A-CSC2102 : DATA STRUCTURES AND ALGORITHM

REPORT

ToeXhek – A Number Guessing Game

GROUP MEMBERS :
MUHAMMAD AHMED / 2412115
MUHAMMAD HUZAIFA / 2412118
Muhammad Ammar/ 2412116

Project Report: ToeXhek - A Number Guessing Game

Abstract

ToeXhek is a two-player number guessing game developed in Java using Swing for an interactive graphical user interface. In the game, Player 1 sets a secret four-digit number, and Player 2 attempts to guess it within a maximum of ten attempts. Each guess is evaluated based on its distance from the secret number, and meaningful feedback is provided to guide the player.

Upon game completion, all guesses are sorted based on their distance using multiple sorting algorithms, and the execution time of each algorithm is displayed for performance comparison. The project emphasizes the practical application of **Data Structures and Algorithms**, utilizing dynamic data structures, searching techniques, multiple sorting algorithms, and time complexity analysis.

Persistent storage is achieved through **CRUD operations using text files and a MySQL database integrated via Maven**, ensuring data durability and scalability. ToeXhek effectively combines gameplay with academic learning, making it a comprehensive DSA-based project.

---

1. Introduction

ToeXhek is a Java-based interactive number guessing game designed to demonstrate real-world applications of **Data Structures and Algorithms (DSA)**. The project focuses on algorithmic thinking, performance analysis, data management, and clean modular design.

The game follows a two-player model:

- **Player 1** enters a secret four-digit number.
- **Player 2** attempts to guess the number within TEN attempts while receiving feedback based on proximity.
- After the game ends, guesses are sorted using selected algorithms and execution time is measured.

The project fulfills course requirements by integrating **efficient data structures**, **multiple algorithms**, **GUI-based interaction**, and **persistent data storage**.

---

2. Objectives

The main objectives of the ToeXhek project are:

- To apply appropriate and efficient data structures for managing game data.
- To implement and compare multiple sorting algorithms.
- To analyze algorithm performance using time complexity measurements.
- To design an intuitive and responsive GUI.
- To implement CRUD operations using text files and a MySQL database.
- To ensure fair gameplay through validation and duplicate prevention.

---

3. Technologies Used

- **Programming Language:** Java (JDK 8+)
- **GUI Framework:** Java Swing
    - Components: JFrame, JPanel, JButton, JTextField, JTextArea
    - Layouts: GridBagLayout, CardLayout
- **Data Structures:**
    - ArrayList<Integer> for storing guesses and distances
    - ArrayList<String> for maintaining history
- **Algorithms:**
    - Bubble Sort
    - Insertion Sort
    - Selection Sort
    - Merge Sort
    - Quick Sort
- **Performance Measurement:** System.nanoTime()
- **Build Tool:** Maven
- **Database:** MySQL (connected via Maven JDBC dependency)

- **Storage Mechanisms:**
    - Text File (game_history.txt)
    - MySQL Database
- **Utilities:**
    - Math.abs() for distance calculation
    - SimpleDateFormat for timestamps
    - JOptionPane for dialogs and alerts

Maven handles dependencies seamlessly, ensuring the project is portable and easy to build. For MySQL, we used Maven to include the connector library, allowing SQL queries for creating, reading, updating, and deleting game records.

---

4. Game Mechanics

4.1 Player 1 – Secret Number Entry

Player 1 inputs a secret four-digit number through a secure password field. The input is validated to ensure it contains exactly four digits. Upon successful validation, control shifts to Player 2.

4.2 Player 2 – Guessing Phase

Player 2 has a maximum of **10 attempts** to guess the secret number.

Validation rules:

- Guess must be exactly four digits.
- Duplicate guesses are not allowed (checked using linear search).

Distance Calculation:

distance = |guess − secret|

Feedback System:

- Distance = 0 → *Correct! Brilliant!!*

- Distance ≤ 50 → *Near of it!! Come Closer*
- Distance ≤ 100 → *High! or Low! But Near!!*
- Distance ≤ 250 → *Not Too Far!!*
- Otherwise → *Far Enough!!*

Each guess is stored along with feedback and timestamp.

4.3 End of Game

The game ends when:

- The correct number is guessed, or
- All 10 attempts are exhausted

At the end:

- Win/Loss message is displayed
- Guesses are sorted by distance (worst first)
- Sorting time is shown
- Options to start a new game or clear history are provided

4.4 Sorting and Performance Analysis

Users can select one of the following sorting algorithms to analyze performance:

Algorithm	**Time Complexity**
Bubble Sort	$O(n^2)$
Insertion Sort	$O(n^2)$
Selection Sort	$O(n^2)$
Merge Sort	O(n log n)
Quick Sort	O(n log n) (average)

Sorting time is calculated using System.nanoTime() and displayed in microseconds.

---

5. DSA Concepts Implemented

Data Structures

- **ArrayList:** Used for dynamic storage of guesses and distances due to fast indexed access and flexibility.
- **Linear Search:** Used to detect duplicate guesses efficiently for small datasets.

Algorithms

- Comparison-based sorting algorithms demonstrate different time complexities.
- Divide-and-conquer techniques (Merge Sort, Quick Sort) showcase optimized sorting.

---

6. CRUD Operations

6.1 Text File-Based CRUD

- **Create:** Append guesses to game_history.txt
- **Read:** Load historical records
- **Update:** Modify stored data when needed
- **Delete:** Clear history file

6.2 MySQL Database (via Maven)

- Integrated using JDBC dependency through Maven
- Stores guess, distance, feedback, and timestamp
- Supports scalable CRUD operations

Example Operations:

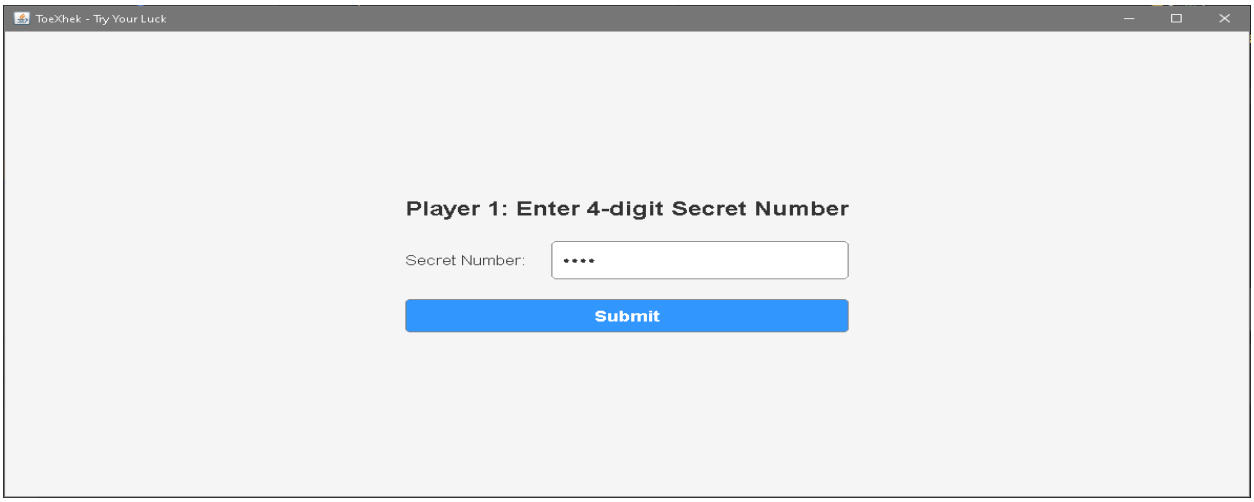
- INSERT game records
- SELECT historical guesses
- UPDATE records
- DELETE history

This hybrid storage approach ensures both simplicity and scalability.

7. Screenshots

The following screenshots illustrate the complete flow of the ToeXhek game, from startup to gameplay, validation, end results with different sorting algorithms, and history management.

1. Player 1: Enter 4-digit Secret Number (Initial Screen)

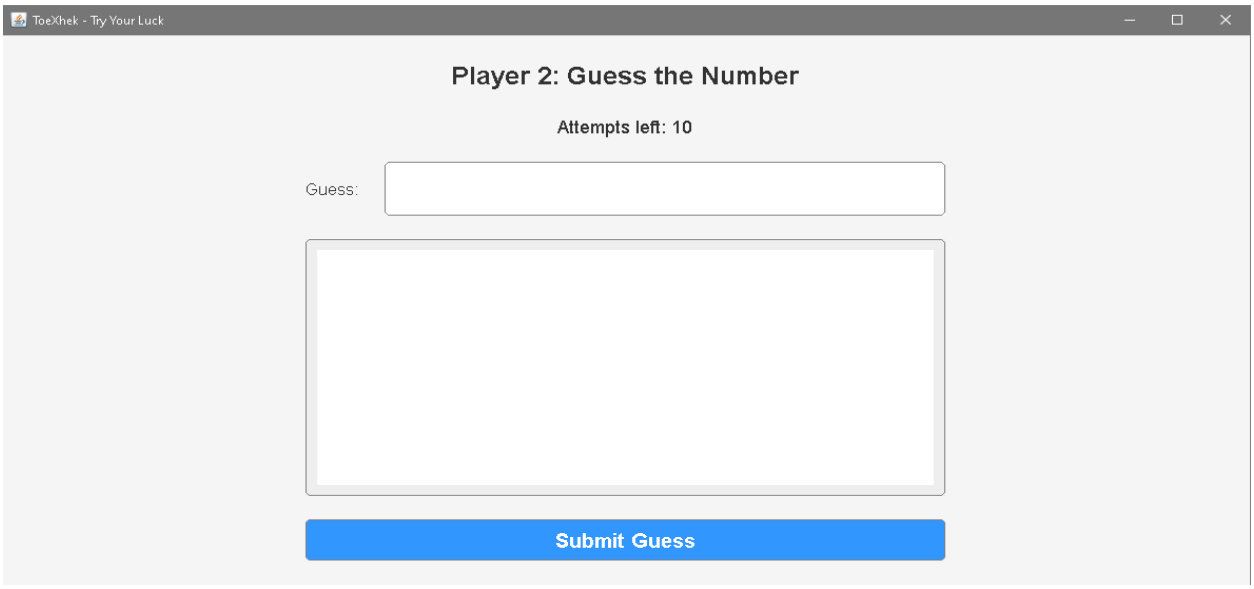


Description: Player 1 screen where the secret 4-digit number is entered using a secure password field.

2. Player 1: Validation Error (Not Exactly 4 Digits)

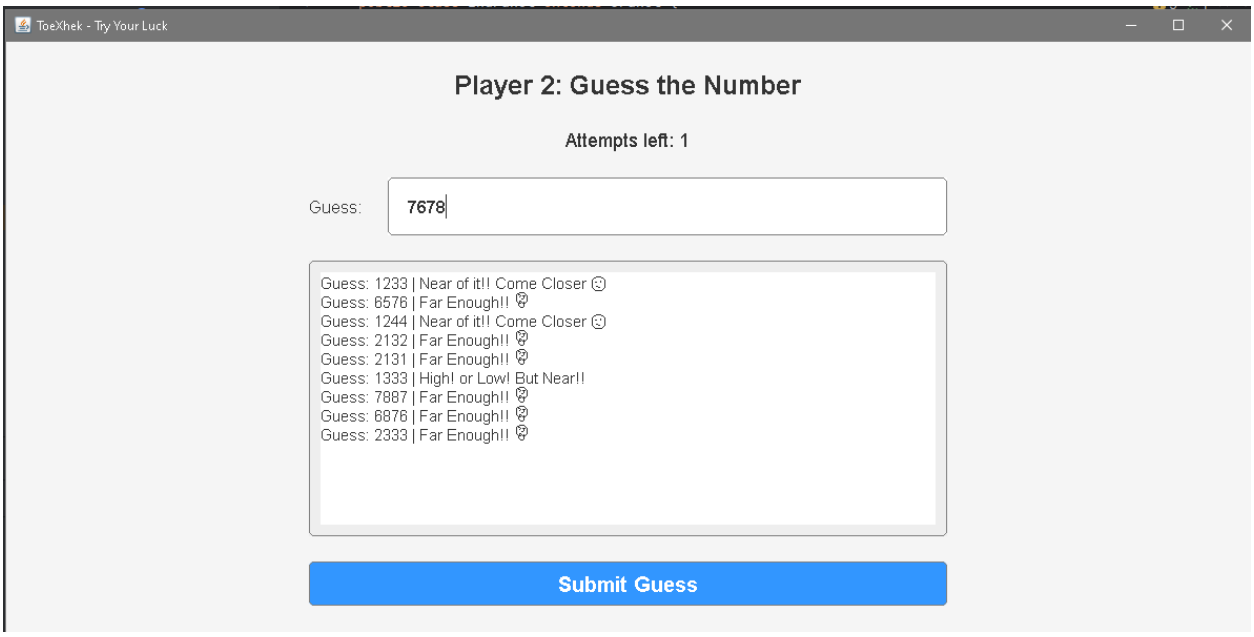
Description: Error popup shown when Player 1 enters fewer or more than 4 digits.

3. Player 2: Guessing Screen (10 Attempts Left)



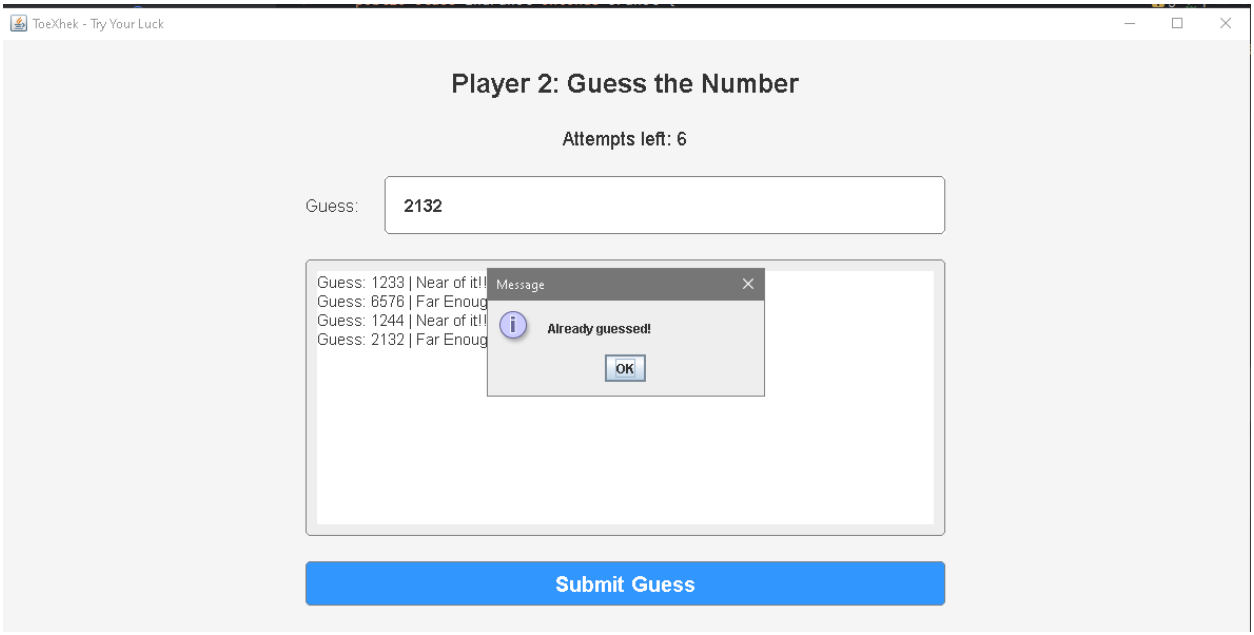
Description: Player 2's main guessing interface at the start of a new game with 10 attempts remaining.

4. Player 2: Making Guesses (Attempts)



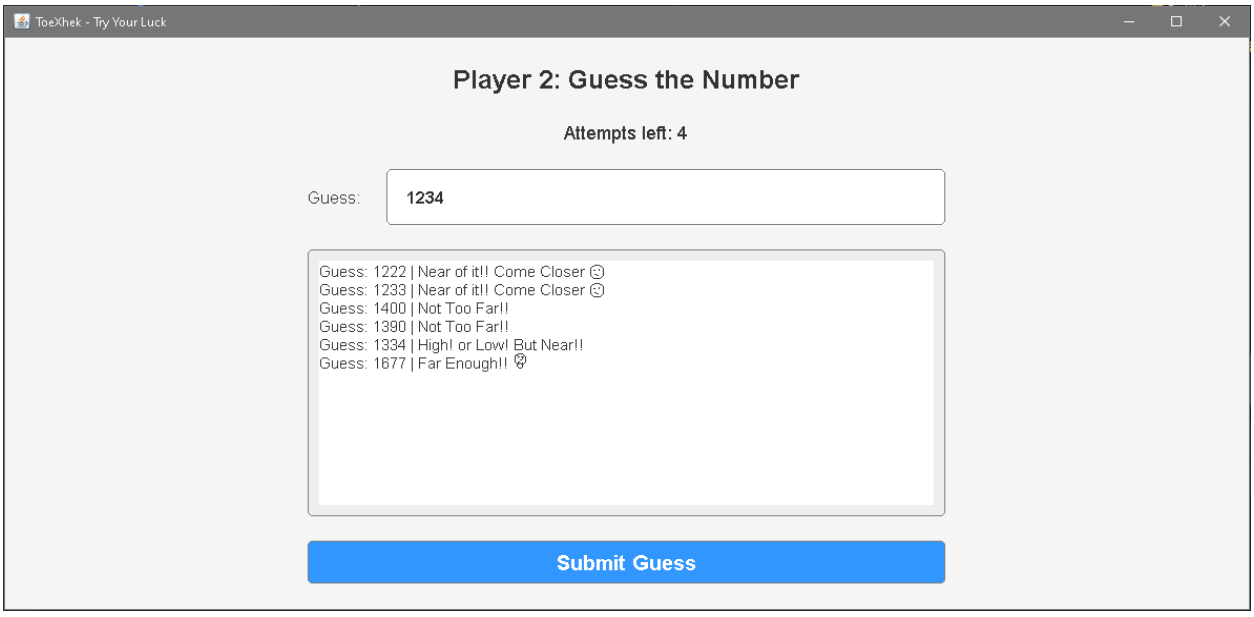
Description: Feedback displayed in real-time as Player 2 makes initial guesses.

5. Player 2: Duplicate Guess Prevention



Description: Alert popup when Player 2 tries to enter a number already guessed.

6. Player 2: Near Correct Guess



Description: Example of feedback when a guess is very close to the secret number.

7. Player 2 Wins! (Sorted with Selection Sort)

Description: End screen showing Player 2 victory with guesses sorted using Selection Sort (time: 35 µs).

8. Player 2 Wins! (Sorted with Merge Sort)

Description: Victory screen using Merge Sort for sorting (time: 52895 µs).

9. Player 2 Wins! (Sorted with Quick Sort)

Description: Victory screen using Quick Sort (time: 55654 µs).

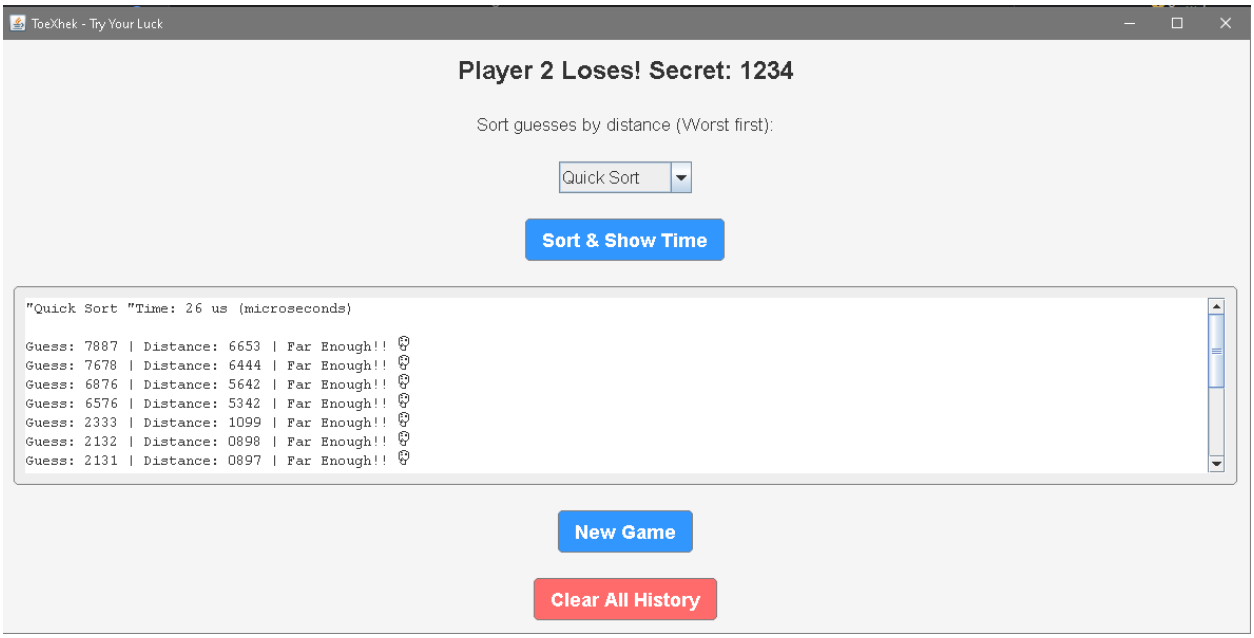
10. Player 2 Wins! (Sorted with Insertion Sort)

Description: Another win scenario sorted using Insertion Sort (time: 10119 µs).

11. Player 2 Wins! (Sorted with Bubble Sort)

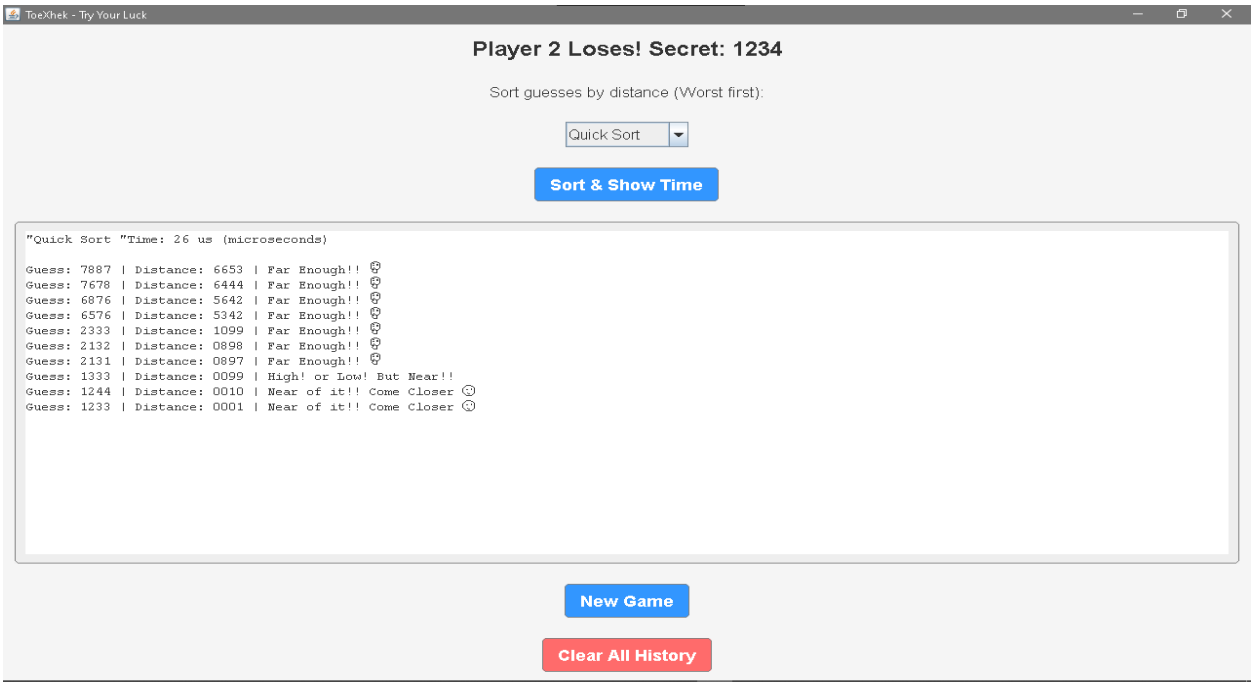
Description: Win screen demonstrating Bubble Sort performance (time: 34781 µs).

12. Player 2 Loses! (Few Guesses, Quick Sort)



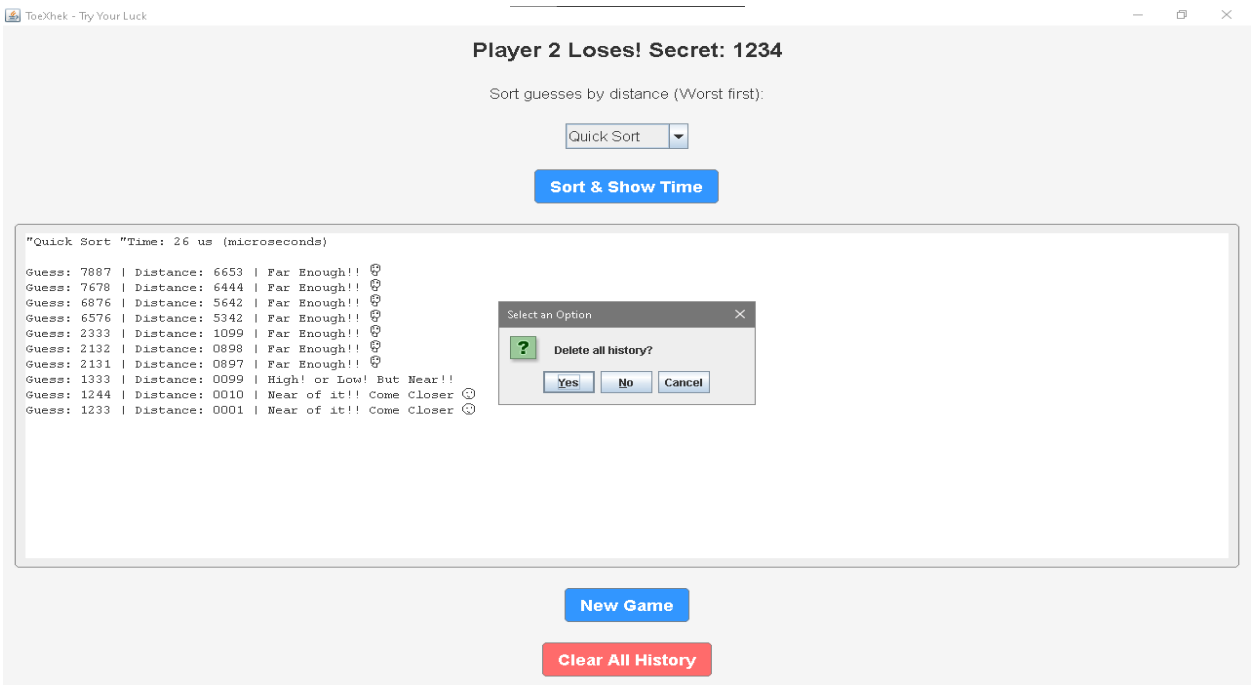
Description: Loss screen with only a few guesses made, sorted quickly using Quick Sort (time: 26 µs).

13. Player 2 Loses! (Full 10 Guesses Exhausted)



Description: Loss after exhausting all 10 attempts, revealing the secret number 1234.

14. Clear All History Confirmation



Description: Confirmation dialog before deleting all saved game history (from both text file and MySQL database).

These screenshots comprehensively demonstrate the user interface, input validation, gameplay mechanics, algorithmic sorting with performance timing, and data management features of ToeXhek.

---

8. Implementation Overview

Key Classes:

- GameFrame – Main window with CardLayout
- Player1Panel – Secret number input
- Player2Panel – Guess handling and feedback
- EndPanel – Sorting results and performance
- GameData – Central data manager
- Sorter & Sorting Classes – Algorithm implementations
- History – File and database persistence
- RoundedButton, RoundedBorder – Custom GUI components

The application follows an event-driven design using ActionListeners.

---

9. Challenges and Learning Outcomes

Challenges

- Managing GUI transitions smoothly
- Preventing duplicate guesses
- Measuring execution time accurately
- Integrating MySQL with Maven
- Handling edge cases in gameplay

Learning Outcomes

- Strong understanding of DSA in practical applications
- Experience with Swing GUI development
- Hands-on use of sorting algorithms and complexity analysis

- Database integration using Maven and JDBC
- Modular and maintainable software design

---

10. Conclusion

ToeXhek successfully integrates entertainment with academic rigor by applying core **Data Structures and Algorithms** concepts in a real-world game scenario. Through dynamic data structures, multiple sorting algorithms, time complexity analysis, and persistent storage using both text files and MySQL (via Maven), the project meets and exceeds course requirements. It demonstrates efficient algorithmic thinking, clean design, and scalability, making it a strong and complete DSA course project.