前，先关注一下语句和分支覆盖率方面的一些问题： 语句覆盖 语句覆盖可以识别在一个方法或类中执行了哪些语句。这是一种计算起来比较简单的量规，现在有许多 开源产品 都可以评测这种覆盖级别。最终，语句覆盖的获益在于其能够识别未执行代码块。然而语句覆盖也存在问题，其无法识别源代码中

控制流结构导致的bug，例如复合条件或者连续开关标签。这意味着在您可以轻松获得100%覆盖率的同时，未发现的明显bug依然存在。 下例中显示了这种问题。此处，returnInput()方法由七条语句组成，它有一个简单的需求：输出应等于输入。 图1. 代码示例 接下来，您可以创建一个满足需求并且具有100%语句覆盖率的JUnit测试用例。 图2. 语句覆盖 在 returnInput() 中有一个明显的bug。如果第一个或第二个决策计算为真而其他的计算为假，返回值则不等于该方法的输入值。精明的软件开发人员会立即注意到这个问题，但语句覆盖报告却显示为100%的覆盖率。如果管理员发现覆盖率为100%，他/她可能会受到虚假的安全感的影响，判定测试已经完成，继而发布错误百出的代码，将之投入生产。 仅仅认识语句覆盖是不够的，开发人员必须进一步使用更为完善的测试技术：分支覆盖。 