Assignment Title: Implementing and Analyzing Cryptographic Hashing Algorithms in Java

---

Assignment Description:

Hashing algorithms are integral to cryptography, providing mechanisms for data integrity, password storage, and digital signatures. In this assignment, you will implement a Java program to compute hash values of given inputs using various cryptographic hashing algorithms. You will also analyze the properties of these algorithms, such as collision resistance and output consistency.

Your program will:

1. Compute the hash of a user-provided input using multiple algorithms (e.g., MD5, SHA-1, SHA-256).
2. Allow users to test collision resistance by hashing multiple inputs.
3. Provide insights into the strength and suitability of each algorithm.

Requirements:

1. **Input and Output**:
    - Accept a plaintext string from the user as input.
    - Display the computed hash values for the input using MD5, SHA-1, and SHA-256.
    - Use Base64 or hexadecimal encoding to display hash outputs.
2. **Hashing Implementation**:
    - Use Java's java.security package to implement hashing algorithms.
    - Ensure consistency by computing hash values with a fixed encoding (e.g., UTF-8).
3. **Collision Testing**:
    - Allow users to provide multiple inputs to test for collisions (e.g., same hash for different inputs).
    - Display a clear message indicating whether a collision occurred.

4. **Algorithm Comparison**:
    - Provide a brief explanation of the strengths and weaknesses of each algorithm in the program output or comments.
5. **Error Handling**:
    - Handle edge cases, such as empty inputs.
    - Provide meaningful error messages for invalid operations.
6. **Documentation and Testing**:
    - Include comments in the code explaining each significant step and component.
    - Provide at least three test cases, including inputs that demonstrate unique outputs and potential collisions.
7. **Optional Enhancements** (Extra Credit):
    - Implement a password hashing function using PBKDF2.
    - Add a feature to hash contents of a file instead of a single string.

Deliverables:

1. **Java Source Code**:
    - Submit .java file(s), ensuring the code is well-documented and follows Java coding standards.
2. **Test Results**:
    - Provide a document summarizing test cases with:
        - Input values.
        - Hash outputs for each algorithm.
        - Notes on collision detection (if applicable).
3. **Readme**:
    - Include a README file in Word or PDF format explaining how to run your program, its dependencies, and any optional features.

4. **Optional Enhancements** (Extra Credit):
    - Demonstrate the use of password hashing or file-based hashing with additional examples.

Submission Guidelines:

1. Submit your .java file(s) and test results via Blackboard.
2. Include a README file in Word or PDF format explaining how to run your program, its dependencies, and any optional features.

10-Point Rubric:

Criteria	Points	Description
Correct Implementation of Hashing Algorithms	2	Successfully implements MD5, SHA-1, and SHA-256 hashing algorithms.
User Input Handling	1	Handles plaintext input with proper validations and error handling.
Collision Detection	1	Accurately detects and reports collisions during testing.
Output Clarity	1	Clearly displays hash outputs in Base64 or hexadecimal format.
Algorithm Comparison	1	Provides clear insights into the strengths and weaknesses of each algorithm.
Error Handling	1	Provides meaningful error messages for invalid inputs or operations.
Documentation/Comments	1	Includes clear comments explaining major code sections and logic.
Test Cases and Results	1	Provides at least three test cases demonstrating correct hashing and collision resistance.
Optional Enhancements	1	Implements password hashing or file hashing as extra features. (Extra Credit)
Overall Functionality	1	Program runs correctly and achieves the goals of the assignment.
