Cody Jones
1001986663
Summary

CSE 4321 Project Summary

1. Introduction

This project was part of the CSE 4321 – Software Testing and Maintenance course at The University of Texas at Arlington, taught by Dr. Jeff Lei. The objective was to test and analyze the provided Java program, Printtokens.java, a 500-line string tokenizer with seeded faults. The program classifies tokens into categories like string constants, numeric constants, and identifiers. Inputs would be given via the a file, and the output lists the token types, one per line. While Eclipse was recommended, I opted to use IntelliJ IDEA for this project.

2. Creation of CFGs

The first step involved creating Control Flow Graphs (CFGs) for methods containing branching decisions, skipping catch blocks as per the project requirements. Each method was divided into basic blocks, and branching conditions were labeled as True (T) or False (F). Using https://app.diagrams.net/, I created CFG diagrams for 15 methods, including the main method. These diagrams, along with their basic block tables and snippet code, were compiled into the document CFG.pdf.

3. Edge Coverage

To achieve edge coverage, I selected test paths for the non-main methods. Additionally, the main method was tested for program-wide edge coverage. The paths ensured that all edges in the CFGs were covered at least once. Test paths and their corresponding test data and expected outputs were documented in Test_Cases.pdf.

4. Generation of Test Cases

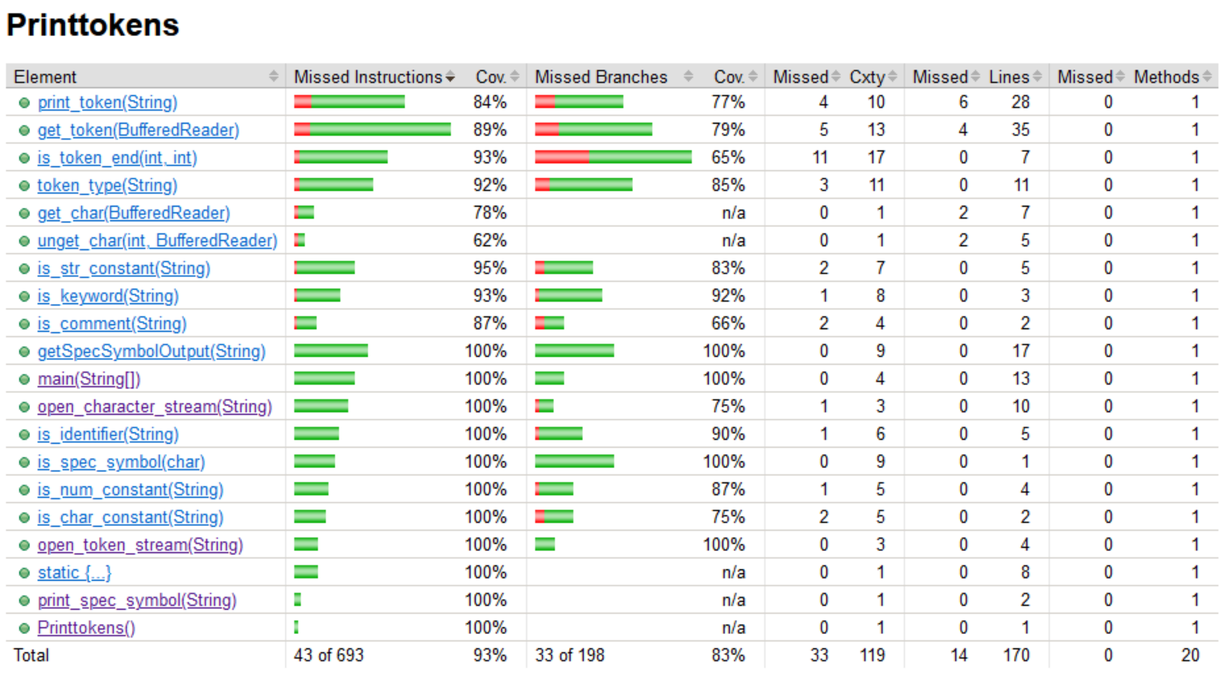
Test cases were generated for each selected test path to ensure edge coverage. Each test case included the test data (inputs) and expected outputs, starting from the Start node and ending at the End node of the CFG.

5. JUnit Testing

JUnit tests were implemented for both non-main and main methods. Achieving 100% coverage for five methods and approximately 96% for get_token. Program-wide tests for the main and non-main method(s) were conducted using PrinttokensTest.java, with inputs provided via the console and files (emptyFile.txt and test.txt).

Cody Jones
1001986663
Summary

JUnit Test Results



Printtokens

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
print_token(String)		84%		77%	4	10	6	28	0	1
get_token(BufferedReader)		89%		79%	5	13	4	35	0	1
is_token_end(int, int)		93%		65%	11	17	0	7	0	1
token_type(String)		92%		85%	3	11	0	11	0	1
get_char(BufferedReader)		78%		n/a	0	1	2	7	0	1
unget_char(int, BufferedReader)		62%		n/a	0	1	2	5	0	1
is_str_constant(String)		95%		83%	2	7	0	5	0	1
is_keyword(String)		93%		92%	1	8	0	3	0	1
is_comment(String)		87%		66%	2	4	0	2	0	1
getSpecSymbolOutput(String)		100%		100%	0	9	0	17	0	1
main(String[])		100%		100%	0	4	0	13	0	1
open_character_stream(String)		100%		75%	1	3	0	10	0	1
is_identifier(String)		100%		90%	1	6	0	5	0	1
is_spec_symbol(char)		100%		100%	0	9	0	1	0	1
is_num_constant(String)		100%		87%	1	5	0	4	0	1
is_char_constant(String)		100%		75%	2	5	0	2	0	1
open_token_stream(String)		100%		100%	0	3	0	4	0	1
static {...}		100%		n/a	0	1	0	8	0	1
print_spec_symbol(String)		100%		n/a	0	1	0	2	0	1
Printtokens()		100%		n/a	0	1	0	1	0	1
Total	43 of 693	93%	33 of 198	83%	33	119	14	170	0	20

The results highlight strong instruction and method coverage while identifying opportunities for improvement in branch testing. Some uncovered branches indicate additional edge cases that could be incorporated in future tests.

6. Branch Coverage Reports

Using IntelliJ's JaCoCo plugin, branch coverage reports were generated to evaluate uncovered branches and refine test cases. Reports for each non-main and main method(s) can be found under target/site/jacoco/com.begintesting/index.html which provides a detailed breakdown of covered and uncovered branches along with instructions.

7. Fault Identification and Fixes

During the CFG creation and test case generation phases, I identified and fixed 14 faults, including:

- Index errors.
- Missing or incorrect logic in branches.
- Typographical issues in the program.

The identified faults and their fixes were documented in Faults_Detected_and_Corrections.pdf. Both the original and corrected versions of the source code were included in the submission.

8. Conclusion

This project provided valuable hands-on experience in software testing and maintenance. Tasks such as CFG creation, edge coverage testing, JUnit test implementation, and code coverage

Cody Jones
1001986663
Summary

analysis demonstrated real-world applications of testing methodologies. By using IntelliJ IDEA, JaCoCo, and JUnit, I gained a practical understanding of modern testing tools and workflows. The project highlighted the importance of systematic testing and prepared me for technical challenges in the software industry.