Assignment 1

Environment

- since soot does not work with Java8, this assignment uses Java7 as the developing kit and environment.

Questions & Answers

Q1: Randoop Related

- Q1-(1): **What is test oracle when conducting random testing via Randoop**
- **Answer**: five built-in **test oracle** are as follows:

  1. Equals to null: `o.equals(null)` should return `false`
  2. Reflexivity of equality: `o.equals(o)` should return `true`
  3. Symmetry of equality: `o1.equals(o2)` implies `o2.equals(o1)`
  4. Equals-hashcode: If `o1.equals(o2)==true`, then `o1.hashCode()==o2.hashCode()`
  5. No null pointer exceptions: NO `NullPointerException` is thrown if no null inputs are used in a test

---

- Q1-(2): **How do you use Randoop to generate test cases? (Please give a detailed descriptions on the steps and the parameter settings.)**
- **Answer**: as follows, in three steps

  - first, enter into Dir `UserfulShells`, and use the following shell script

```
RANDOOP_CLASSPATH="../Local-Jars/randoop-all-3.0.4.jar"
SRC_CLASSPATH="../AssignmentSubject/bin"
IO_ARGS="--classlist=my_classes.txt --junit-output-dir=../AssignmentSubject/t
est_src --junit-package-name=randoop_test"
LITERAL_ARGS="--literals-file=literals.txt"
TIME_LIMIT_ARGS=" --timelimit=80"

java -cp $RANDOOP_CLASSPATH:$SRC_CLASSPATH randoop.main.Main gentests $IO_ARG
S $LITERAL_ARGS $TIME_LIMIT_ARGS
```

  - second, run the shell in Dir `UserfulShells`

```
./use_randoop_gen_tests.sh
```

Q2: Coverage Related

- Q2-(1): **Please specify the settings of Randoop**
- **Answer**: it is what is elaborated in Q1(2), the shell used is as follows:

```
RANDOOP_CLASSPATH="../Local-Jars/randoop-all-3.0.4.jar"
SRC_CLASSPATH="../AssignmentSubject/bin"
IO_ARGS="--classlist=my_classes.txt --junit-output-dir=../AssignmentSubject/test_s
rc --junit-package-name=util_test"
LITERAL_ARGS="--literals-file=literals.txt"
TIME_LIMIT_ARGS=" --timelimit=600"
java -cp $RANDOOP_CLASSPATH:$SRC_CLASSPATH randoop.main.Main gentests $IO_ARGS $LI
TERAL_ARGS $TIME_LIMIT_ARGS
```

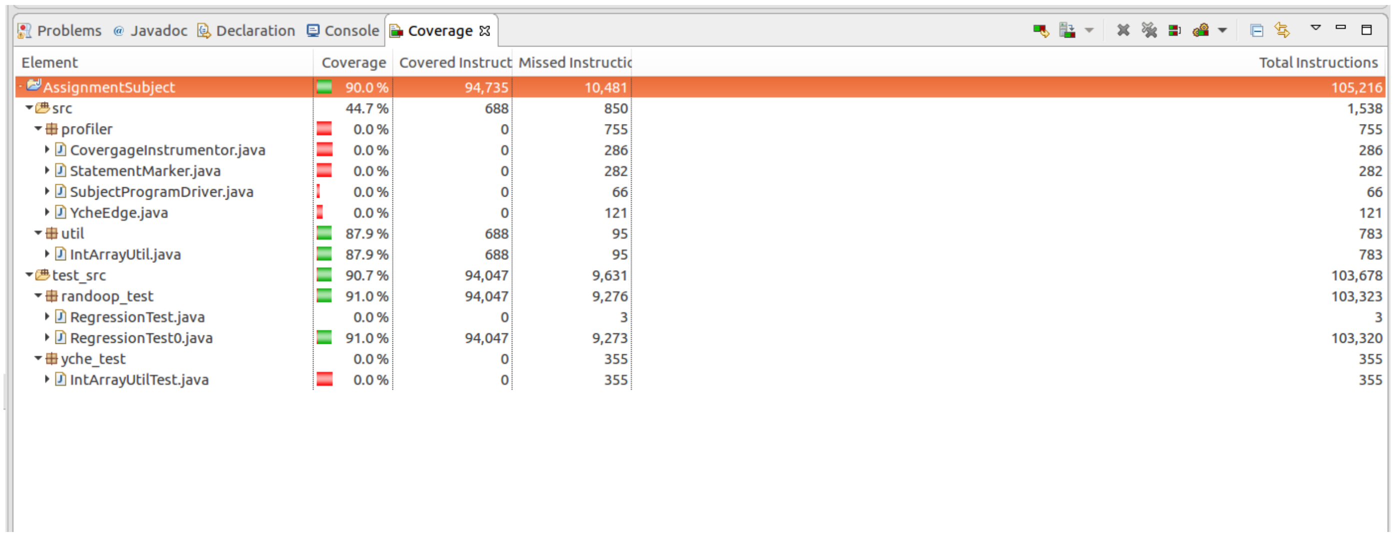
- Q2-(2): **What are the statement coverage and branch coverage in your random testing**
- **Answer**: the statement coverage and branch coverage collected by EclEmma, underlying using Jacoco are as follows.
- **Statement Coverage**

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
IntArrayUtil.java	87.9%	688	95	783

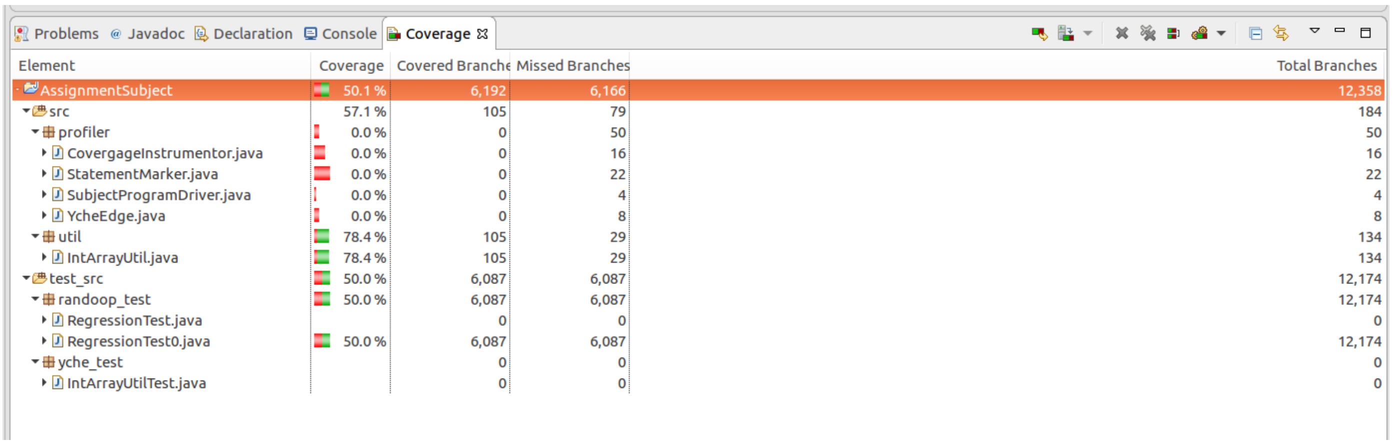
- **Branch Coverage**

Element	Coverage	Covered Branch	Missed Branch	Total Branches
IntArrayUtil.java	78.4%	105	29	134

- Screenshot:statement coverage and branch coverage



Element	Coverage	Covered Instruct	Missed Instructic	Total Instructions
AssignmentSubject	🟩 90.0 %	94,735	10,481	105,216
src	44.7 %	688	850	1,538
profiler	🟥 0.0 %	0	755	755
CovergageInstrumentor.java	🟥 0.0 %	0	286	286
StatementMarker.java	🟥 0.0 %	0	282	282
SubjectProgramDriver.java	🟥 0.0 %	0	66	66
YcheEdge.java	🟥 0.0 %	0	121	121
util	🟩 87.9 %	688	95	783
IntArrayUtil.java	🟩 87.9 %	688	95	783
test_src	🟩 90.7 %	94,047	9,631	103,678
randoop_test	🟩 91.0 %	94,047	9,276	103,323
RegressionTest.java	0.0 %	0	3	3
RegressionTest0.java	🟩 91.0 %	94,047	9,273	103,320
yche_test	0.0 %	0	355	355
IntArrayUtilTest.java	🟥 0.0 %	0	355	355



Element	Coverage	Covered Branche	Missed Branches	Total Branches
AssignmentSubject	50.1 %	6,192	6,166	12,358
src	57.1 %	105	79	184
profiler	0.0 %	0	50	50
CovergageInstrumentor.java	0.0 %	0	16	16
StatementMarker.java	0.0 %	0	22	22
SubjectProgramDriver.java	0.0 %	0	4	4
YcheEdge.java	0.0 %	0	8	8
util	78.4 %	105	29	134
IntArrayUtil.java	78.4 %	105	29	134
test_src	50.0 %	6,087	6,087	12,174
randoop_test	50.0 %	6,087	6,087	12,174
RegressionTest.java		0	0	0
RegressionTest0.java	50.0 %	6,087	6,087	12,174
yche_test		0	0	0
IntArrayUtilTest.java		0	0	0

Statement & Branch Coverage Measurement Program

Codes Usages

- Dependency: the jars are put in Local-Jars.
- Codes Organization: the codes could be found in AssignmentSubject, the main profiler codes are in AssignmentSubject/src/profiler, the test codes generated by randoop is in AssignmentSubject/test_src, I write a test case in AssignmentSubject/test_src/yche_test
- First, the eclipse helps me compile the codes, the `*.class` are put in AssignmentSubject/bin.
- Second, enter the UserfulShells, run the use_soot_driver.sh, and then copy the related classes with UserfulShells/cp_related_class_files.sh.
- Third, run the instrumented programs, use shells UserfulShells/run_instrumented_subject_program_randoop_test.sh, and UserfulShells/run_instrumented_subject_program_yche_test.sh.

Output

- the specific statement covered and branch covered are stored in UserfulShells/randoop_coverage.txt and UserfulShells/yche_coverage.txt.
- the whole statement count and branch count are stored in UserfulShells/util.IntArrayUtil_branches_num.txt and UserfulShells/util.IntArrayUtil_statements_num.txt
- In the experiments I found the statements number got by EclEmma may be not accurate, which may include the instrumented Instructions.

Understanding

- here, in the jimple code, arguments passing should be skipped, i.e, `JIdentityStmt` , including `this` and other arguments
- **statement coverage**, is the vertices, represented as a statement, for the ratio, dividing it by whole if
- **branch coverage**, is the edges between nodes starting from a `JIfStmt` , destinating in another statement e.g, `goto label` , and the else is the statement chained

after the `JIfstmt` for the ratio

Part1:Statement Coverage & Part2:Branch Coverage(Bonus)

The basic implementation is made in AssignmentSubject/src/profiler/StatementMarker.java, and the instrumented program will invoke the reflected functions in helper class. In order to hold the branch info, I introduce
a user-define type AssignmentSubject/src/profiler/YcheEdge.java.

References

- Tutorial
- Assignment Requirements
- Assignment Faq
- Soot Doc