软件测试与质量保证

课程作业报告

北京理工大学计算机学院

指导老师：刘辉

小组成员：

1120172169 杨训迪

1120170121 黄舒心

1120171197 孙浩辰

1120171210 张帜航

2020-01-18

一、 实现思路

1. 爬虫思路

小组讨论后决定选取acm代码为实验数据来源。经过对acm网站的了解，最终选取codeforces作为爬取代码和用例的来源，使用Python完成爬虫代码。

首先获取题解代码，根据codeforces题目列表的网址规律，如第一页为https://codeforces.com/problemset/page/1，以便获取题目列表网页html并解析，此后页数仅改动最后一个数字即可。解析每一页题目列表时，利用soup找到title=”Participants solved the problem”，的部分，获取各个题目提交列表网址并保存，然后根据获取到的网址访问各个提交列表。这里需要注意的是，我们仅需要Java8（语言选择中Java有Java8和Java11，由于本次实验比较Java7与Java8，故选择Java8）语言的题解代码，这里我们使用到了selenium模拟浏览器表达提交，完成仅选择Java8题解功能。在某个题目的Java题解列表中选取第一个，获取其ID，方便之后对具体代码的获取。如https://codeforces.com/problemset/submission/1288/68827182， 这是题目集1288的某个题解代码网址，最后的68827182即为题解代码ID，所以获取某个题目的题解ID后，便可以很容易的得到题解代码网址。

然后还需获取测试用例，这可以在具体题目网页获取。由于测试用例部分有特定的标签类名“inoput”和“output”，利用soup可以容易地获取到。

将爬取数据以统一格式保存，并清洗不完整文件夹，爬取数据部分完成。

2. 插桩代码实现思路

(1) SrcFileSearch.java说明

首先，通过Python爬取Java7和Java8源码下的所有.java文件，并将文件名分别保存在jre1.7.json和jre1.8.json下。

第二，提供文件名的字符串，调用SrcFileSearch.java，在当前JRE环境对应的Json文件中进行搜索，存在则返回true,反之false。

(2) Parser.java说明

首先，调用StaticJavaParser.parse()解析输入源码，生成语义树。

第二，存取类中所有字段。两个字符串数组，reference[]存取对象所对应的类名，variable[]存取对象名。具体实现过程为遍历所有的类型声明，判断是否为字段申明。若是，则获取字段的类型(调用FieldDeclaration.getElementType())，转为字符串，调用SrcFileSearch类，若返回值为true，则将字段类型存入数组reference，字段名(通过

FieldDeclaration. getVariables()获取)存入数组 variable。

第三，处理语法树的各个节点。主要需要处理的节点有四种：赋值访问，变量定义，函数定义，函数返回值。对于“赋值访问”，首先判断是否为方法调用表达式(MethodCallExpr)，若是，则获取该节点下的调用类名，方法名，返回值，调用第四点，判断是否为方法调用；对于“变量定义”，需要处理两种情况，第一种，无方法调用，第二种，存在方法调用。对于第一种情况，获取变量类型，变量名，调用 SrcFileSearch 类，若返回值为 true，则将变量类型存入数组 reference，变量名存入数组 variable。对于第二种情况，调用 VariableDeclarator.getInitializer() 获取右值，判断类型是否为 MethodCallExpr，若是，则获取该节点下的调用类名，方法名，返回值，调用第四点，判断是否为方法调用；对于“函数定义”，获取形式参数类型，形式参数名，调用 SrcFileSearch 类，若返回值为 true，则将形式参数类型存入数组 reference，形式参数名存入数组 variable；对于“函数返回值”，调用 ReturnStmt. getExpression() 获取右值，判断类型是否为 MethodCallExpr，若是，则获取该节点下的调用类名，方法名，返回值，调用第四点，判断是否为方法调用，并标记此次调用为“函数返回值”情况。

第四，判断是否为方法调用。首先，对于传来的类名，区分是否为真实的类名还是对象名，若为对象名，遍历数组 variable[],获取真实类名，调用 SrcFileSearch 类，若返回值为 false，则返回，否则，将类名置换为真实类名。此时可以确定该句应为插桩点，将此时的插桩点对应的所有的插桩语句存入数组 stubLine[]，插桩点的行数存入数组 stubLineNum，特殊地，若为“函数返回值”情况调用该函数，即标记 addToFront 为 true，则还需将插桩点的前一行的行数存入数组 stubLineNumFront[]。

第五，插桩。调用 BufferedReader.reaLine(),一行行读取文件中的内容，若此时读取行数为数组 stubLineNum[]的行数，则将对应数组 stubLine[]插入该行的末尾；若此时读取行数为数组 stubLineNumFront []的行数，则将对应数组 stubLine[]插入该行的前面。调用 BufferedWriter.write(),一行行写入新文件中。

3. 运行代码思路

对于代码的批量处理与运行，采取如下方式实现：

(1) 首先，递归遍历所给路径下所有文件夹，获取相应文件。

若获取到的内容为文件夹，则进行下一步递归；

若获取到的内容为文件，首先判断其文件类型；当为 java 源代码，则将代码重命名为 solution+题号.java 形式，并调用 parser 接口实现代码的插桩。

同时，将相应的管道命令“java solutionXX.java < solutionXX_input.txt > solutionXX_output.txt”形式写入 bat 脚本文件，以实现代码的批量编译运行。因为文件名的修改，进入代码中将相应代码的类名改成与文件名一致，并统一移至 code_init 文件夹中；当为输入的 txt 文件，将源文件移动至源码文件所在目录下。

(2) **获取爬虫所得源代码后，运行 GenerateTestCase 类，实现如上所述的代码插桩和移动；**

(3) **完成后进入相应目录，打开 cmd，在不同环境下将 bat 文件拖入后即可运行生成 output 文件。**

（针对一些自动插桩代码，存在插桩错误，所以需进行手动插桩）

4. 比较输出结果

每次读取一行存储为字符串，识别字符串中含有“Call method”的字符串时存储下一行为方法名。识别字符串中含有“Return”的情况下，拼接之后到 The Line 之前的每一行。然后进行比较。如果不同就存储为 DiffDescribe，然后存储到 List 中，最后输出。

二、 实验结果概述

本实验通过运行 tem 和 SoftwareTest 文件夹下的代码，获得最终 java7 和 java8 的比较结果(output_diff.txt)，通过对相应的代码进行分析，获取 14 个 java7 和 java8 输入相同，输出结果不同的接口。有些接口没有返回值，但本组经过分析发现其对于最终输出结果产生了影响，所以这些接口也相应的包含在其中。 14 个接口分别为： String.split()、 Map.put()、 DecimalFormat.format()、 Class.getDeclaredMethods()、 Set.add()、 String(bytes[],Charset)、 NumberFormat.format()、 Method.getMethods()、 Matcher.replaceAll()、 ZipEntry.setTime()、 Locale.getDisplayScript()、 SimpleDateFormat.parse()、 Iterator.remove()、String(byte bytes[], String charsetName)

本实验，提取与最终比较结果所对应的中间结果，存至./output/bug。

三、 差异函数具体说明

1. String.split()

(1) 类名:String

(2) 方法名:split

(3) 输入:abc

(4) Java7 输出:

a
b
c

(5) Java8 输出:

a
b
c

(6) 注释:

1. SplitTest.java Line7

2. Map.put()

(1) 类名:Map

(2) 方法名:put

(3) 输入:

VAR1=var
VAR2=var1

(4) Java7 输出:{VAR=var1, VAR1=var}

(5) Java8 输出:{VAR1=var, VAR=var1}

(6) 注释:

1. HashMapTest.java Line11-12
2. Map.put()本身没有返回值，本组选择通过 System.out.println(map)将其打印

3. DecimalFormat.format()

(1) 类名:DecimalFormat

(2) 方法名:format

(3) 输入:83.65

(4) Java7 输出:83.6

(5) Java8 输出:83.7

(6) 注释:

1. DecimalFormatTest.java Line12

4. Class.getDeclaredMethods()

(1) 类名:Class
(2) 方法名:getDeclaredMethods
(3) 输入:interface GetDeclaredMethods$Derived
(4) Java7 输出:
public abstract
GetDeclaredMethods$Derived$DerivedFooGetDeclaredMethods$Derived.f
oo() //一行
(5) Java8 输出:
public abstract GetDeclaredMethods$Derived$DerivedFoo
GetDeclaredMethods$Derived.foo() //第一行
public default GetDeclaredMethods$Base$BaseFoo
GetDeclaredMethods$Derived.foo() //第二行
(6) 注释:
1. GetDeclaredMethods.java Line23

5. Set.add()

(1) 类名:Set
(2) 方法名:add
(3) 输入:qwe rtz 1234 123121
(4) Java7 输出:[rtz, , 1234, qwe, 123121]
(5) Java8 输出:[, 1234, rtz, 123121, qwe]
(6) 注释:
1. HashSetTest.java Line11-16
2. Set.add()本身没有返回值，本组选择通过
System.out.println(stringSet)将其打印

6. String(bytes[],Charset)

(1) 类名:String
(2) 方法名:String
(3) 输入：new String(b, StandardCharsets.UTF_8)，其中 b = [-5, -122, -28]
(4) Java7 输出:?
(5) Java8 输出:???
(6) 注释:
1. Java87String.java Line7

7. NumberFormat.format()

(1) 类名: NumberFormat

(2) 方法名：format

(3) 输入：83.65

(4) Java7 输出：83.6

(5) Jav8 输出：83.7

(6) 注释:

1. NumberFormatTest.java Line13
2. 测试前提是将 NumberFormat 类实例的 MinimumFractionDigits 和 MaximumFractionDigits 设置为 1

8. Method.getMethods()

(1) 类名：Method

(2) 方法名：getMethods()

(3) 输入：无

(4) Java7 输出：

```
public abstract GetMethodsTest$Derived$DerivedFoo
GetMethodsTest$Derived.foo()//第一行
public abstract GetMethodsTest$Base$BaseFoo
GetMethodsTest$Base.foo() //第二行
```

(5) Java8 输出：

```
public abstract GetMethodsTest$Derived$DerivedFoo
GetMethodsTest$Derived.foo()//第一行
public default GetMethodsTest$Base$BaseFoo
GetMethodsTest$Derived.foo() //第二行
```

(6) 注释：

1. GetMethodsTest.java Line25

9. Matcher.replaceAll()

(1) 类名：Matcher

(2) 方法名：replaceAll()

(3) 输入：""

(4) Java7 输出：Aa123 \/*-.ac!e('"][{

(5) Java8 输出：Aa123 \/*-.ac!e§('"][{

(6) 注释：

1. PatterMatcherTest.java Line17。
2. Java7和Java8区别是Java8结果包含§而Java7不包含。例子中matcher内容为"Aa123 \/*-+.ac!e§('"~'][{^"，Pattern.compile内容为"[^a-zA-Z0-9\\p{P}\\s]*"

10. ZipEntry.setTime()

(1) 类名：ZipEntry

(2) 方法名：setTime()

(3) 输入：10

(4) Java7输出：无返回值，调用getTime()得到315504000000

(5) Java8输出：无返回值，调用getTime()得到10

(6) 注释：

1. ZipEntryTest.java Line15

11.Locale.getDisplayScript()

(1) 类名：Locale

(2) 方法名：getDisplayScript

(3) 输入：无

(4) Java7输出：Simplified Han

(5) Java8输出：简体中文

(6) 注释：

1. LocalTest.java Line25

12.SimpleDateFormat.parse()

(1) 类名：SimpleDateFormat

(2) 方法名：parse

(3) 输入：01/01/ 12

(4) Java7输出：With Space: 01/01/0012

(5) Java8输出：With Space: 01/01/2012

(6) 注释：

1. TwoDigitYearWithSpaceBug.java Line19

13. Iterator.remove()

(1) 类名：Iterator

(2) 方法名：remove

(3) 输入：无

(4) Java7输出：

keys.get: c

key: c

keys.get: b

key: b

keys.get: a

key: c

(5) Java8输出：

keys.get: c

key: c

keys.get: b

key: b

keys.get: a

key: a

(6) 注释：

1. TreeMapIteratorBug.java Line23
2. 方法本身无返回值，调用方法后，指针发生变化，此处将调用函数后值的变化作为输出

14. String(byte bytes[], String charsetName)

(1) 类名：String

(2) 方法名：String

(3) 输入：

[-4 -128 -128 -113 -65 -65]

UTF8

(4) Java7输出：fffd

(5) Java8输出：fffd fffd fffd fffd fffd fffd

(6) 注释：

1. RegTest.java Line7