**Monoalphabetic 암호화 알고리즘 구현**

컴퓨터학부 2016110879 유재인

**1) 프로젝트 설명**

1. 암호화 및 복호화 프로그램

- 주요 기능은 1) key 생성, 2) 암호화, 3) 복호화이다.

- key 생성 기능을 선택하면, 임의의 key를 생성하여 파일에 저장한다.

- 암호화 기능을 선택하면, key 파일과 원문 텍스트 파일을 읽어서, 암호문 텍스트 파일을 생성한다.

- 복호화 기능을 선택하면, key 파일과 암호문 텍스트 파일을 읽어서, 원문 텍스트 파일을 생성한다.

- 암호와 복호의 대상은 텍스트 파일에서 알파벳만 치환한다.

2. 암호문 공격 프로그램

- 암호문 텍스트 파일을 읽어 들여서 1) 알파벳의 빈도수 출력과 2) key와 원문을 10개 예측한다.

- 알파벳 빈도수 기반으로 적합한 key를 추측하여, 그에 해당하는 원문을 출력한다.

- 예측한 key와 원문을 각각 파일로 저장한다. (예: key1.txt, plaintext1.txt)

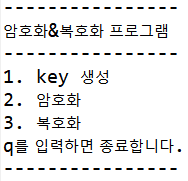
- 가장 적합할 것으로 예측하는 key와 원문을1개 이상 생성한다.

**2) 코드 설명**

1. 암호화 및 복호화 프로그램

이 프로그램은 1개의 메인 메소드와 3개의 메소드로 이루어져 있습니다.

1. public static void main(String[] args): 메인 메소드
2. public static void makeKey(): 키 생성 메소드
3. public static void encryption(): 암호화 메소드
4. public static void decryption(): 복호화 메소드
5. public static void main(String[] args)



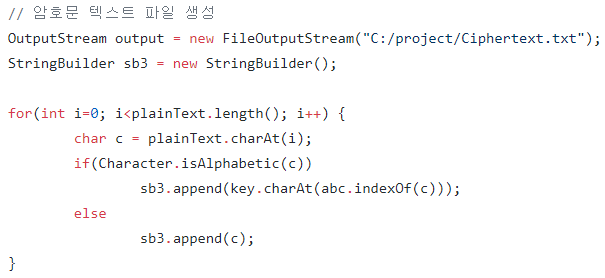
프로그램의 메뉴를 보여주고 사용자에게 입력을 받은 후에 해당 작업을 실행합니다.

1. public static void makeKey()



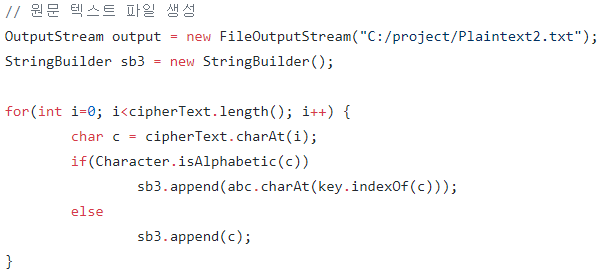
Random r을 이용하여 알파벳 26자를 무작위로 중복 없이 선택합니다. 완성된 String을 C:/project/Key.txt 파일로 저장합니다.

1. public static void encryption()



C:/project/Key.txt 파일을 읽고 String key에 저장합니다. 암호화시킬 Plaintext.txt 파일을 읽고 String plainText에 저장합니다. 암호문은 key와 알파벳이 순서대로 나열된 String abc를 대응하여 만듭니다. 완성된 암호문을 Ciphertext.txt 파일로 저장합니다.

(4) public static void decryption()



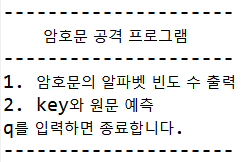
C:/project/Key.txt 파일을 읽고 String key에 저장합니다. 복호화시킬 Ciphertext.txt 파일을 읽고 String cipherText에 저장합니다. 원문은 key와 알파벳이 순서대로 나열된 String abc를 대응하여 만듭니다. 완성된 원문을 Plaintext2.txt 파일로 저장합니다.

2. 암호문 공격 프로그램

이 프로그램은 1개의 메인 메소드와 3개의 메소드로 이루어져 있습니다.

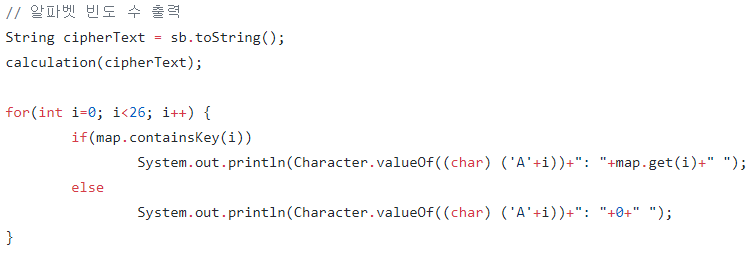
1. public static void main(String[] args): 메인 메소드
2. public static void frequencyAnalysis(): 알파벳 빈도 수 출력 메소드
3. public static void calculation(String cipherText): 알파벳 빈도 수 계산 메소드
4. public static void attack(): 키, 원문 예측 메소드

(1) public static void main(String[] args)



프로그램의 메뉴를 보여주고 사용자에게 입력을 받은 후에 해당 작업을 실행합니다.

(2) public static void frequencyAnalysis()

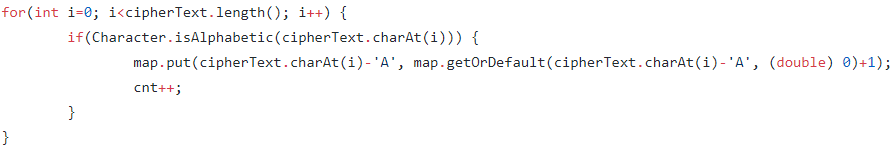


C:/project/Key.txt 파일을 읽고 String cipherText에 저장합니다. 암호문 알파벳 빈도 수를 계산하는 메소드 public static void calculation(String cipherText) 을 실행합니다. 알파벳의 빈도 수가 0인 경우 따로 출력을 실행합니다.

(3) public static void calculation(String cipherText)

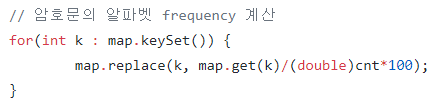


알파벳(0~25)과 빈도 수를 계산하기 위해 HashMap<Integer, Double> map을 사용합니다. 전체 알파벳 개수는 int cnt 에 저장합니다.

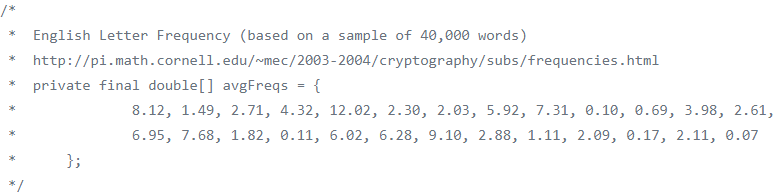


매개변수 cipherText의 i번째 문자가 알파벳인 경우 map의 value를 갱신합니다. 그리고 전체 알파벳 개수 카운터인 cnt를 1 증가시킵니다.

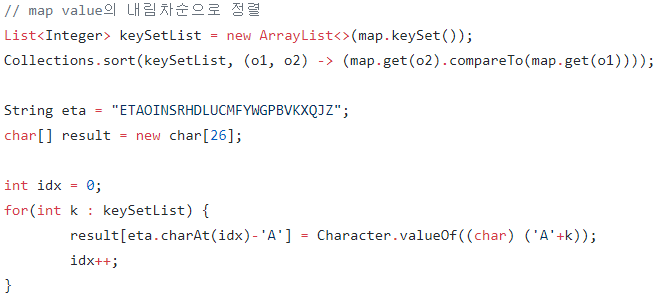
(4) public static void attack()



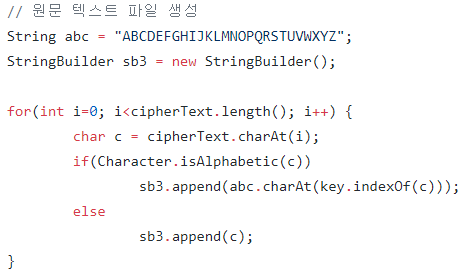
HashMap<Integer, Double> map을 사용하여 빈도 수의 백분율을 계산합니다. 이 때 전체 알파벳 개수를 이용합니다.



English Letter Frequency를 이용하여 키와 원문을 예측합니다. 알파벳 26자를 빈도 수의 내림차순으로 정렬했을 때 ETAOINSRHDLUCMFYWGPBVKXQJZ가 됩니다. 이를 String eta에 저장합니다.



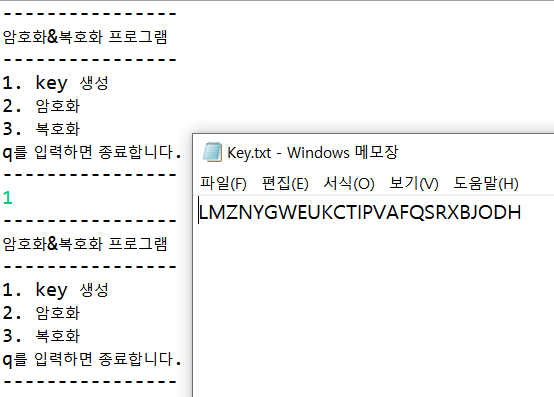
마찬가지로 암호문에서도 빈도 수의 내림차순으로 알파벳을 정렬합니다. 정렬 결과를 eta와 대응하여 키를 예측합니다.



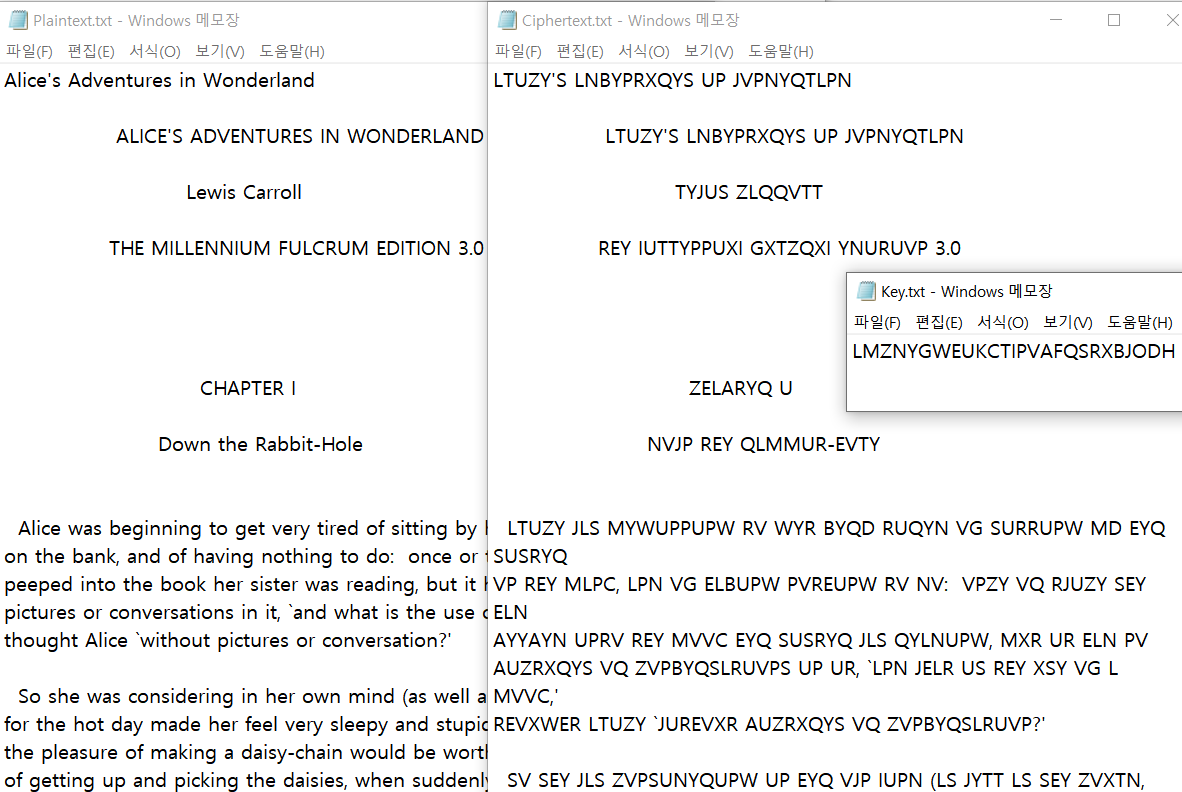
예측한 키를 기준으로 암호문의 복호화를 수행합니다. 이후, 예측한 키와 원문을 각각 Key1.txt, Plaintext1.txt로 저장합니다.

**3) 결과**

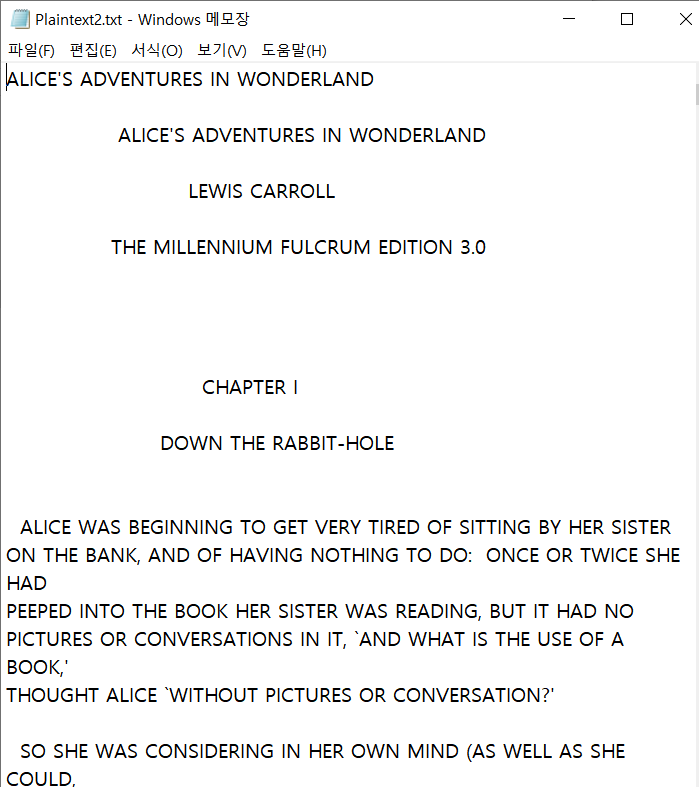
1. 암호화 및 복호화 프로그램



알파벳 대문자 26자로 키가 생성되었다.

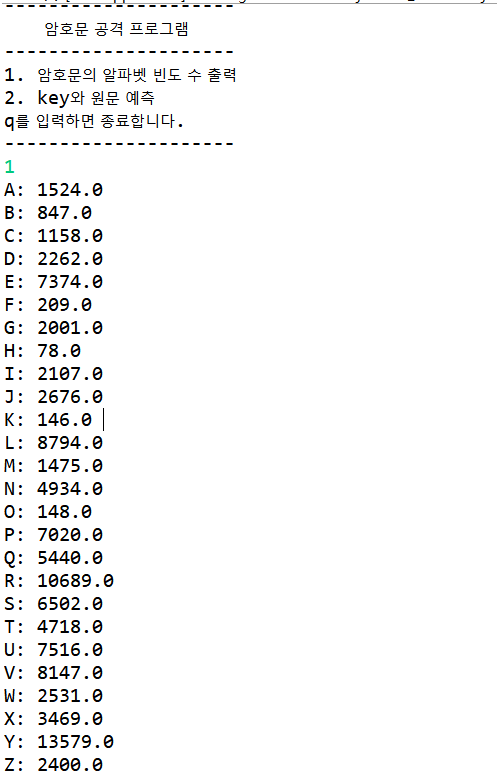


원문이 암호화되었다.

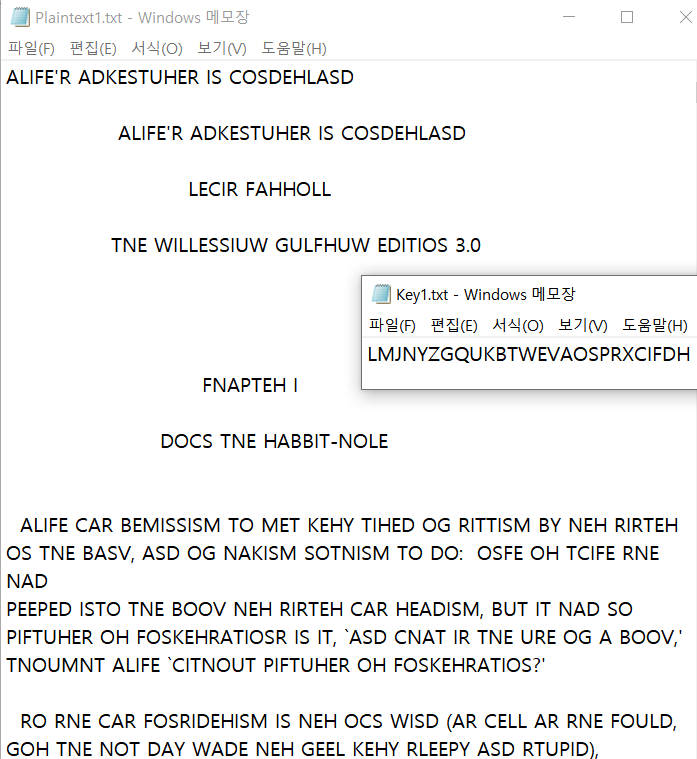


암호문이 복호화되었다.

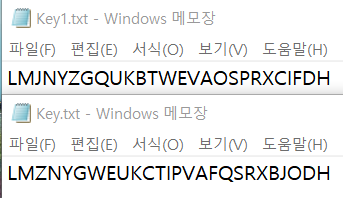
2. 암호문 공격 프로그램



암호문의 알파벳 빈도 수를 출력했다.



빈도 수를 대응하여 예측한 키와 원문이다.



원래 키와 예측한 키이다.