Alexander Crowther
Cpt_S 422
Project Milestone 3

Fault Models:

For the comment black box

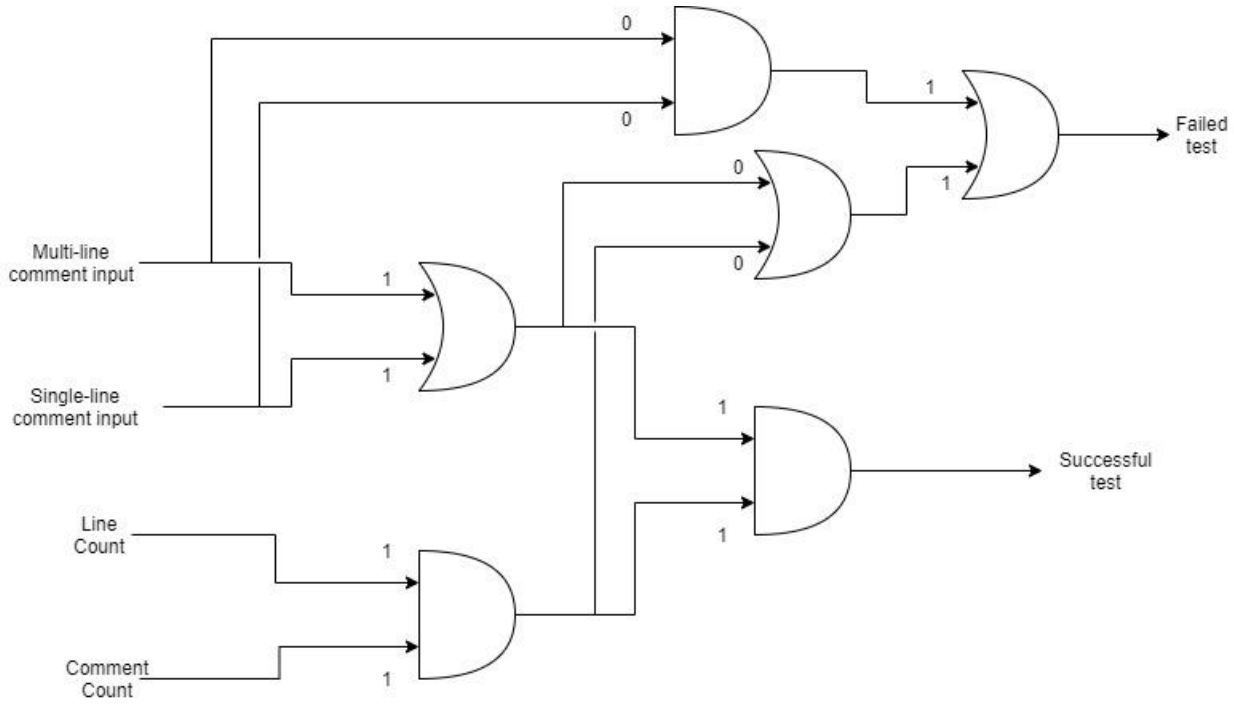


Diagram 1: Comment black box checks fault model

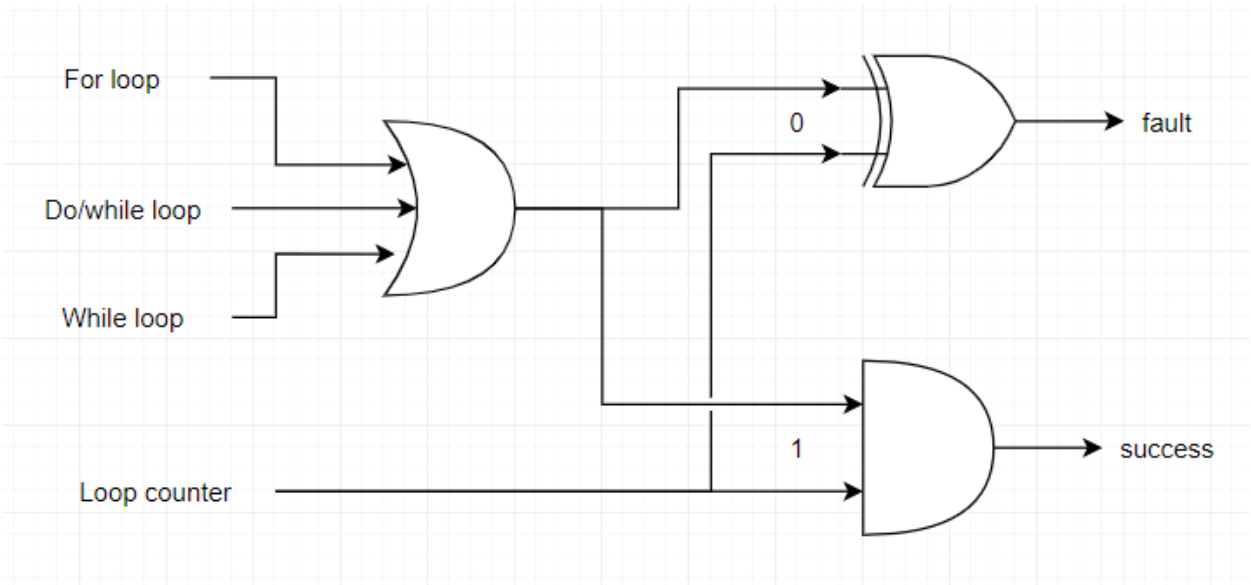


Diagram 2: Loop counter black box checks fault model

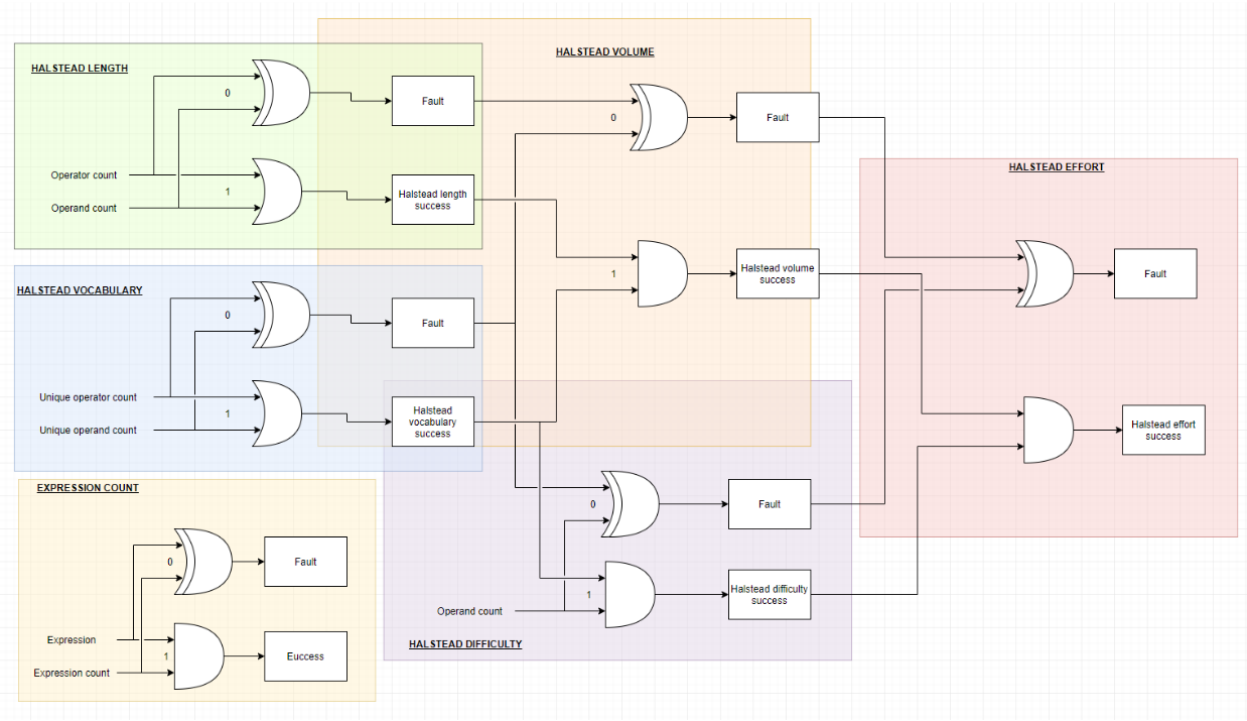


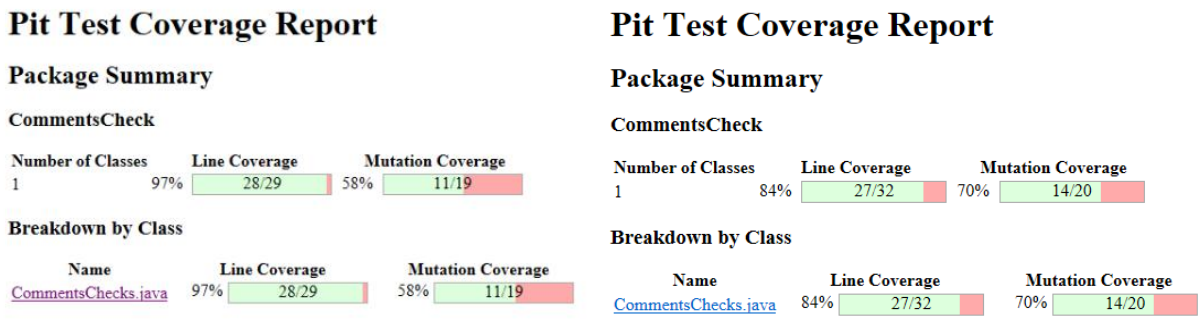
Diagram 3: Halstead black box checks fault model

Pit Test:

I made the mistake of capturing only the package summaries from milestone 2. This greatly limits the analysis I can provide. For this paper I will instead talk about the coverage report for milestone 3.

CommentsCheck

For the CommentsCheck I ended up with less coverage. Adding in a getResults() method with a try/catch statement led to the exception to not be covered. It also appears that adding getResults() makes the finishTree() method to not be covered. I was experimenting with how the @Tests retrieve the log information from CommentsCheck using getResults(). I did not attempt this with the other checks, rather just printing the log information. I fear that the solution I employed does not properly enter the finishTree() method, resulting in no coverage. Pit Test is also not covering the printStackTrace() I added when I added a try/catch block in the getResults() method. While adding more overall mutations, I was able to reduce survived mutations by two. The getDefaultTokens() and getRequiredTokens() methods survive when substituted with a NULL value. At this time, I have no idea why these two methods fail for CommentsCheck and not the other checks (written the same way for all three).



Pit Test Coverage Report

Package Summary

CommentsCheck

Number of Classes	Line Coverage		Mutation Coverage	
1	97%	28/29	58%	11/19

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
CommentsChecks.java	97%	28/29	58%	11/19

Pit Test Coverage Report

Package Summary

CommentsCheck

Number of Classes	Line Coverage		Mutation Coverage	
1	84%	27/32	70%	14/20

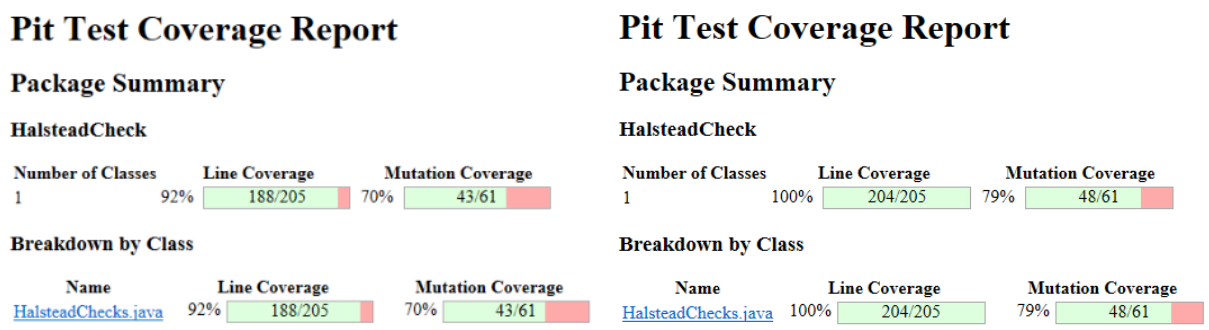
Breakdown by Class

Name	Line Coverage		Mutation Coverage	
CommentsChecks.java	84%	27/32	70%	14/20

Chart 1: Pit test results for CommentsChecks

HealsteadCheck

Surprisingly, I improved line coverage to 100%. I also improved mutation coverage by 9%. The beginTree() survived mutation when removing the initializeVariables() call. Further, in the initializeVariables() two list.clear() calls survived mutation. I will have to conduct further research on a better way to initialize variables in Checkstyle. For the list.clear() I am attempting to ensure the list is empty. I see this surviving as lists ideally start empty anyway. Removing the calls probably will not impact the Checkstyle. The rest of the surviving mutations involve putting all the Checkstyle results in the log. I believe removing the log entries and emulating a getResults() method similar to the CommentCheck and removing the log entries will resolve this.



Pit Test Coverage Report

Package Summary

HalsteadCheck

Number of Classes	Line Coverage		Mutation Coverage	
1	92%	188/205	70%	43/61

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
HalsteadChecks.java	92%	188/205	70%	43/61

Pit Test Coverage Report

Package Summary

HalsteadCheck

Number of Classes	Line Coverage		Mutation Coverage	
1	100%	204/205	79%	48/61

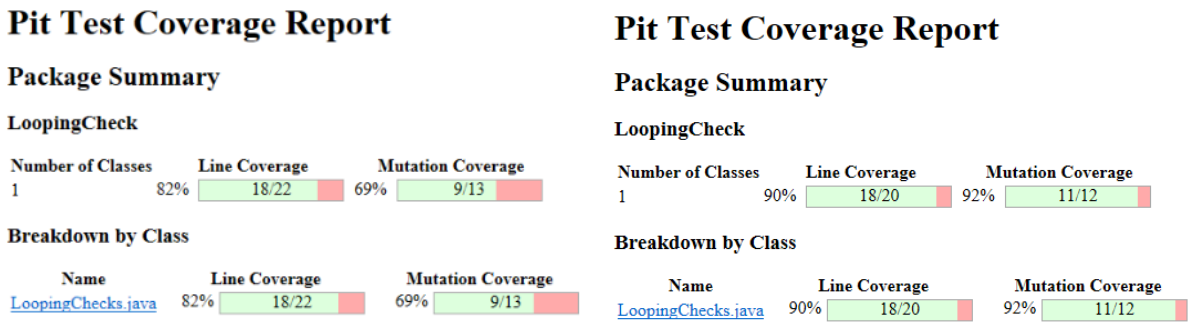
Breakdown by Class

Name	Line Coverage		Mutation Coverage	
HalsteadChecks.java	100%	204/205	79%	48/61

Chart 2: Pit test results for HalsteadChecks

LoopingCheck

For this Pit Test I was unable to cover the logging done in the finishTree() method. The report summary does not show where the mutations survived in the Pit Test. I assume that removing the log statement call for the NO_COVERAGE also resulted in the finishTree() surviving the mutation as it only calls for the log, which does not really influence anything else in the program.



Pit Test Coverage Report

Package Summary

LoopingCheck

Number of Classes	Line Coverage		Mutation Coverage	
1	82%	18/22	69%	9/13

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
LoopingChecks.java	82%	18/22	69%	9/13

Pit Test Coverage Report

Package Summary

LoopingCheck

Number of Classes	Line Coverage		Mutation Coverage	
1	90%	18/20	92%	11/12

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
LoopingChecks.java	90%	18/20	92%	11/12

Chart 3: Pit test results for LoopingChecks