# KECCAK을 이용한 자바 난독화 프로그램 개발

**DID_Capston_Design**

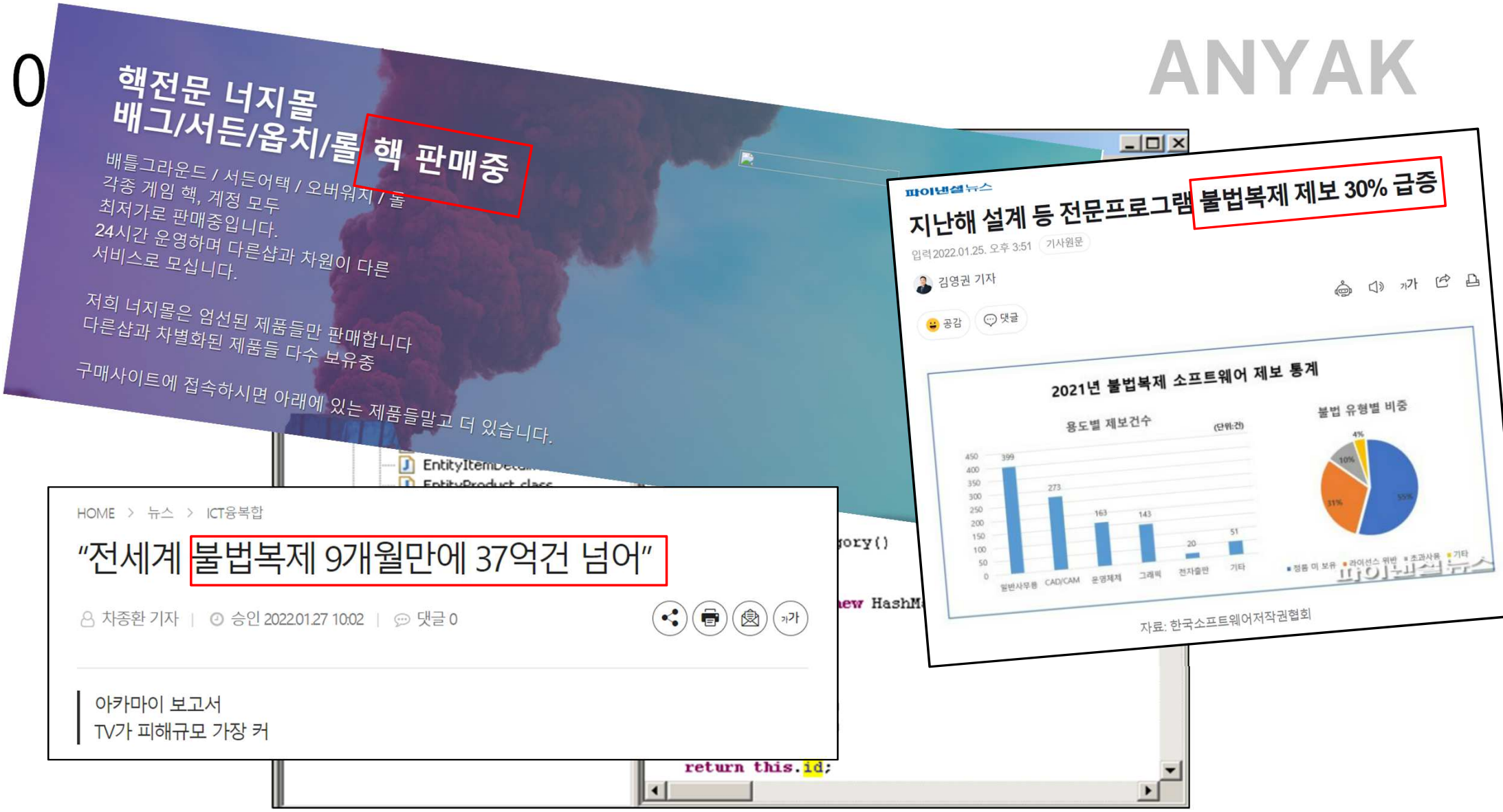
**2022년 12월 07일**

**동명대학교 정보보호학과**

**조원 : 김 *원, 김 *원, 김 * 현, 김 * 형**

ANYAK

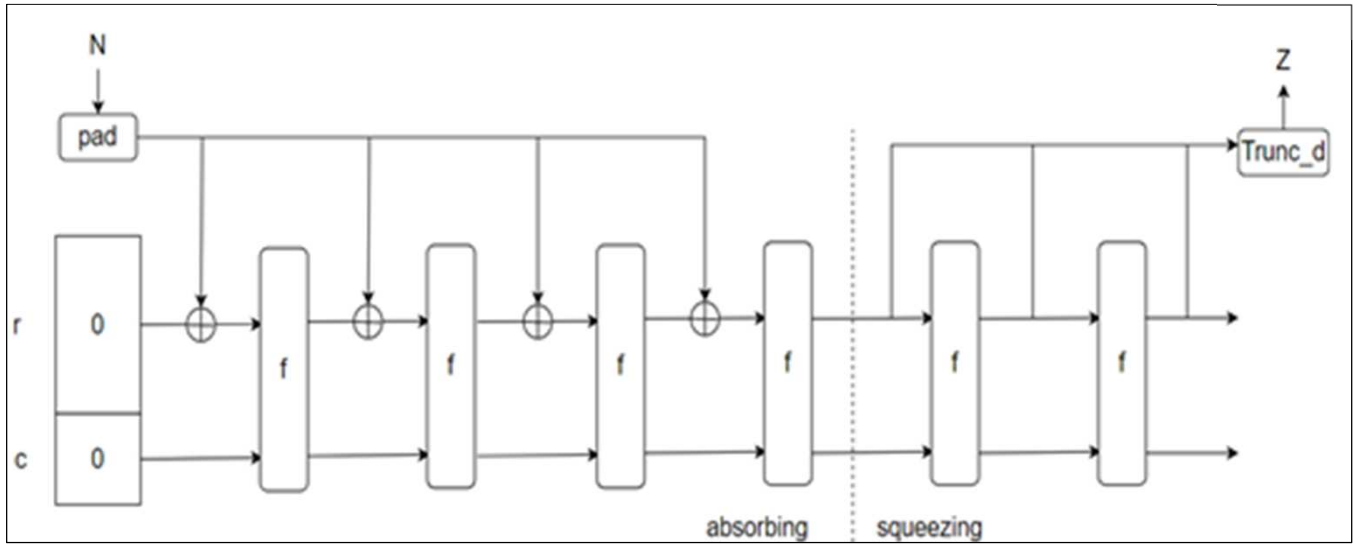
# 목차




출처 : http://sjava.net/2010/01/another-java-decompiler/

# 02. 관련 연구 - KECCAK

ANYAK

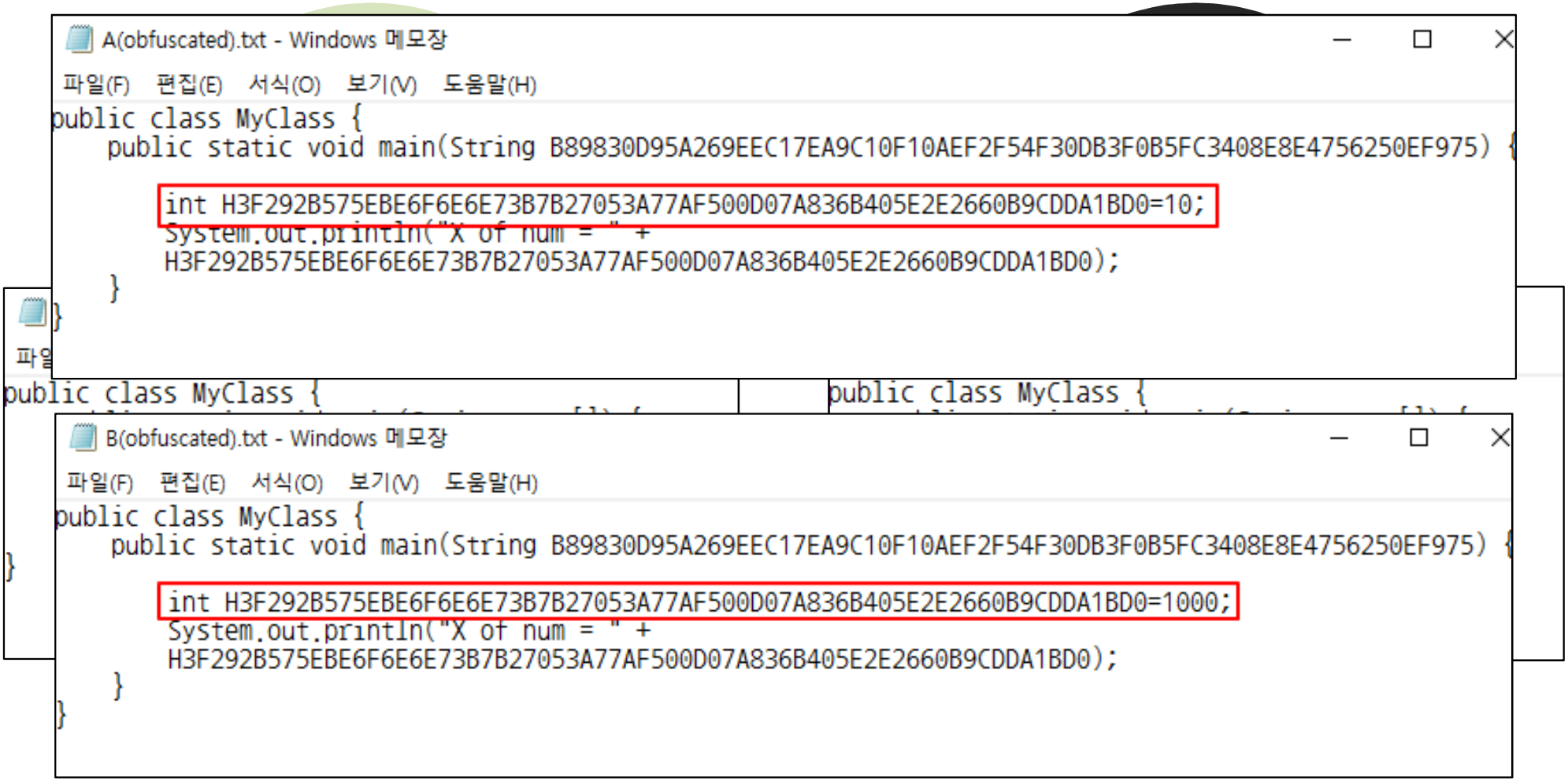



- 미국 국립 표준 기술 연구소(NIST)는2012년 1월 KECCAK를 SHA-3 해시 알고리즘으로 선정했다.
- KECCAK은 스펀지 구조를 가지며, 스펀지 구조는 흡수(absorbing) 과정과 압착(squeezing) 과정으로 이뤄진다.
- KECCAK은 **역상 저항성, 제2 역상 저항성, 충동 저항성을** 만족하며 수학적으로 증명 가능한 안정성이 보장된다.

참고문헌 KISA, “SHA-3 해시함수 알고리즘에 대한 소스코드 활용 매뉴얼”, 한국인터넷진흥원, pp1~2, 2020,02

# 02. 관련 연구 - 의사 난수 생성기

ANYAK



# 03. 프로그램 개발 - 개발 환경

ANYAK

| 항목 | 버전 |
| --- | --- |
| OS | Window 10 64bit |
| 언어 | C++ |
| 개발 플랫폼 | Visual Studio 2022 |

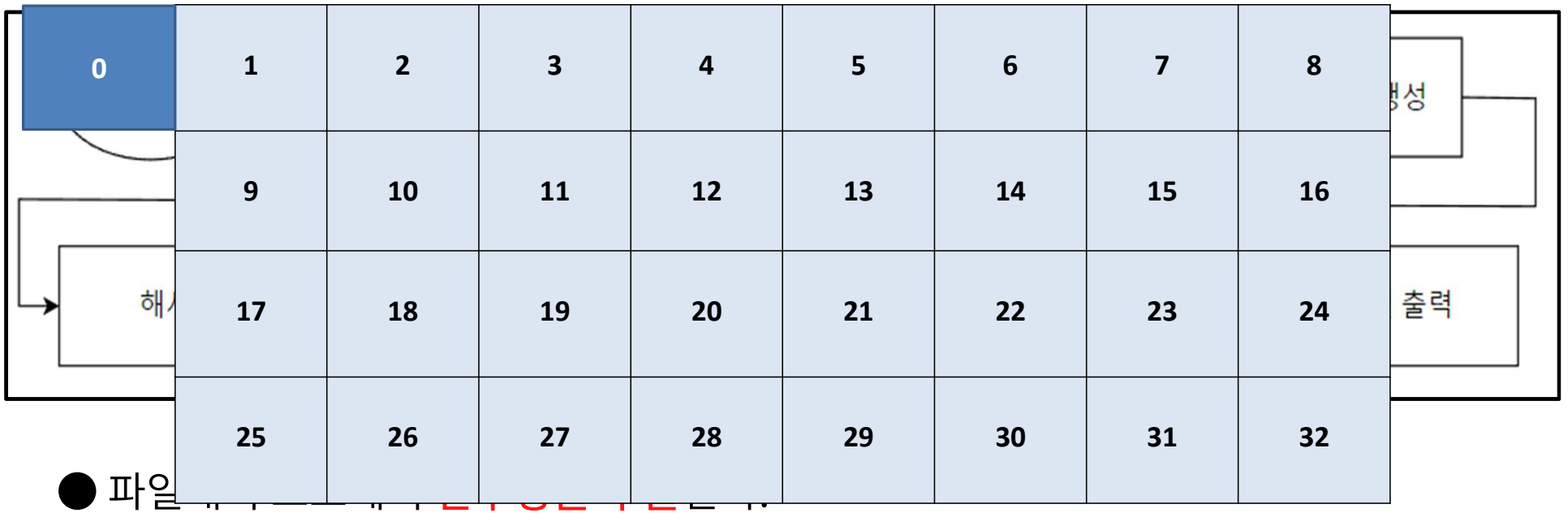
- 프로그램 동작 환경
  윈도우(Window)를 대상으로 한다.

- 난독화 프로그래의 개발 언어
  언어는 C++을 사용한다.

- 개발 플랫폼
  Visual Studio 2022를 사용한다.

# 03. 프로그램 개발 - 시스템 동작 과정

ANYAK




- 파일
- 레인보우 테이블을 이용한 공격을 방지하기 위해 변수명 앞에 의사난수를 추가한다.
- 의사난수를 추가한 변수명을 KECCAK알고리즘을 사용하여 난독화 함.

# 03. 프로그램 개발 - 프로그램 테스트

ANYAK

https://blog.naver.com/wogud3426/222941280376

# 04. 결론 - 비교 분석

ANYAK

```
import java.util.Scanner;

public class test {
    public static void main(String[] args) {
        String name;
        int age;
        double height;
        String intro;
        String buffer;

        Scanner sc = new Scanner(System.in);
        System.out.println("이름을 입력하세요");
        name = sc.next();
        System.out.println("나이를 입력하세요");
        age = sc.nextInt();
        System.out.println("키를 입력하세요");
        height = sc.nextDouble();
        System.out.println("자기소개를 입력하세요");
        buffer = sc.nextLine();
        intro = sc.nextLine();

        System.out.println("이름은 "+name+"나이는 "+age+",키는 "+height+"입니다.");
        System.out.println(intro);
    }
```

난독화 전

```
Y5788E682E1AD8B3C0CC9F67D2A82016D42236153F5B9BFABBE617ECAE110411B;int
MC5E3AB367959A198849C50EB906738B71DEDB4234899D36FC8D72AC47278BA39;String
M9CC47D5CA9FEEE88F3FB8CBF8C50284829FD503AB91B720FF8BD2927971FEB55;String
B6D016A00D02FCA010E4DE350CD53403FA5D8E411F475CB9E915D4637937D9557;Scanner
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725 = new Scanner
(System.in);System.out.println("이름을
입력하세요");Y5788E682E1AD8B3C0CC9F67D2A82016D42236153F5B9BFABBE617ECAE110411B =
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725.next
();System.out.println("나이를
입력하세요");ZA30B94FFF640802A90C381787F78CA24558182F50B484350D9705205A2C91FB3 =
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725.nextInt
();System.out.println("키를
입력하세요");MC5E3AB367959A198849C50EB906738B71DEDB4234899D36FC8D72AC47278BA39 =
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725.nextDouble
();System.out.println("자기소개를
입력하세요");M9CC47D5CA9FEEE88F3FB8CBF8C50284829FD503AB91B720FF8BD2927971FEB55 =
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725.nextLine
();B6D016A00D02FCA010E4DE350CD53403FA5D8E411F475CB9E915D4637937D9557 =
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725.nextLine
();System.out.println("이름은
"+Y5788E682E1AD8B3C0CC9F67D2A82016D42236153F5B9BFABBE617ECAE110411B+"나이는
"+ZA30B94FFF640802A90C381787F78CA24558182F50B484350D9705205A2C91FB3+",키는
"+MC5E3AB367959A198849C50EB906738B71DEDB4234899D36FC8D72AC47278BA39+"입니다.");Syste
m.out.println(M9CC47D5CA9FEEE88F3FB8CBF8C50284829FD503AB91B720FF8BD2927971FEB55);}}
```

난독화 후

# 04. 결론 - 개선사항

ANYAK

Calculator.class

```
import java.awt.event.ItemEvent;
import java.util.function.Consumer;
import java.util.regex.Pattern;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JTextField;

public class Calculator {
  private static final int L3D8F6579BE2CE59D3CEDA4D2E13FE65FFD78A340C784A93076FCC0C222DC8B33 = 410;

  private static final int E7EC4E41555928AA4E4D0294338CFFC8CAE8638684555B5177912143AAECAC0D3 = 600;

  private static final int DA3A76E2AB2A86D6173F36EBF9BA44F3106037A0A71D53F5BFF0CEFFE9B35223E = 80;

  private static final int P7AECF5A206CA0E7132EEC87BF266226D8DE59E09A1D695691C1CCFC073164258 = 70;

  private static final int NC16D10971BABEA08D356BFECA17755D339F30FADE62FB9EFAC8049FAA4E600BB = 20;

  private static final int F83840608E908396F1EEFC3EFFB34F05516A61D28F638F4F9BC6EF722E1D6E7AC = 60;

  private JFrame window;

  private JComboBox<String> comboCalcType;

  private JComboBox<String> comboTheme;
```

# 04. 결론

ANYAK

- KECCAK을 이용한 **코드 난독화 프로그램을 개발**하였다.
- KECCAK으로 인해 난독화된 코드에서 원본 코드로 복원하기 어렵다.
- 가독성이 현저히 떨어지는 것을 확인할 수 있다.
- 이러한 방식의 난독화 기법은 악의적인 사용자의 코드분석을 방해하여 프로그램 **무단 복제, 게임 핵 제작** 등과 같은 악용사례의 빈도를 줄이는 효과를 기대한다.
- 디컴파일로 복원된 코드에서도 **줄 바꿈과 공백을 제거한다면** 더욱 완벽한 코드 난독화가 가능할 것으로 기대된다.

# ANYAK

감사합니다.