Technical Documentation for Enigma Machine Simulator on Java.

Nikita Tkachyov

0.Prerequisites for Enigma Machine Simulator Project

1. Software Requirements

Java Development Kit (JDK) 17:

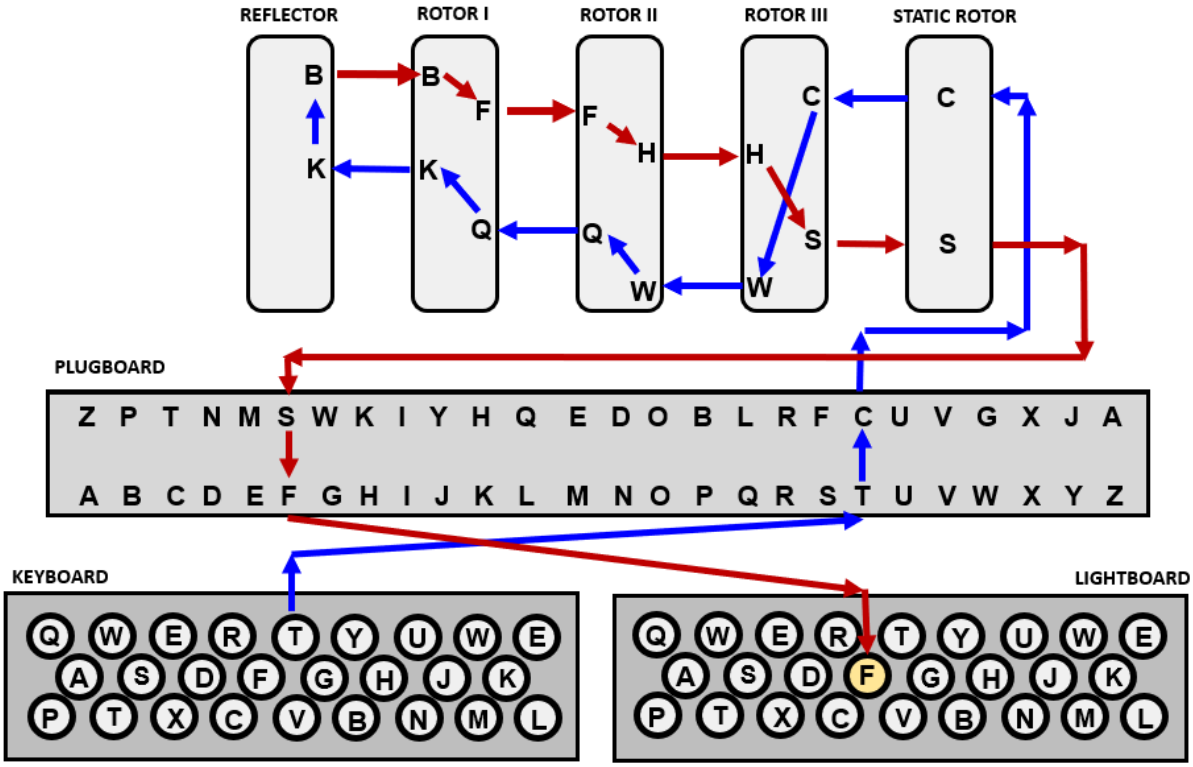
The project is configured to use Java 17, as specified in the pom.xml file (maven.compiler.source and maven.compiler.target are set to 17). Ensure that JDK 17 is installed on your machine. You can download it from the official Oracle JDK website or use an open-source alternative like OpenJDK.

Apache Maven:

Maven is used for project management and build automation. It is essential for compiling the project and managing dependencies. You can download Maven from the Apache Maven website and follow the installation instructions.

1. Overview

The Enigma Machine Simulator is a Java-based application that simulates the functionality of the historical Enigma machine, used for encrypting and decrypting messages. This simulator replicates various aspects of the original Enigma, including plugboard settings, rotor configurations, and reflector choices, offering a comprehensive tool for both educational and cryptographic experimentation.



The path taken by a letter through an Enigma machine as it is encrypted

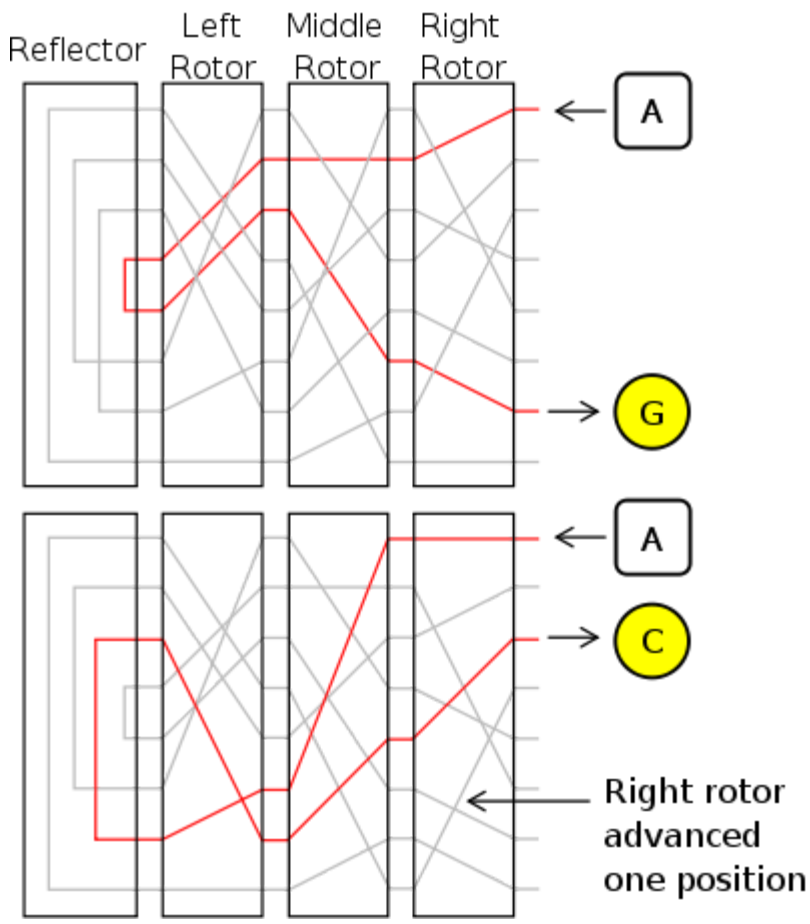
2. Main Features

2.1 Libraries Used

- Java Utility Library (java.util.*): Utilized for data structures such as Maps and Lists that store the configurations and state of the Enigma machine.
- Java Regex (java.util.regex.*): For validating and processing string patterns, especially useful in setting up the plugboard connections.
- Java Stream API: Employs stream operations to simplify complex data transformations, particularly in the mapping of rotor configurations.

2.2 Components

- Plugboard (Steckerbrett): Allows for the configuration of letter swaps, simulating the plugboard of the Enigma machine.
- Rotors (Walzen): Simulate the rotating disks with wiring that permutes the letters. Includes functionality to choose between three and four rotors setups, including the Kriegsmarine (naval) version with a thin rotor.
- Reflectors (Umkehrwalze): Implements the reversing mechanism of the Enigma, which is essential for the encryption and decryption processes.
- Rotor Settings and Configuration: Users can customize the initial settings of the rotors, including the ring settings (Ringstellung) and the rotor position (Grundstellung).



3. Backend Implementation

3.1 Methods and Functionalities

- **Program Class**

1. **main(String[] args):**

   Manages the overall flow of the program, including initializing the plugboard and rotors, handling user input, and encoding messages.

2. **create(String name, boolean kriegsmarine, Scanner scanner):**

   Guides the user through the process of configuring a rotor, including setting the ring position and offset.

3. **encode(char lett):**

   Encodes a single character by passing it through the configured rotors, reflector, and plugboard.

4. **rotate(char ch, int n, int z):**

Adjusts a character's position within the alphabet based on a specified shift and offset.

3.2 Rotor Class

1. **Constructor:**
   Initializes the rotor with specific wiring, turnover positions, and ring settings.
2. **Getter Methods:**
   Retrieve the rotor's wiring, current position, turnover positions, ring setting, and reverse wiring configuration.
3. **Setter Methods:**
   Adjust the rotor's current position.

3.3 Library Class

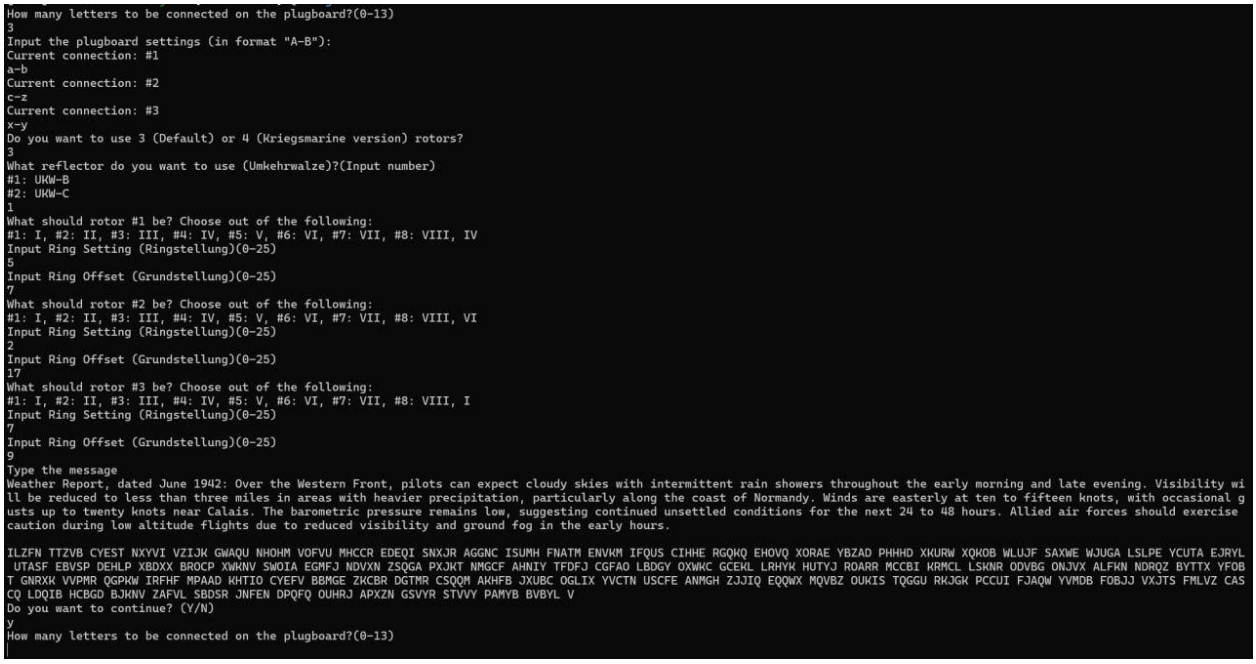
1. **masterDictR:**
   Contains mappings for standard reflectors (e.g., UKW-B, UKW-C).
2. **masterDictKR:**
   Contains mappings for Kriegsmarine thin reflectors.
3. **reflector & reflectorK:**
   Lists of reflector configurations.
4. **masterDictT & masterDictKT:**
   Contains turnover positions for standard and Kriegsmarine rotors.
5. **masterDict & masterDictK:**
   Contains wiring configurations for standard and Kriegsmarine rotors.

4. Compilation and Execution

- Compile the Java code: Use “mvn clean compile” to compile the project.

```
PS D:\Enigma_Machine - Copy> mvn clean compile
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< org.example:enigma-machine >--------------------
[INFO] Building enigma-machine 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ enigma-machine ---
[INFO] Deleting D:\Enigma_Machine - Copy\target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ enigma-machine ---
[INFO] Copying 0 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.10.1:compile (default-compile) @ enigma-machine ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to D:\Enigma_Machine - Copy\target\classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.135 s
[INFO] Finished at: 2024-05-23T14:27:51+02:00
[INFO] ------------------------------------------------------------------------
PS D:\Enigma_Machine - Copy> mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< org.example:enigma-machine >--------------------
[INFO] Building enigma-machine 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec:3.1.0:java (default-cli) @ enigma-machine ---
How many letters to be connected on the plugboard?(0-13)
3
```

- Run the simulator: Execute java org.example.Program with “mvn exec:java” to start the simulation. Follow on-screen prompts to configure and use the machine.



5. Future Enhancements

- GUI Implementation: Developing a graphical user interface can make the simulator more user-friendly and visually appealing.
- Extended Rotor and Reflector Configurations: Adding more rotor and reflector configurations could provide more encryption variations and historical accuracy.
- Persistence: Saving and loading configurations can allow users to resume their work or experiments without needing to reconfigure the settings.
- Uhr: Implement the Steckerbrett addon called “Uhr” or “Clock” for additional encryption complexity.