

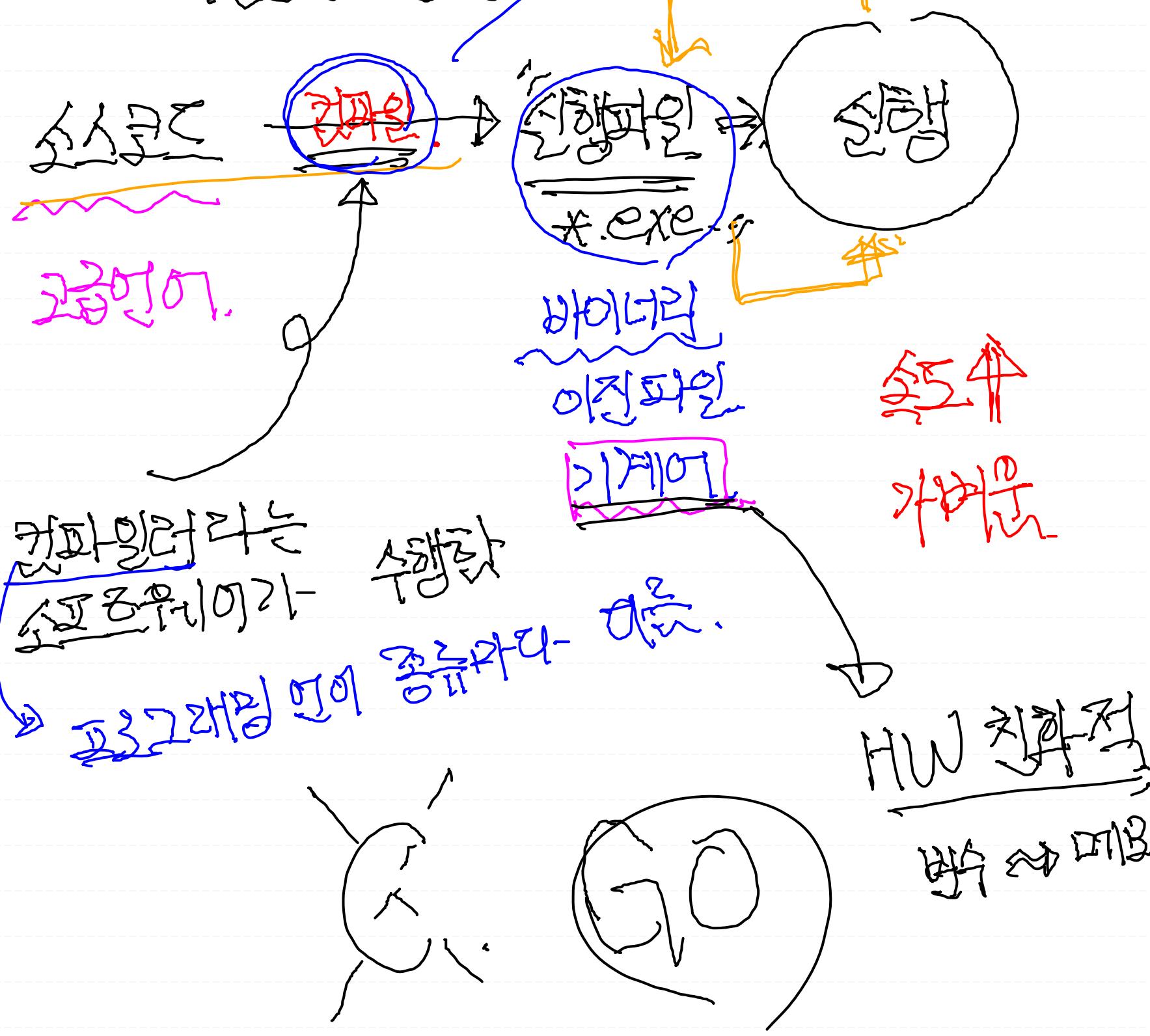
# 1. 프로그래밍 언어의 종류

CPU

~~한국어~~ | 제1 죽은 물고기

수행

# 1) 컴파일 언어



## 2) 인터프리터 언어 (스크립트 언어)

A hand-drawn diagram of a neural network. It consists of three layers: an input layer with four nodes (labeled 1, 2, 3, 4), a hidden layer with two nodes (labeled 5, 6), and an output layer with one node (labeled 7). The connections between layers are shown as black lines. The input layer has arrows pointing to the hidden layer, and the hidden layer has an arrow pointing to the output layer. The output node has a small orange arrow pointing downwards.

~~해석기 = 인문학의 한~~

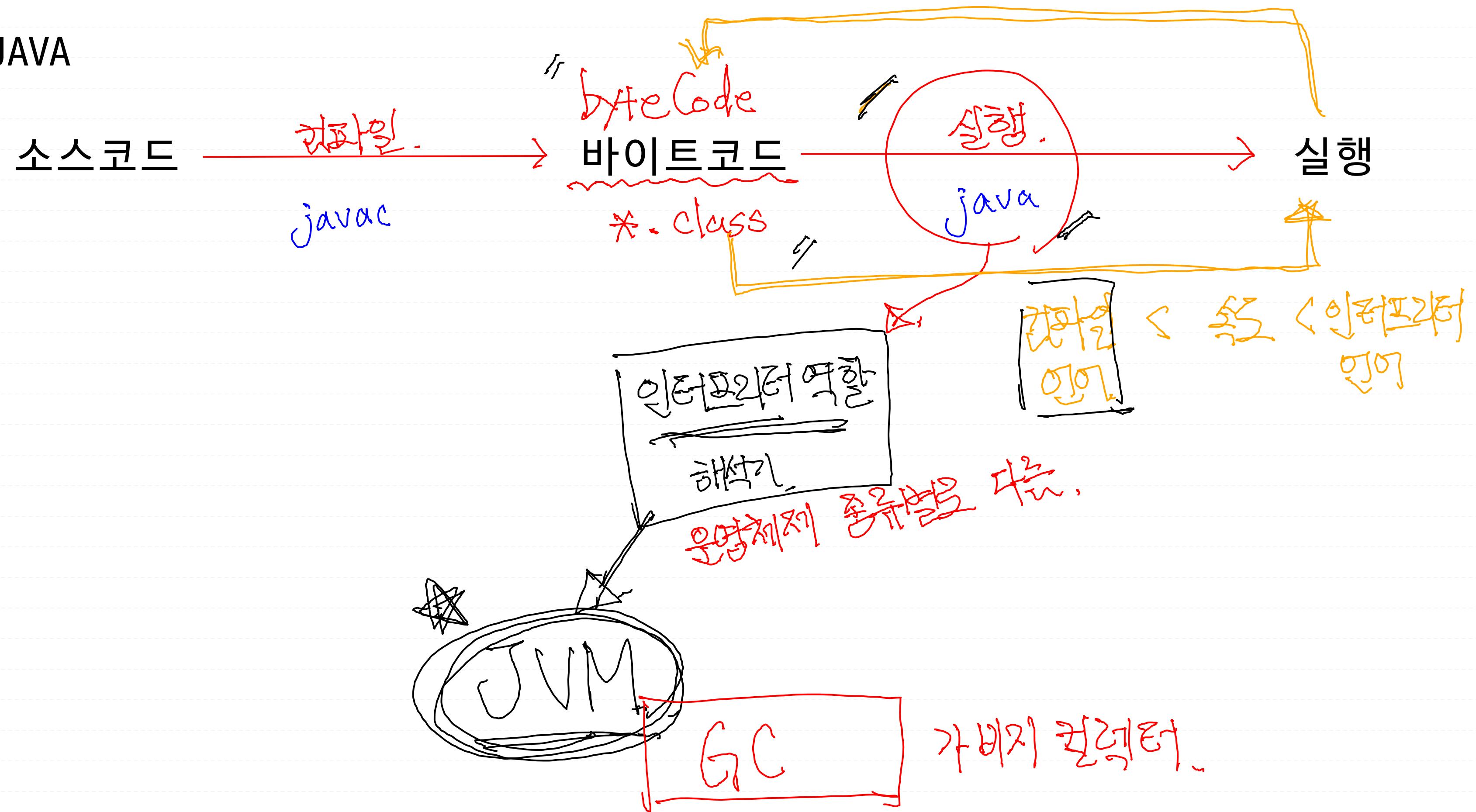
도 그 대 왕 이어 종 을 마 자 나 는.

4) Web Browser { JS.

## 2) Node.js

JS - Python.

# 3) JAVA



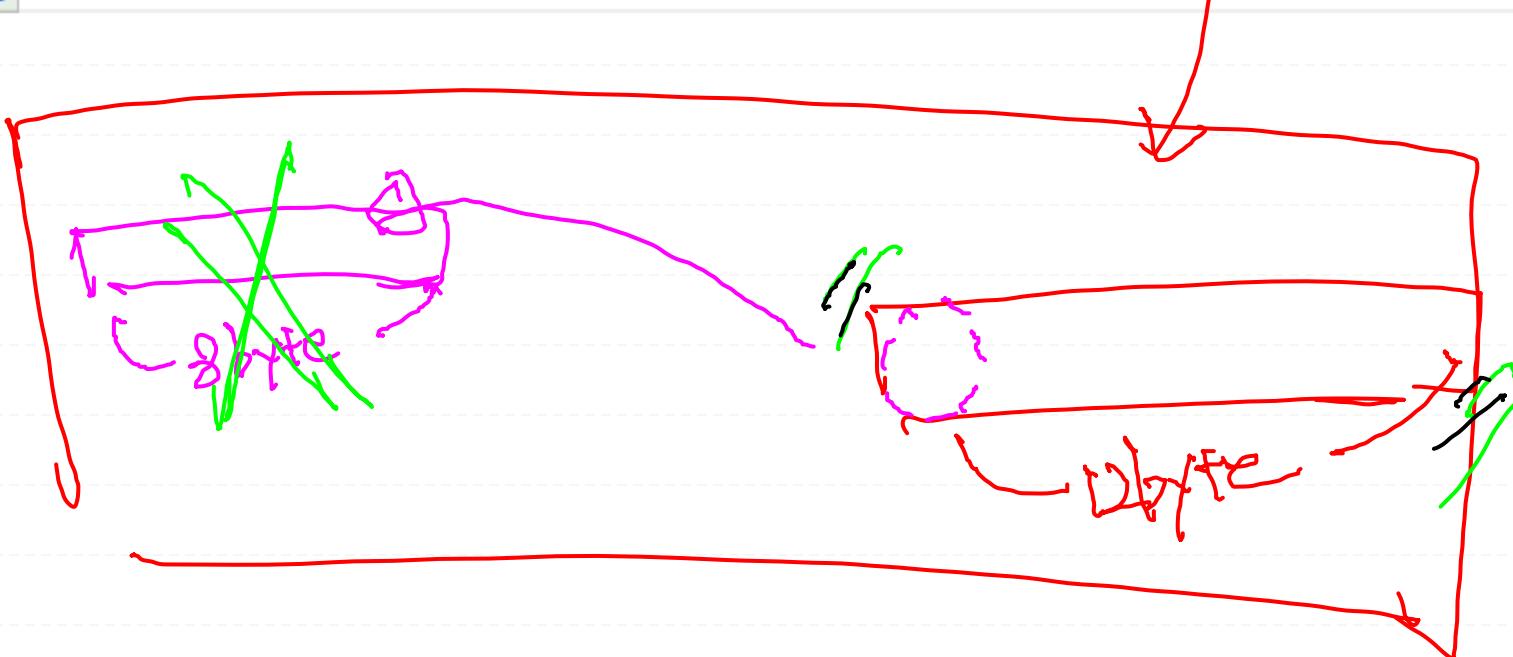
## 2. 가비지 컬렉터????

C언어

```
1 struct Member {  
2     int age;  
3     int height;  
4     int weight;  
5 }
```

구조체  $\Rightarrow$  서로 다른 type의 멤버들을.

```
7 int main() {  
8     struct Member* member;  
9  
10    member = (struct Member*)malloc(sizeof(struct Member));  
11    member->age = 20;  
12    member->height = 180;  
13    member->weight = 70;  
14    memfree(member);  
15    return 0;  
16 }  
17
```



사용이

종료(Destructor 수행 종료) 된

후로 안의 객체이기 때문에  
자동으로 메모리를 반납.

Java.

```
1 class Member {  
2     int age;  $\rightarrow$  4 byte  
3     int height;  $\rightarrow$  4 byte  
4     int weight;  $\rightarrow$  4 byte  
5 }
```

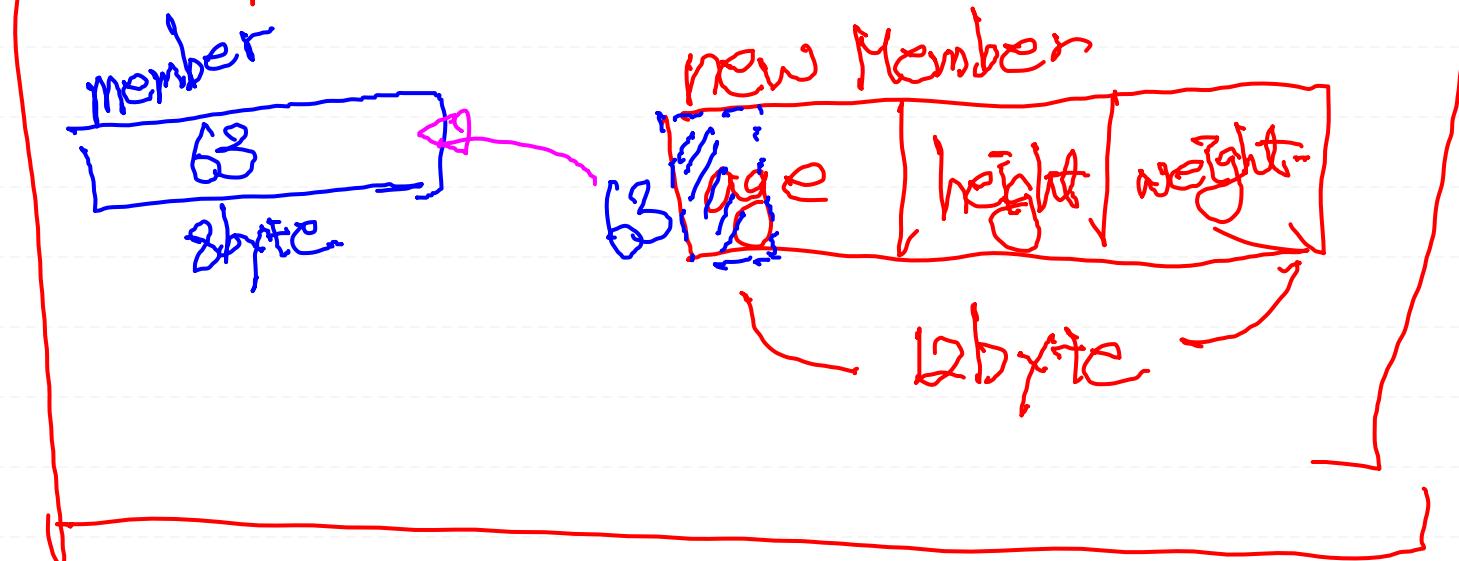
8 byte.

```
7 public class HelloWorld {  
8     public static void main(String[] args) {  
9         Member member = new Member();  
10        member.age = 20;  
11        member.height = 180;  
12        member.weight = 70;  
13    }  
14 }
```

자동적 참조변수

12 byte  
0x1000

Memory



### 3. Go 언어 시작

```
1 import "fmt" → 패키지 참조.  
2  
3 func main() {  
4     fmt.Println("Hello world")  
5 }
```

↓  
함수.  
Go

```
func main() {
```

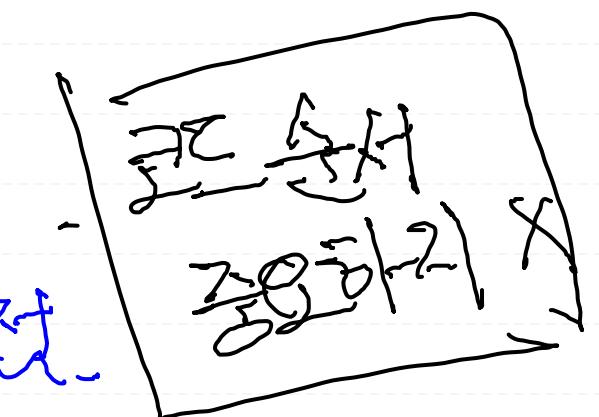
```
    hello();
```

```
}
```

func hello() {  
}

컴파일 언어 특!

main 함수가  
→ 프로그램의 시작점.



JS,  
foo(): 대체



인터프리터 언어 특!

function foo()

Main 함수 없음.  
→ 솔루션으로 실행.

## 4. Go 언어 함수

함수: 프로그램의 기능 단위. ==> 프로그램 명령어의 집합

```
1 import "fmt"
2 // 파라미터
3 // ① 매개변수 타입, 리턴 타입은 이름 뒤에 지정해줍니다
4 func add1(x int, y int) int {
5     return x + y
6 }
7
8 // ② 매개변수 x, y가 같은 타입일 때에는 타입을 한 번만 명시해 줄 수 있습니다.
9 func add2(x, y int) int {
10    return x + y
11 }
12
13 func main() {
14     fmt.Println("add1(x int, y int)의 결과: ", add1(42, 13))
15     fmt.Println("add2(x, y int)의 결과: ", add2(42, 13))
16 }
```

타입 태입

메모리에 사용할  
포인터.

결과값 반환  
= 리턴

55

## 5. 함수의 리턴

```
1 import "fmt"
2
3 //① return 뒤에 리턴 타입을 정해주는 방법
4 func divide1(dividend, divisor int) (int, int) {
5     var quotient = (int)(dividend/divisor) => 3
6     var remainder = dividend%divisor => 1
7     return quotient, remainder
8 }
9
10 //② return 뒤에 리턴할 변수를 선언하는 방법. ①과는 달리 함수 내부에서 `quotient`를 `var`로 선언하지 않고 바로 씁니다.
11 func divide2(dividend, divisor int) (quotient, remainder int) {
12     quotient = (int)(dividend/divisor)
13     remainder = dividend%divisor
14     return //return이라고만 적으면 미리 return값으로 정해 놓은 quotient와 remainder를 return합니다.
15 }
16
17 func main() {
18     //①로 한 번에 여러개의 결과를 return 받는 부분
19     var quotient, remainder int
20     quotient, remainder = divide1(10, 3)
21     fmt.Println("①의 결과:", quotient, remainder)
22
23     //②로 한 번에 여러개의 결과를 return 받는 부분
24     quotient, remainder = divide2(10, 3)
25     fmt.Println("②의 결과:", quotient, remainder)
26 }
```

- 1) 배열을 표현
- 2) Beans을 표현

# 6. 여러가지 변수의 선언

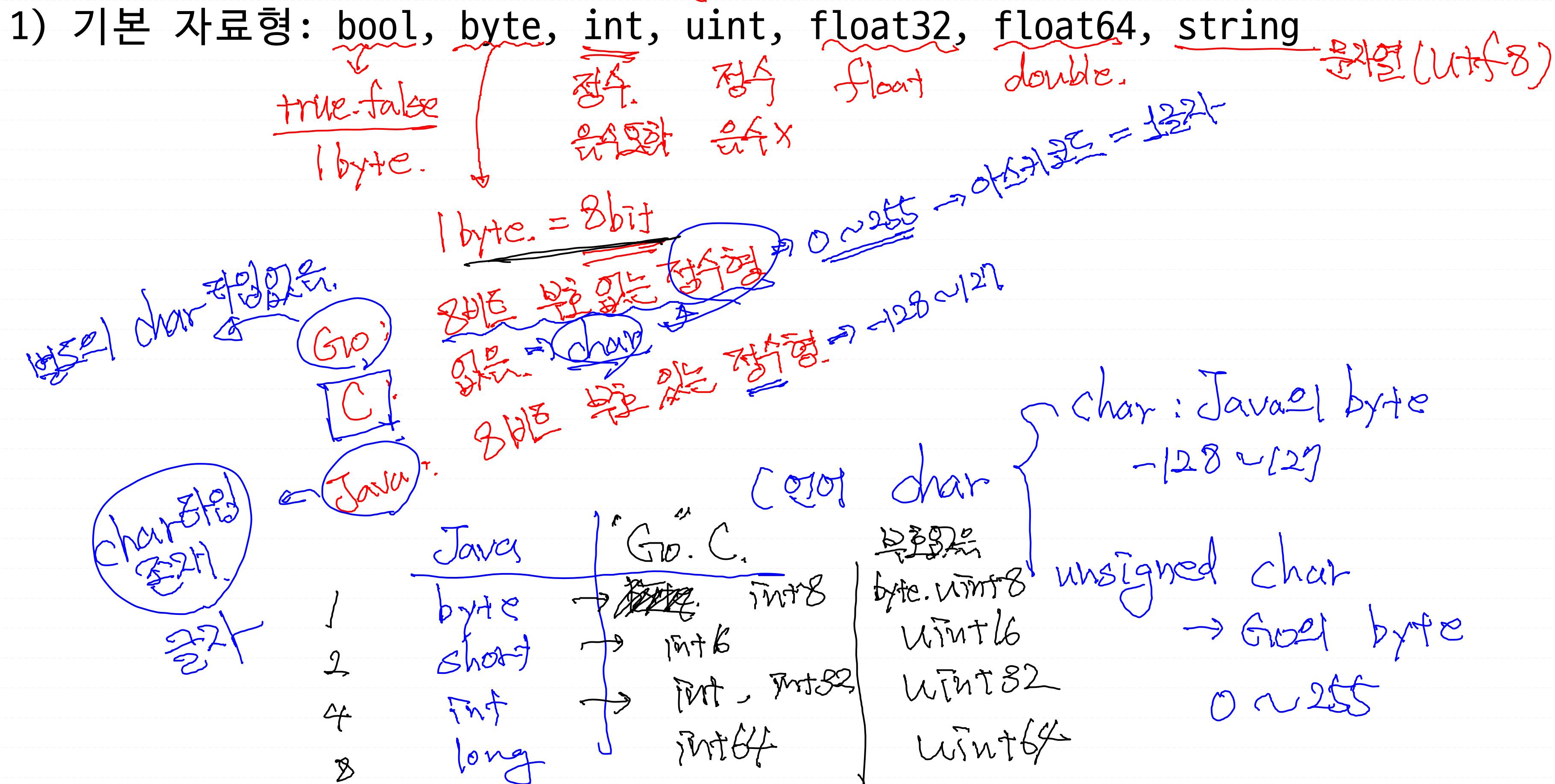
```
1 import "fmt"
2
3 //① 변수를 하나 선언
4 var num1 int,
5
6 //② 같은 타입을 가지는 변수를 여러 개 선언
7 var num2, num3 int
8
9 //③ 여러 변수에 한 번에 값을 초기화 : 선언과 동시에 값을 초기화하면 타입을 명시할 필요가 없습니다.
10 var num4, num5, str1 = 4, 5, "example"
11
12 //④ 함수 밖에서는 :=을 쓸 수 없습니다.
13 errorvar := str1
14
15 //⑤ 다른 타입을 가지는 변수를 여러 개 선언
16 var (
17     i int
18     b bool
19     s string
20 )
21
22 func main(){
23     fmt.Println("①", num1)
24     fmt.Println("②", num2, num3)
25     fmt.Println("③", num4, num5, str1)
26
27 //④ 함수 안에서는 :=을 쓰면 var과 타입을 지정하지 않고 변수를 선언과 동시에 초기화할 수 있습니다.
28     num6 := 6
29     fmt.Println("④", num6)
30
31     fmt.Println("⑤", i, b, s)
32 }
```

var i int  
var b bool  
var s string

var num6 int  
num6 = 6  
var num6 := 6  
num6 := 6

## 7. DataType

플랫폼에 따라 크기 다른 (4byte or 8byte)



## 7. DataType (계속..)

```
1 import (
2     "fmt"
3     "math/cmplx"
4 )
5
6 var (
7     i int ← 정수
8     f float64 ← 실수
9     MaxInt uint64 ← 정수 ← 64bit
10    z complex128 = cmplx.Sqrt(-5 + 12i)
11 )
12
13 func main() {
14     const format = "%T(%v)\n"
15     fmt.Printf(format, MaxInt, MaxInt)
16     fmt.Printf(format, z, z)
17
18     f = i ← 정수 ⇒ 형변환 필요
19     f = float64(i),
20 }
```

int a: ← Java : 4byte. = x8 ⇒ 32bit  
128bit → 32bit

var a int64 ← Go : 64byte → 8byte  
128bit → 64bit.



%d ⇒ 정수

%S ⇒ 문자열.

Java

f=(float) i

%T ⇒ Type.

%v ⇒ value

printf(-----)

## 7. Zero values

- 선언만 하고 초기화 하지 않은 변수

- \* 숫자타입은 0
- \* boolean 타입은 false
- \* string 타입은 ""

으로 자동 초기화 됨

Java  $\Rightarrow$  { 숫자  $\Rightarrow$  0      boolean  $\Rightarrow$  false }

String  $\Rightarrow$  null.

JS  $\Rightarrow$  ?    undefined

## 8. 상수선언

const 기호.

```
1 import (
2     "fmt"
3 )
4
5 var (
6     i int // zero value = 0
7     f float64 // zero value = 0
8     b bool // zero value = false
9     s string // zero value = ""
10 )
11
12 func main() {
13     fmt.Printf("int의 zero value: %v\n", i)
14     fmt.Printf("float64의 zero value: %v\n", f)
15     fmt.Printf("boolean의 zero value: %v\n", b)
16     fmt.Printf("string의 zero value: %q\n", s)
17 }
```

## 9. 조건문

if 조건식 {

}

( ) 가 필요.  
아니.

if 조건식 {

} else {

}

if 조건식 {

} else if 조건식 {

} else {

}

조건식이 반드시

true/false로 판명 나야함.

```
1 import (
2     "fmt"
3     "math"
4 )
5
6 func sqrt(x float64) string { // 정의
7     if x < 0 {
8         return sqrt(-x) + "i"
9     }
10    v else { // else문은 if문이 닫히는 {} 줄과 함께 쓰여야 합니다.
11        return fmt.Sprint(math.Sqrt(x))
12    }
13 }
14
15 func pow(x, n, lim float64) float64 {
16     if v := math.Pow(x, n); v < lim {
17         return v
18     }
19     // v는 if문 내부에서만 쓸 수 있고, 여기부터는 쓸 수 없습니다.
20
21     return lim
22 }
23
24 func main() {
25     fmt.Println(sqrt(2), sqrt(-4)) // 정의
26
27     fmt.Println(
28         pow(3, 2, 10),
29         pow(3, 3, 20),
30     )
31 }
```

```
1 package main
2
3 func main() {
4     var name string
5     var category = 1
6
7     switch category {
8         case 1: // 1번 카테고리
9             name = "Paper Book"
10        case 2:
11            name = "eBook"
12        case 3, 4:
13            name = "Blog"
14        default:
15            name = "Other"
16    }
17    println(name)
18
19    // Expression을 사용한 경우
20    switch x := category << 2; x - 1 {
21        //...
22    }
23 }
```

case 1: // 1번 카테고리

break는 꼭 X

case 5까지 X

```
func grade(score int) {  
    switch {  
        case score >= 90:  
            println("A")  
        case score >= 80:  
            println("B")  
        case score >= 70:  
            println("C")  
        case score >= 60:  
            println("D")  
        default:  
            println("No Hope")  
    }  
}
```

if ~ else if ~ else  
else if  
else.

fmt.Println

func println(s string){  
 fmt.Println(s)  
}

V<sub>i</sub><sup>o</sup> =

```
switch v.(type) {  
    case int:  
        println("int")  
    case bool:  
        break // 차등.  
        println("bool")  
    case string:  
        println("string")  
    default:  
        println("unknown")  
}
```

타입별 문가 가눙.

```
import "fmt"  
  
func main() {  
    check(2)  
}  
  
func check(val int) {  
    switch val {  
        case 1:  
            fmt.Println("1 이하")  
            fallthrough // 다음 cases 넘어감  
        case 2:  
            fmt.Println("2 이하")  
            fallthrough  
        case 3:  
            fmt.Println("3 이하")  
            fallthrough  
        default:  
            fmt.Println("default 도달")  
    }  
}
```

## 10. 반복문 ==> for문만 존재함

```
package main

func main() {
    sum := 0
    for i := 1; i <= 100; i++ {
        sum += i
    }
    println(sum)
}
```

( var i int  
i = 1  
 $\Rightarrow \text{var } i \text{ int} = 1$   
 $\Rightarrow i := 1$

package main

```
func main() {  
    n := 1  
    for n < 100 {  
        n *= 2  
    }  
    println(n)  
}
```

Go의 for문은 초기식, 증감식 중 하나  
혹은 모두를 생각 가능

↑ 초기식  
↑ 조건식  
↑ 증감식

) like while.

$O(\log n)$

```
for {  
    println("Infinite loop")  
}
```

⇒ 무한루프

```
package main
func main() {
    var a = 1
    for a < 15 {
        if a == 5 {
            a += a
            continue // for루프 시작으로
        }
        a++
        if a > 10 {
            break //루프 빠져나옴
        }
    }
    if a == 11 {
        goto END //goto 사용예
    }
    println(a)
}

END:
    println("End")
}
```

← 바흐.

# 11. 배열

```

package main
func main() {
    var a [3]int //정수형 3개 요소를 갖는 배열 a 선언
    a[0] = 1
    a[1] = 2
    a[2] = 3
    println(a[1]) // 2 출력
}

```

int[3]

*배열 크기를 알지 못해*

배열의 초기화

```

var a1 = [3]int{1, 2, 3}
var a3 = [...]int{1, 2, 3} //배열크기 자동으로

```

리스트의 특성 또한

len(배열이름) ⇒ 길이

Java : 배열이름.length

```

func main() {
    var a = [2][3]int{
        0 {1, 2, 3},
        1 {4, 5, 6}, // 끝에 콤마 추가
        2
    }
    println(a[1][2]) ⇒ 6
}

```

var a []int  
a = append(a, 100) 원소추가

$$\frac{1}{2} + \frac{3}{4} = \frac{1 \times 4 + 3 \times 2}{2 \times 4} = \frac{4 + 6}{8} = \frac{10}{8} = \frac{5}{4}$$

A handwritten calculation showing the addition of two fractions,  $\frac{1}{2}$  and  $\frac{3}{4}$ . The first fraction is shown with a red circle around the numerator 1 and a blue circle around the denominator 2, with a red X drawn through them. The second fraction is shown with a blue circle around the numerator 3 and a red circle around the denominator 4, with a blue X drawn through them. The numerators are multiplied by their respective denominators to find a common denominator of 8, resulting in the equation  $\frac{1 \times 4 + 3 \times 2}{2 \times 4} = \frac{4 + 6}{8}$ . This is then simplified to  $\frac{10}{8}$ , which is further simplified to  $\frac{5}{4}$ .

$$\left( \frac{\text{numer1}}{\text{denom1}} + \frac{\text{numer2}}{\text{denom2}} \right)$$

A general formula for adding two fractions. It shows two fractions with numerators numer1 and numer2 and denominators denom1 and denom2. The formula is enclosed in large blue parentheses.