**We implemented bonus #3**

**SUBMITTERS**

**Name**: Shani Shira Grego

**Id**: 318919388

**Email**: Shanishiragr@mta.ac.il

**Name**: Adi Rozenberg

**Id**: 207380221

**Email**: [adirz@mta.ac.il](mailto:adirz@mta.ac.il)

**ABOUT BREAKING THE ENIGMA PROJECT - EX02**

In this exercise we created 5 modules:

1. EnigmaMachine: (main packages and classes)

includes components package:

this package includes 3 classes:

**Rotor class** - we chose to implement the rotor class with two ArrayList<Character> one for left and the other for right. there are more data members such as the id of the rotor, number of signs, start position of rotor and position of notch.   
The class's main functions are move rotor, move notch, mapping left to right and the opposite direction - mapping right to left, and more getters and setters.

**Reflector class** - we chose to implement the reflector class with data members of Map<Integer, Integer> for the content of the reflector - each in integer has an out integer, and also an id of reflector data member. this class's main function has a mapping function.

**Plug class** - we chose to implement plug class with two data members - char sign1, char sign2.

EngiaMachine module also includes a Configuration class, a Coding interface and an EnigmaMachine class. Coding interface has two functions - activate and init.  
EnigmaMachine class implements coding interface, and implements the machine by defining data members of ArrayList<Rotor> rotors, Reflector reflector, ArrayList<Plug> plugs, and more. this class implements functions such as change according to plugs, init machine and more.

1. Engine: (main packages and classes)

includes engine package:

this package includes 3 classes:

**ConfigurationDto class -** this class mediates between logic and ui.

**Engine class** - this class implements part of the logic layer. It has a reference to the EnigmaMachine class, a dto, and many more data members.  
this class implements engineering interface functions and many more functions, that take the cte enigma machine, and make it to the enigma machine

**Engineering interface** - this interface includes initialize, activate and createMachine functions.

includes a factory package - that include only one class **- Factory,** that its goal is to create our enigma machine from the cte machine we've got from xml file.

1. Ui – for exercise 01 : (main packages and classes)

includes playMachine package - that includes an enum of manu options, and also a **PlayMachine class** - this class has a reference of the Engine, and gets all inputs from user, and handles it.  
it also includes functions of print messages to the screen, and catch exceptions and gives a specific message to the screen. this class has a play function that the whole enigma machine works with it.

1. **Brute Force module:** (main packages and classes)

This module manages all the brute force. It has a DM class – which has all data member and functions that do the brute force process. This class also has a copy of the enigma machine as a data member. We have also an AgentTask class, and from that class we create tasks.  
we also have the Engima Dictionary class in that modulw, that contain all details that belongs to the dictionary, like all of the words we've got from the xml file.

1. **tryJavaFX module:** (main packages and classes)

this module contains all packages that belong to the new screen we built to the machine throughout SceneBuilder.

We divided the screen into components:  
1. Machine.

2. encrypt / decrypt.

3. brute force.

4. header.

5. tabs – includes all 1-3 components.

6. main app – includes tabs and header.

Each component has its own package with a controller class that manage it, and make it possible to handle with all buttons and more.

It also has a CSS file that makes it possible to design the screen.