1.9 Class and Overloading

In my video, I will show

1. Write a class “MultiplyClass” and a method “Multiply” with matA (3x4), matB(4x2), matC(2x3) and scalar 2, 3.5
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)

   2.4Multiply (matA, matB)

3. Make an object(instance) “MultiplyObj1” using class MultiplyClass

   Call MultiplyObj1 .Multiply with 4 different inputs.

   3.1 MultiplyObj1 .Multiply(2, 3.5)

   3.2 MultiplyObj1 .Multiply(2.7 , matA)

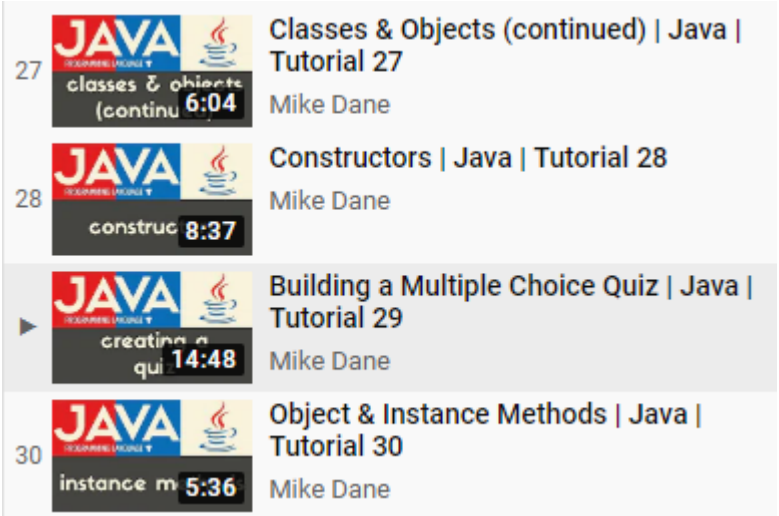
   3.3 MultiplyObj1 .Multiply(matA, 2.7)

   3.4 MultiplyObj1 .Multiply (matA, matB)

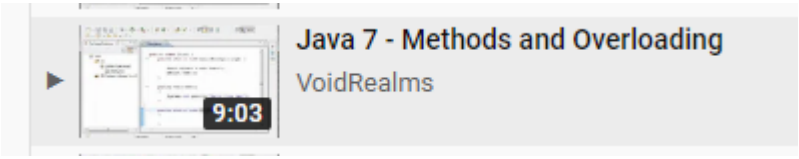
4. Make another object “MultiplyObj2” using class MultiplyClass. matD(4x4)

   print(matC*3.5*matA*2*inverse(matD)*matB) using “MultiplyObj2.Multiply”

Watch the videos



https://www.youtube.com/watch?v=hGRSyIvoIT4&list=PL0C156F1D01276C7C&index=7



Class and Object

- Do not use “static” to create objects

```
public class MultiplyClass {
//Multiply scalar * scalar2
public double Multiply(double scalar, double scalar2) {
return scalar* scalar2;
}

public class HW19 {
public static void main(String[] args) {
          MultiplyClass MultiplyObj1 = new MultClass();
          System.out.println(MultiplyObj1.Multiply(3,4));
          System.out.println(MultiplyObj2.Multiply(matA,4));

          MultiplyClass MultiplyObj2 = new MultClass();
        for(   ){
              for(   ){
                      System.out.println(MultiplyObj2.Multiply(matA,4));
            }
        }
```

Overloading

- We may use the same nave for different methods

```
//Multiply scalar * scalar2
public double Multiply(double scalar, double scalar2) {
return scalar* scalar2;
}

//Multiply scalar * mat A
public double[][] Multiply(double scalar, double[][] matA) {
return scalarMat(scalar, matA);
}
```

(Optional) Deep Learning for Java

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