Your Name _____

ISM Exam January 31, 2025 (Java - JCA)

- **The Java solution is developed in the given single .java file.**
- **Rename the given package with your name**
- **For each requirement the implementation must produce CORRECT results to be evaluated (as you are allowed to use laboratory examples).**
- **You are responsible for defining in that file all the classes that you need to solve the problem. If the solution depends on external source code files (not Java libraries) it will not be compiled and it will not be evaluated.**
- **The use of Bouncy Castle library is optional (my recommendation is NOT to use it)**
- **All submitted solutions must be without compiler errors (0 errors)**
- **All the solutions will be cross-checked with MOSS from Stanford and very similar source code files (more than 50%) will not be evaluated.**

All Java requirements have 4,5 points, all C++ requirements have 4,5 points, 1 point is for showing up

Each student will use the values given in the "SAP 2025 Exam.xlsx" file. Search for your name in that file.

The tasks are sequentially dependent, so you should test each step thoroughly before moving to the next

Subject: Secure Document Manager for SecureDocs Inc.

Scenario: You are a developer for **SecureDocs Inc.**, a company specializing in secure digital document storage. Your task is to create a secure document management system to store, authenticate, and encrypt company documents. Each requirement must be implemented in sequence to ensure the data is correctly processed and secured.

Step-by-Step Requirements ()

1. **Document Integrity Check (1 point)**:
    - **Scenario**: SecureDocs Inc. needs to ensure that documents stored in their system have not been tampered with.
    - **Task**: Implement the *generateFilesDigest()* method that takes the path of the local "messages" folder and computes the **Message Digest** (MD5) of all the files in that location. **For each file** save the digest as **a hex string** in a separate file called *<filename>.digest*.
    - **For example**, for the file *message_10_5emaqc.txt* you should generate a file named *message_10_5emaqc.digest* with the next value "CEC3C3 ..." (the case does not matter)

Your Name _____

2. **Secure Document Transfer with Authentication (1 point)**:
    - **Scenario**: Documents transferred between systems must be verified for authenticity.
    - **Task**: Implement an **HMAC** (using SHA-1) to authenticate the document. The HMAC should be based on a shared secret key provided by SecureDocs Inc. Check the given Excel file for your name. Save the HMAC value of each file, **as Base64 text**, in *<filename>.hmac*. Do this for all the files in the given folder. This step relies on the integrity check done in Step 1 to confirm the file is authentic. The HMAC value is stored as Base64 text to provide portability between different systems.
3. **Document Retrieval and Integrity Check (1 point)**:
    - **Scenario**: During retrieval, SecureDocs Inc. must verify the integrity and authenticity of the document.
    - **Task**: Implement the step 3 method verifies the document's **HMAC**, and re-generates the **Message Digest** to compare it with the original values stored in the .hmac and .digest files. If the HMAC and digest match, display the document content. The method must recompute the 2 values and compare them with the ones from the files (you can compare them at byte level or hex string or Base64 level – your choice)
4. **Generate key for encryption (1 point)**
    - **Scenario**: Documents need to be encrypted before storage. An encryption key must be generated from the shared secret
    - **Task**: Implement the *generateSecretKey* method by processing the given shared secret **by flipping a specific bit** (if the byte is 1 make it 0, if is 0 make it 1. The specific bit is given in the Excel file. The method should return a byte array 128 bits
5. **Secure Document Storage with Encryption (0,5 points)**:
    - **Scenario**: Documents need to be encrypted before storage.
    - **Task**: Using **AES encryption in ECB mode** and the previous generated key, encrypt a given document content. Save the encrypted content to <filename>.enc. Use PKCS 5 padding.