PR 2.0 Class and Constructor

In my video, I will show

1. 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

1.

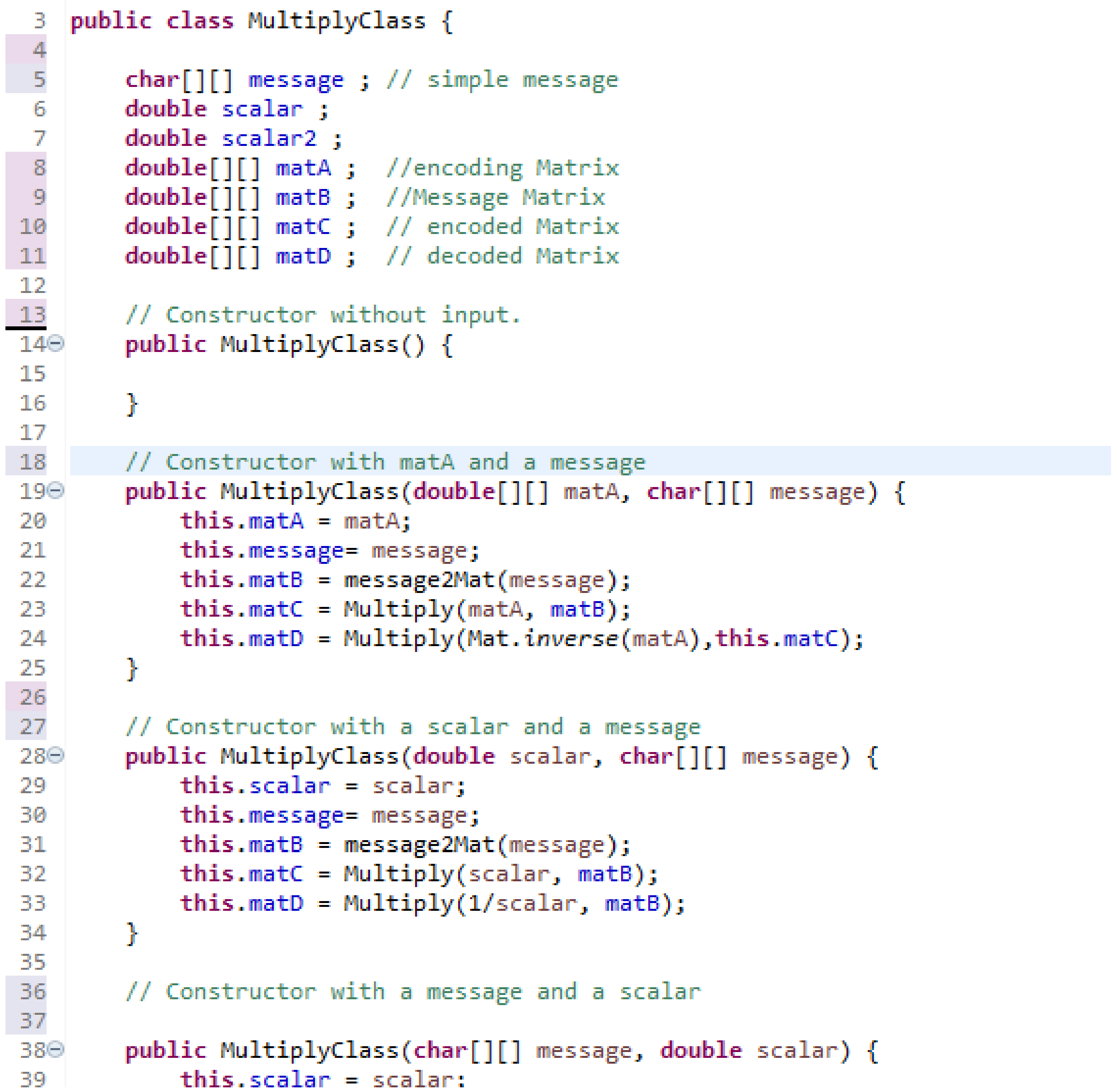
```
public class MultiplyClass {
     public MultiplyClass(double[][] matA, char[][] message) {
     this.matA = matA;
     this.message= message;
     this.matB = message2Mat(message);
     this.matC = Multiply(matA, matB);
     this.matD = Multiply(Mat.inverse(matA),this.matC);

}
public MultiplyClass(double scalar, char[][] message) {

    ……

}
```

Watch the videos

https://www.youtube.com/watch?v=jbcng9VhaSY&list=PLEBtfn2xvyj5fy9JLBnH2dCc1kAMZj5ey&index=9

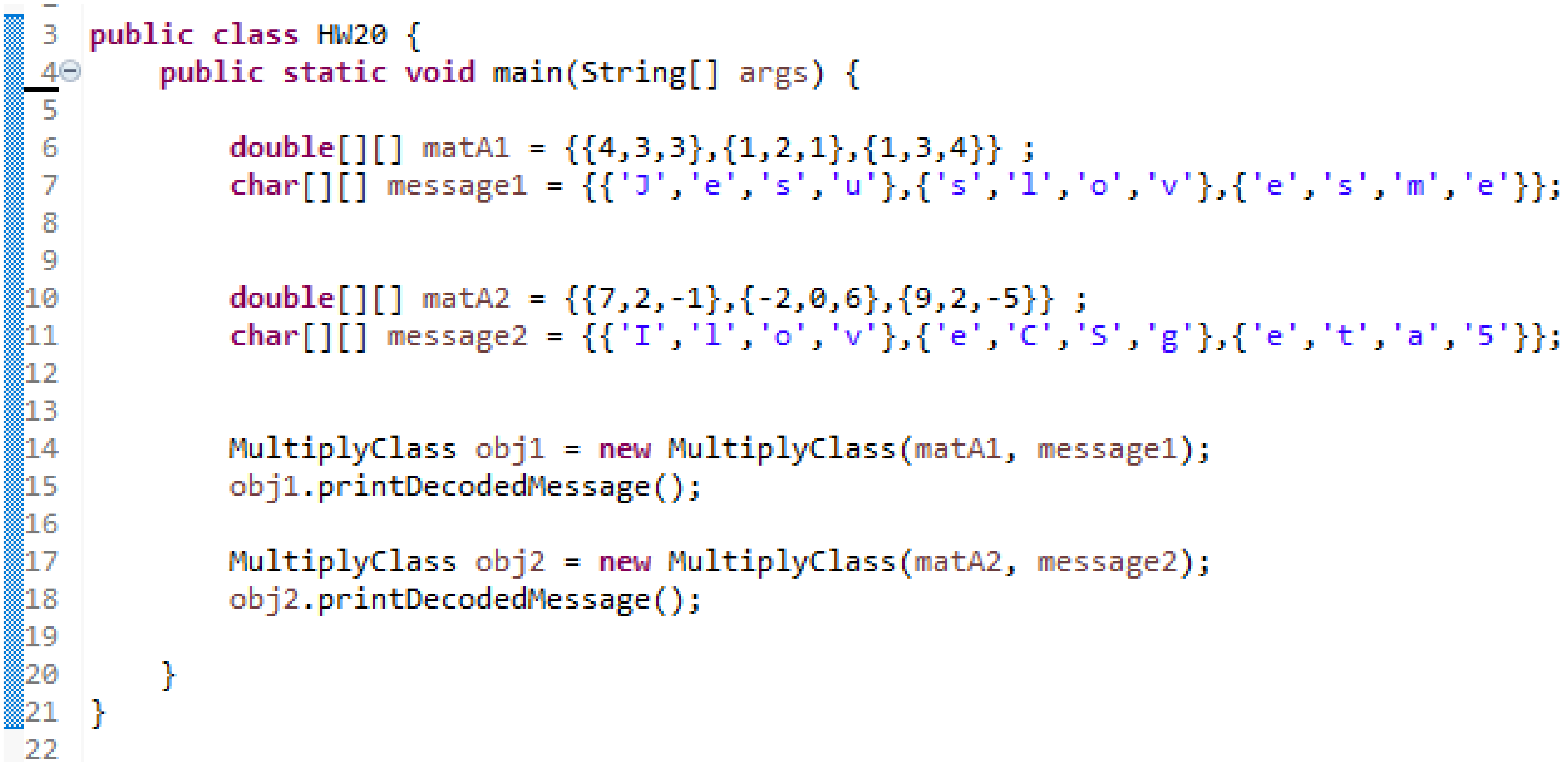
https://www.youtube.com/watch?v=MK2SMJZbUmU

Constructors and This



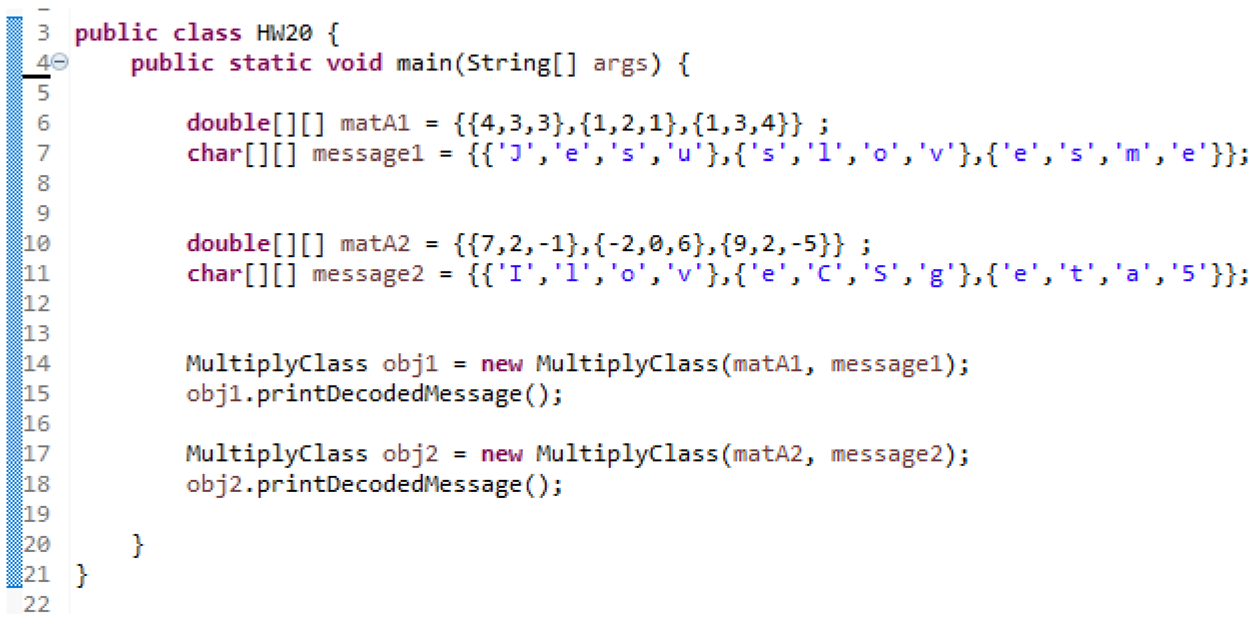
```
public class MultiplyClass {

    char[][] message ; // simple message
    double scalar ;
    double scalar2 ;
    double[][] matA ;  //encoding Matrix
    double[][] matB ;  //Message Matrix
    double[][] matC ;  // encoded Matrix
    double[][] matD ;  // decoded Matrix

    // Constructor without input.
    public MultiplyClass() {

    }

    // Constructor with matA and a message
    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);
    }

    // Constructor with a scalar and a message
    public MultiplyClass(double scalar, char[][] message) {
        this.scalar = scalar;
        this.message= message;
        this.matB = message2Mat(message);
        this.matC = Multiply(scalar, matB);
        this.matD = Multiply(1/scalar, matB);
    }

    // Constructor with a message and a scalar

    public MultiplyClass(char[][] message, double scalar) {
        this.scalar = scalar;
```

Constructors and This

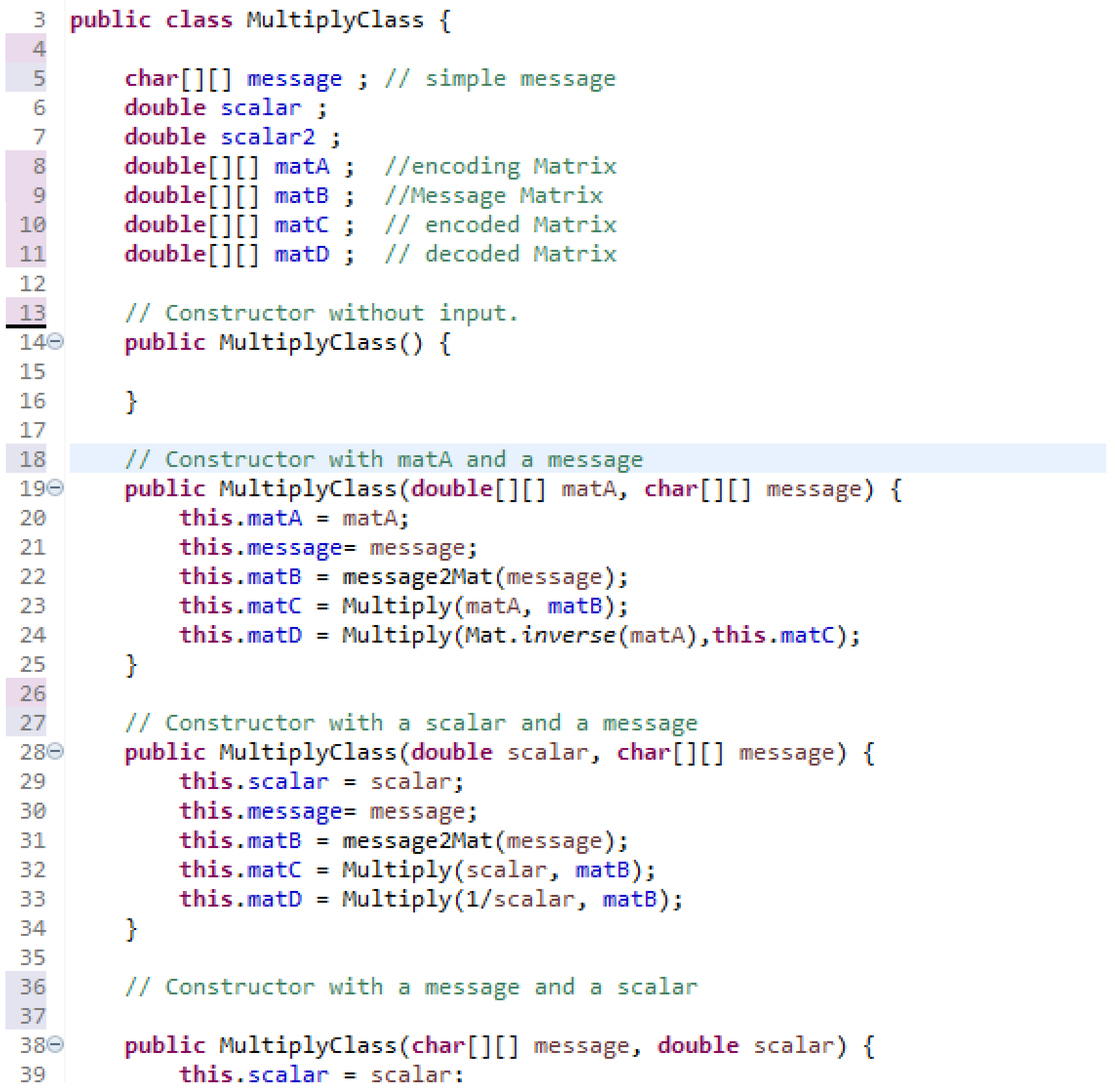


```java
public class HW20 {
    public static void main(String[] args) {

        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}} ;
        char[][] message2 = {{'I','l','o','v'},{'e','C','S','g'},{'e','t','a','5'}};


        MultiplyClass obj1 = new MultiplyClass(matA1, message1);
        obj1.printDecodedMessage();

        MultiplyClass obj2 = new MultiplyClass(matA2, message2);
        obj2.printDecodedMessage();

    }
}
```

Constructors and This



```java
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}} ;
        char[][] message2 = {{'I','l','o','v'},{'e','C','S','g'},{'e','t','a','5'}};


        MultiplyClass obj1 = new MultiplyClass(matA1, message1);
        obj1.printDecodedMessage();

        MultiplyClass obj2 = new MultiplyClass(matA2, message2);
        obj2.printDecodedMessage();

    }
}
```



```java
public class MultiplyClass {

    char[][] message ; // simple message
    double scalar ;
    double scalar2 ;
    double[][] matA ;  //encoding Matrix
    double[][] matB ;  //Message Matrix
    double[][] matC ;  // encoded Matrix
    double[][] matD ;  // decoded Matrix

    // Constructor without input.
    public MultiplyClass() {

    }

    // Constructor with matA and a message
    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);
    }

    // Constructor with a scalar and a message
    public MultiplyClass(double scalar, char[][] message) {
        this.scalar = scalar;
        this.message= message;
        this.matB = message2Mat(message);
        this.matC = Multiply(scalar, matB);
        this.matD = Multiply(1/scalar, matB);
    }

    // Constructor with a message and a scalar

    public MultiplyClass(char[][] message, double scalar) {
        this.scalar = scalar;
```

(Optional) Deep Learning for Java

- Install “DL4J” Library
- Test and run AI codes