PR 2.0 Class and Constructor

In my video, I will show

- Write a class "MultiplyClass" and a method "Multiply" that can multiply matA (MxN) and and scalar (double)
- 2. Write 4 methods with the same name "Multiply" doing
 - 2.1 Multiply(scalar, scalar2)
 - 2.2 Multiply(scalar, matA)
 - 2.3 Multiply(matA, scalar)
 - 2.4 Multiply (matA, matB)
 - 2.5 message2Mat()
 - 2.6 encode(matA, message)
 - 2.7 decode(matA,matC), decode(1/scalar, matC)
 - 2.8 decodePrintMessage()

In my video, I will show

- 3. Write 3 constructors.
 - 3.1 MultiplyClass(scalar, matB)
 - 3.2 MultiplyClas(matA, scalar)
 - 3.2 MultiplyClass(matA, MatB)
- 4. In PR20, make 4 different objects(instance) "cryptObj1", " cryptObj2", "cryptObj3" using class MultiplyClass

Call MultiplyObj1 .Multiply with 3 different inputs.

- 4.1 Multiply cryptObj1 = new Multiply(5, message)
- 4.2 Multiply cryptObj2 = new Multiply(message, 5)
- 4.3 Multiply cryptObj3 = new Multiply(matA1, message1)
- 4.4 Multiply cryptObj4 = new Multiply(matA2, message2)
- 5. Print 4 decoded messages

Objects and Constructors

```
public class MultiplyClass {
    public MultiplyClass(double[][] matA, char[][] message) {
        this.matA = matA;
        this.message= message;
        this.matB = message2Mat(message);
        this.matC = Multiply(matA, matB);
        this.matD = Multiply(Mat.inverse(matA),this.matC);
}
public MultiplyClass(double scalar, char[][] message) {
    ......
}
```

Watch the videos

https://www.youtube.com/watch?v=jbcng9VhaSY&list=PLEBtfn 2xvyj5fy9JLBnH2dCc1kAMZj5ey&index=9

https://www.youtube.com/watch?v=MK2SMJZbUmU

Constructors and This

```
public class MultiplyClass {
        char[][] message ; // simple message
        double scalar ;
        double scalar2 ;
        double[][] matA ; //encoding Matrix
        double[][] matB ; //Message Matrix
        double[][] matC ; // encoded Matrix
11
        double[][] matD ; // decoded Matrix
12
13
        // Constructor without input.
14⊖
        public MultiplyClass() {
15
16
17
18
        // Constructor with matA and a message
19⊖
        public MultiplyClass(double[][] matA, char[][] message) {
20
            this.matA = matA;
21
            this.message= message;
22
            this.matB = message2Mat(message);
23
            this.matC = Multiply(matA, matB);
24
            this.matD = Multiply(Mat.inverse(matA),this.matC);
25
26
27
        // Constructor with a scalar and a message
28⊖
        public MultiplyClass(double scalar, char[][] message) {
            this.scalar = scalar;
29
30
            this.message= message;
31
            this.matB = message2Mat(message);
32
            this.matC = Multiply(scalar, matB);
33
            this.matD = Multiply(1/scalar, matB);
34
35
36
        // Constructor with a message and a scalar
37
38⊖
        public MultiplyClass(char[][] message, double scalar) {
39
            this.scalar = scalar:
```

Constructors and This

```
3 public class HW20 {
        public static void main(String[] args) {
             double[][] matA1 = \{\{4,3,3\},\{1,2,1\},\{1,3,4\}\}\};
             char[][] message1 = {{'J','e','s','u'},{'s','l','o','v'},{'e','s','m','e'}};
             double[][] matA2 = \{\{7,2,-1\},\{-2,0,6\},\{9,2,-5\}\};
11
             char[][] message2 = {{'I','I','o','v'},{'e','C','S','g'},{'e','t','a','5'}};
12
13
14
             MultiplyClass obj1 = new MultiplyClass(matA1, message1);
15
16
             obj1.printDecodedMessage();
17
18
             MultiplyClass obj2 = new MultiplyClass(matA2, message2);
             obj2.printDecodedMessage();
19
20
22
```

Constructors and This

```
public class HW20 {
        public static void main(String[] args) {
             double[][] matA1 = {{4,3,3},{1,2,1},{1,3,4}};
             char[][] message1 = {{'J', 'e', 's', 'u'}, {'s', 'l', 'o', 'v'}, {'e', 's', 'm', 'e'}};
 10
             double[][] matA2 = {{7,2,-1},{-2,0,6},{9,2,-5}};
             char[][] message2 = {{'I', 'l', 'o', 'v'}, {'e', 'C', 'S', 'g'}, {'e', 't', 'a', '5'}};
 11
 12
             MultiplyClass obj1 = new MultiplyClass(matA1, message1);
 15
             obj1.printDecodedMessage();
 16
17
             MultiplyClass obj2 = new MultiplyClass(matA2, message2);
 18
             obj2.printDecodedMessage();
19
20
21
22
```

```
public class MultiplyClass {
4
5
        char[][] message ; // simple message
        double scalar ;
        double scalar2;
        double[][] matA ; //encoding Matrix
        double[][] matB ; //Message Matrix
        double[][] matC ; // encoded Matrix
        double[][] matD ; // decoded Matrix
12
        // Constructor without input.
        public MultiplyClass() {
14⊖
15
16
17
18
        // Constructor with matA and a message
        public MultiplyClass(double[][] matA, char[][] message) {
19⊜
            this.matA = matA;
20
            this.message= message;
21
            this.matB = message2Mat(message);
22
            this.matC = Multiply(matA, matB);
23
            this.matD = Multiply(Mat.inverse(matA),this.matC);
24
25
26
        // Constructor with a scalar and a message
27
        public MultiplyClass(double scalar, char[][] message) {
28⊝
            this.scalar = scalar;
29
            this.message= message;
30
            this.matB = message2Mat(message);
31
            this.matC = Multiply(scalar, matB);
32
            this.matD = Multiply(1/scalar, matB);
33
34
35
        // Constructor with a message and a scalar
36
37
        public MultiplyClass(char[][] message, double scalar) {
38⊜
            this.scalar = scalar:
39
```

(Optional) Deep Learning for Java

- Install "DL4J" Library
- Test and run Al codes