

Ministry of Education, Culture and Research of the Republic of Moldova

Technical University of Moldova

Department of Software and Automation Engineering

**REPORT**

Laboratory work No. 1

**Discipline**: Cryptography and Security

Elaborated: FAF-221, Cusnir Grigorii

Checked: asist. univ., Dumitru Nirca

Chișinău 2023

**Introduction**

This report discusses the implementation of the Caesar cipher in Java, covering two tasks that aim to provide encryption and decryption functionality with specific constraints. The Caesar cipher is a substitution cipher where each letter in the plaintext is shifted by a fixed number of positions down the alphabet. The two tasks include:

**Task 1:** Implementing the Caesar cipher for the English alphabet, where the user can encrypt or decrypt a message using a key between 1 and 25.

**Task 2:** Extending the Caesar cipher to use two keys—one numeric key for shifting the letters and a second key consisting of Latin letters of at least 7 characters long, used to create a custom alphabet.

**Task 1: Caesar Cipher with Single Key**

**Objective:**

Implement the Caesar cipher for the English alphabet, where:

* The encryption key is between 1 and 25.
* The message consists only of uppercase and lowercase English letters (A-Z, a-z).
* The message is converted to uppercase before encryption.
* Spaces in the message are removed.
* The user can choose between encryption and decryption.

**Process:**

**Input Validation:**

* The user enters a message and a key (between 1 and 25).
* Before encryption, the message is converted to uppercase, and spaces are removed.
* If the user enters characters outside the allowed range ('A' to 'Z' and 'a' to 'z'), the program will prompt for a valid input.

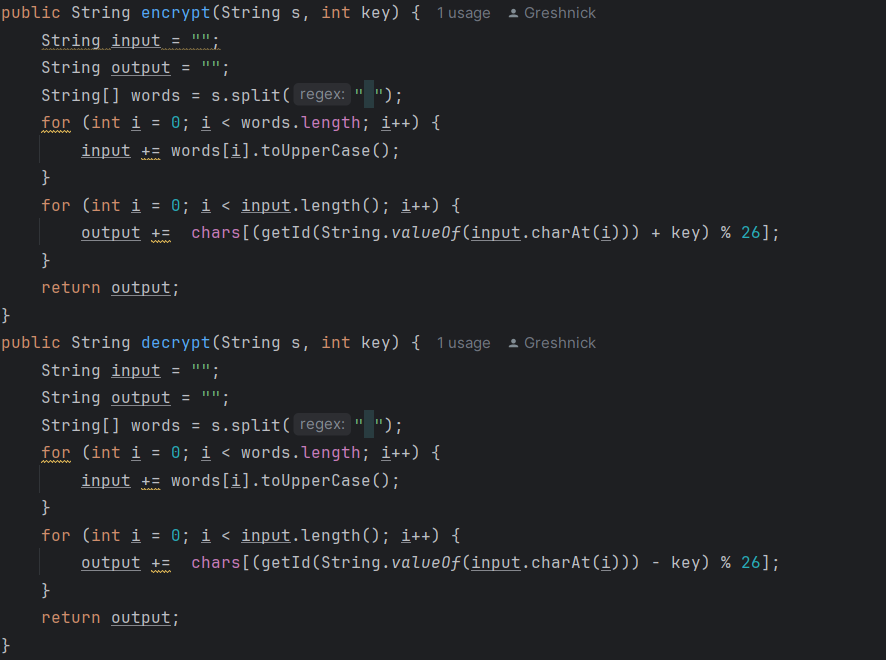
**Encryption/Decryption:**

* For encryption, each character's position in the alphabet is shifted by the key value.
* For decryption, the reverse process is applied, shifting the characters by the negative of the key.

**Output:**

The program outputs either the encrypted message (cryptogram) or the decrypted message based on the user's choice.

**Code:**



**Example:**

For an input message "HELLO" and a key of 3:

Encryption: The result would be "KHOOR".

Decryption: The result would be "HELLO".

**Task 2: Caesar Cipher with Two Keys**

**Objective:**

Extend the Caesar cipher to use two keys:

* The first key is a numeric value between 1 and 25, which is used to shift characters in the message.
* The second key is a string consisting of Latin alphabet characters (minimum length of 7), which is used to generate a custom alphabet for the cipher.

**Process:**

Key Validation:

* The first key (numeric) must be between 1 and 25.
* The second key must be at least 7 characters long and contain only alphabetic characters.
* The custom alphabet is generated by using the characters from the second key (without repetitions) followed by the unused letters from the alphabet.

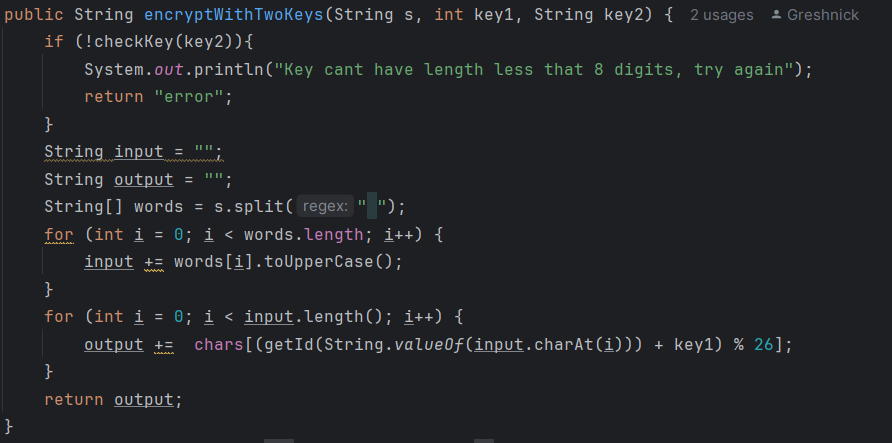
**Input Conversion:**

* The user enters a message and two keys: the numeric key and the string key.
* The message is converted to uppercase, and spaces are removed.

**Encryption/Decryption:**

* The first key is used to shift characters based on the custom alphabet generated from the second key.
* For encryption, each character's position in the custom alphabet is shifted by the first key.
* For decryption, the reverse process is applied by shifting the character back.

**Code:**



**Output:**

The program outputs either the encrypted message (cryptogram) or the decrypted message based on the user's choice.

**Conclusion**

The implementation of the Caesar cipher in Java fulfills the requirements of both tasks effectively:

**Task 1:** The Caesar cipher allows users to encrypt and decrypt messages using a single numeric key, with input validation to ensure only valid characters are processed.

**Task 2:** The Caesar cipher with two keys provides additional flexibility and security by allowing the use of a second key to generate a custom alphabet. It ensures that the second key meets the required conditions and provides a secure way to encrypt and decrypt messages.

Both implementations are modular and easy to use, providing a clear example of how substitution ciphers can be extended and customized in a real-world application.