CSE364 Software Engineering
Individual Assignment

Instructor: Mijung Kim

Due: Monday, May 22, 2023 at 11:59pm on Blackboard

Submission Guidelines

- Submit a PDF report.

General Information

- This assignment has 120 points in total and is worth 10% of your total grade.
- This assignment is an individual assignment, and thus must be solved independently (i.e., without assistance from others.
- You will get 70% and 50% of the grade if you turn it in within 24 and 48 hours, respectively, after the regular deadline. Please note that an assignment that is re-submitted after the regular deadline will be counted as late even if you have an earlier submission. After the late submission deadline has passed, you won't be able to submit your assignment and will get 0 points.
- If you have any questions, please actively use Pizza at `http://piazza.com/unist.ac.kr/spring2023/cse364` or TA's Office Hours.

Instructions

The goal of this individual assignment is to study automated test generation tools learned in class and experience their practicality and weakness of them. In this assignment, you are expected to detect ***one*** bug of a Java class in an open-source project using test generation tools, Randoop[1] and Evosuite[2].

Suppose that you are a developer of an open-source library called Apache Lang. The source code of this library is located at `library_under_test` directory of the `CSE364_individual_assignment.zip`. Suppose you are developing `org.apache.commons.lang3.StringUtils` class and want to use some test generation tools for testing rather than manually writing tests.

You should write a report containing the sections below:

1. **Detect and localize a bug (50 points)**
    - Run Randoop and Evosuite `org.apache.commons.lang3.StringUtils` class as provided in Randoop and Evosuite Tutorials attached below.
    - Attach a failing test case that manifests the bug. This means the failing test case code along with the stack trace.
    - Attach a code snippet of the bug location in the source code.
      The source code of `org.apache.commons.lang3.StringUtils` class is located at `src/java/org/apache/commons/lang3/StringUtils.java`.
    - Explain how this bug occurred and its root cause.
    - Which tool did you use to reveal the bug? Did the other one succeed to reveal the same bug as well?

[1] `https://randoop.github.io/randoop/`
[2] `https://www.evosuite.org/`

- *tip)* When you run Randoop, you will probably be overwhelmed by a tremendous amount of failing tests which are mostly false alarms. A false alarm refers to a failing test case that produces an expected failure. Usually, such a false alarm fails just because the test input is invalid (i.e., illegal) rather than because it reveals the real bug in the program. One possible way to effectively investigate failing tests is to run Evosuite first and detect a true alarm, and then find out whether Randoop detects the same bug or not.
- *tip)* Due to the randomness of test generation, it is possible that the generated failing tests do not reveal a true bug. In that case, you can consider rerunning test generation or adjusting the time budget by command line option(Randoop: `--time-limit=<`*time budget*`>`, Evosuite: `-Dsearch_budget=<`*time budget*`>`).

2. **Measure coverage of generated tests (20 points)**
   - Referring to the Tutorials below, run `jacoco` with generated tests.
   - Attach coverage reports of Randoop and Evosuite tests.
3. **Describe strengths and weaknesses of Randoop and Evosuite (30 points)**
   - Use the results of 1 and 2 above.
4. **Improve Test Generation (20 points)**
   - Think about how to address the weakness of Randoop and Evosuite that you have described in 3.
   - Describe your idea on how to improve the bug detection ability of Randoop and Evosuite.

Randoop and Evosuite Tutorials (Linux/macOS/Linux Docker[3])

Unzip `CSE364_individual_assignment.zip`. All dependencies are self-contained so you should only make sure to use JAVA8[4]. Once Setting Environment Variables is done, you can conduct Running Randoop and Running Evosuite in any order.

Setting Environment Variables

Set variables for brevity. Those paths are used in Running Randoop and Running Evosuite. The below commands should be executed in `CSE364_individual_assignment` directory.

```
# Path to dependent libraries
export JAVA_HOME=/your/java/home/path
export RANDOOP=${PWD}/randoop-4.3.2/randoop-all-4.3.2.jar
export EVOSUITE=${PWD}/evosuite/evosuite-1.0.6.jar
export EVOSUITE_RUNTIME=${PWD}/evosuite/evosuite-standalone-runtime-1.0.6.jar
export JUNIT=${PWD}/junit/junit-4.11.jar
export JACOCOAGENT=${PWD}/jacoco/org.jacoco.agent-0.8.10-runtime.jar
export JACOCOCLI=${PWD}/jacoco/org.jacoco.cli-0.8.10-nodeps.jar

# Path to library under test.
export LUT=${PWD}/library_under_test
export TARGET_CLASS=org.apache.commons.lang3.StringUtils
export TARGET_CLASSPATH=${LUT}/target/classes
```

Running Randoop

```
mkdir ${LUT}/src/test/randoop
cd ${LUT}/src/test/randoop
```

[3]You can use pre-built image with JDK-8 from `https://hub.docker.com/r/rtfpessoa/ubuntu-jdk8`.

[4]When you switch to another JAVA version, you'd better to use an automatic way instead of manual manipulation which is prone to miss some dependencies. One such way is `update-java-alternatives` command. Please refer to the following link. `https://askubuntu.com/questions/740757/switch-between-multiple-java-versions`

```
# Run Randoop.
java -classpath ${TARGET_CLASSPATH}:${RANDOOP} randoop.main.Main gentests \
--testclass=${TARGET_CLASS} --flaky-test-behavior=output --usethreads --clear=10000 \
--string-maxlen=5000 --forbid-null=false --null-ratio=0.1 \
--only-test-public-members=false --omit-methods=HashCodeAndEqualsSafeSet.of \
--no-regression-assertions=true --checked-exception=ERROR --unchecked-exception=ERROR \
--testsperfile=20000 --maxsize=10 --no-regression-tests=true --time-limit=60

# Compile generated test files.
javac -classpath .:${TARGET_CLASSPATH}:${JUNIT} ErrorTest*.java -sourcepath .:${LUT}

# Run compiled tests. It will report lots of failures.
# You should find a real bug among them by manually checking each unit test.
java -javaagent:${JACOCOAGENT} -classpath .:${TARGET_CLASSPATH}:${JUNIT} \
org.junit.runner.JUnitCore ErrorTest

# Get a coverage report. A report would be stored in `./report/index.html`.
java -jar ${JACOCOCLI} report jacoco.exec \
--classfiles ${TARGET_CLASSPATH}/org/apache/commons/lang3/StringUtils.class \
--sourcefiles ${TARGET_DIR}/src/java/org/apache/commons/lang3/StringUtils.java \
--html report
```

Running Evosuite

```
mkdir ${LUT}/src/test/evosuite
cd ${LUT}/src/test/evosuite

# Run Evosuite.
java -jar ${EVOSUITE} -class ${TARGET_CLASS} \
-projectCP ${TARGET_CLASSPATH} -criterion branch -Djunit_check=false \
-Dfilter_assertions=false -Dassertions=false -Dshow_progress=false \
-Dtest_comments=false -mem 2048 -heapdump -Dcatch_undeclared_exceptions=false \
-Duse_separate_classloader=false -Dsearch_budget=60

# Compile generated test files.
javac -classpath ${TARGET_CLASSPATH}:${EVOSUITE_RUNTIME}:${PWD}/evosuite-tests:${JUNIT} \
evosuite-tests/org/apache/commons/lang3/*.java

# Run compiled tests. It will report lots of failures.
# You should find a real bug among them by manually checking each unit test.
java -classpath ${TARGET_CLASSPATH}:${EVOSUITE_RUNTIME}:${PWD}/evosuite-tests:${JUNIT} \
-javaagent:${JACOCOAGENT} org.junit.runner.JUnitCore ${TARGET_CLASS}_ESTest

# Get a coverage report. A report would be stored in `./report/index.html`.
java -jar ${JACOCOCLI} report jacoco.exec \
--classfiles ${TARGET_CLASSPATH}/org/apache/commons/lang3/StringUtils.class \
--sourcefiles ${TARGET_DIR}/src/java/org/apache/commons/lang3/StringUtils.java \
--html report
```