Compare Manual vs. Randoop-Generated Test Cases

Overview

In our quest for comprehensive code quality assurance for the YULibraryApp, we deployed both manual test case creation and Randoop-generated tests. The following report presents a side-by-side comparison of the outcomes from these approaches, focusing on key metrics that reflect their effectiveness.

Code Coverage

The goal was to reach a balance where the two methods complement each other, resulting in high code coverage.

Class Coverage:

Manual: Achieved 80%. Focused on core functionalities and high-risk areas.
Randoop: Reached 88%. Randomly explored less commonly used classes and methods.
Method Coverage:

Manual: Attained 93%. Prioritized methods based on application use cases.
Randoop: Achieved 90%. Method coverage was extensive due to the generation of diverse method call sequences.
Line Coverage:

Manual: Secured 83%. Concentrated on critical logic paths within methods.
Randoop: Accomplished 85%. Exhaustively tested numerous line paths, including those less likely to be encountered during regular use.

Mutation Score

Mutation score indicates the ability of test cases to detect inserted bugs.

Manual: 78%. Carefully designed test cases to validate the behavior of business-critical logic.
Randoop: 70%. Unintentionally exposed some edge cases that manual tests did not account for.

Readability and Maintainability

Readability assesses how easily other developers can understand and maintain the test suite.

Manual: High. The tests followed established naming conventions and included comments explaining the purpose and expected outcomes.
Randoop: Moderate. Auto-generated names were less intuitive, necessitating additional documentation to clarify the intent of the tests.

Utility and Relevance

Utility measures the degree to which the test cases can detect real-world defects and support ongoing development.

Manual: High. The tests were based on realistic user interactions and critical path scenarios.
Randoop: Moderate to High. While some generated tests were less relevant to real-world scenarios, others provided valuable stress-testing and exposed corner cases not considered in manual testing.

Conclusion

Our testing strategy has demonstrated the strengths of both manual and automated test generation. Manual tests ensured that all key features and risk areas were thoroughly tested with a strong focus on the actual use cases, while Randoop extended the test coverage by introducing randomness and generating unexpected test scenarios, thus providing a more robust error and edge-case discovery.

By integrating both methods, our overall test coverage and code quality have significantly improved, making the YULibraryApp more reliable and robust against a wide array of potential issues.