SOEN 6011 – Software Engineering Process

ETERNITY – Transcendental Function Calculator (F5 - ab^x)

Harshavardhana Mudduluru - 40231250

INTRODUCTION

The **Eternity – Transcendental Function Calculator** is a software project designed to perform complex mathematical calculations involving transcendental functions. The project aims to provide an intuitive and user-friendly interface that allows users to input parameters for the function ab^x, where a, b, and x can be any real numbers. This function can model exponential growth or decay processes, making it applicable to numerous real-world scenarios such as population dynamics, radioactive decay, and interest calculations in finance.

Beyond its practical applications, the project serves as an educational tool for understanding the principles of GUI design, user input validation, and error handling in software development.

FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

FUNCTIONAL REQUIREMENTS:

User Input Validation: The calculator must validate user inputs to ensure they fall within acceptable ranges. It should provide real-time feedback if an input is invalid, helping users correct errors before computation.

Computation of Transcendental Function: The application should accurately calculate the value of the function ab^x using the provided inputs. The computation must handle edge cases such as negative exponents, fractional values, decimal points and zero bases correctly.

Display of Results: The results of the computation, along with the time taken to perform the calculation, must be displayed in a clear and concise manner, allowing users to understand the outcomes easily.

Error Handling: The system should gracefully handle any errors that occur during computation, such as division by zero or invalid number formats, by providing informative error messages.

NON-FUNCTIONAL REQUIREMENTS:

Usability: The application should be intuitive and easy to navigate, enabling users with varying levels of expertise to operate it without difficulty. This includes a clear layout and straightforward interaction design.

Performance: The calculator must perform computations efficiently, providing results promptly even for complex calculations involving large numbers or fractional exponents.

Reliability: The system should operate reliably under different conditions, ensuring consistent and accurate results across all use cases.

Maintainability: The codebase should be well-documented and adhere to coding standards to facilitate easy maintenance and future enhancements. Tools like Checkstyle are used to enforce consistent coding practices.

DESIGN AND IMPLEMENTATION

Java Swing for GUI: The graphical user interface (GUI) is designed using Java Swing, providing a responsive and interactive platform for user input and feedback. The interface includes real-time error handling, guiding users to input valid data and alerting them of any discrepancies.

Semantic Versioning: The application follows semantic versioning to ensure that updates and improvements are systematically documented, facilitating easier maintenance and collaboration.

Robust Error Handling: The project employs rigorous input validation and error-handling mechanisms to manage edge cases, such as zero, negative, and fractional inputs. This ensures that calculations are accurate and meaningful, even under challenging conditions.

Coding Standards: To maintain high code quality, the project adheres to coding standards using tools like Checkstyle. This practice not only enhances readability and maintainability but also ensures that the codebase remains consistent and free from common pitfalls.

PROJECT CHALLENGES AND SOLUTIONS:

Challenge 1: Managing fractional, decimal-point and negative inputs without losing precision.

Solution 1: Custom algorithms were developed to perform accurate calculations for power and base manipulations.

SUN MICROSYSTEM'S STYLING TOOL

The Sun Checks styling tool, part of Checkstyle, was employed in this project to enforce Java coding standards and best practices. It was used to identify and correct style violations such as line length, magic numbers, and missing documentation, ensuring consistent code quality throughout the development process.

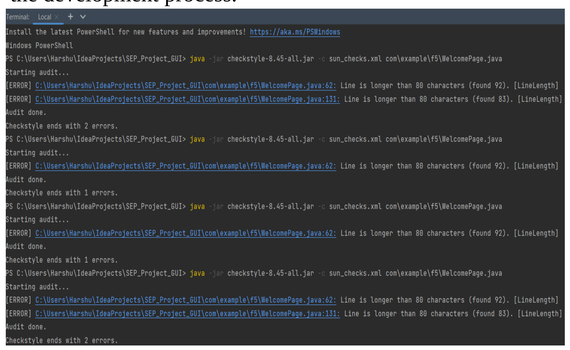

Fig 1 Errors found while using sun_checks coding standards

Fig 2 Errors fixed using sun_checks coding standards

RESULTS

DEVELOPED GUI INTERFACE:

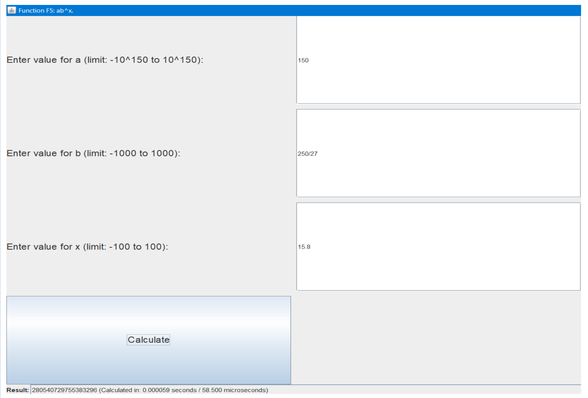

Fig 3 Function Calculator

TESTING AND VALIDATION:

- Implemented unit tests using JUnit for method 'calculate F5'.
- Ensured all edge cases are covered, including positive, negative, and zero values.
- Regular use of Checkstyle to maintain code quality and consistency.

DEBUGGER USED: JDB

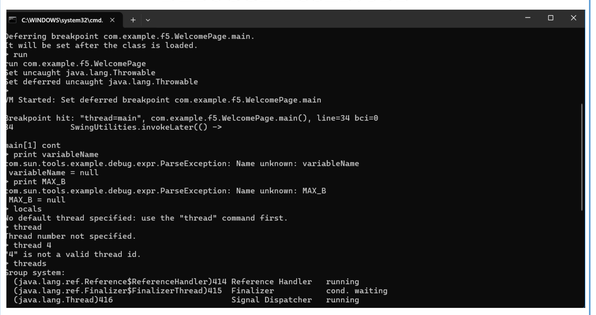

Fig 4 JDB Debugging commands

STATIC CODE ANALYZER: SONARLINT

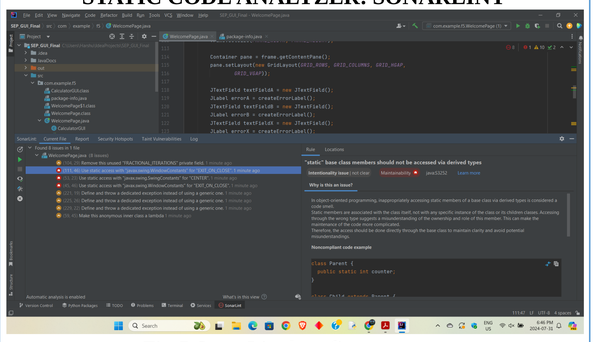

Fig 5 SonarLint bug fixes

CONCLUSIONS

The application successfully calculates the value of the function ab^x using user-provided inputs.

Real-time input validation ensures that users enter values within the specified limits, providing immediate feedback for errors.

JUnit tests validate the core functionalities of the application, ensuring robustness and accuracy.

The application utilizes semantic versioning to track changes and updates, ensuring that users have access to the latest features and bug fixes.

Regular use of Checkstyle improved code readability and maintainability, reinforcing the value of consistent coding practices.

The testing process highlighted the importance of iterative development, allowing for continuous improvements and refinements based on feedback and test results.

The Java Debugger (JDB) was used to step through code execution, allowing for in-depth analysis and troubleshooting of runtime behavior. It helped in identifying and resolving logical errors and improving code efficiency by examining the flow of execution and variable states.

SonarLint was integrated into the development environment to provide real-time feedback on code quality and potential issues. It was used to detect code smells, bugs, and vulnerabilities, facilitating proactive resolution of problems and ensuring adherence to best coding practices.

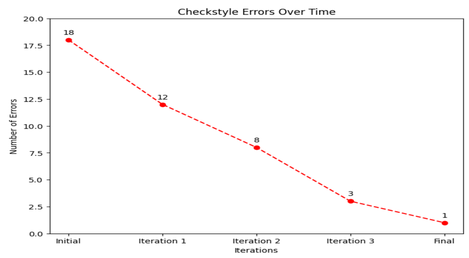

Fig 6 CheckStyle Errors over time while fixing

FEEDBACK AND IMPROVEMENTS

User feedback highlights the application's effectiveness in solving transcendental function calculations.

Suggestions for future enhancements include expanding the range of supported mathematical operations and integrating additional user interface features for improved interaction.

REFERENCES

- https://mathinsight.org/exponential_function
- https://softwareengineering.stackexchange.com/questions/72761/how-do-you-identify-edge-cases-on-algorithms
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2655855/
- https://www.nngroup.com/articles/error-message-guidelines/
- https://cheesecakelabs.com/blog/edge-cases/