Assignment Title: Implementing the Bell-LaPadula Model in Java

---

Assignment Description:

The Bell-LaPadula (BLP) model is a cornerstone of access control in information security, focusing on confidentiality. In this assignment, you will implement a Java program to enforce the principles of the BLP model: the *simple security property* ("no read up") and the *-property* ("no write down"). Your program will simulate a secure system with users, objects, and access levels, enforcing BLP policies for read and write operations.

Your program will:

1. Define users and objects with distinct security levels.
2. Enforce BLP rules for access requests.
3. Provide an interface for users to perform read and write operations.

Requirements:

1. **System Components**:
    - Define a set of security levels (e.g., Unclassified, Confidential, Secret, Top Secret).
    - Define users and objects, each assigned a specific security level.
2. **Access Control Rules**:
    - Enforce the *simple security property*: users cannot read objects with a higher security level.
    - Enforce the *-property*: users cannot write to objects with a lower security level.
3. **Input and Output**:
    - Allow the user to simulate operations by specifying:
        - User ID
        - Object ID
        - Operation (read/write)

    - Display whether the operation is permitted or denied based on BLP rules.

4. **Error Handling**:
    - Handle invalid inputs (e.g., non-existent users or objects).
    - Provide meaningful error messages for denied access.
5. **Documentation and Testing**:
    - Include comments explaining each part of the code.
    - Provide test cases to demonstrate the enforcement of BLP rules.

Deliverables:

1. **Java Source Code**:
    - Submit the .java file(s), ensuring code is clean, well-structured, and documented.
2. **Test Results**:
    - Provide a document summarizing test cases, including:
        - User and object security levels.
        - Attempted operations (read/write).
        - Whether the operations were allowed or denied.
3. **Readme**:
    - Include a short README file in Word or PDF format explaining how to run your program and any dependencies.
4. **Optional Enhancements** (Extra Credit):
    - Implement dynamic role changes for users and objects, updating security levels.
    - Add a logging feature to track all access requests and their outcomes.

Submission Guidelines:

1. Submit your .java file(s) and test results via Blackboard.

2. Include a short README file in Word or PDF format explaining how to run your program and any dependencies.

10-Point Rubric:

Criteria	Points	Description
Correct Enforcement of Simple Security	2	Accurately prevents users from reading objects at higher security levels.
Correct Enforcement of -Property	1	Accurately prevents users from writing to objects at lower security levels.
Security Level Implementation	1	Properly defines and applies security levels to users and objects.
User Input Handling	1	Handles input for user IDs, object IDs, and operations with appropriate validations.
Output Clarity	1	Clearly indicates whether operations are allowed or denied and why.
Error Handling	1	Provides meaningful error messages for invalid inputs or denied 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 enforcement of BLP rules.
Optional Enhancements	1	Implements dynamic role changes or an access log (extra credit).
Overall Functionality	1	Program runs correctly and enforces BLP rules as specified.