Lab8 report

11510352 李子强

folder hierarchy

```
.
├── hamcrest-core-1.3.jar
├── junit-4.12.jar
└── triangle
    ├── TestSuite.java
    └── Triangle.java
```

shell commands

```
# compile
javac -cp ".:./junit-4.12.jar:./hamcrest-core-1.3.jar:$CLASSPATH"
triangle/Triangle.java triangle/TestSuite.java

# analysis
java -cp ".:./junit-4.12.jar:./hamcrest-core-1.3.jar:$CLASSPATH"
daikon.DynComp triangle.Triangle

java -cp ".:./junit-4.12.jar:./hamcrest-core-1.3.jar:$CLASSPATH"
daikon.Chicory --daikon --comparability-file=Triangle.decls-DynComp
triangle.Triangle
```

Triangle running results

```
entered daikon.chicory.Runtime.setDtrace(./Triangle.dtrace.gz, false)...
Chicory warning: ClassFile: org.junit.Assert - classfile version (49) is
out of date and may not be processed correctly.
Chicory warning: ClassFile: org.junit.ComparisonFailure - classfile version
(49) is out of date and may not be processed correctly.
Chicory warning: ClassFile: org.junit.internal.ArrayComparisonFailure -
classfile version (49) is out of date and may not be processed correctly.
Chicory warning: ClassFile: org.hamcrest.Matcher - classfile version (49)
is out of date and may not be processed correctly.
Chicory warning: ClassFile: org.hamcrest.SelfDescribing - classfile version
(49) is out of date and may not be processed correctly.
Daikon version 5.6.4, released April 3, 2018;
http://plse.cs.washington.edu/daikon.
Reading declaration files Processing trace data; reading 1 dtrace file:

================================================================================
```

```
org.junit.Assert.assertEquals(java.lang.Object, java.lang.Object):::ENTER
expected == actual
expected.getClass().getName() == actual.getClass().getName()
expected.getClass().getName() == triangle.Triangle$Type.class
================================================================================
org.junit.Assert.assertEquals(java.lang.Object, java.lang.Object):::EXIT
================================================================================
org.junit.Assert.assertEquals(java.lang.String, java.lang.Object,
java.lang.Object):::ENTER
expected == actual
expected.getClass().getName() == actual.getClass().getName()
message == null
expected.getClass().getName() == triangle.Triangle$Type.class
================================================================================
org.junit.Assert.assertEquals(java.lang.String, java.lang.Object,
java.lang.Object):::EXIT112
================================================================================
org.junit.Assert.assertEquals(java.lang.String, java.lang.Object,
java.lang.Object):::EXIT
================================================================================
org.junit.Assert.equalsRegardingNull(java.lang.Object,
java.lang.Object):::ENTER
expected == actual
expected.getClass().getName() == actual.getClass().getName()
expected.getClass().getName() == triangle.Triangle$Type.class
================================================================================
org.junit.Assert.equalsRegardingNull(java.lang.Object,
java.lang.Object):::EXIT127
================================================================================
org.junit.Assert.equalsRegardingNull(java.lang.Object,
java.lang.Object):::EXIT127;condition="return == true"
================================================================================
org.junit.Assert.equalsRegardingNull(java.lang.Object,
java.lang.Object):::EXIT
return == true
================================================================================
org.junit.Assert.equalsRegardingNull(java.lang.Object,
java.lang.Object):::EXIT;condition="return == true"
================================================================================
org.junit.Assert.isEquals(java.lang.Object, java.lang.Object):::ENTER
expected == actual
expected.getClass().getName() == actual.getClass().getName()
expected.getClass().getName() == triangle.Triangle$Type.class
================================================================================
org.junit.Assert.isEquals(java.lang.Object, java.lang.Object):::EXIT
return == true
================================================================================
org.junit.Assert.isEquals(java.lang.Object,
java.lang.Object):::EXIT;condition="return == true"
================================================================================
triangle.Triangle$Type:::OBJECT
================================================================================
triangle.Triangle$Type.Type(java.lang.String, int):::ENTER
================================================================================
```

```
triangle.Triangle$Type.Type(java.lang.String, int):::EXIT
arg0.toString == orig(arg0.toString)
==========================================================================
triangle.Triangle.classify(int, int, int):::ENTER
==========================================================================
triangle.Triangle.classify(int, int, int):::EXIT21
return == triangle.Triangle$Type.INVALID
return has only one value
orig(arg0) one of { 0, 1 }
orig(arg1) one of { 0, 1, 1301 }
orig(arg2) one of { -665, 0, 1 }
==========================================================================
triangle.Triangle.classify(int, int, int):::EXIT31
return == triangle.Triangle$Type.INVALID
orig(arg1) == orig(arg2)
return has only one value
orig(arg0) == 1108
orig(arg1) == 1
==========================================================================
triangle.Triangle.classify(int, int, int):::EXIT35
return == triangle.Triangle$Type.EQUILATERAL
orig(arg0) == orig(arg1)
orig(arg0) == orig(arg2)
return has only one value
orig(arg0) == 582
==========================================================================
triangle.Triangle.classify(int, int, int):::EXIT
triangle.Triangle$Type.INVALID has only one value
triangle.Triangle$Type.SCALENE has only one value
triangle.Triangle$Type.EQUILATERAL has only one value
triangle.Triangle$Type.ISOSCELES has only one value
triangle.Triangle$Type.$VALUES has only one value
triangle.Triangle$Type.$VALUES.getClass().getName() ==
triangle.Triangle$Type[].class
triangle.Triangle$Type.$VALUES[] contains no nulls and has only one value,
of length 4
triangle.Triangle$Type.$VALUES[].getClass().getName() ==
[triangle.Triangle.Type, triangle.Triangle.Type, triangle.Triangle.Type,
triangle.Triangle.Type]
triangle.Triangle$Type.$VALUES[].getClass().getName() elements ==
triangle.Triangle.Type.class
size(triangle.Triangle$Type.$VALUES[]) == 4
return in triangle.Triangle$Type.$VALUES[]
triangle.Triangle$Type.INVALID in triangle.Triangle$Type.$VALUES[]
triangle.Triangle$Type.SCALENE in triangle.Triangle$Type.$VALUES[]
triangle.Triangle$Type.EQUILATERAL in triangle.Triangle$Type.$VALUES[]
triangle.Triangle$Type.ISOSCELES in triangle.Triangle$Type.$VALUES[]
==========================================================================
triangle.Triangle.main(java.lang.String[]):::ENTER
arg0 has only one value
arg0.getClass().getName() == java.lang.String[].class
arg0[] == []
arg0[].toString == []
==========================================================================
```

```
triangle.Triangle.main(java.lang.String[]):::EXIT
arg0[] == orig(arg0[])
arg0[] == []
arg0[].toString == []
Exiting Daikon.
```

Results analysis

From the above report, function `Triangle.classify` return values must be in enum class `Triangle.Type`. The enum class only have 4 elements.