

The Testing Project

CSE 4321

Motivation

- Provides an opportunity to practice the basic concepts, principles, and techniques covered in this course
- In particular, you will apply control flow testing to a program of moderate size
 - Printtokens.java: a Java class that implements a string tokenizer (available in Canvas).
 - about 500 lines of code and with 15 seeded faults

Input and Output

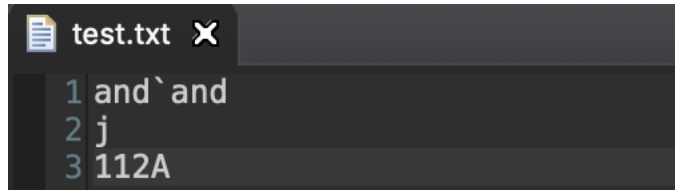
- Input
 - Location of a file (i.e., input to the main() method) that contains the tokens to be classified
 - If a location is not given, the user is expected to type the tokens to be classified on the console.
- Output
 - Type of each token, one per line

Type of Tokens

- **Keyword:** and, or, if, xor, lambda, =>
- **Special Symbol:** “(”, “)”, “[”, “]”, “””, “””, “,” (lparen, rparen, lsquare, rsquare, quote, bquote, comma)
- **Identifier:** begins with a letter and contains only letters and digits, e.g., a, aa, a1, a2.
- **Number Constant:** e.g., 123, 1, 321
- **String Constant:** e.g., “asd”, “123”
- **Character Constant:** e.g., #a, #b, #c, #d
- **Comment:** Anything starts with “;”
- **Error:** Everything else

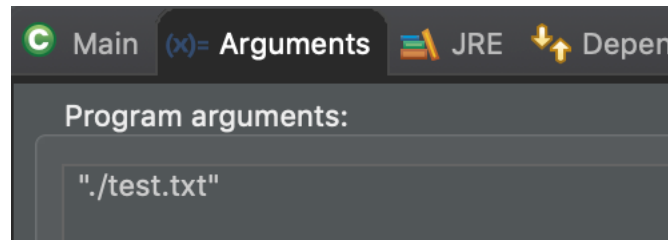
Example

Input:



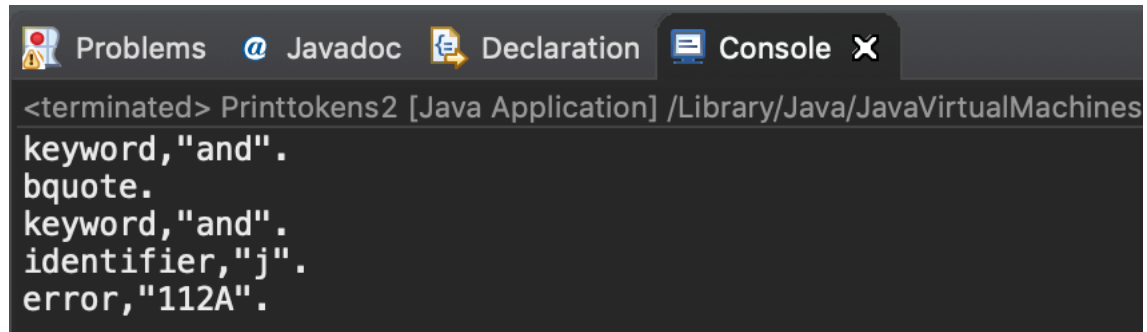
```
test.txt x
1 and`and
2 j
3 112A
```

Arguments:



```
Main (x)= Arguments JRE Depen
Program arguments:
"./test.txt"
```

Output:



```
Problems Javadoc Declaration Console x
<terminated> Printtokens2 [Java Application] /Library/Java/JavaVirtualMachines
keyword,"and".
bquote.
keyword,"and".
identifier,"j".
error,"112A".
```

Major Steps

- Create the Control Flow Graph (CFG) for each method that has at least one branching decision
 - Three methods, `get_char`, `unget_char`, `unget_error`, do not have any branching decision and can be skipped
 - The code in the `catch` clauses can be skipped, i.e., these clauses do not need to be represented in the CFG
- Select test paths from CFG to achieve edge coverage
 - For the CFG of each of the following methods, `get_token`, `is_token_end`, `is_keyword`, `is_num_constant`, `is_str_constant`, `is_identifier`, achieve edge coverage for the method, i.e., cover all the edges in CFG. (Unit Testing)
 - For the CFG of the `main` method, achieve edge coverage for the entire program, i.e., cover all the edges in all the CFGs. (Program-Level Testing)
 - Other methods can be skipped
- Generate test cases, including test data and expected output, for each test path
 - If a test path you selected in the previous step is infeasible, a new test path needs to be selected, and test data and expected output need to be selected for the new test path
- Use JUnit to implement and execute the generated test cases.
- Provide branch coverage reports for tests executed.
 - One report for the unit tests of individual methods, and one report for the program-level tests of the main method

Deliverables

- 20%: CFGs along with the corresponding source code, and the basic block table that identifies each basic block
- 30%: Test cases, including each test path, the corresponding test data to execute the test path, and the expected output
- 25%: JUnit source code
- 5%: Code coverage reports
- 10%: Faults detected and corrections
- 5%: Summary and discussion.
- 5%: A ReadMe file that explains how to execute your JUnit code

Must-have files

- Your project should be submitted as a zipped file with file name formatted as “FirstName-LastName- 100*****.zip”. E.g. “John-Doe-1000123456.zip”
- Must contain these but not limited to these files.
 - CFG.pdf
 - Test Cases.pdf
 - Source Code
 - Code Coverage Reports.zip
 - Faults Detected and Corrections.pdf
 - Summary and Discussion.pdf
 - ReadMe.txt
- No hand-written reports, or scan of hand-written reports, will be accepted.

Evaluation

- Faults detected and code coverage are not the main focus of our evaluation.
- If you follow the approach correctly, you could expect to detect at least 10 bugs.
- Focus on the “approach”, instead of the “faults”

Recommendations

- IDE: Eclipse; Code Coverage: JaCoCo/EclEmma
 - Refer to the video titled “Get started with the project” in Canvas
 - Other tools could be used, but you may be asked to give a demo for the TA
- Fix a fault after you find it before you continue; this could help to discover more faults.
- If fixing a fault alters the control flow, you are not required to construct a new CFG.
- Do not try to use CFG generation tools for Java programs; you will receive 0 on the project if you do.
 - Many tools generate CFGs based on byte code, not source code. The output could be incorrect w.r.t the project specification.
- Carefully read these slides and revisit as needed, try not to lose points for not understanding the project description.