The Enigma Cipher

As long ago as the Ancient Greeks, warring armies have encrypted their communications in an attempt to keep their battle plans a secret from their enemies. However, just as one side invented an ingenious new way to encipher its messages, so would its enemies discover a clever way of cracking that code. The result has been that codes and ciphers have become more and more complex and increasingly difficult to crack over time, as, throughout history, an intellectual battle has raged between code makers and code breakers.

The battle of wits was never keener than during the Second World War, when the Germans used the famous Enigma machine - which they believed uncrackable - to encode messages due to there being approximately 150,000,000,000,000 - that is, 150 million million - possible combinations for configuring an enigma machine and enciphering text.

History of the Enigma Machine

Up till the Second World War, the most advanced forms of encryption involved simple paper and pencil techniques. But security blunders on both sides during the First World War highlighted a need for a higher level of secrecy, with more advanced methods of enciphering messages. Both the Allies and the Axis countries were looking for a new way to encrypt messages - a way that would result in complete security.

In 1915 two Dutch Naval officers had invented a machine to encrypt messages. This encryption tool became one of the most notorious of all time: the Enigma cipher machine. Arthur Scherbius, a German businessman, patented the Enigma in 1918 and began selling it commercially to banks and businesses.

The Enigma machine's place in history was secured in 1924 when the German armed forces began using a specially adapted military version to encrypt their communications. They continued to rely on the machine throughout the Second World War.

Deciphering Enigma

When the Enigma machine is used, the Enigma machine itself is the algorithm; the way in which it is set up is the key. Just as with any other type of cipher, as long as the recipient knows the key, the process of deciphering an Enigma encrypted message is incredibly simple. A German soldier receiving an enciphered message simply had to type the ciphertext letters into his own Enigma machine. If his machine was set up exactly in exactly the same way as the message sender's, then the plaintext letters would appear on a lamp board -- similar to a keyboard, except the buttons would light up showing the decrypted character.

However, just as with any other type of cipher system, if you don't know the key it is very difficult to read the message - even if you know which system was used to encipher it.

The German army was distributed a Key Sheet each month, describing the settings required for each day of that month.

A common format for enigma was characters were sent in sets of 5, ie `HOMER SIMPS ON`

Faults of Enigma

Even today, it would take months to decipher an enigma-encryped message with modern computers. This assumes two things:

- Messages are unique and do not contain same text.
    - The German Army often sent their messages starting with the word *Spruchnummer* (message number)
    - Weather Reports containing *Keinebesondere Ereignisse* (nothing to report)
    - Messages always ended with *Heil Hitler*
- You don't have a physical machine and/or the key sheet

The only other downfall is a character can never be encoded as itself.

With knowing some key words, eventually a computer or a physical machine (like in WWII) can guess what the configuration is for the enigma machine. With the settings known, all is needed is an engima machine.

Components of an Enigma Machine

When a plaintext letter was typed on the keyboard, an electric current would pass through the different scrambling elements of the machine and light up a ciphertext letter on the "lamp board". What made the Enigma machine so special was the fact that every time a letter was pressed, the movable parts of the machine would change position so that the next time the same letter was pressed, it would most likely be enciphered as something different. This meant that it wasn't possible to use traditional methods to try and crack the notorious cipher.

To make things even more difficult, different parts of the machine could be set up in different ways, with each setting producing a unique stream of enciphered letters. Unless you knew the exact settings of the machine, you couldn't decipher the messages.

Rotors

Army issue Enigma machines had three revolving "wheels" or "rotors" that could be taken out and changed about.The first task for an Enigma operator would be to decide which rotor went in which position. There were five rotors to choose from and they could be inserted into three positions on the Enigma machine.

Each rotor had numbers, 1 - 26, listed on the face of the wheel -- like an odometer in your car. Each number corrosponded to a character, for example, the letter `A` would corrospond to the number `1`. `B` could be `2`, `C` `3`, all the way to `Z` being number `26`. The circuits, however, were never mapped this way. They were always scrambled, so `A` could map to the number `23`, `B` to `2`, `C` to `24` etc. This is the first phase of the enigma cipher.

Rotor Example

Consider this to be the configuration for a rotor: `EKMFLGDQVZNTOWYHXUSPAIBRCJ`

On the face of the rotor (like an odometer) it would look like this:

Face	Circuit Map
1	E
2	K
3	M
4	F

and so on.

Rotor Configuration

Though the rotor operates similar to an odometer, it doesn't exactly. When an odometer operates, after it completes a cycle (going from 0 to 10) the next ring rolls over to the next number. With enigma, we can change when the next ring will roll over. For example: instead of having my odometer roll the next ring over after it surpasses number 9, we can change it to roll over at the number 4 with enigma.

Below is what three rotors in an enigma machine would look like, instead of `ABCDEFG...` it would have numbers. This provides a clear image of what enigma is doing with its circuit mapping.

```
          ABCDEFGHIJKLMNOPQRSTUVWXYZ
  Rotor 1: EKMFLGDQVZNTOWYHXUSPAIBRCJ

          ABCDEFGHIJKLMNOPQRSTUVWXYZ
  Rotor 2: AJDKSIRUXBLHWTMCQGZNPYFVOE

          ABCDEFGHIJKLMNOPQRSTUVWXYZ
  Rotor 3: BDFHJLCPRTXVZNYEIWGAKMUSQO
```

For example: if we press the letter `A` on a keyboard, it would be encoded as `E` on rotor 1. That `E` character is then passed to rotor 2, where it'll be encoded as `S`, then from `S` to `G` on rotor 3.

So the final encoding of the character `A` when going through the rotors is:

```
  Character: A
  ----

  A - E
  E - S
  S - G

Encoded: G
```

However, the rotor(s) rotate after each key press. The amount of rotors that rotate depends on the ring settings. The example above and below is ring-setting 1 (meaning `A` maps to `E`). The first rotor always rotates on a keypress. We encoded the character `A` to `G`, now we rotate the first rotor:

```
          BCDEFGHIJKLMNOPQRSTUVWXYZA
  Rotor 1: EKMFLGDQVZNTOWYHXUSPAIBRCJ

          ABCDEFGHIJKLMNOPQRSTUVWXYZ
  Rotor 2: AJDKSIRUXBLHWTMCQGZNPYFVOE

          ABCDEFGHIJKLMNOPQRSTUVWXYZ
  Rotor 3: BDFHJLCPRTXVZNYEIWGAKMUSQO
```

If we press the `A` key again, `A` now maps to `J` on the first rotor. `J` maps to `B` on rotor 2, and `B` maps to `D` on rotor 3.

```
  Character: A
  ----

  A - J
  J - B
  B - D

Encoded: D
```

So if we pressed the letter `A` twice, the encoded result would be `GD` if we utilized just our rotors.

The Ring Setting

The last aspect of rotors is the ring setting. This is the position in which will cause the next rotor to roll over (like when we press the `A` character once, rotor 1 rolls over but 2 did not). this is because we did not change the ring setting. Currently we would have to press a key/keys 26 times for rotor 2 to roll over to the next character.

For example, say I set my ring setting on rotor 1 to 25, initially it would look like this:

```
           ZABCDEFGHIJKLMNOPQRSTUVWXY
  Rotor 1: EKMFLGDQVZNTOWYHXUSPAIBRCJ

           ABCDEFGHIJKLMNOPQRSTUVWXYZ
  Rotor 2: AJDKSIRUXBLHWTMCQGZNPYFVOE

           ABCDEFGHIJKLMNOPQRSTUVWXYZ
  Rotor 3: BDFHJLCPRTXVZNYEIWGAKMUSQO
```

If I press another key, rotor 2 will trip:

```
           ABCDEFGHIJKLMNOPQRSTUVWXYZ   <--- pressed a key
  Rotor 1: EKMFLGDQVZNTOWYHXUSPAIBRCJ

           ZABCDEFGHIJKLMNOPQRSTUVWXY   <--- notice the change here
  Rotor 2: AJDKSIRUXBLHWTMCQGZNPYFVOE

           ABCDEFGHIJKLMNOPQRSTUVWXYZ
  Rotor 3: BDFHJLCPRTXVZNYEIWGAKMUSQO
```

This would also be the case for rotor 3 after rotor 2 hits an increment notch / ring.

Reflector and Plugboard

THe last step of enigma is to "reflect" or encoded the encoded character in the rotors again, but reversed.

A reflector is like a rotor, but it does not increment. Instead, we can use a plugboard to mix and match characters. We're not going to be implementing a plugboard, so it's going to be a standard A-Z configuration.

Going with our initial example, we're going to add a reflector and encode a character:

```
          ABCDEFGHIJKLMNOPQRSTUVWXYZ
Rotor 1: EKMFLGDQVZNTOWYHXUSPAIBRCJ

          ABCDEFGHIJKLMNOPQRSTUVWXYZ
Rotor 2: AJDKSIRUXBLHWTMCQGZNPYFVOE

          ABCDEFGHIJKLMNOPQRSTUVWXYZ
Rotor 3: BDFHJLCPRTXVZNYEIWGAKMUSQO

             ABCDEFGHIJKLMNOPQRSTUVWXYZ
reflector: EJMZALYXVBWFCRQUONTSPIKHGD
```

Let's encode the character `A`.

- We press `A` and rotor 1 encodes as `E`.
- `E` is encoded to `S` in rotor 2
- `S` is encoded to `G`
- `G` is mapped to Y in the reflector
- `Y` in rotor 3, is mapped to the character `O` (reflection is reversed)
- `O` in rotor 2, is mapped to `Y`
- `Y` in rotor 1, is mapped to `O`

```
A
----

A - E
E - S
S - G

ref: G - Y

Y - O
O - Y
Y - O
```

Rotor 1 would then rotate to map the circuits to `ZABCDEFGHIJKLMNOPQRSTUVWXY`.

```
         ZABCDEFGHIJKLMNOPQRSTUVWXY
Rotor 1: EKMFLGDQVZNTOWYHXUSPAIBRCJ

         ABCDEFGHIJKLMNOPQRSTUVWXYZ
Rotor 2: AJDKSIRUXBLHWTMCQGZNPYFVOE

         ABCDEFGHIJKLMNOPQRSTUVWXYZ
Rotor 3: BDFHJLCPRTXVZNYEIWGAKMUSQO

             ABCDEFGHIJKLMNOPQRSTUVWXYZ
reflector: EJMZALYXVBWFCRQUONTSPIKHGD
```

Decoding

To decode our character, we have to set our rotors to the initial settings as when we started with the encoding process:

```
          ABCDEFGHIJKLMNOPQRSTUVWXYZ
Rotor 1: EKMFLGDQVZNTOWYHXUSPAIBRCJ

          ABCDEFGHIJKLMNOPQRSTUVWXYZ
Rotor 2: AJDKSIRUXBLHWTMCQGZNPYFVOE

          ABCDEFGHIJKLMNOPQRSTUVWXYZ
Rotor 3: BDFHJLCPRTXVZNYEIWGAKMUSQO

              ABCDEFGHIJKLMNOPQRSTUVWXYZ
reflector: EJMZALYXVBWFCRQUONTSPIKHGD
```

Our encoded character was `O`. So let's decode it:

- `O` maps to `Y` in rotor 1
- `Y` maps to `O` in rotor 2
- `O` maps to `Y` in rotor 3
- `Y` is mapped to `G` in the reflector
- `G` is mapped to `S` in rotor 3 (remember reflection means we have to go backwards)
- `S` is mapped to `E` in rotor 2
- `E` is mapped to `A` in rotor 1

Here we encoded the character `A` to `O` and decoded `O` back to `A`.

Your Extra Credit Opportunity

This assignment is an opportunity for you to demonstrate some of the competencies in the course in which you may not have met your own personal expectations. This assignment is worth *up to 150 extra credit points* if you follow the instructions. If you do not complete this assignment, you will still earn points based on the competencies you were able to demonstrate.

What I'm looking for:

- You used the Java Programming Language
- You demonstate proper Object-Oriented programming and Design
- You utilize Git effectively
- You write supporting unit tests before production code
- You properly implement an enigma machine that can decode a message using these techniques
    - Console-based application only, no UI
    - No user input needed, just hard-code the message I want you to decode at the bottom of this

document

There is a small catch, enigma is what we call a base-26 encoding, meaning it does not support anything but characters. I also want you to include a whitespace in your reflector

Submitting your assignment

You must submit your assignment before class on October 18th. You will email me the link to your github repository containing your code. If you are able to fully implement the Enigma cipher, also include the message I'll ask you to decode at the bottom of this document.

Message to Decode

Settings:

```
    Rotor 1: 'EKMFLGDQVZNTOWYHXUSPAIBRCJ '
    Rotor 2: 'AJDKSIRUXBLHWTMCQGZNPYFVOE '
    Rotor 3: 'BDFHJLCPRTXVZNYEIWGAKMUSQO '

    Reflector: 'EJMZALYXVBWFCRQUONTSPIKHGD '

Ring Settings:
    - Rotor 1: Position 10
    - Rotor 2: Position 1
    - Rotor 3: Position 1
```

Message:

```
"AYYTC IMLGF UUEFW LAOGU TZOXE CRMIM JFMPY QRJDW DKBLO JPCFV UDDSF RPJFA RZOCU BXXMY
TAJLC IOMQY ADZUS GYIOC MPFKM HRTFX N"
```

Note: The quotes (" ") are just to show that there isn't any whitespace starting/ending the message.