Cracking The Enigma, version 2.

# **Description**

Version 2 of this project allows a user to load a working Enigma machine (for a detailed explanation on the machine, please refer to the following link: <https://www.youtube.com/watch?v=ybkkiGtJmkM>), configure a loaded machine, encrypt messages and attempt to crack an encrypted code as a JavaFX application.

# **Changes from VER 1**

Version 2 of Cracking The Enigma lets the user attempt to decrypt a message encrypted under the TAB ‘Brute Force’.

# **How to run ‘Cracking the Enigma Ver 2’**

Run the Run.BAT file located under the folder ‘Executable’. This operation requires Java 8 installed on the system to run.

# **How to use ‘Cracking the Enigma Ver 2’**

Start by loading a new machine using the upload button and upload an XML file of the machine. Configure the machine (either randomly using the toggle switch or manually). Press ‘SET’ afterwards. Once the machine has been set the user is then able to visit the ‘Encryption’ and ‘Brute Force’ tabs.

**Encryption tab** – allows the user to insert any sentence that matches the machine’s acceptable ABC (mentioned in the XML file) and encrypt the sentence. The user can also do it on manual mode by seeing each letter being encrypted as he types and see a nice animation of the keyboard lighting up!

**Brute Force tab –** a user can attempt to decrypt a message. For the sake of simplicity, the user is only allowed to insert words (separated by spaces) from a given dictionary mentioned in the XML file of the machine. As the user types a popup will appear with suggested words to insert. Once a message has been encrypted, the user can configure how to decrypt the message: choose amount of ‘agents’ amount of tasks each agent will perform and the difficulty.

**Easy:** Only the rotor’s configuration is unknown, the decryption process will try all configuration combinations.

**Medium:** Rotor’s configuration and reflector are unknown, the decryption process will try all reflectors and all configuration combinations in every reflector.

**Hard:** Rotor’s configuration, reflectors and the order of the rotors chosen are unknown, the process includes trying all positionings of the rotors, all reflector combinations for each positioning and all configuration combinations for each reflector.

**Impossible:** Not a single part is known, all possible part combinations will be tried.

**\*Plugs are not used in the decryption process due to inflation of possibilities.**

The user can then start the decrypting process which might take some time, therefore the process is done using multithreading to allow the user to continue using the application. While waiting for the process to finish, candidate words/sentences that were found will be displayed in the Table View below the decryption’s configuration along with the parts that were used in the machine.

# **Technologies used**

* Written in **Java 8** using **Intelij** ide.
* Used **JAXB** for xml file loading.
* Used **JavaFX** and **SceneBuilder** to develop the GUI of the application. Written in **MVC** fashion.
* Used **Multithreading** for the ‘Brute force’ process. The agents represent a thread pool corresponding the amount of agents chosen. This along with a BlockingQueue allowed a new thread to push new tasks to agents waiting in the pool to attempt to find candidates and push them to a different BlockingQueue which has a different thread listening to it to send the results to the **JAT** and update the UI with a possible candidate.