Java理论与实践:平衡测试，第2部分（二） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022_Java_E7_90_86_E8_AE_BA_c104_145096.htm 编写 RuntimeException 捕获检测器 正如您在 上一期 中所学的，编写 bug 模式的第一个步骤是清楚地标识 bug 模式。在这里，bug 模式是捕获 Exception 的 catch 块，这时不存在用于 RuntimeException 的相应捕获块，并且尝试块中的任何方法调用或 throw 语句都不会抛出 Exception。要检测此 bug 模式，则需要知道 try-catch 块的位置、try 块可能抛出的内容以及在 catch 块中将捕获的内容。 标识捕获的异常 像上个月的操作一样，您可以通过创建 BytecodeScanningDetector 基础类（可实现 Visitor 模式）的子类启动 bug 检测器。在 BytecodeScanningDetector 中有一个 visit(Code) 方法，并且在每次发现 catch 块时，该实现都会调用 visit(CodeException)。如果重写 visit(Code)，并从那里调用 super.visit(Code)，则当超类 visit(Code) 返回时，它将调用用于该方法中所有 catch 块的 visit(CodeException)。清单 4 了显示实现 visit(Code) 和 visit(CodeException) 的第一步，它将积累方法中所有 catch 块的信息。每个 CodeException 都包含相应 try 块的起始和终止的字节码偏移量，这样您可以方便地确定哪一个 CodeException 对象与 try-catch 块对应。 清单 4. 第一版 RuntimeException 捕获检测器可以收集某一方法中抛出的异常信息public class RuntimeExceptionCapture extends BytecodeScanningDetector { private BugReporter bugReporter. private Method method. private OpcodeStack stack = new OpcodeStack(). private List catchList. private List throwList. public

void visitMethod(Method method) { this.method = method. super.visitMethod(method) } public void visitCode(Code obj) { catchList = new ArrayList(). throwList = new ArrayList(). stack.resetForMethodEntry(this). super.visitCode(obj). // At this point, weve identified all the catch blocks // More to come... } public void visit(CodeException obj) { super.visit(obj). int type = obj.getCatchType(). if (type == 0) return. String name = getConstantPool().constantToString(getConstantPool().getConstant(type)). ExceptionCaught caughtException =new ExceptionCaught(name, obj.getStartPC(), obj.getEndPC(), obj.getHandlerPC()). catchList.add(caughtException). }} 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