Example and analysis: a codified-MR-based test has higher test coverage than a developer-written test

Project: alibaba/one-java-agent

Class under test: com.alibaba.oneagent.utils.FeatureCodec, which is available at <https://github.com/alibaba/one-java-agent/blob/1f399a2299a8a409d15ea6111a7098629b8f1050/one-java-agent-plugin/src/main/java/com/alibaba/oneagent/utils/FeatureCodec.java>

* Original developer-written test case: com.alibaba.oneagent.utils.FeatureCodecTest.test(), which is available at https://github.com/alibaba/one-java-agent/blob/1f399a2299a8a409d15ea6111a7098629b8f1050/one-java-agent-plugin/src/test/java/com/alibaba/oneagent/utils/FeatureCodecTest.java. (or see figure 1)
* Line coverage: **0.7857142857142857**

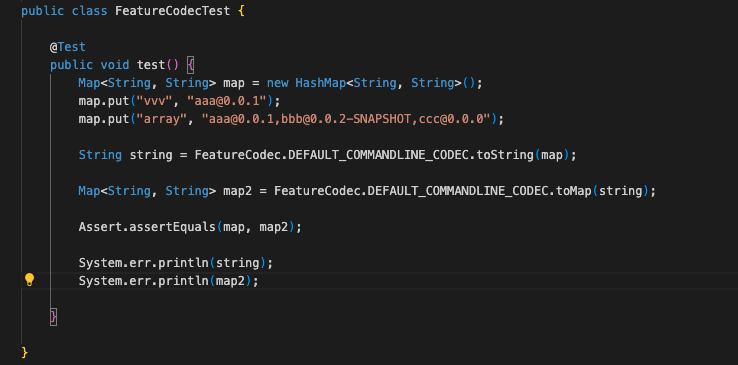


Fig. 1 the existing developer-written test case

* Synthesized codified MR: the *test\_AutoMR(Map<String, String> map)* in fig. 2
* Codified-MR-based test case: the *newtest( )* in fig. 2
* Line coverage: **0.9081632653061225**



Fig. 2 a codified MR based test case

The code snippet accounting for the additional coverage: see fig. 3

* The **blue** lines are lines **covered** by the test case, and the **right** lines are **not covered**.
* **left** part: coverage from the **codified-MR-based test case**,
* **right** part: coverage from the **developer-written test case**.

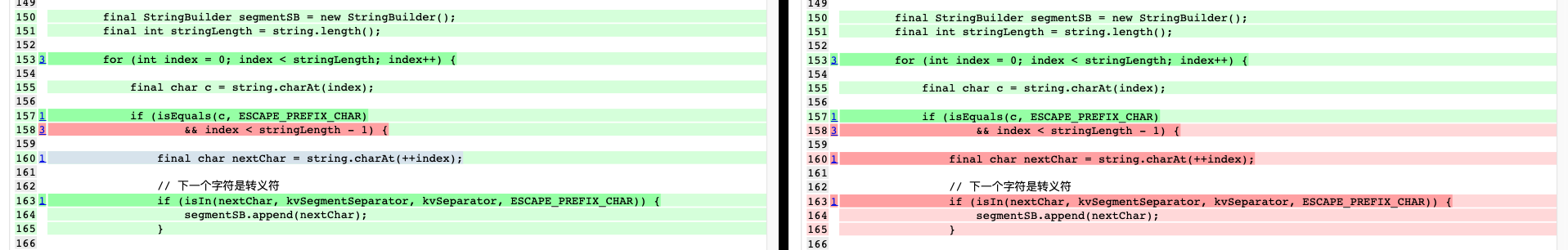




Fig. 3 additional covered code

The additional covered lines are as follows:

162

// 下一个字符是转义符

163

if (isIn(nextChar, kvSegmentSeparator, kvSeparator, ESCAPE\_PREFIX\_CHAR)) {

164

segmentSB.append(nextChar);

165

}

…

…

200

// 匹配到转义前缀符

201 1

if (isEquals(c, ESCAPE\_PREFIX\_CHAR)) {

202

203

decodeStack.push(c);

204 3

if (index < stringLength - 1) {

205 1

final char nextChar = string.charAt(++index);

206

decodeStack.push(nextChar);

207

}

We can see that the additional covered lines are mainly focus on “escape character” (“转义符”). The test input of codified MR based test is with an escape character “\”, but the developer-written test inputs are not.

* + Codified-MR-based test input (in fig. 2): *hashMap0.put("k\*HG]4}Z=Q?syY", "t~\"&u");*
  + Developer-written test inputs (in fig. 1):
    - *map.put("vvv", "aaa@0.0.1");*
    - *map.put("array", "aaa@0.0.1,bbb@0.0.2-SNAPSHOT,ccc@0.0.0");*

Example2: a codified-MR-based test has higher test coverage than a EvoSuite-generated test

Project: leonchen83/redis-replicator

Class under test: com.moilioncircle.redis.replicator.util.Lzf, which is available at https://github.com/leonchen83/redis-replicator/blob/3bdefd9d3160853c482d7d14d74b7b9b4205eb24/src/main/java/com/moilioncircle/redis/replicator/util/Lzf.java

* The Evosuite-generated test case(or see figure 1)
* Line coverage: **0.7857142857142857**

图形用户界面, 文本

中度可信度描述已自动生成

Fig. 1 the EvoSuite-generated test case

* Synthesized codified MR: the *test\_AutoMR(Map<String, String> map)* in fig. 2
* Codified-MR-based test case: the *newtest( )* in fig. 2
* Line coverage: **0.9081632653061225**

文本

描述已自动生成

Fig. 2 a codified MR based test case

The code snippet accounting for the additional coverage: see fig. 3

* The **blue** lines are lines **covered** by the test case, and the **right** lines are **not covered**.
* **left** part: coverage from the **codified-MR-based test case**,
* **right** part: coverage from the **developer-written test case**.

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

Fig. 3 additional covered code

The additional covered lines are as follows:

162

// 下一个字符是转义符

163

if (isIn(nextChar, kvSegmentSeparator, kvSeparator, ESCAPE\_PREFIX\_CHAR)) {

164

segmentSB.append(nextChar);

165

}

…

…

200

// 匹配到转义前缀符

201 1

if (isEquals(c, ESCAPE\_PREFIX\_CHAR)) {

202

203

decodeStack.push(c);

204 3

if (index < stringLength - 1) {

205 1

final char nextChar = string.charAt(++index);

206

decodeStack.push(nextChar);

207

}

We can see that the additional covered lines are mainly focus on “escape character” (“转义符”). The test input of codified MR based test is with an escape character “\”, but the developer-written test inputs are not.

* + Codified-MR-based test input (in fig. 2): *hashMap0.put("k\*HG]4}Z=Q?syY", "t~\"&u");*
  + Developer-written test inputs (in fig. 1):
    - *map.put("vvv", "aaa@0.0.1");*
    - *map.put("array", "aaa@0.0.1,bbb@0.0.2-SNAPSHOT,ccc@0.0.0");*