

SHANGHAI JIAO TONG UNIVERSITY



课程名称： 高级软件测试

学生姓名/学号: 杨铭/117037910041

学生姓名/学号: 翟煜/117037910003

学生姓名/学号: 华逸君/117037910061

学生姓名/学号: 韩易忱/117037910058

专 业: 软件工程

指导教师: 姚建国

学院(系): 电子信息与电气工程学院

1. 概述
2. 编写目的

本文档为高级软件测试课程第二次作业的测试报告和工具说明，为手工变异测试的测试结果进行总结，并且对自动化变异测试工具进行介绍

1. 测试范围

Calculate类和Transformer类中的所有函数

1. 开发环境和运行需求

开发环境

|  |  |
| --- | --- |
| 环境 | 名称 |
| 操作系统 | windows10 |
| 开发框架 | NodeJS |
| 开发语言 | Javascript/css/html |
| 图形界面 | Electron |
| 前端框架 | VueJS |
| css库 | Iview |

运行需求

|  |  |
| --- | --- |
| 环境 | 名称 |
| 操作系统 | windows7/10,Linux,MacOS |

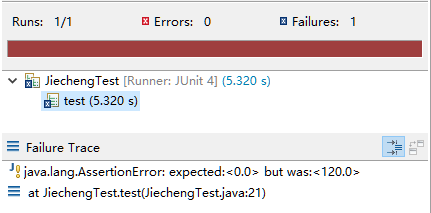
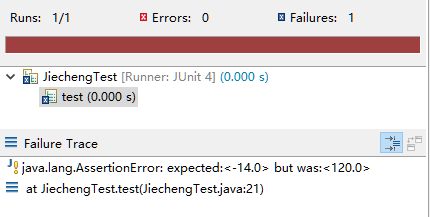
1. 人工错误注入
2. 注入方法

我们对Calculate和Transformer的每个函数手工注入了5个错误（Calculate中的exp较为简单，只注入了3个错误），生成相应的变异体，并且对注入的错误进行了标注。再使用作业一的用例对这些变异体进行测试

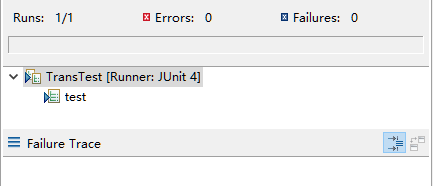
1. 结果报告

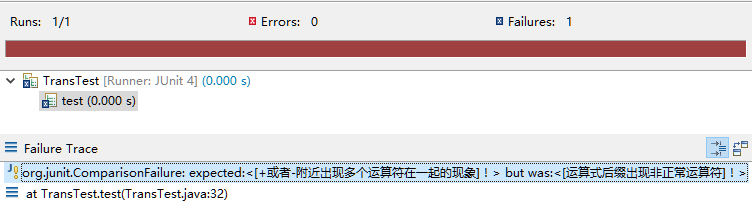
作业一中的用例杀死了所有的变异体，证明作业一中的用例较为全面。部分结果截图如下

jiecheng函数变异体测试结果

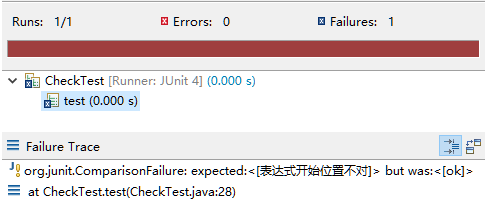
 

trans函数变异体测试结果





Check函数变异体测试结果



1. 自动化变异工具
2. 简介

我们使用了NodeJS和Electron框架进行自动化变异工具的开发，工程的文件目录如下

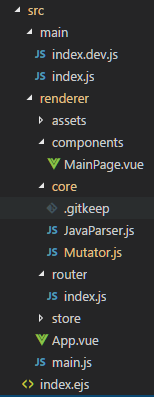


图 1.开发文件目录图

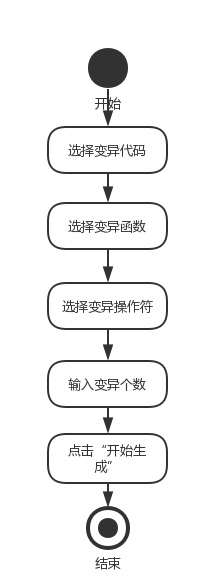
其中主要逻辑和页面在\render文件夹中。\render\core中的JavaParser.js负责对读取的.java文件进行简单的语法解析，并记录所有的变异操作符。Mutator.js则负责选择JavaParser记录的变异操作符进行变异，生成变异体文件。页面的布局和用户交互逻辑则在\components文件夹中的MainPage.vue文件中。

1. 支持变异操作符

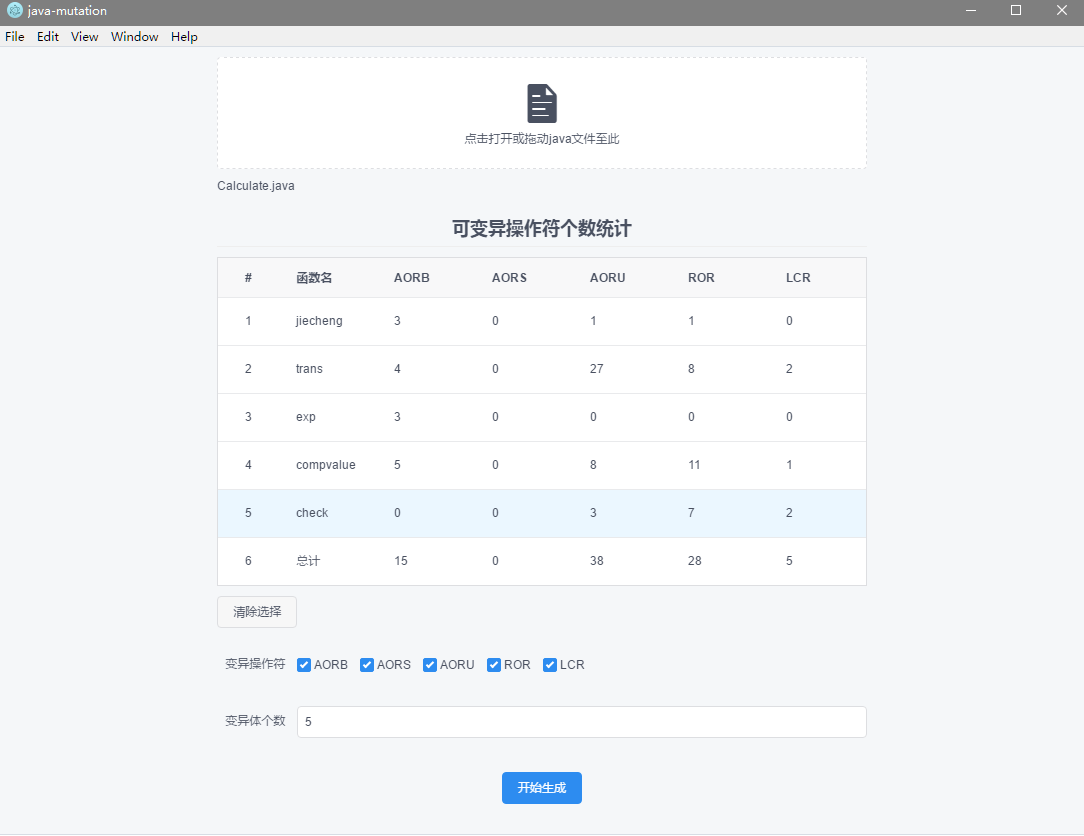
我们参考了Offutt A J的五个变异操作符选择策略[1]，并进一步精简为3大类操作符：AOR,ROR,LCR，其中AOR可以进一步被分为AORB,AORS,AORU三种。变异操作符的缩略语见下表

|  |  |  |
| --- | --- | --- |
| 缩略语 | 定义 | 例子 |
| AORB | 二元算数运算符 | a+b |
| AORS | 算数运算符缩写 | a+=b |
| AORU | 一元算数运算符 | a++ |
| ROR | 关系运算符 | a>b |
| LCR | 逻辑运算符 | a&&b |

1. 使用流程



1. 运行截图



1. 引用

[1] Offutt A J. An experimental determination of sufficient mutant operators[J]. Acm Transactions on Software Engineering & Methodology, 1996, 5(2):99-118.