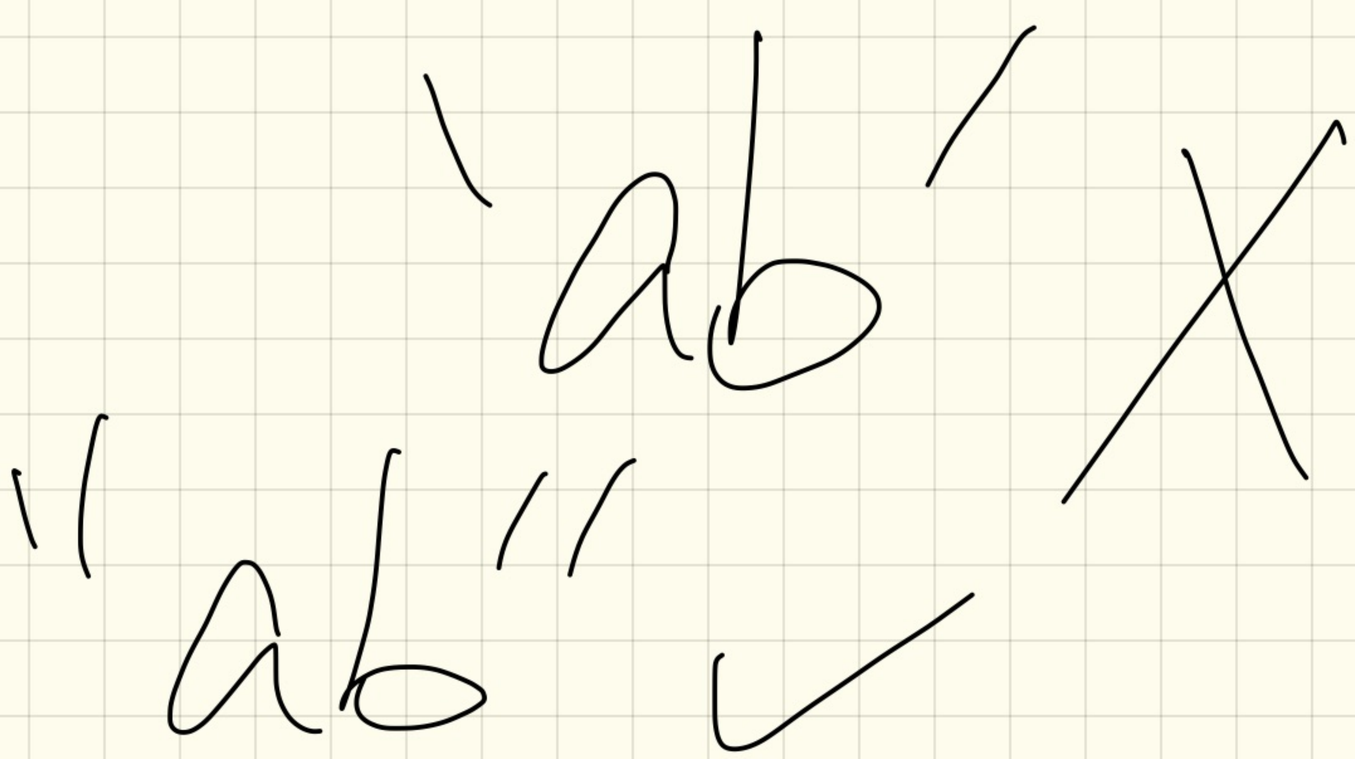
EECS 1022 Winter 2018

Programming for Mobile Computing

Monday, Jan. 8

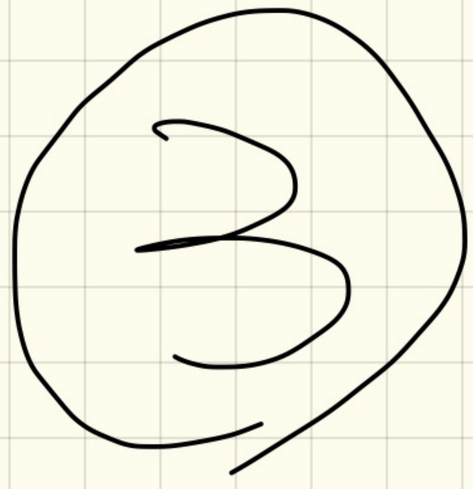
Lecture 1

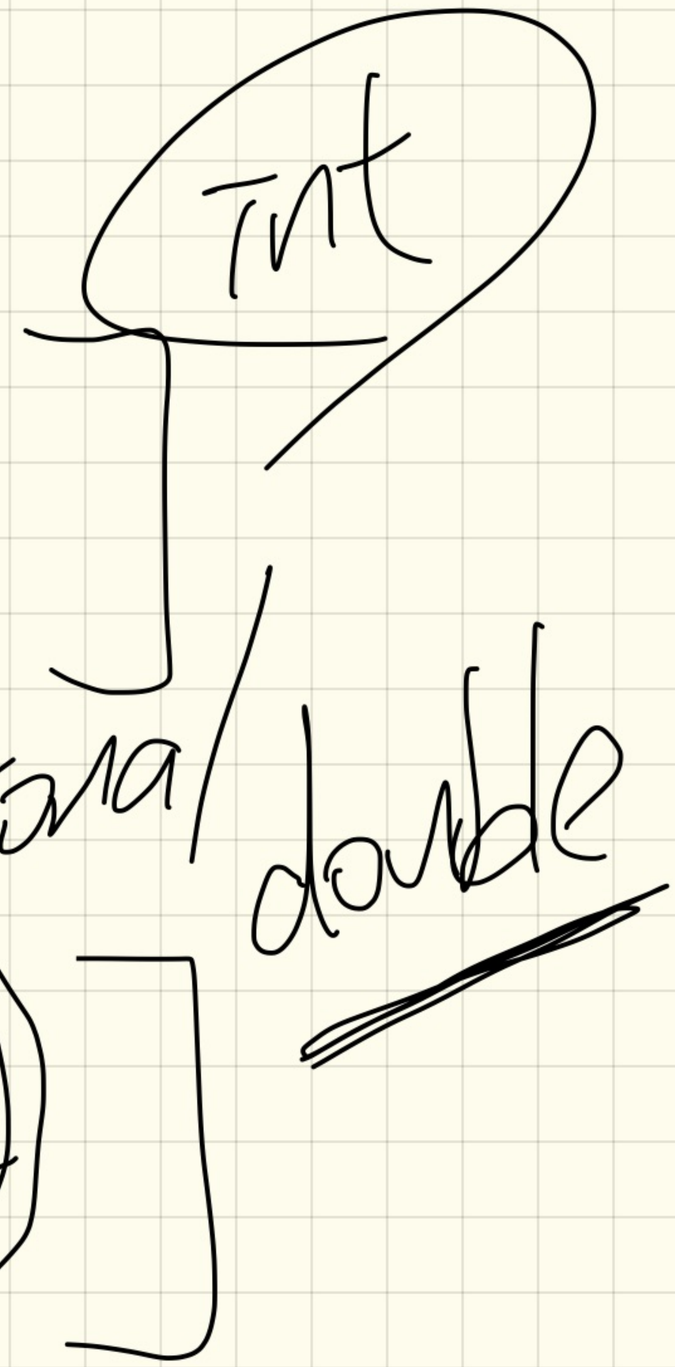


Math

Integer

Real



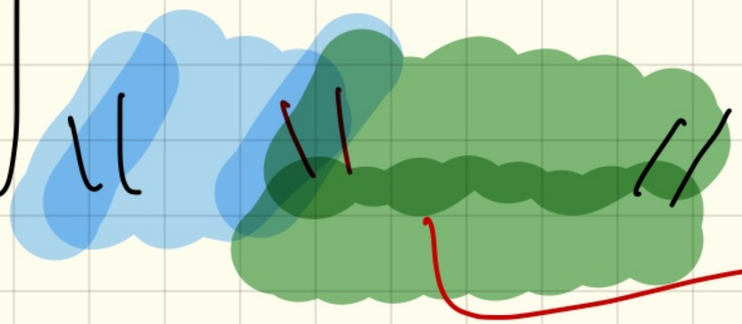


Java

2 integer

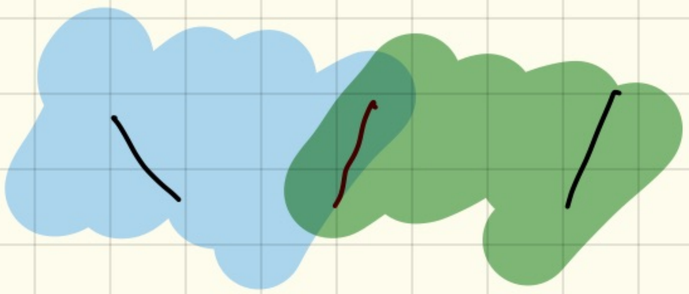
2.0 double

String literal



(enclosed within matching double quotes)

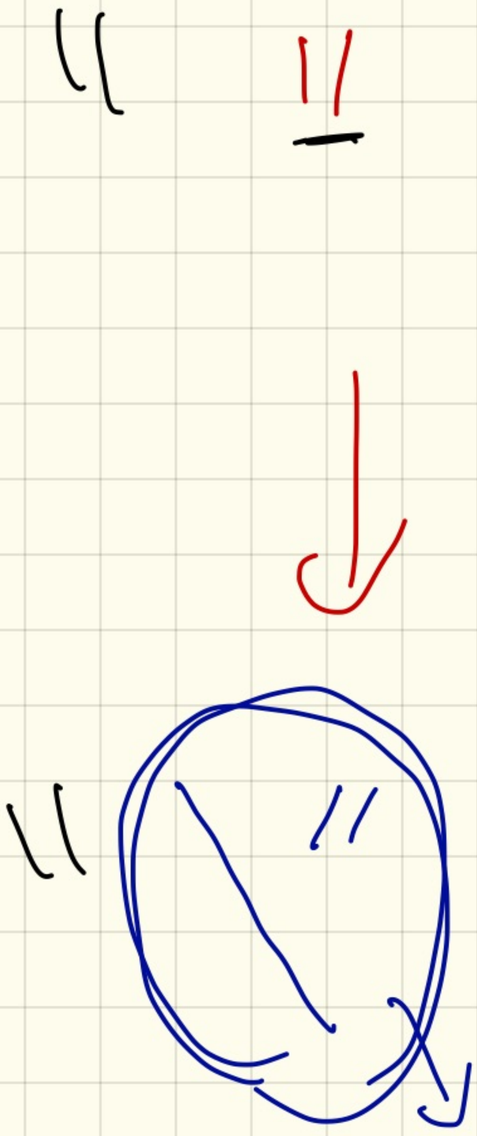
Character literal



⇒ The compiler gets confused

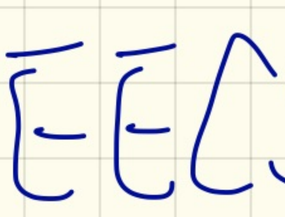
⇒ disambiguate usage escape seq.

" " " "



has a special meaning to the compiler: a double quote

EECS1022



York University

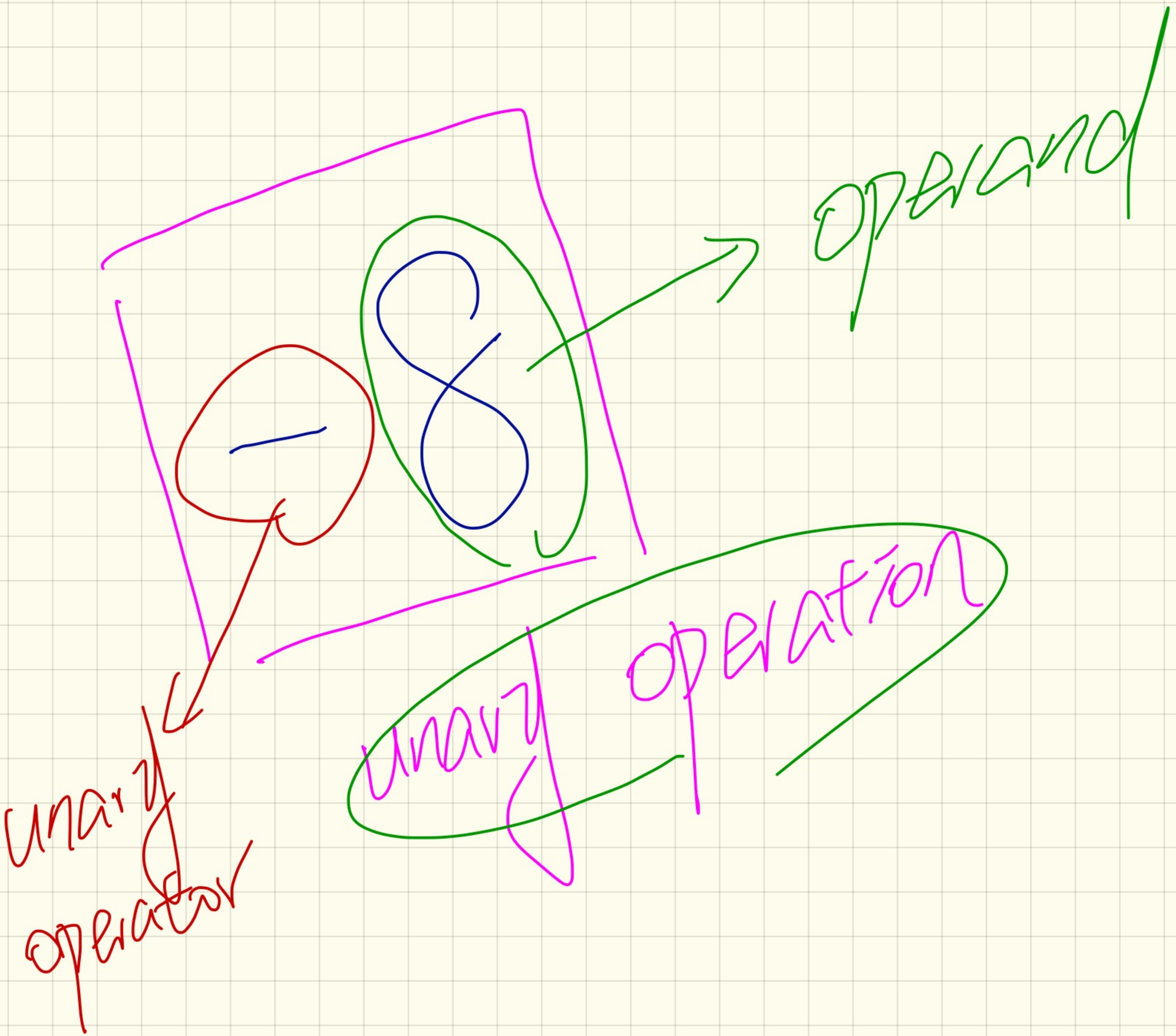
addition operation

left operand

right operand

3 + 4

binary operator


-8
operand
unary operation
unary operator

① 13 / 4

3

② 13.0 / 4 3.25

3

3.25

Division in Java

LO / RO

① If the LO and RO are both integers, evaluate the quotient.

② Otherwise, evaluate mathematically.

13/4

3

13.0/4

3.25

13/4.0

3.25

12.0/4.0

3.25

13 / 4 = 3

quotient

13 % 4

modulo

/

remainder.

$$(13 / 4) * 4 + \frac{(13 \% 4)}{1}$$

$$\frac{3}{12}$$

13

Given integers a, b

$(a / b) * b + (a \% b)$

=

a

Math

=

>

<

>=

<=

Java

==

>

<

>=

<=

The plus (+) operation has two possible meanings:

① 2 + 3 = 5

int + int

addition

② "a" + "b" = "ab"

"1" + "2" = "12"

concatenation

(2)

String

"EECS" + 1022 → EECS1022

"1022" + 23 → 102223

Variable Declaration

int c;

c = 23;

c = 46;

c = "hi";

String

int → type

c → variable name

c → constrains what can be stored in 'c'

c → 46

final double PI = 3.14;

constant

double radius;

names

data types

PI = 6.28; X

3.14

PI

int ? int ?

(1 + 2) ≠ (23 % 5)

int ?

int ?

("(" + ")") * (46 % 4)

String

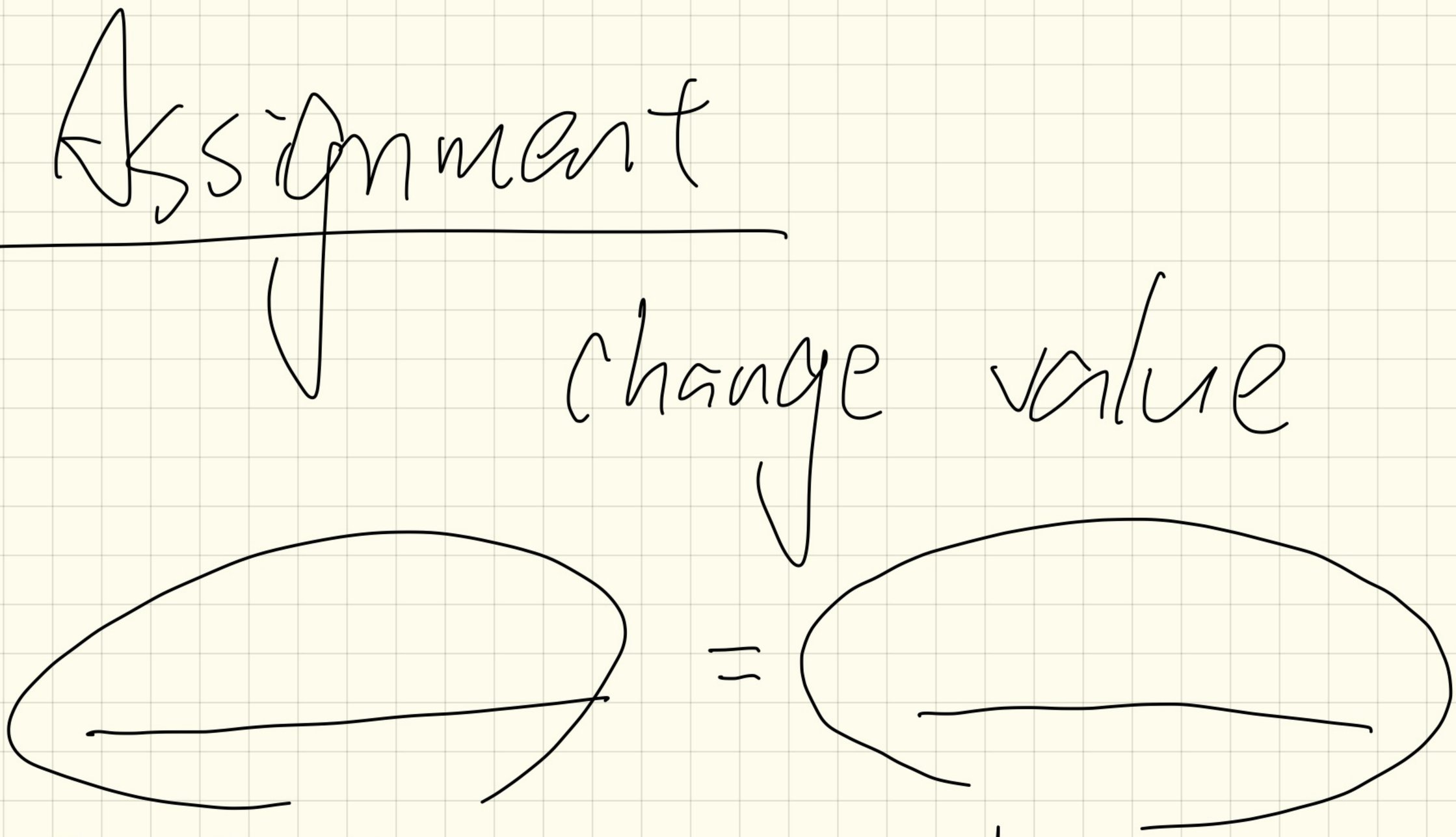
int

Monday Jan. 15

Lecture 2

Assignment

change value



① int i
-final double PI

② i =

any expr. with LRS whose value is com.

①

int i = 23 * 4; (92)

final double PI = 3.14;

200

~~184~~

---

② trace

int j = i * 2; (92)

100

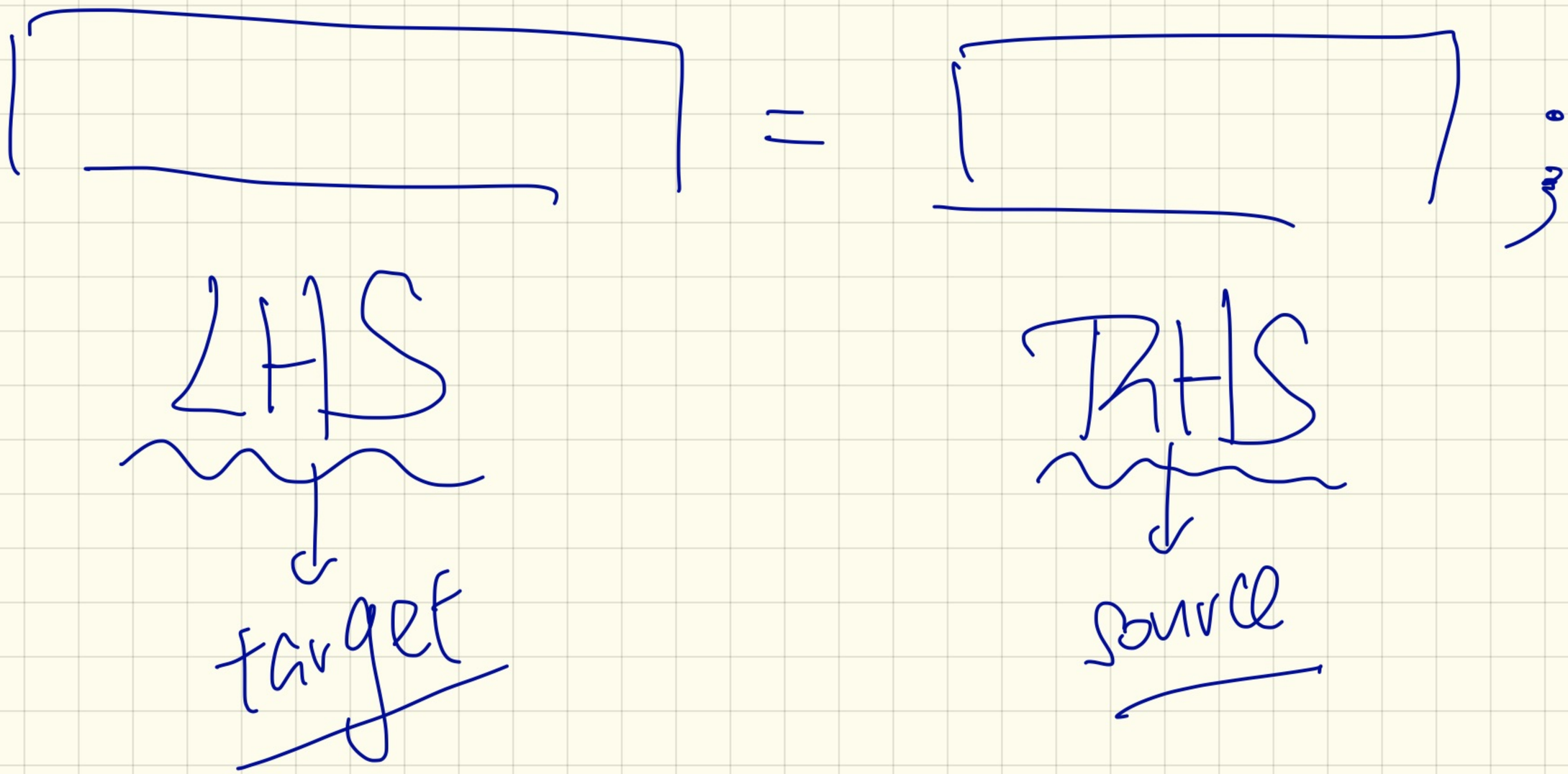
~~92~~

i

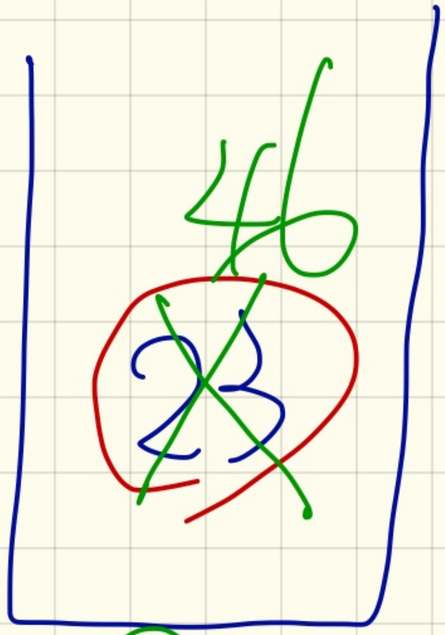
i = 100;

j = i * 2; (200)

Assignment



int i = 23;



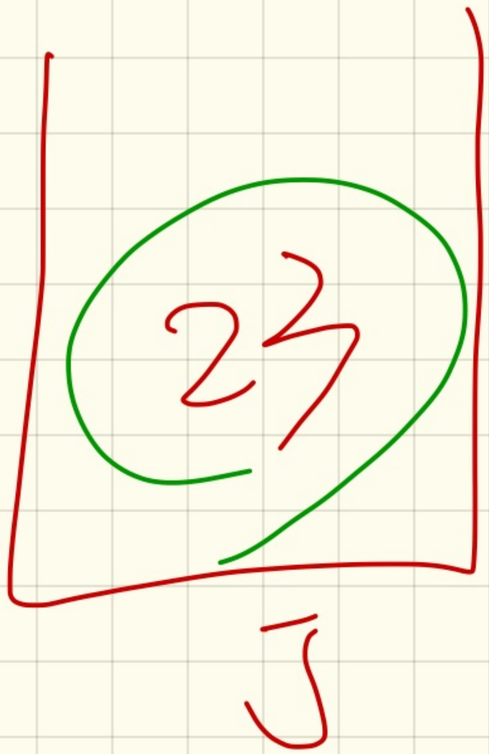
i integers only

int j = i;

src

tar:

i = 46;



double value1 = 2 * 4.5;

int

double

Coercion

↓

promotion

3.0

double

int value1 = 3 * 4.5

↓ promotion coercion

3.0

13.5

13

5

0

13.5 X

not compile

:(you cannot store 13.5 into an int variable.

int value1 = 3 * 4.5;

135

0.5

13

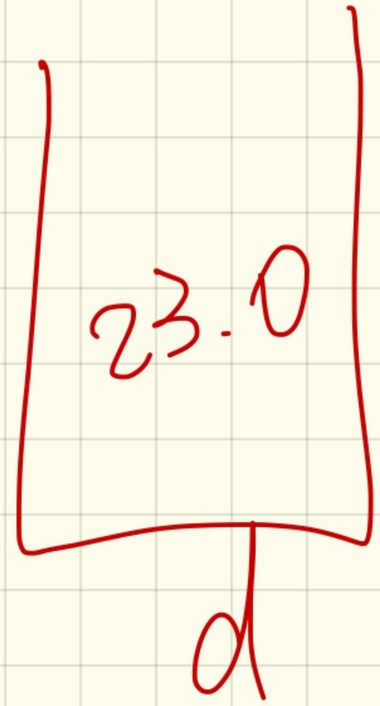
int value1 = (int)3 * 4.5;

13

value1

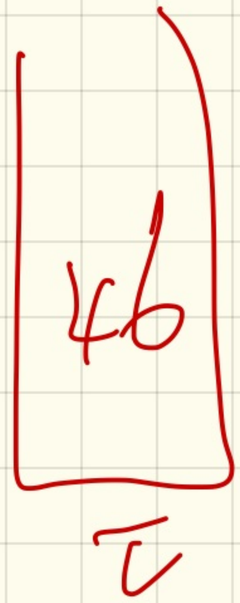
after cast, .5 is truncated.

① double d = 23;



② ~~int~~ int c = 46.23;

int c = (int) 46.23;



println(1 / 2) ; → 0 (%)

1 int, 2 int

1.0 / 2 → quotient

println((double) 1 / 2) ; → 0.5

println(1 / (double) 2) ; → 0.5

println((double) 1 / (double) 2) ; → 0.5

1.0 / 2.0

int i = 23;

int j = 5;

println(i / j); 4

int

println((double) i / j); 4.6

23.0

int

5

int i = 10;

int j = 4;

casting has higher precedence than arithmetic operation.

① println((double) i / j); 2.5

② println((double) (i / j)); 2.0

2

(3+4) * b

3 + 4 * 6

27

lower precedence

Precedence

higher precedence

|||

3 + (4 * 6)

int i = 10;

int j = 4;

(double)

printf((double) i / j);

2.5

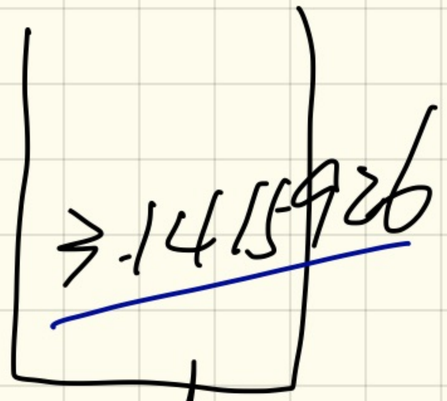
111

printf(((double) i) / j);

Trace.

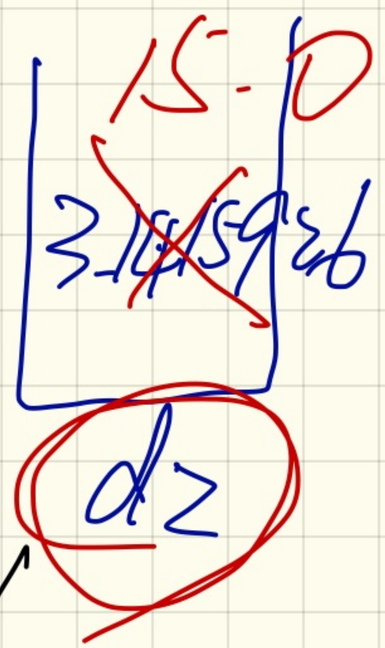
① double d1 = 3.1415926;

printf(d1); // 3.1415926



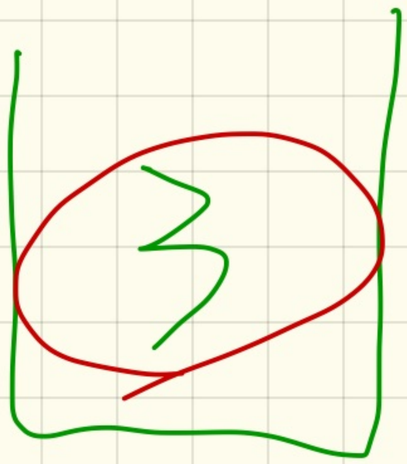
② double dz = d1;

printf(dz); // 3.1415926



③ int i1 = (int) d1;

printf(i1); //



④ dz = i1 * 5;

printf(dz); 15.0

```
int i = 23;
```

```
i = i + 46;
```

```
i += 46;
```

```
int j = 46;
```

```
j = j * 2;  ⟶  j *= 2;
```

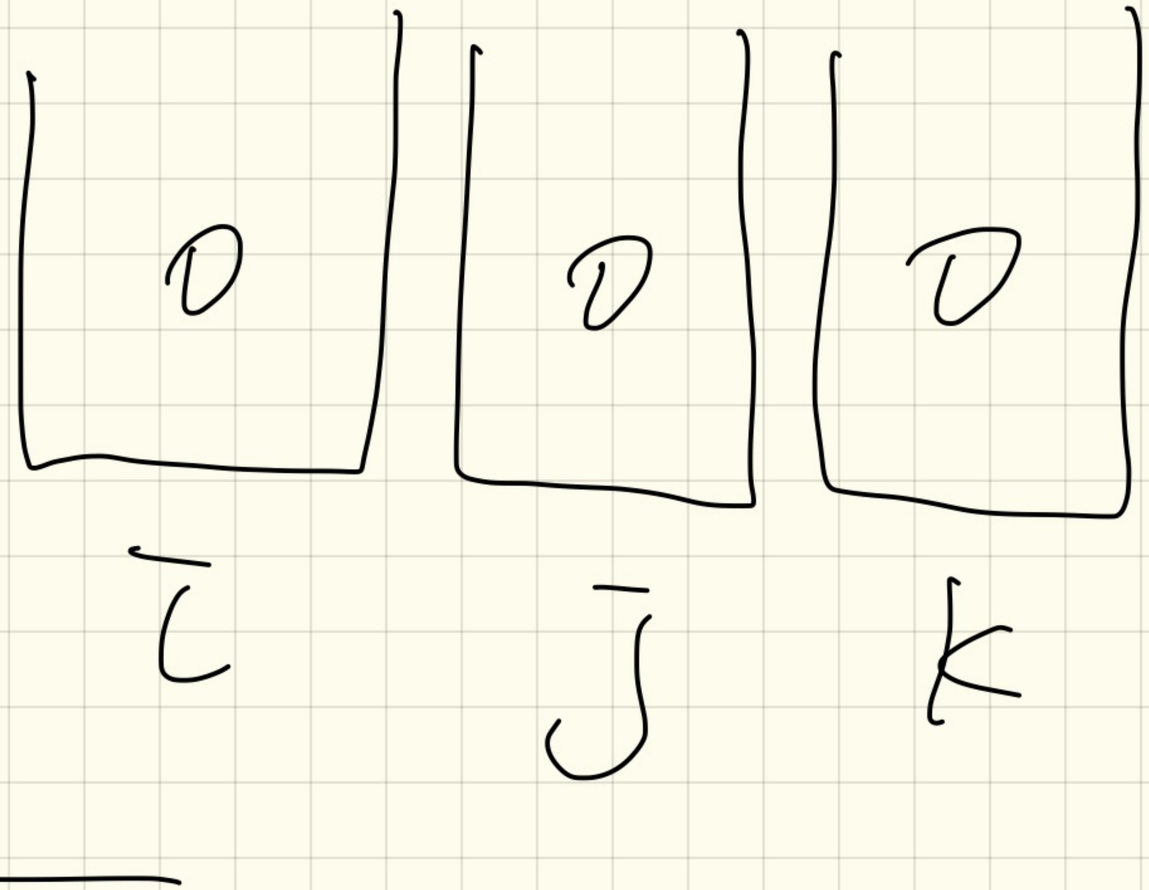
Int k = 23;

k = k - 1;

k -- ;

k --

```
int i = 0;
int j = 0;
int k = 0;
```




```
k = i++;
k = i++;
k = j++;
k = j++;
```

Selections

boolean expression

① int i = 23;

if (i > 0) {

}

② int j = -24;

if (j > 0) {

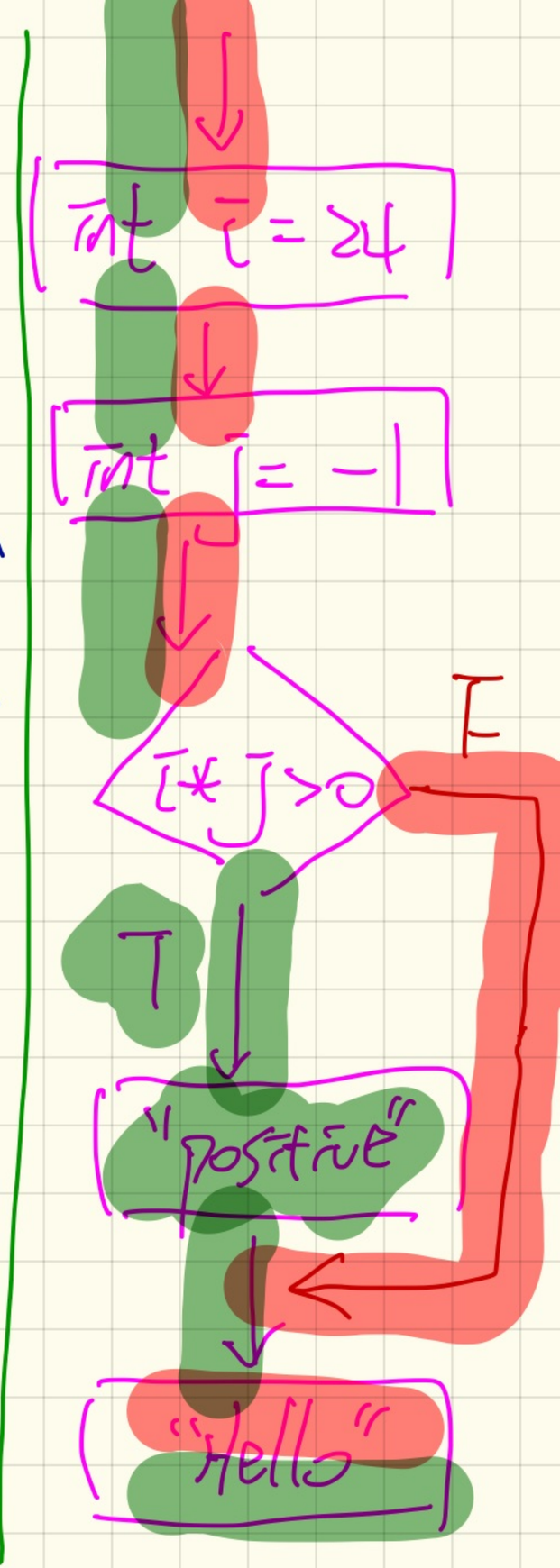
}

e.g.

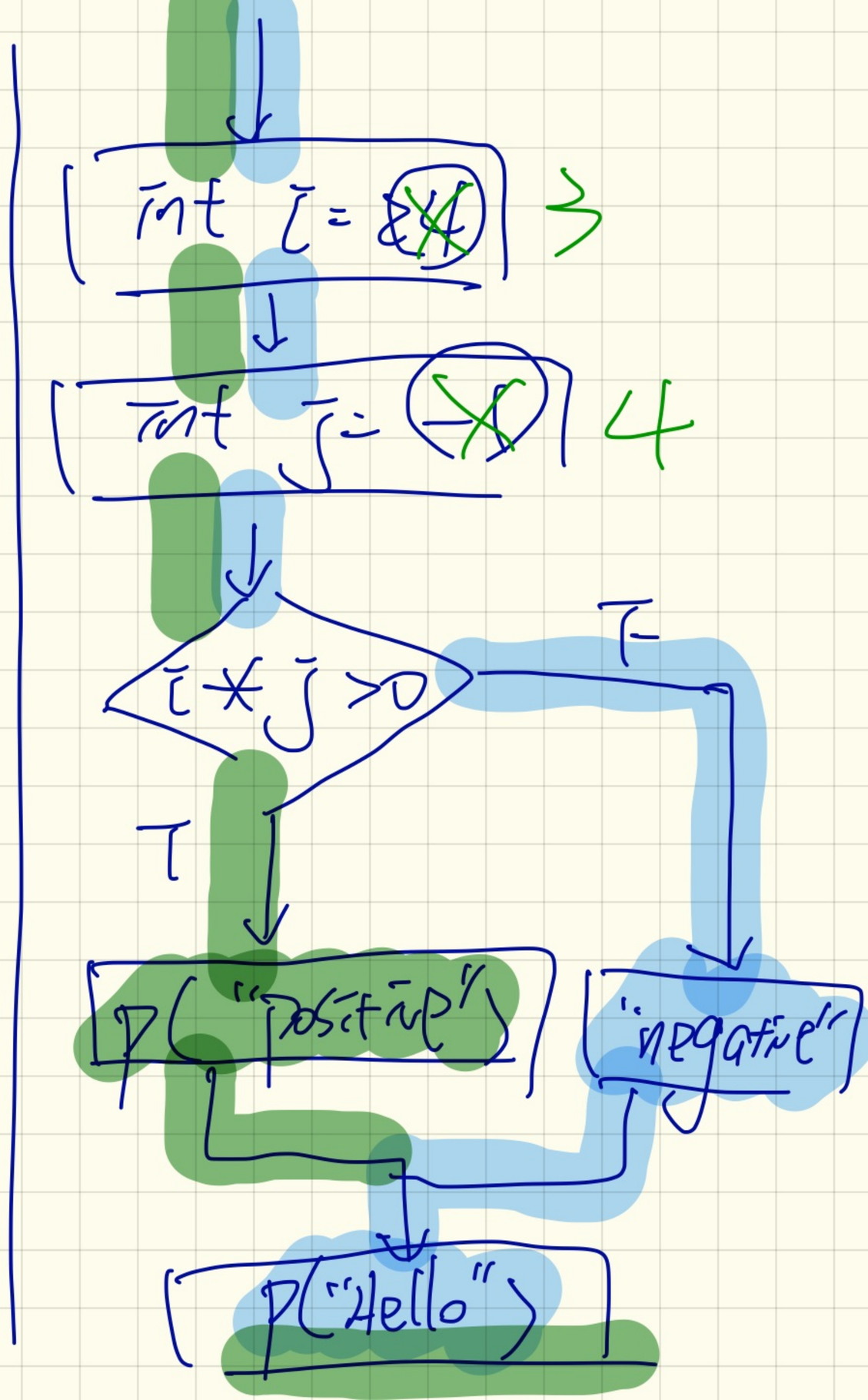
```
int i = 24;
int j = -1;
if (3*i + 4*j > 0) {
    println("positive");
}
println("Hello");
```

Hello

positive
Hello



```
int i = 3;
int j = 4;
if (i * j > 0) {
    println("positive");
} else {
    println("negative");
}
println("Hello");
```



Monday Jan. 22

Lecture 3

T

[x <= y

3 5

x = 3

y = 5

!(x > y)

3 5

F T

```
if ( x < y )
{
    int i = 3;
    printf( i * 2 );
}
```

branching condition

body of branch

if-statement

if-statements

1

```
if (          ) {
}
```

2

```
if (          ) {
}
else if (          ) {
}
```

3

```
if (          ) {
}
else {
}
```

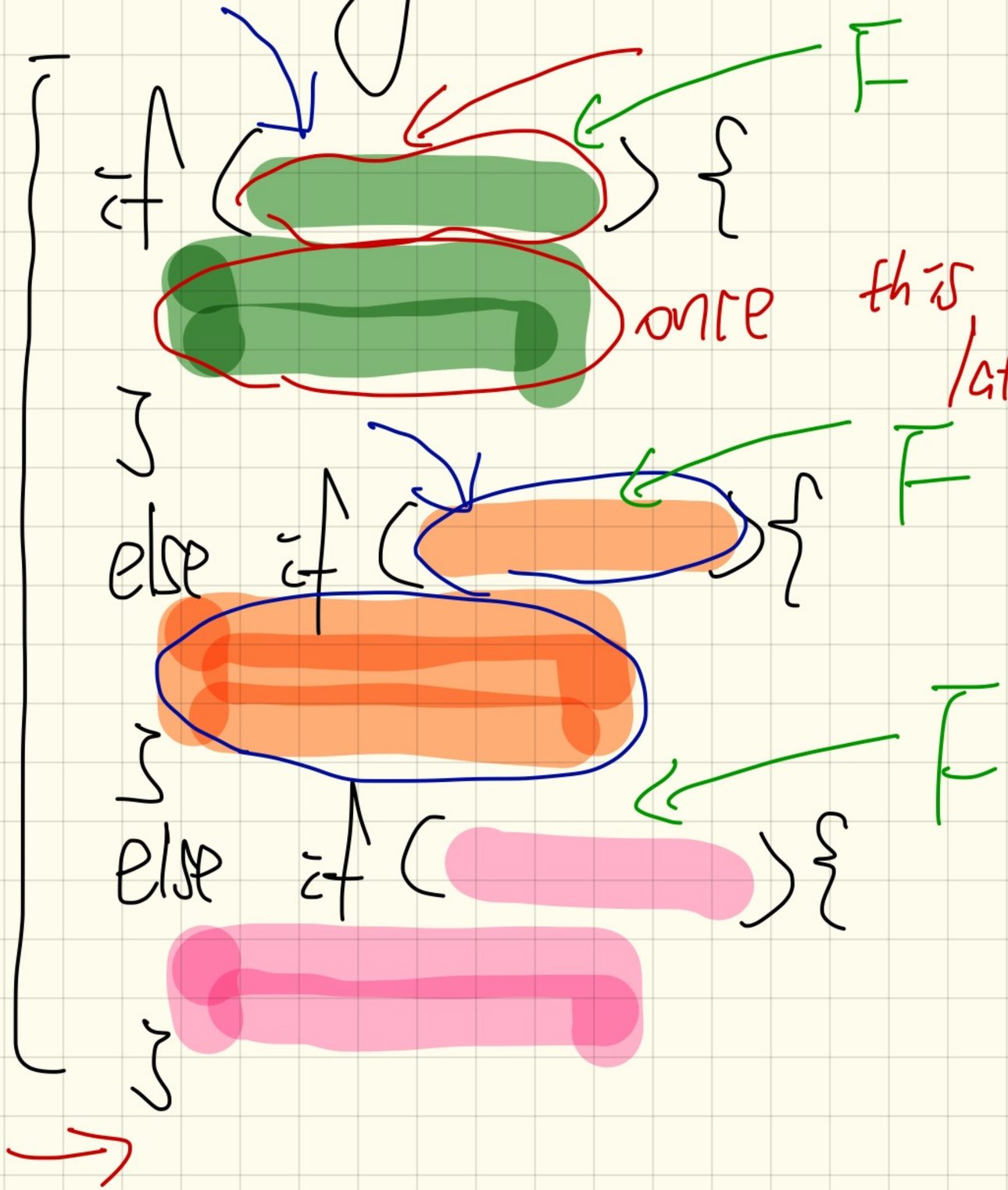
Q1

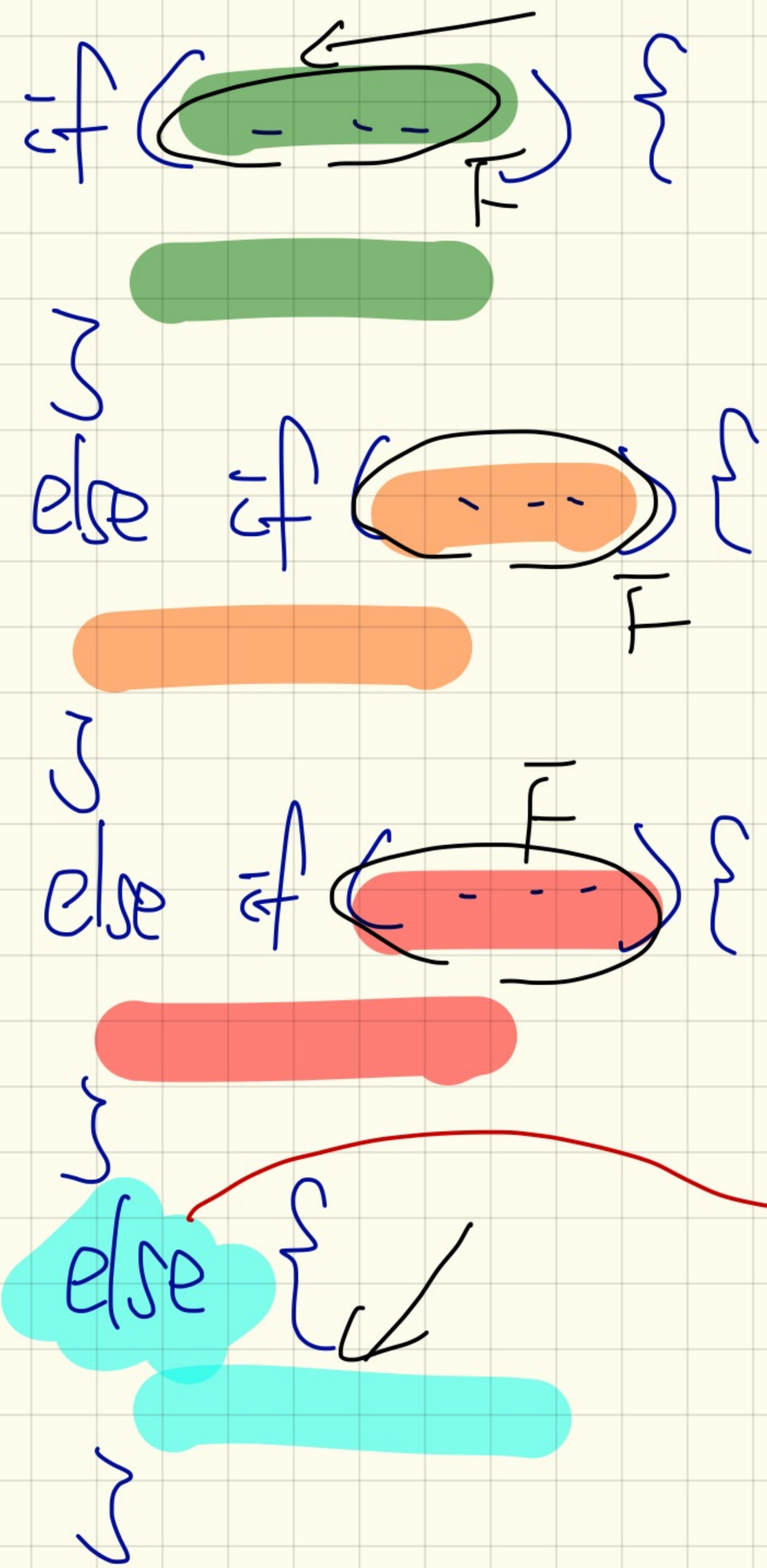
```
if ( . . . ) {
    . . .
}
else if ( . . ) {
    . . .
}
else if ( . . ) {
    . . .
}
```

Q2

```
if ( . . . ) {
    . . .
}
if ( . . . ) {
    . . .
}
else if ( . . ) {
    . . .
}
if ( . . . ) { . . }
```

Executing an if-statement





default branch:
when all previous branching conditions evaluate to F, this branch is expected.

```java
int i = 5;
if(i < 0) {
  System.out.println("i is negative");
}
else if(i < 10) {
  System.out.println("i is less than than 10");
}
else if(i == 10) {
  System.out.println("i is equal to 10");
}
else {
  System.out.println("i is greater than 10");
}
```

5 < 0 → F

5 < 10 → T

i is less than 10

```
int i = 12;
if(i < 0) {
  System.out.println("i is negative");
}
else if(i < 10) {
  System.out.println("i is less than than 10");
}
else if(i == 10) {
  System.out.println("i is equal to 10");
}
```

```
int i = 12;
if(i < 0) {
  System.out.println("i is negative");
}
else if(i < 10) {
  System.out.println("i is less than than 10");
}
else if(i == 10) {
  System.out.println("i is equal to 10");
}
else {
  System.out.println("i is greater than 10");
}
```

12 < 0 F

12 < 10 F

12 == 10 F

radius < 0 invalid

valid ?

! (radius < 0)

radius >= 0

read input from user

① valid condition

if (input is valid) {

do comp.

} else {

print some error

}

error condition

if (input is invalid) {

print some error

} else {

do comp.

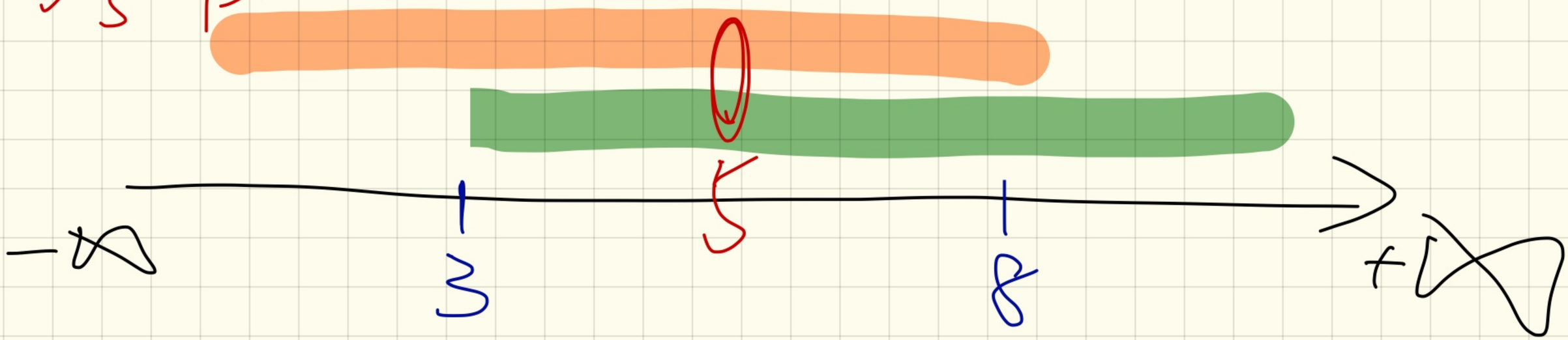
}

1 if-statement

```
int i = 5;
if(i >= 3) {System.out.println("i is >= 3");}
else if(i <= 8) {System.out.println("i is <= 8");}
```

2 if-statements

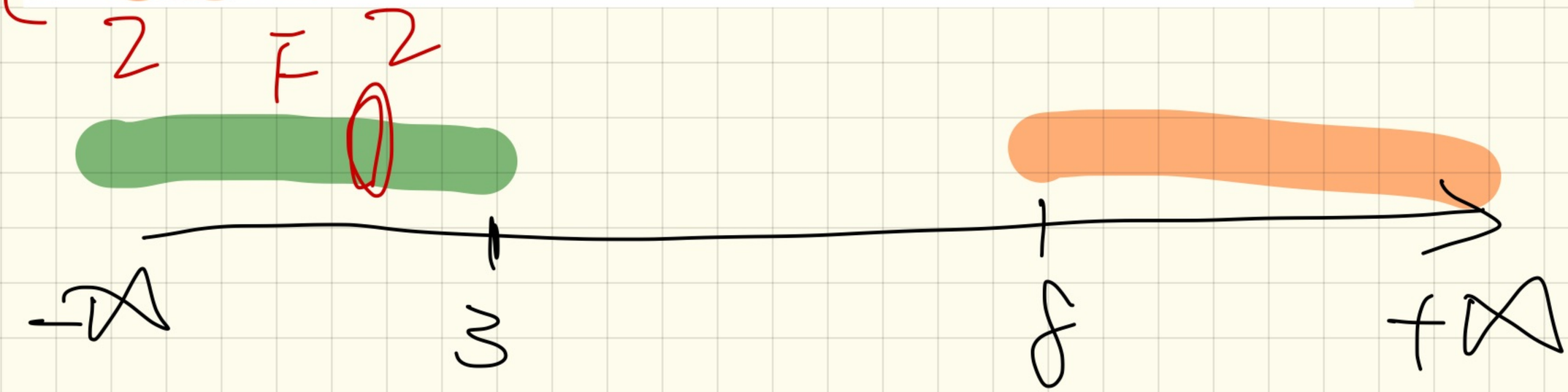
```
int i = 5;
if(i >= 3) {System.out.println("i is >= 3");}
if(i <= 8) {System.out.println("i is <= 8");}
```



① ②

```
int i = 2;
if(i <= 3) {System.out.println("i is <= 3");}
else if(i >= 8) {System.out.println("i is >= 8");}
```

```
int i = 2;
if(i <= 3) {System.out.println("i is <= 3");}
if(i >= 8) {System.out.println("i is >= 8");}
```



Boolean Expressions

- true
  false
  ] Boolean literals

- x <= y
  y > z
  ] relational expressions

-] logical operations.

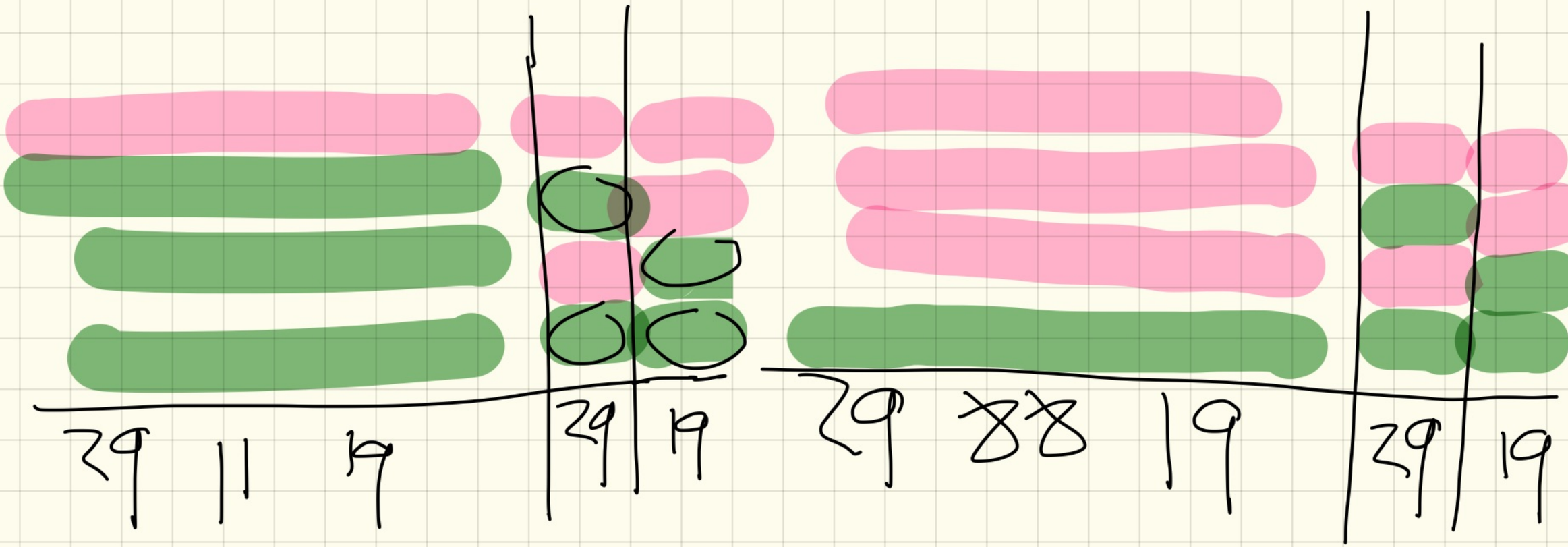
Logical negation (truth table)

unary operator

single operand

b	!b
[illegible]	[illegible]
[illegible]	[illegible]

```
double radius = input.nextDouble();
boolean isPositive = radius > 0;
if ( !isPositive ) {
    error
}
else {
    comp.
}
```


29
11
19
29
19
29
88
19
29
19

boolean oldEnough = age >= 45;

boolean notTooOld = age < 65;

oldEnough && notTooOld

T

T

F

F

50

70

$45 \leq age \leq 65$

Math

$$25 \le \bar{L} \le 35$$

40

40 False

Inconsistent

Java

$$25 <= \bar{L} <= 35$$

T F

① $25 <= \bar{x}$ (40) && $\bar{x} <= 35$ (40)

② $25 <= \bar{x}$ (40) || $\bar{x} <= 25$ (40)

T F

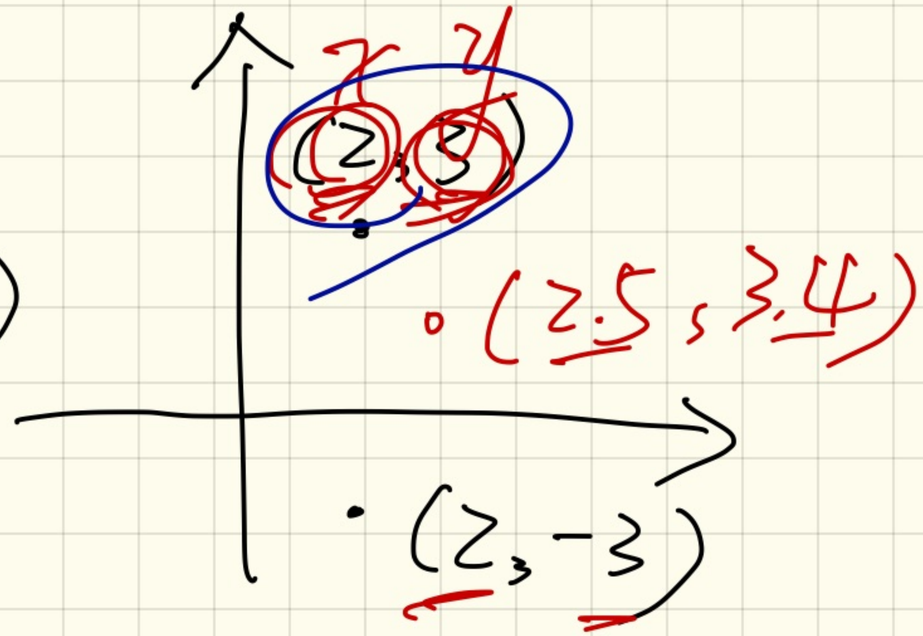
T F

T

class Point {

- attributes (class variables)
- constructors
- accessor methods (questions)
- mutator methods (changes)

}

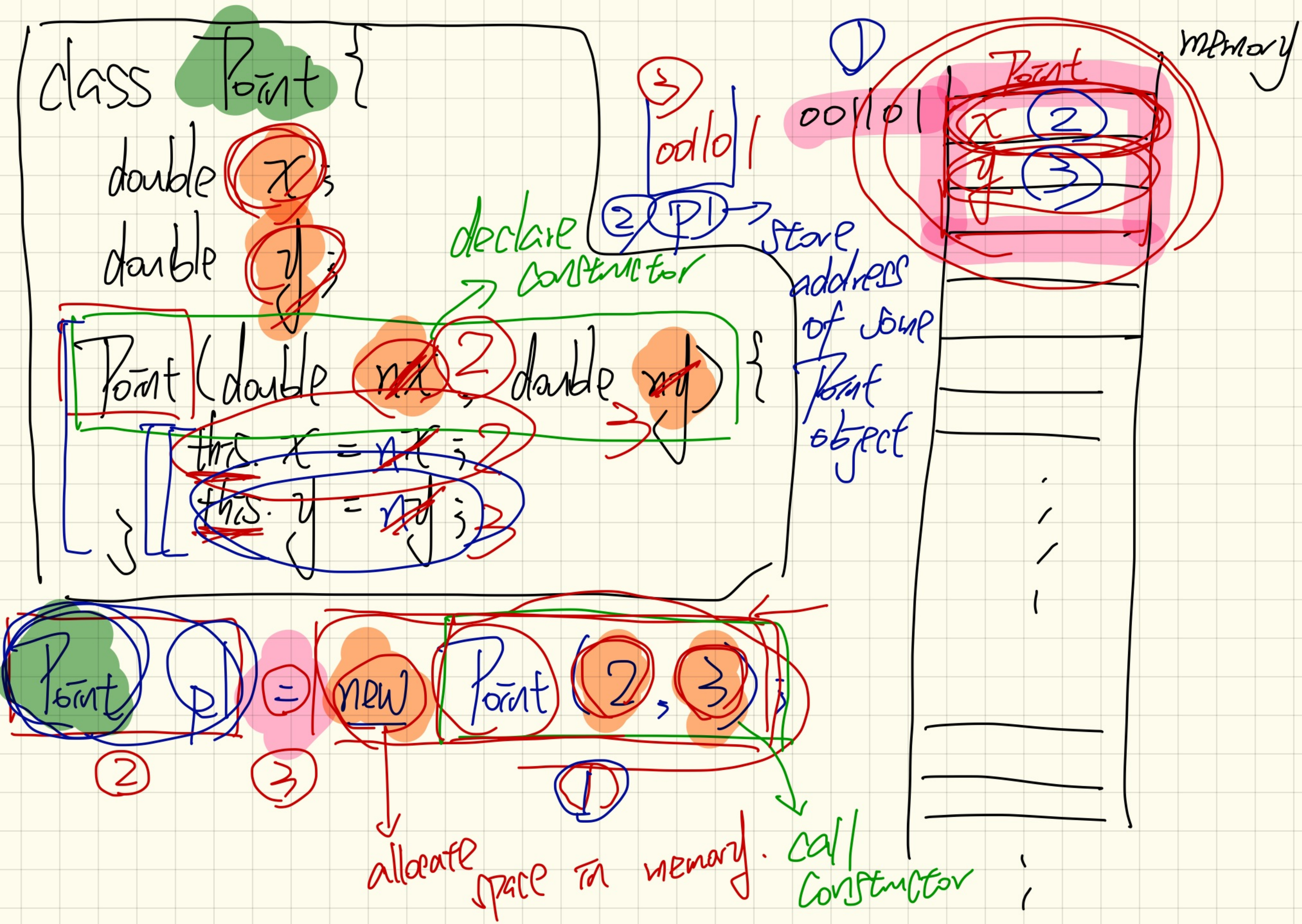


```
class Point {
    double x;
    double y;

    Point(double nx, double ny) {
        this.x = nx;
        this.y = ny;
    }
}

Point p1 = new Point(2, 3);
```

declare constructor

store address of some Point object

allocate space in memory. call constructor



3 4

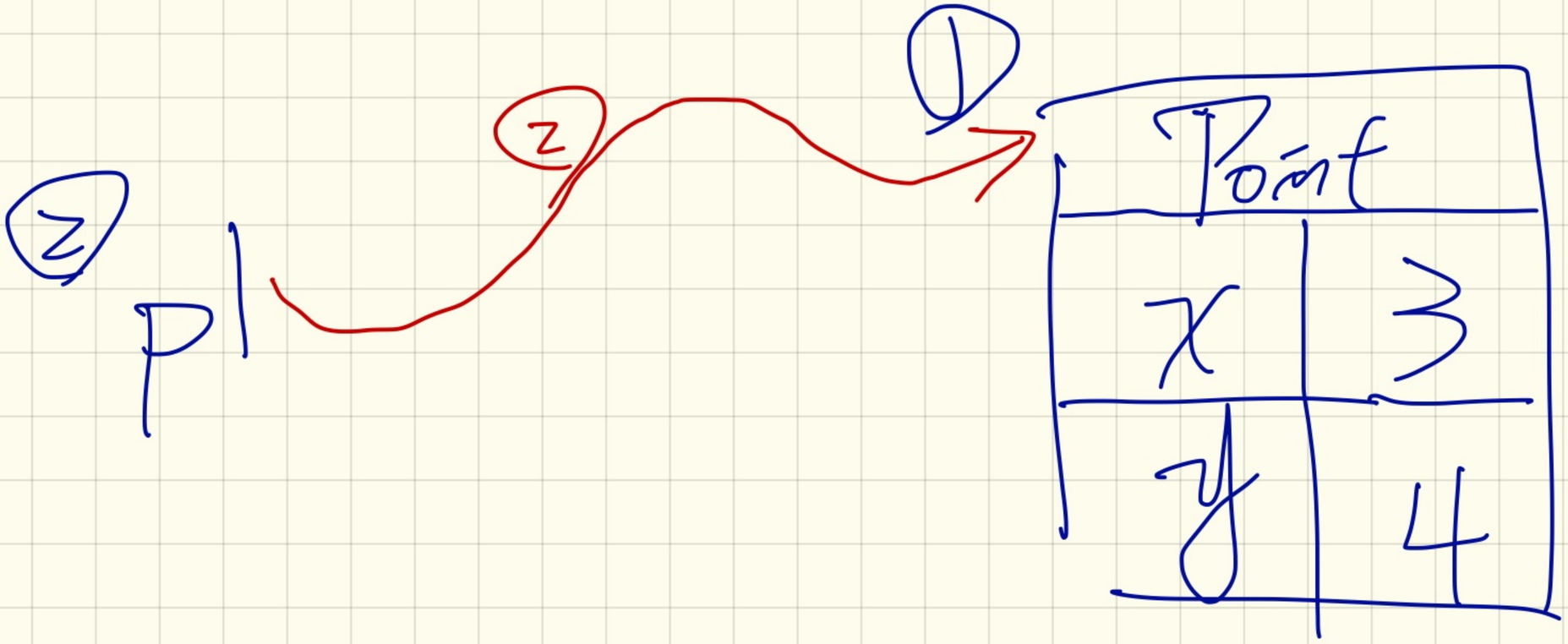
add(x, y) = x + y

declaration

add(3, 4) 7

Point p1 = new Point(3,4);

② ①



```
if (score >= 80.0) {
  System.out.println("A"); }
else { /* score < 80.0 */
  if (score >= 70.0) {
    System.out.println("B"); }
  else { /* score < 70.0 */
    if (score >= 60.0) {
      System.out.println("C"); }
    else { /* score < 60.0 */
      System.out.println("F");
    }
  }
}
```

Flow Chart?

```java
if (score >= 80.0) {
  System.out.println("A");
}
else if (score >= 70.0) {
  System.out.println("B");
}
else if (score >= 60.0) {
  System.out.println("C");
}
else {
  System.out.println("F");
}
```

Flow Chart?

```
String lettGrade = "F";
if (score >= 80.0) {
  letterGrade = "A";
}
else if (score >= 70.0) {
  letterGrade = "B";
}
else if (score >= 60.0) {
  letterGrade = "C";
}
```

Inputs:

score = 85

75

65

55

```
1  int x = input.nextInt();
2  int y = 0;
3  if (x >= 0) {
4    System.out.println("x is positive");
5    if (x > 10) { y = x * 2; }
6    else if (x < 10) { y = x % 2; }
7    else { y = x * x; }
8  }
9  else {  /* x < 0 */
10   System.out.println("x is negative");
11   if(x < -5) { y = -x; }
12 }
```

How many if-statement are in the above program?

Monday Jan. 29

Lecture 4

MVC

View

1234

"1234"

.xml

textView
Spinner

textfield

buttons

breakpoints
debugger

IDE

Tester

Person.java

TP.java

Frame.java

Model

new

Person user =

Person(...)

1.

Controller

ActivityJava

user.toString()

user.getBMI

// class Person{

Person(...){

double getBMI(){

① Retrieve inputs

② Conversion

③ Connecting model

Object Orientation (OO)

Observe

Entities

mutator

change weight (90 kg)

bmi ?

accessor

question triggered

"Jim"

80 kg

180 cm

name

weight

height

attributes

x y

1 1

"Jonathan"

85 kg

175 cm

name

weight

height

x y

3 4

Model

class ... {

[attributes

constructors

accessors

mutators

}

Execute

- allocate space in mem. for new objects

- manipulate objects (call methods)

```
class Point{
    double x;
    double y;
    Point(double nx, double ny){
        x = nx;
        y = ny;
    }
};
```

Scope of variables

```
class Foo {
    int i;

    void m1(int i, int j) {
        int k = i * j;
    }

    void m2() {
        int k = i * j;
    }
}
```

attribute ↓ class-level variable

scope of m1

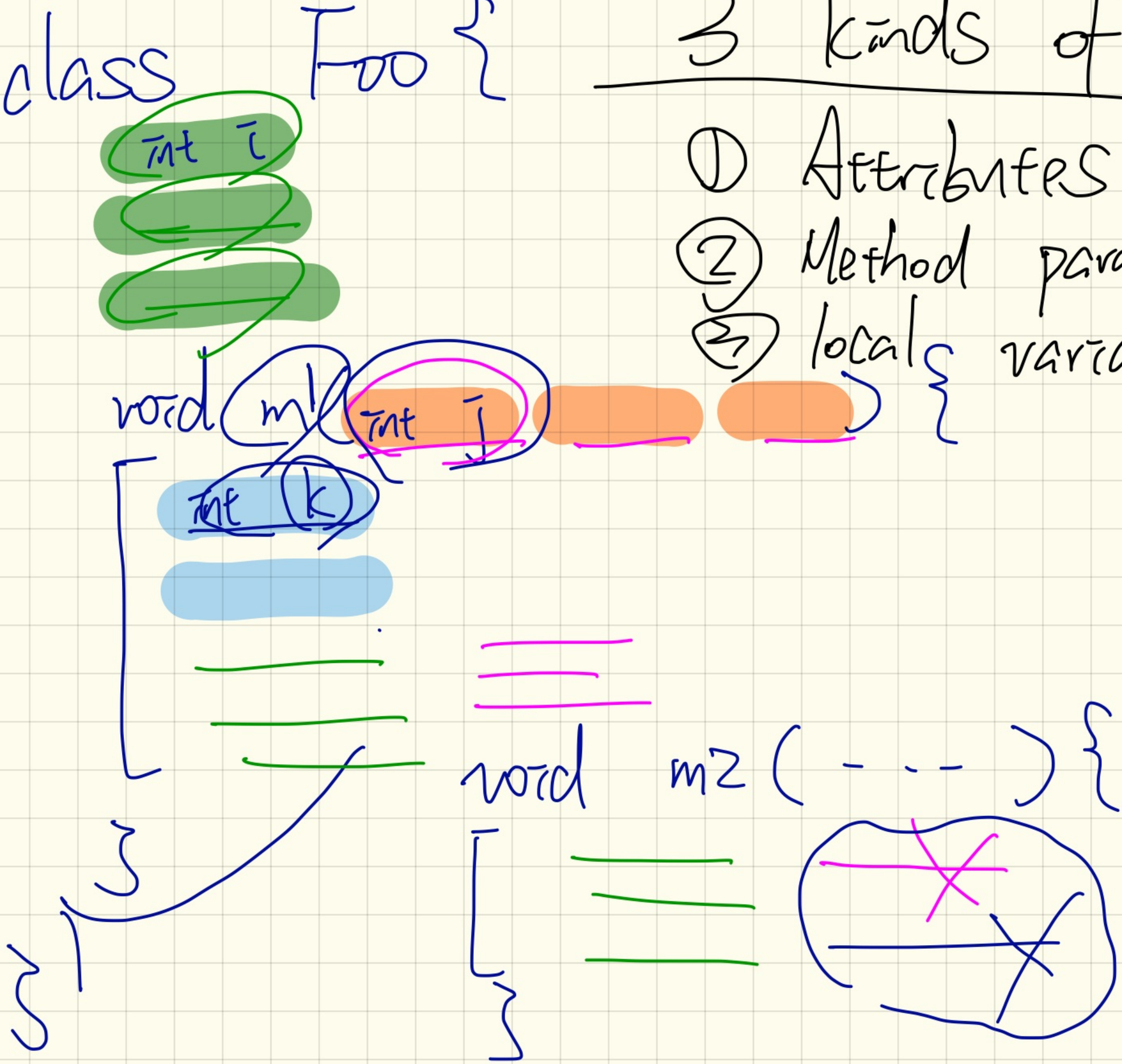
scope of m2

input parameter shadows attribute i.

does not compile.

3 kinds of variables

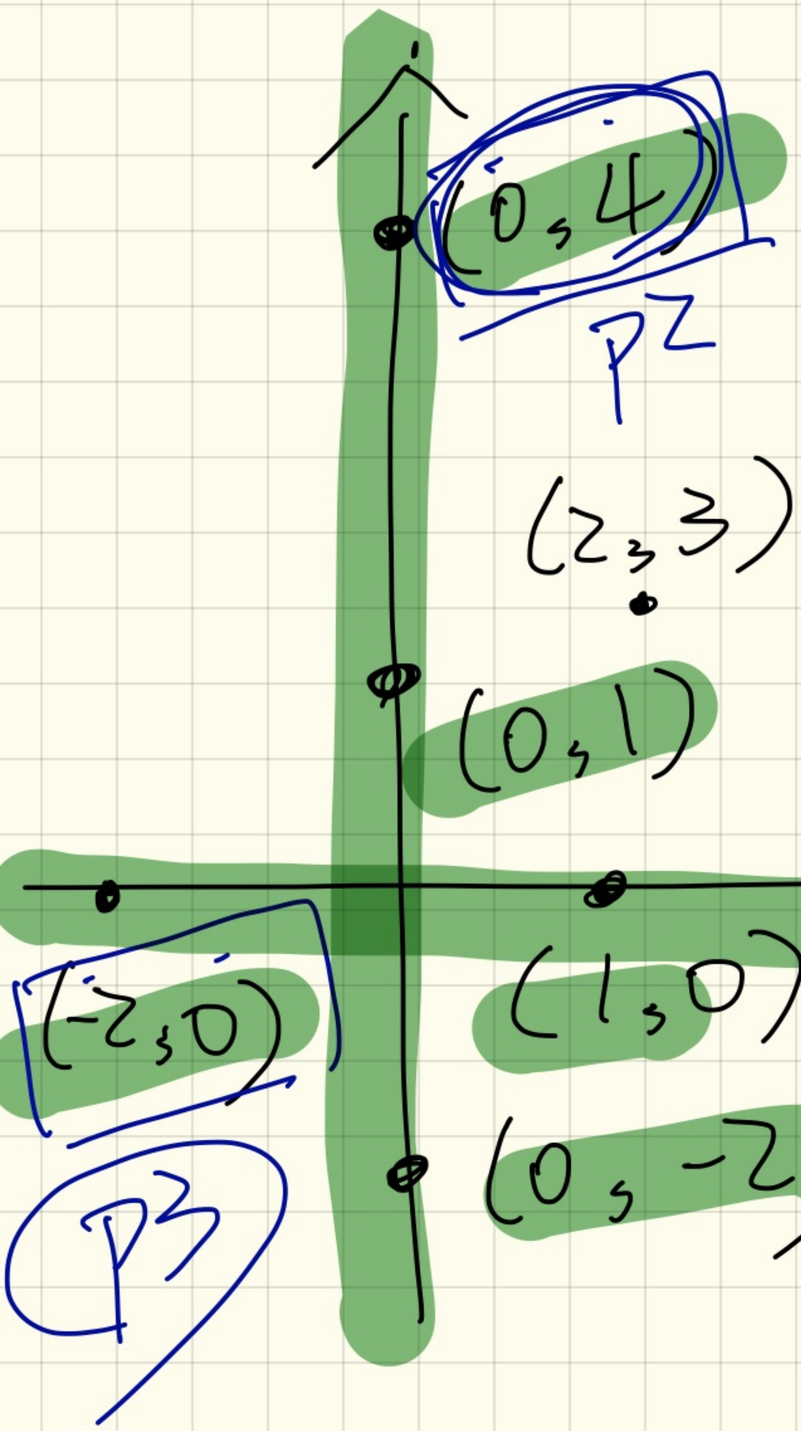
① Attributes (class-level)
② Method parameters
③ local variables.



```
class Foo {
    int a1;
    void m1() {
        int i = 23;
        if (i > 24) {
            int j = i * 2;
        } else {
            int k = i * 4;
            j = 46;
        }
        j = 46;
    }
}
```

visible to the entire "m1"

visible to the entire "if" branch




```
Point p1 = new Point(2, 3);
Point p2 = new Point('y', 4);
Point p3 = new Point('x', -2);
```

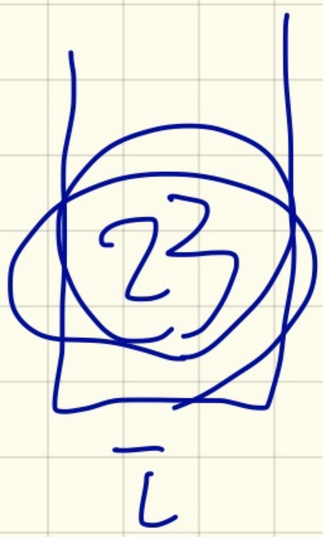
```
class Point {
    Point(char axis, double val) {
        if(axis == 'x') { this.x = val; }
        else if(axis == 'y') { this.y = val; }
    }
}
```

double

int i = 23;

all-lower case

primitive type (value)




```
class Point {
    double x;
    double y;
    Point( --- ) { --- }
}
```

Point p = new Point(2, 3);

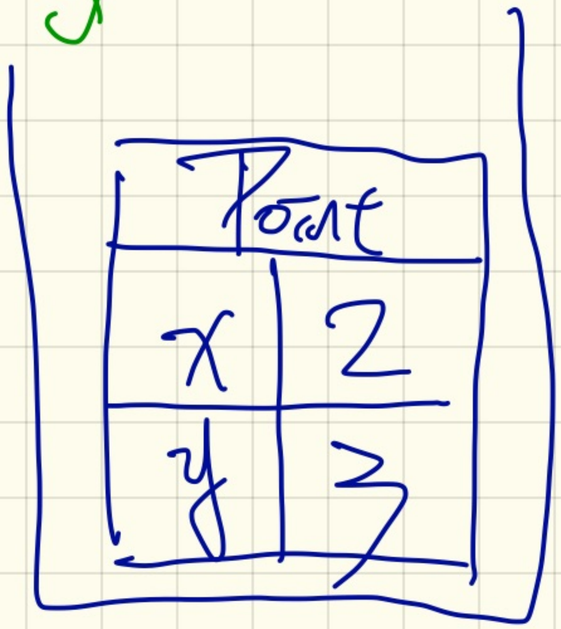
Capitalized

reference type (address) of composite structure

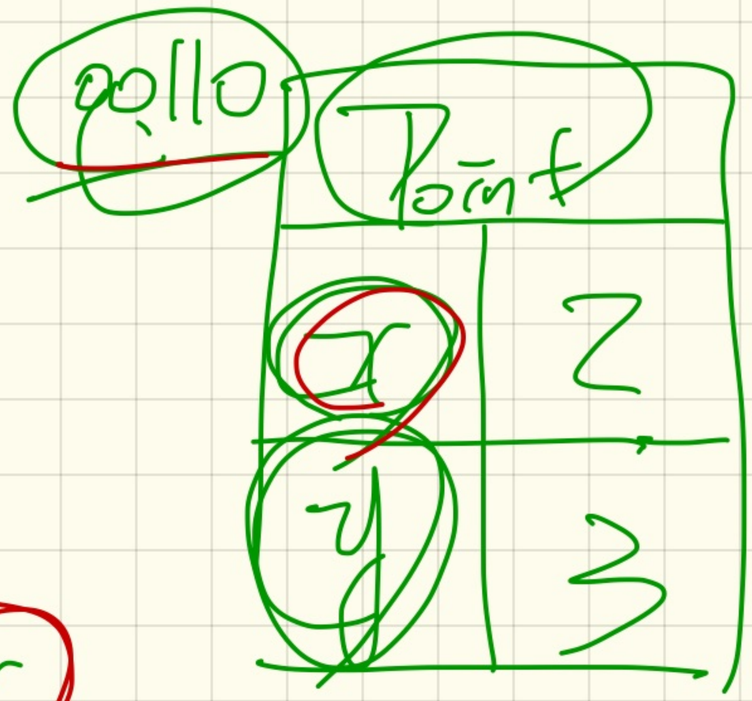
stores address of some Point object.

Point p = new Point(2,3);

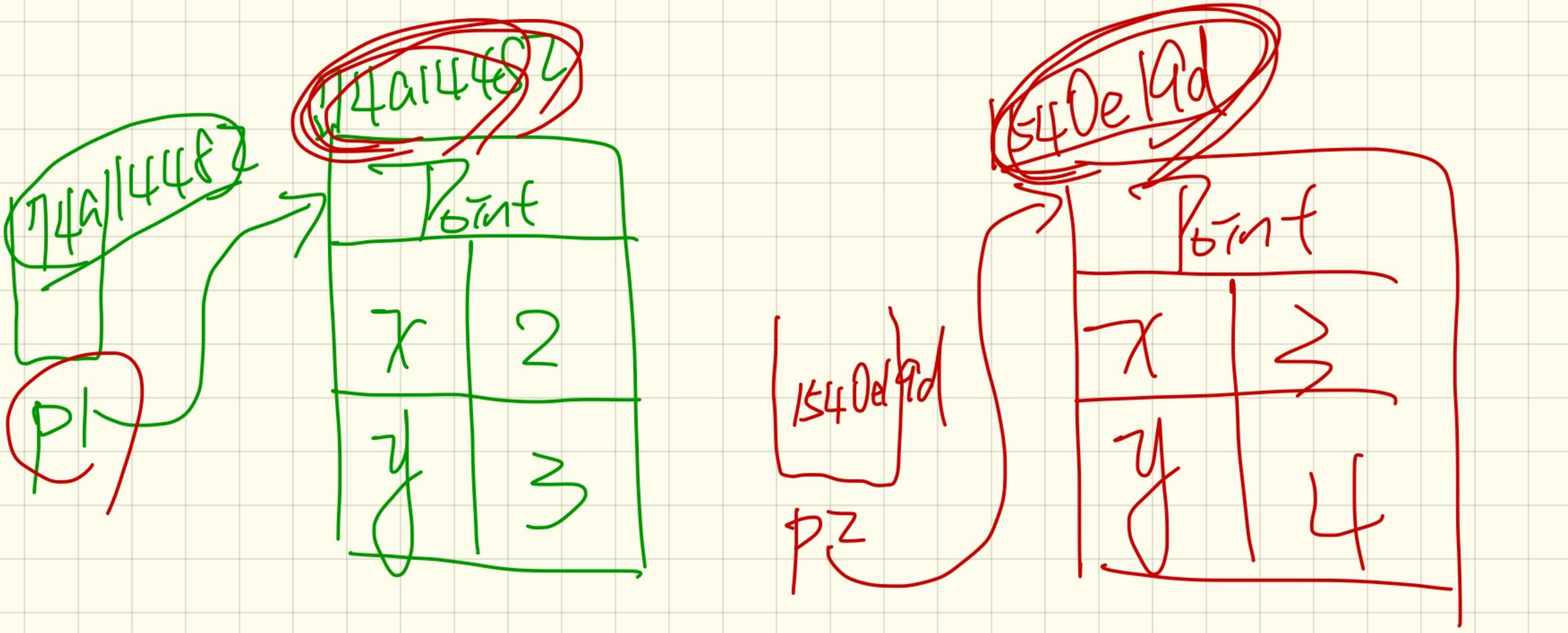
Wrong



Correct address (EECS 2021)



```
Point p1 = new Point(2, 3);
Point p2 = new Point(3, 4);
```



Monday, Feb. 5

Lecture 5

```
class Person {
    int age;
    String nationality;
    double weight;
    double height;

    Person(int a, String n, double w, double h) {
        this.age = a;
        this.nationality = n;
        this.weight = w;
        this.height = h;
    }

    double getBMI() {
        double bmi = this.weight / (this.height * this.height);
        return bmi;
    }

    void gainWeightBy(double units) {
        this.weight = this.weight + units;
    }
}
```

attribute

constructor

accessor (return required)

mutator

return type

change attribute values

boolean pork(char c) {

accesso

```
    boolean result = false;
    if( c == 'p' || c == 'k' ){
        result = true;
    }
    return result;
}
```

X

not Compile

'; ' missing

return

statement

e.g. 'a'

```
boolean pOrk(char c) {
    if(c == 'p' || c == 'k') {
        return true;
    }
    // missing return statement
}
```

'a'

'a'

X not compile

when the if-condition is false.

boolean pOrk(char 'p' c) {

if(c == 'p' || c == 'a' || c == 'a' || c == 'k') {

return true;

}

else {

not compile unreachable statement

return false;

}

boolean result = c == 'p' || c == 'k';

return result;

}

```
class Person {
    int age;
    String nationality;
    double weight;
    double height;

    Person(int a, String n, double w, double h) {
        this.age = a;
        this.nationality = n;
        this.weight = w;
        this.height = h;
    }

    double getBMI() {
        double bmi = this.weight / (this.height * this.height);
        return bmi;
    }

    void gainWeightBy(double units) {
        this.weight = this.weight + units;
    }
}
```

this.age = 50

50 "British"

Jim

context object (this)

Person

a. 50

this.age n. "British"

w.

this.n.

h. 1.8

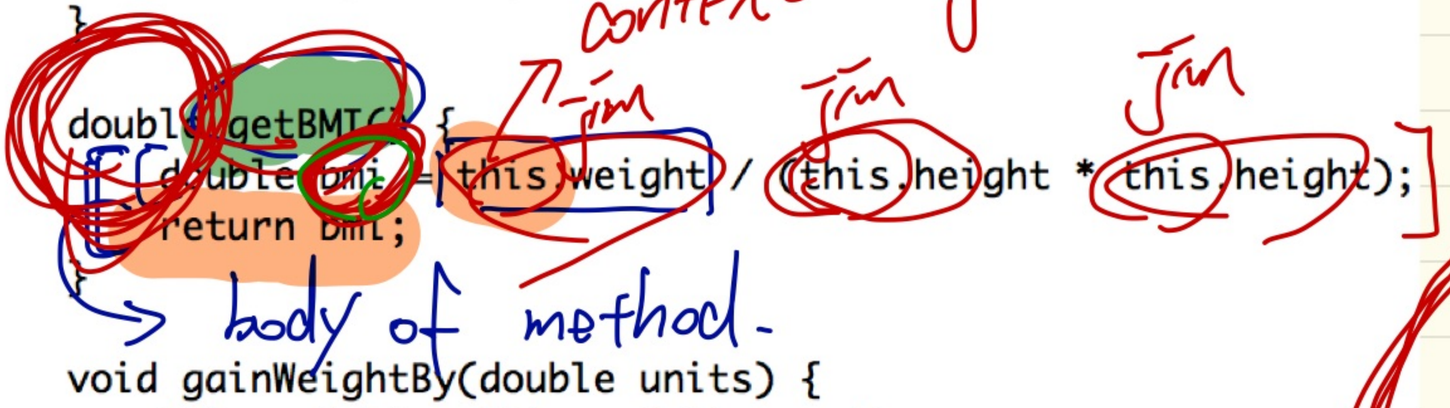
type of Jim is Person

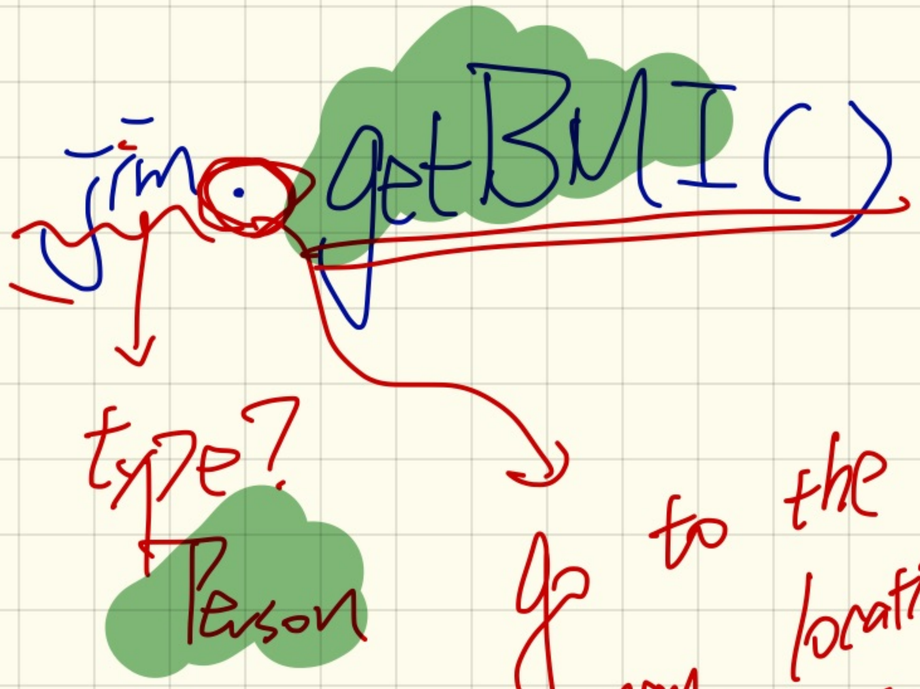
Person Jim = new Person(50, "British", 80, 1.8);

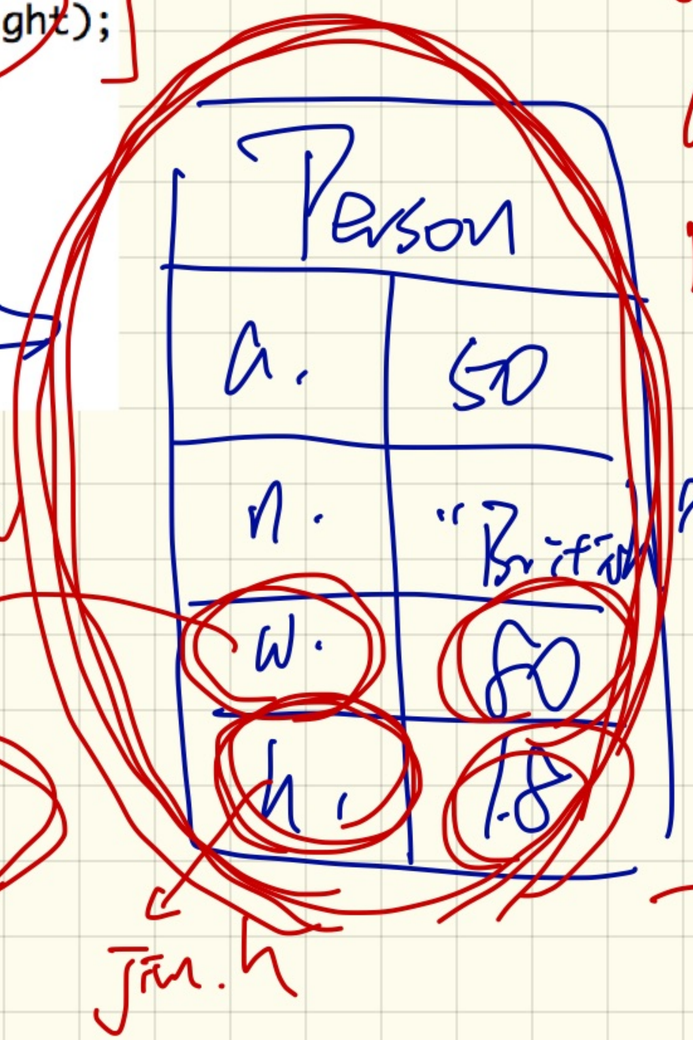
class name

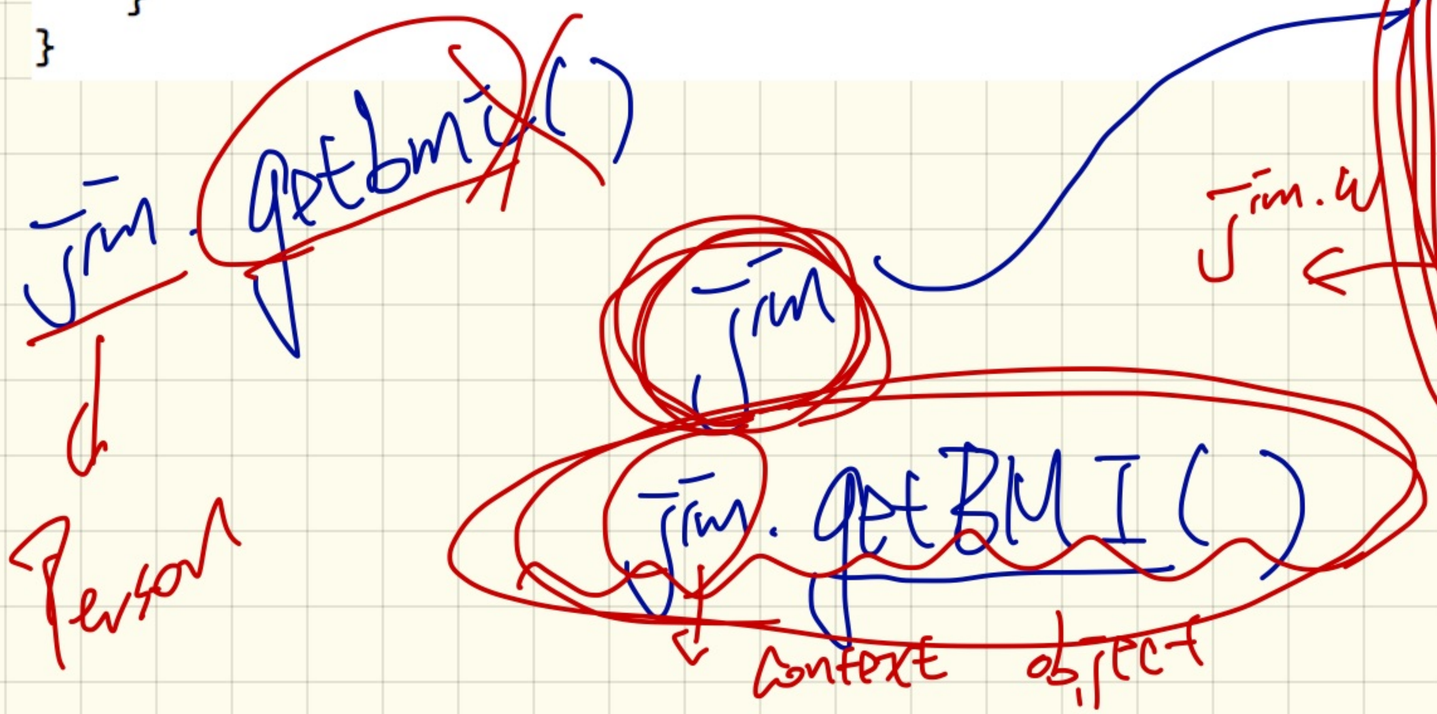
variable → store address of Jim Person object

```
class Person {
    int age;
    String nationality;
    double weight;
    double height;

    Person(int a, String n, double w, double h) {
        this.age = a;
        this.nationality = n;
        this.weight = w;
        this.height = h;
    }

    double getBMI() {
        double bmi = this.weight / (this.height * this.height);
        return bmi;
    }

    void gainWeightBy(double units) {
        this.weight = this.weight + units;
    }
}
```









Store/Catch return value from an accessor method call

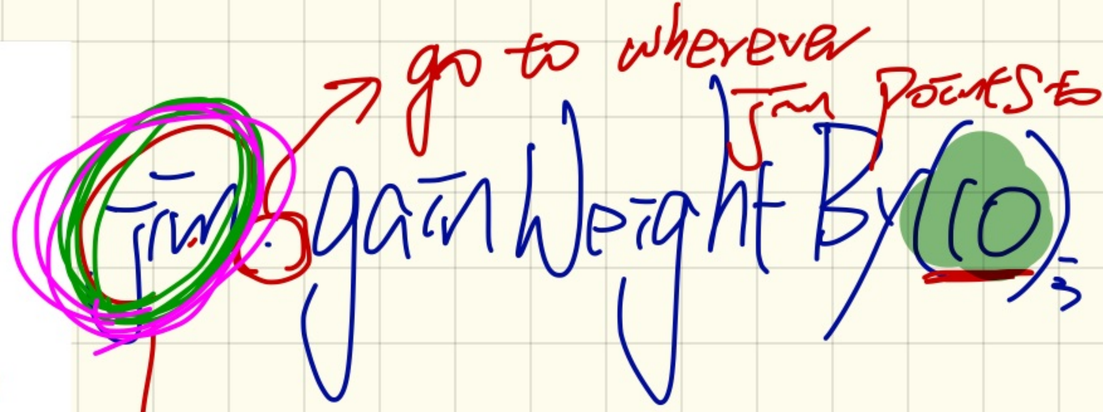
① println(jim.getBMI() 80/1.8*1.8)

jim: context object

getBMI(): execute def. of getBMI according to jim.w and jim.h

② double result = jim.getBMI(); → double value

```java
class Person {
    int age;
    String nationality;
    double weight;
    double height;

    Person(int a, String n, double w, double h) {
        this.age = a;
        this.nationality = n;
        this.weight = w;
        this.height = h;
    }

    double getBMI() {
        double bmi = this.weight / (this.height * this.height);
        return bmi;
    }

    void gainWeightBy(double units) {
        this.weight = this.weight + units;
    }
}
```



accessor method call

println(jim.getBMI());

double result = jim.getBMI();

mutator method call

println(jim.gainWeightBy(10));

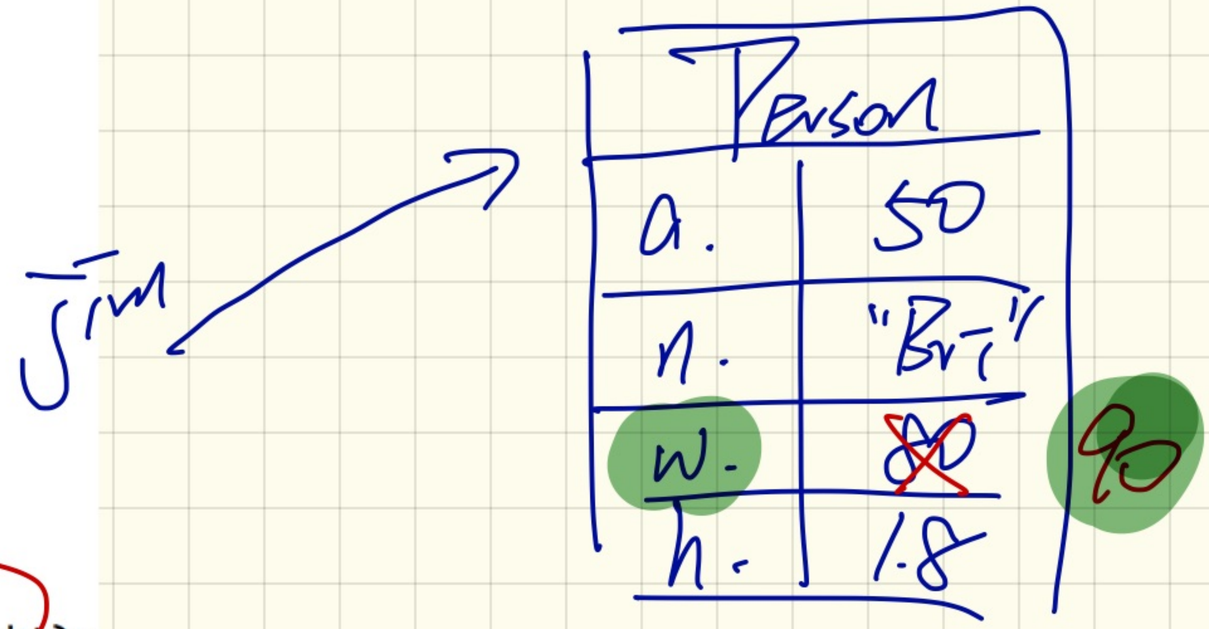
does not compile

void

double result = jim.gainWeightBy(10);

void

```
class Person {
    int age;
    String nationality;
    double weight;
    double height;

    Person(int a, String n, double w, double h) {
        this.age = a;
        this.nationality = n;
        this.weight = w;
        this.height = h;
    }

    double getBMI() {
        double bmi = this.weight / (this.height * this.height);
        return bmi;
    }

    void gainWeightBy(double units) {
        this.weight = this.weight + units;
    }
}
```



① Person jim = new Person(50, "Bri", 80, 1.8);

② println(jim.getBMI());

③ jim.gainWeightBy(10);

④ println(jim.getBMI());

$80/1.8^2$

$90/1.8^2$

jim.getBMI();

jim.w
jim.h * jim.h

jim.gainWeightBy(10);

jim.weight increased by 10

jim.getBMI();

jim.w
jim.h * jim.h

class PersonTester {

main(..){

this.gainWeightBy(10);

PersonTester

}

Person jim = new Person(..);

jim.m();

context object

}

class Person {

void gainWeightBy(..){

}

Person void m(){

this.gainWeightBy(10);

}

}

For Loop

```
for(                ;                ;                ){
    body of loop
}
```

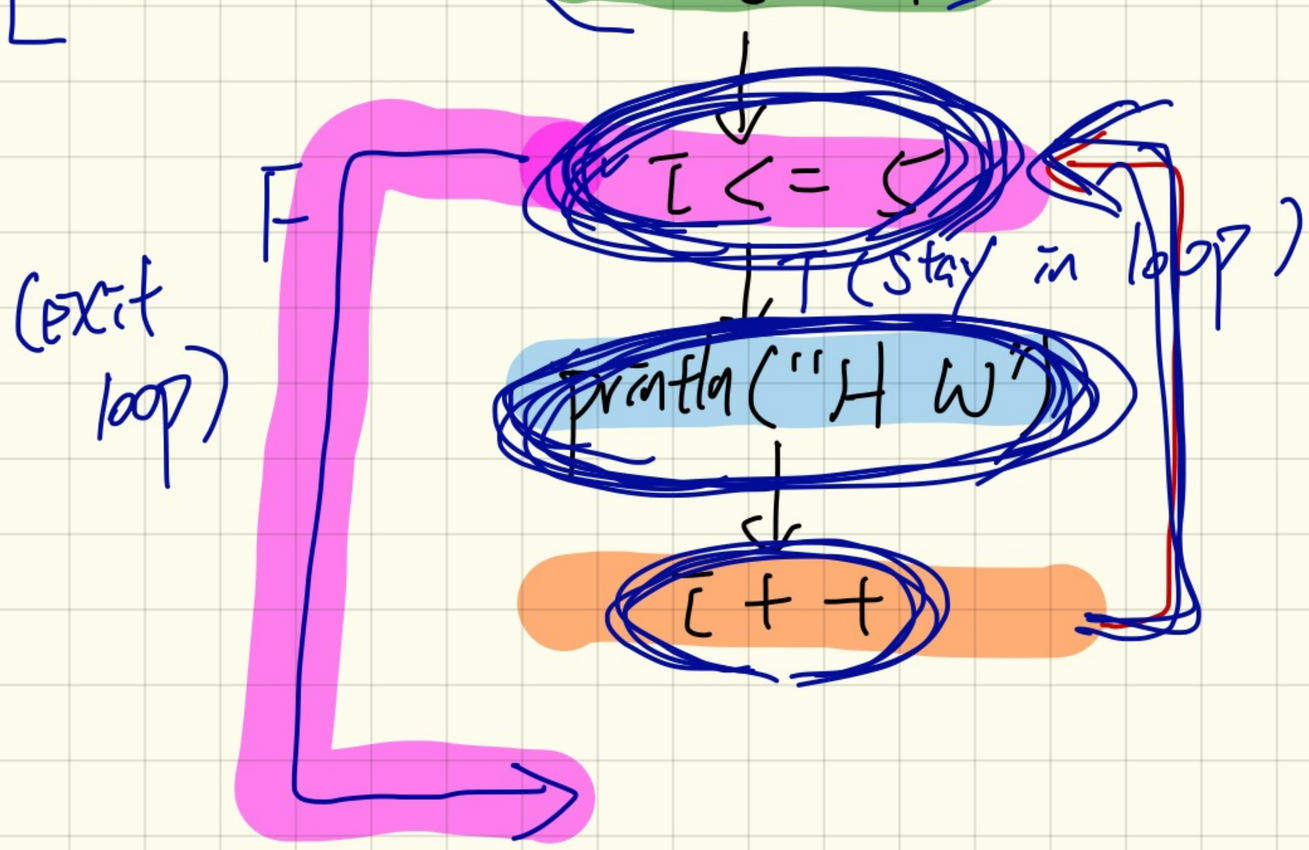
- initialization (once)
- boolean expression
- stay condition (SC)
- as long as SC is true, loop repeating
- update (at the end of each repetition)

```
for (int i = 1 ; i <= 5 ; i++) {
    System.out.println("Hello World");
}
```

loop counter

stay condition

update



i	i<=5	action
1	T	print i++
2	T	print i++
3	T	print i++
4	T	print i++
5	T	print i++
6	F	

infinite loops (terminating)

~~never terminating~~

loop

① for (int i = 1 ; i <= 5 ; i++) {
    println ("H W");
}

② int i = 1;
for (; i <= 5 ; i++) {
    println ("H W");
}

③

```
int i = 1;
for ( ; i <= 5 ; ) {
    println("Hi W");
    i++;
}
```

① ≡ ② ≡ ③

④

```
int i = 1;
for( ; i <= 5; ){
    i++;
    System.out.println("Hello");
}
```

④ ≠ ③

```
for( int i=0; i<100 ; i++){
    System.out.println("Welcome");
}
```

Q1. How many times println() will be executed?

Q2. How many times i<100 will be checked?

Monday Feb. 12

Lecture 6

Utilities {

int avg(int[] is){

}

UT

main(--){

}

```
if (score >= 80.0) {
   System.out.println("A"); }
else { /* score < 80.0 */
   if (score >= 70.0) {
      System.out.println("B"); }
   else { /* score < 70.0 */
      if (score >= 60.0) {
         System.out.println("C"); }
      else { /* score < 60.0 */
         System.out.println("F");
      }
   }
}
```

score

75

Flow Chart?

F

f(C) {

X [

} else { // !C

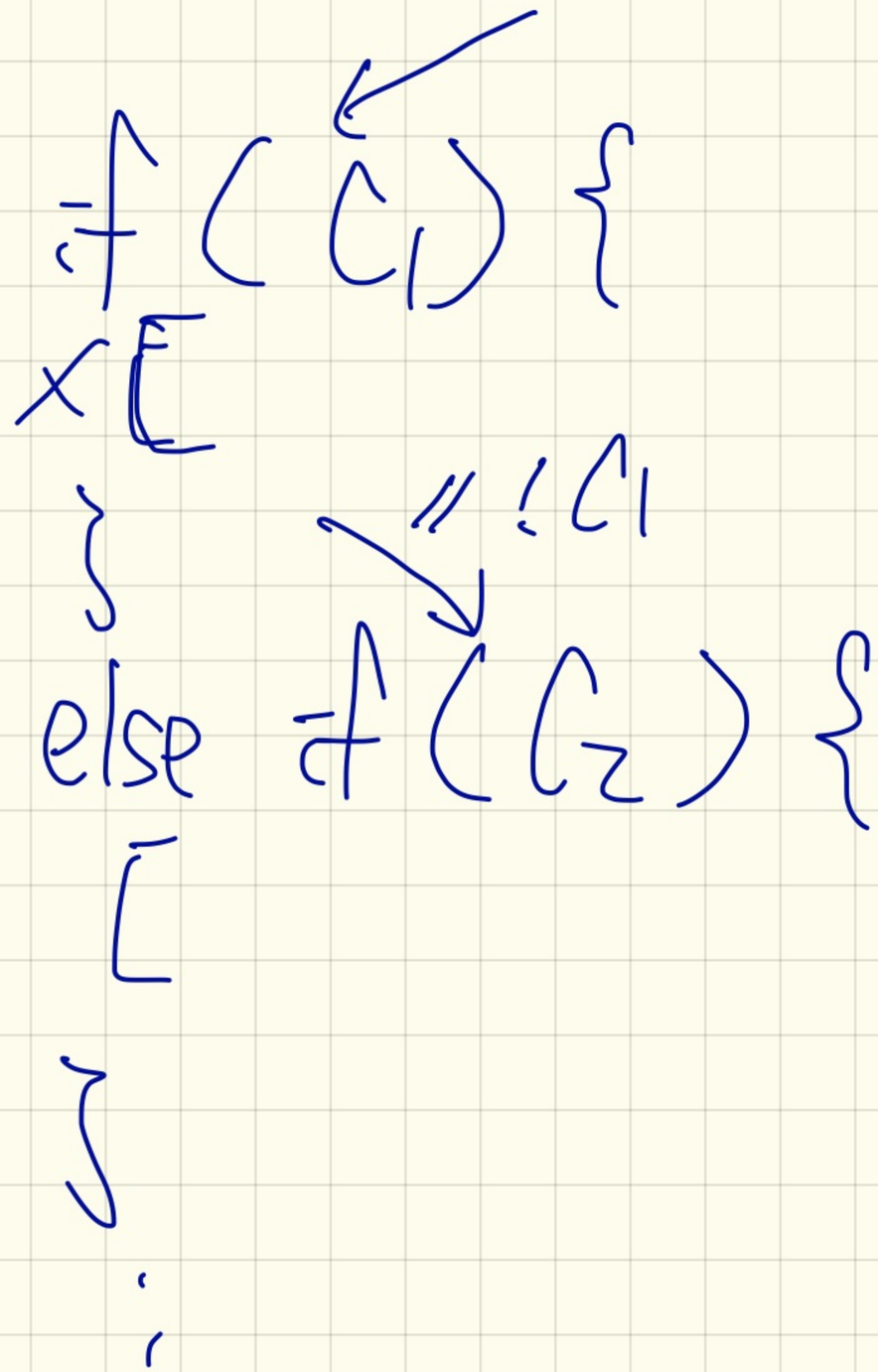
[

}

```java
if (score >= 80.0) {
  System.out.println("A");
}
else if (score >= 70.0) {
  System.out.println("B");
}
else if (score >= 60.0) {
  System.out.println("C");
}
else {
  System.out.println("F");
}
```

Score
75

Flow Chart?



```
String letterGrade = "F";
if (score >= 80.0) {
   letterGrade = "A";
}
else if (score >= 70.0) {
   letterGrade = "B";
}
else if (score >= 60.0) {
   letterGrade = "C";
}
```

missing else

Inputs:

score = 85
75
65
→ 55
45

score

If you have missed the else branch, make sure your variable is initialized.

```
1  int x = input.nextInt();
2  int y = 0;
3  if (x >= 0) {
4    System.out.println("x is positive");
5    if (x > 10) { y = x * 2; }
6    else if (x < 10) { y = x % 2; }
7    else { y = x * x; }
8  }
9  else {  /* x < 0 */
10   System.out.println("x is negative");
11   if(x < -5) { y = -x; }
12 }
```

Compound

!(x >= 0)

111

x < 0

-b

b

-b*-1

How many if-statement are in the above program?

Q. Give a value of x, s.t. L11 is executed.

```
for (int i=0; i<100; i++) {
    System.out.println("HW");
}
```

S.C.

How many times to check S.C.?

[0, 99]

99 - 0 + 1 = 100

i	i<100
0	T
1	T
...	...
99	T
-100	F

```
for (int i=1; i<20|; i+=2) {
    System.out.println("HW");
}
```

S.C.

iterations

How many times to check S.C.?

100 iterations

i	i<20|
1	T
3	T
5	T
...	...
19	T
21	F

$$i \quad j$$

$$1 \text{ —— } 1$$

$$i = 2j - 1$$

$$3 \text{ —— } 2$$

$$5 \text{ —— } 3$$

$$\vdots$$

$$199 \text{ —— } 100$$

100

```
for (int count = 0; count < 100; count++){
    println("Hw" + count);
}
```

100 iterations

Hw 0
Hw 1
Hw 2
⋮
Hw 99

```
for (int count = 1; count <= 100; count++){
    println("Hw" + count);
}
```

Hw 1
Hw 2
Hw 3
⋮
Hw 100

100 iterations

```
int i = 1;
for( ; i <= 5 ; ){
    i++;
    println(i);
}
```

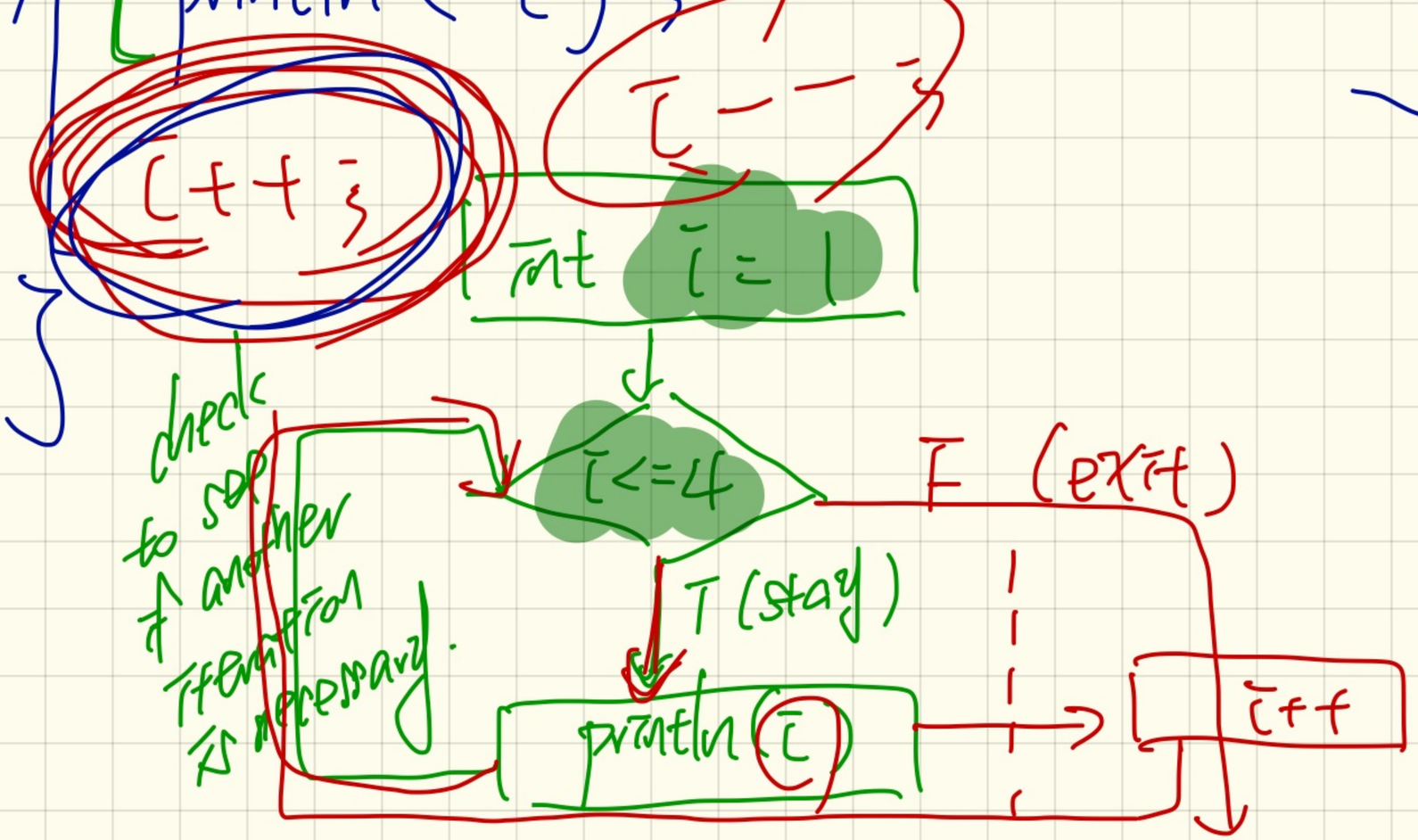
3 →

i	i<=5	p(i)
1	T	2
2	T	3
3	T	4
4	T	5
5	T	6
6	F	

while loop

```
int i = 1;
while ( i <= 4 ) {
    S.C.
    println( i );
    i++;
}
```

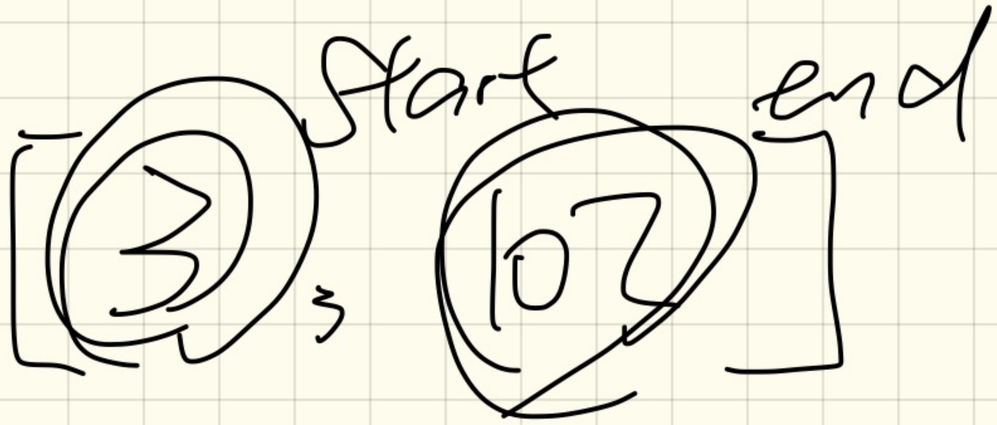
Body



≡

for loop

```
for (int i = 1; i <= 4; i++) {
    println(i);
}
```



3, 4, - - - 102

end − start + 1

for(- — -) {
    while(- -) {
    }
}

for(- - -) {
    if(- -) {
    }
}

if(- -) {
    for(- -) {
    }
}

Input values

-4

5

-2

```
1  System.out.println("Enter a radius value:");
2  double radius = input.nextDouble();
3  boolean isNegative = radius < 0;
4  while (!isNegative) {
5    double area = radius * radius * 3.14;
6    System.out.println("Area is " + area);
7    System.out.println("Enter a radius value:");
8    radius = input.nextDouble();
9    isNegative = radius < 0; }
10 System.out.println("Error: negative radius value.");
```

input values

-4

5

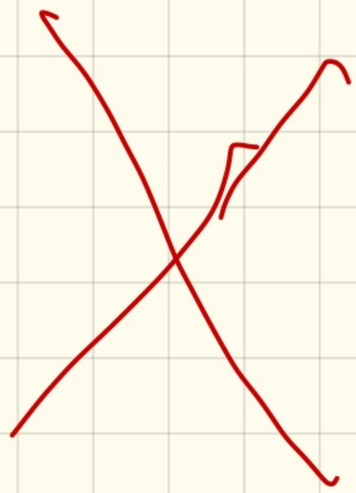
-2

```
1  System.out.println("Enter a radius value:");
2  double radius = input.nextDouble();
3  boolean isPositive = radius >= 0;
4  while (isPositive) {
5    double area = radius * radius * 3.14;
6    System.out.println("Area is " + area);
7    System.out.println("Enter a radius value:");
8    radius = input.nextDouble();
9    isPositive = radius >= 0; }
10 System.out.println("Error: negative radius value.");
```

as long as radius is positive, keep going.

not negative

```
int c = 1;
for (; c <= 10;) {
    printf("%d", c);
    c++;
}
```

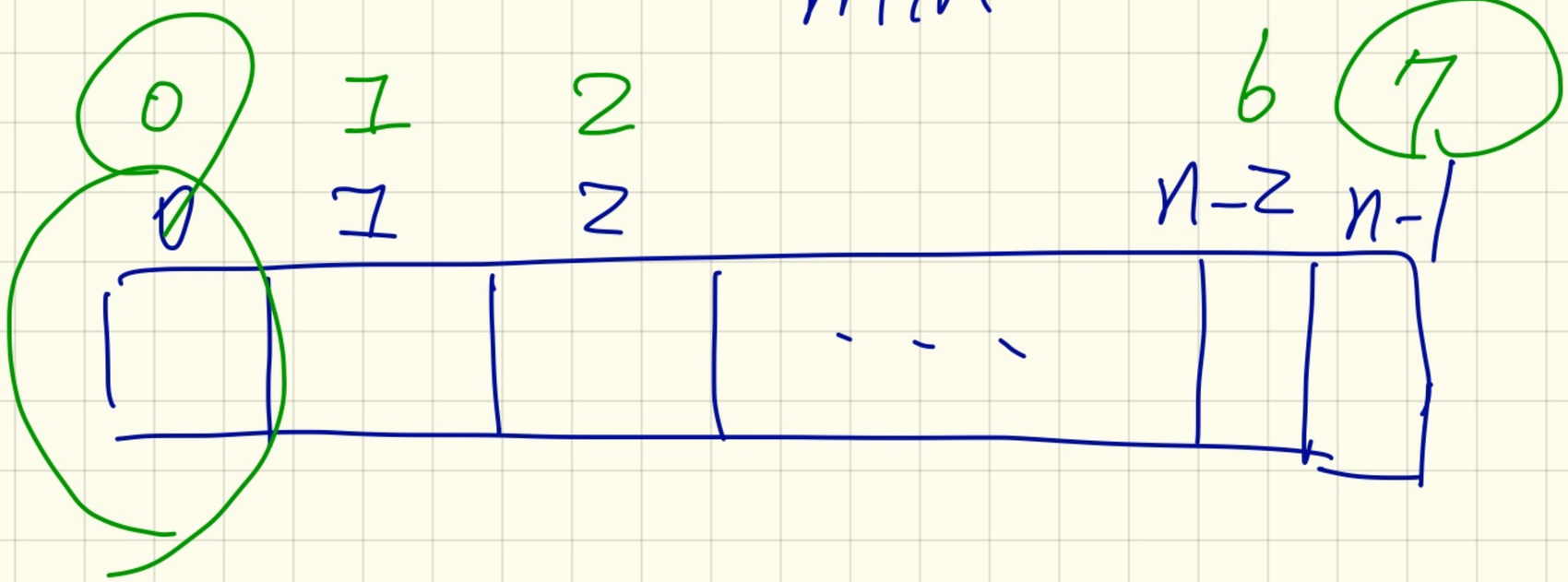


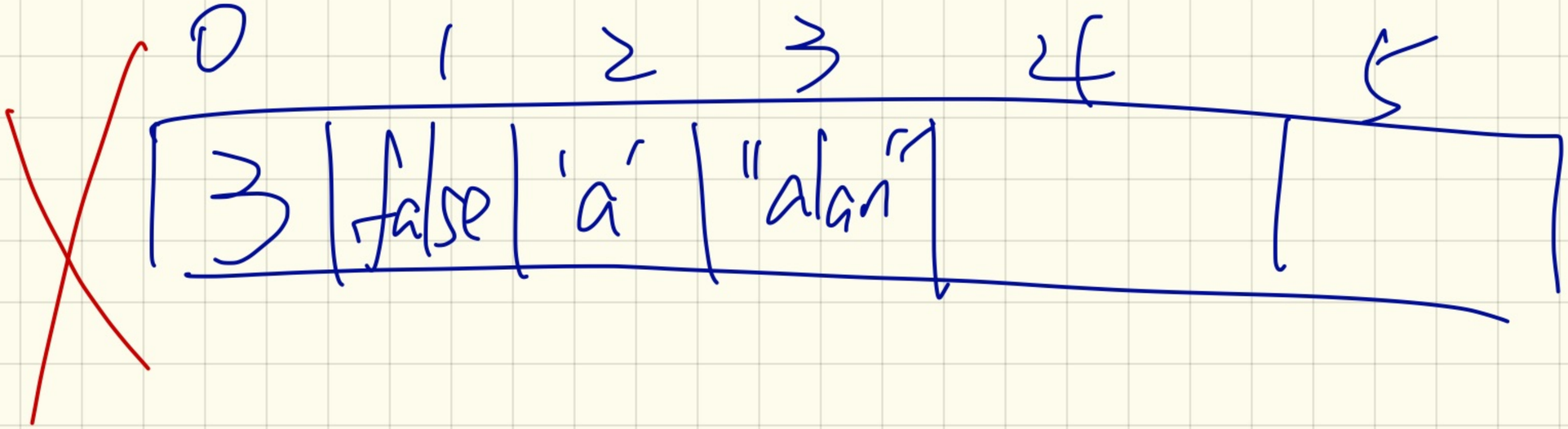
```
while (int i = 1; i < 10; i++){
    printf(i);
}
```

Given an array of size n (8)

what's the max index?

min




0
1
2
3
4
5
3
false
'a'
"alan"

Syntax of array

Declare

int[] ia;

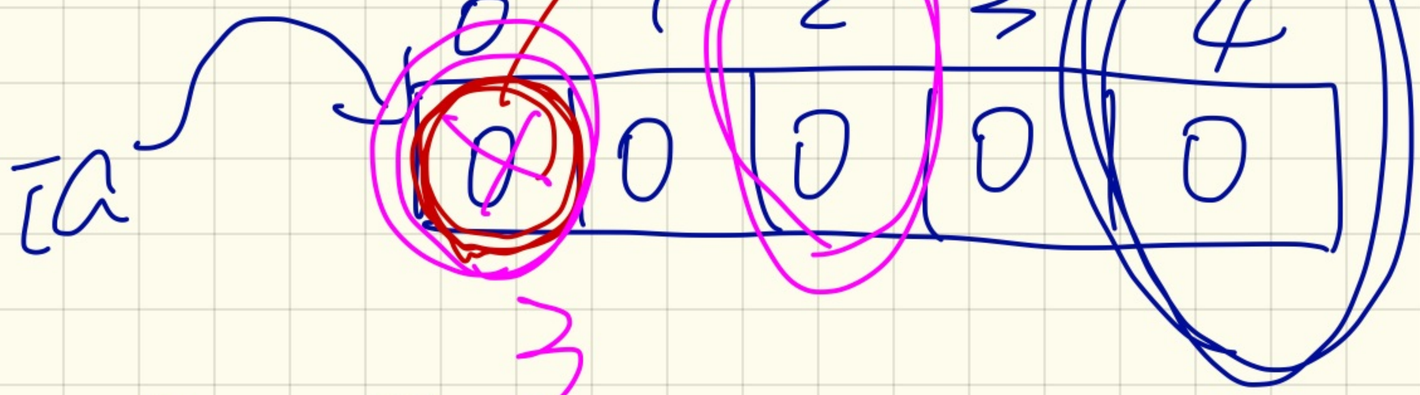
Assign

① ia = new int[5];

size

ia.length 5

ia[0]



access

ia[0] ~ ia[ia.length - 1]

modify

ia[0] = 3

Monday Feb. 26

Lecture 7

- Lab 5
  - New tutorial series
  - Tutorial document
- Practice lab test 2 solution

negation (!)

b	!b

At runtime:

Left to right

1. b1 && b2

b1 → false

b2 → evaluation by-passed

2. b1 || b2

b1 → True by passed.

Conjunction (&&)

b1	b2	(b1) && (b2)

disjunction (||)

b1	b2	b1 || b2

Short-Circuit

$p \wedge true \equiv p$
$p \vee false \equiv p$
$p \wedge false = false$
$p \vee true = true$

guarding

① b1 && b2

① false

② true

✓ b1 || b2

① true

② false

bypassed

evaluate b2

bypassed

evaluate b2

int x = 0;

int y = 10;

guard condition

① boolean b1 = x != 0 && (y / x > 2)

false

false

② boolean b2 = (y / x > 2) && x != 0

crash

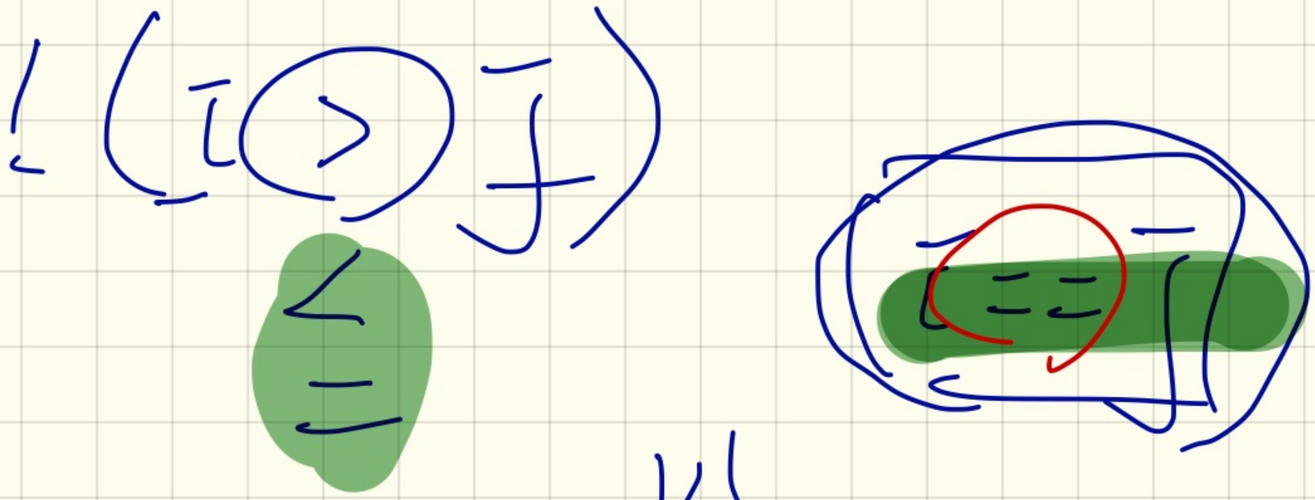
---

③ boolean b3 = x == 0 || (y / x > 2)

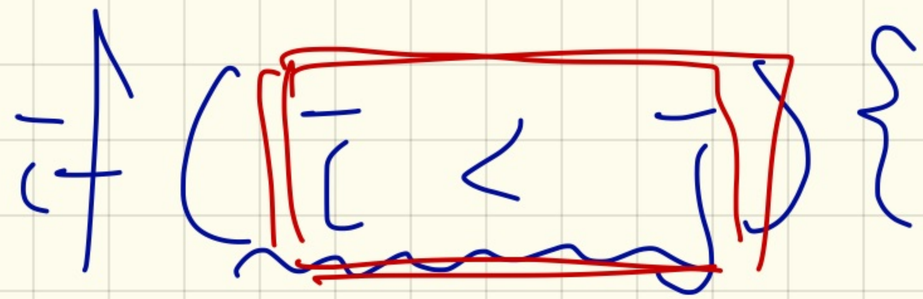
true

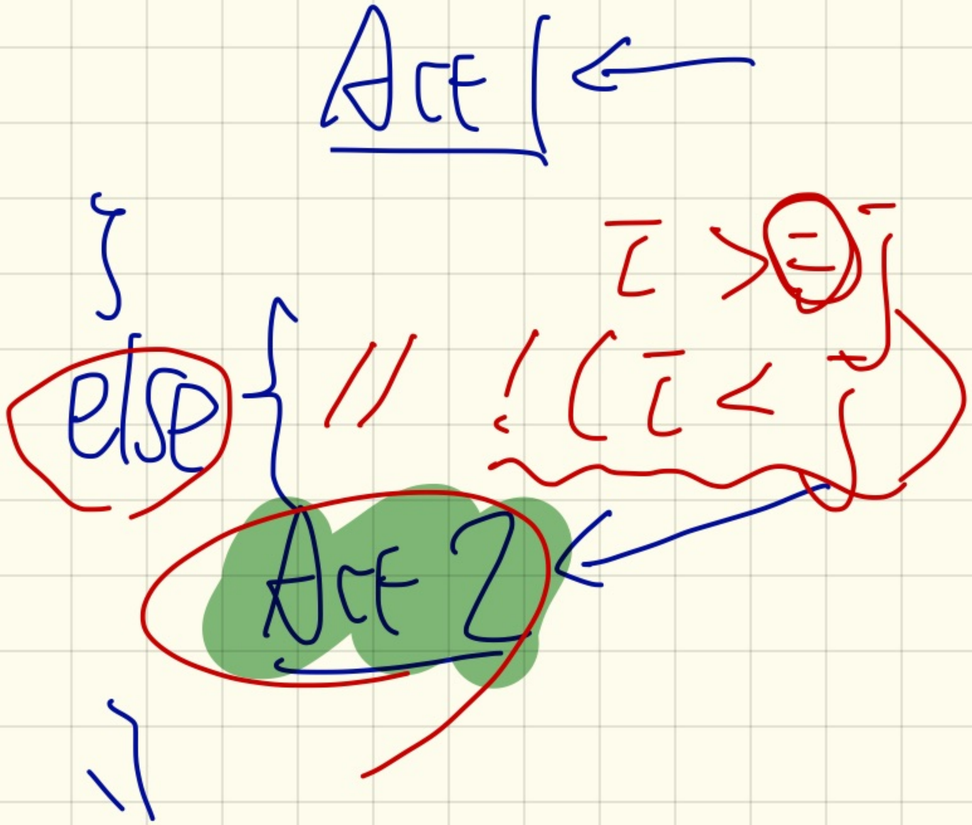
④ boolean b4 = (y / x > 2) || x == 0

crash.

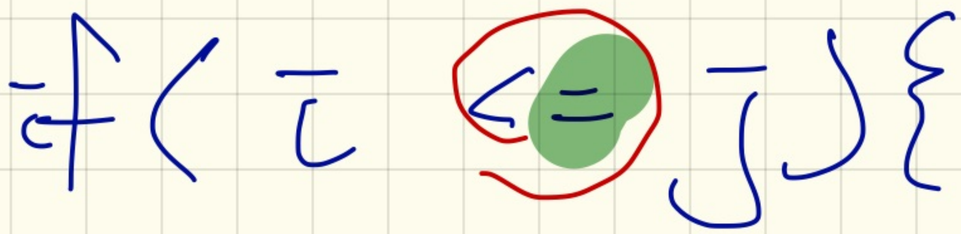


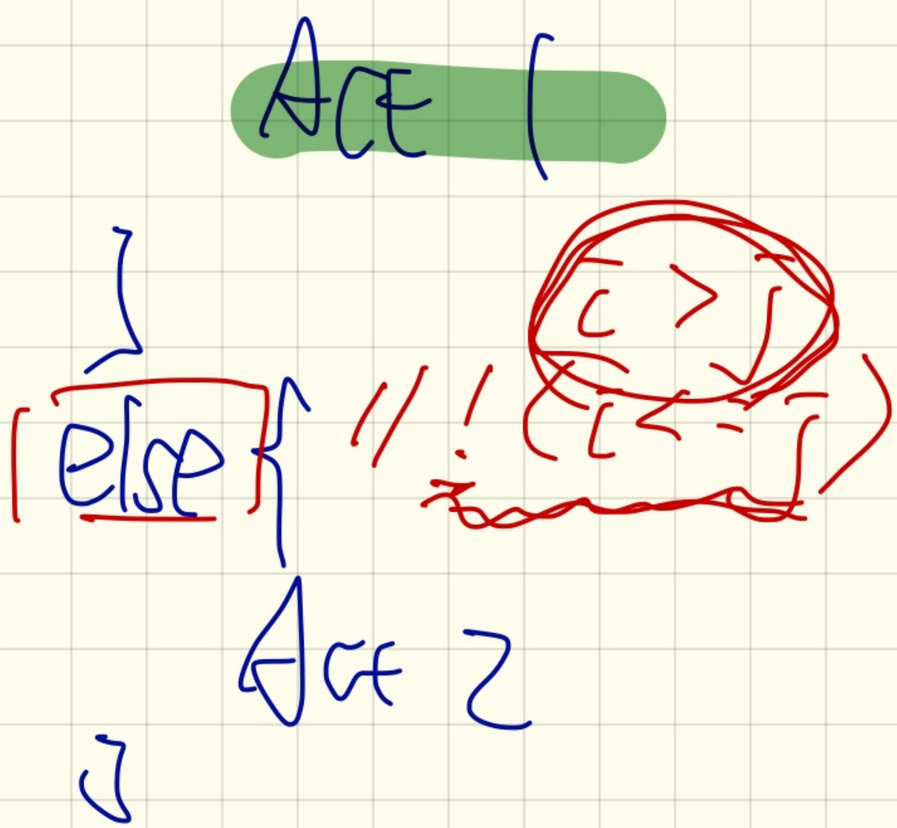
v1.





v2.





de morgan's

$!(b1 \;\&\&\; b2) \equiv !b1 \;||\; !b2$

$!(b1 \;||\; b2) \equiv !b1 \;\&\&\; !b2$

b1 true

or

b2 true

if (0 <= i && i <= 10) {

F

C1

act 1.

} else {

// ? ? ?

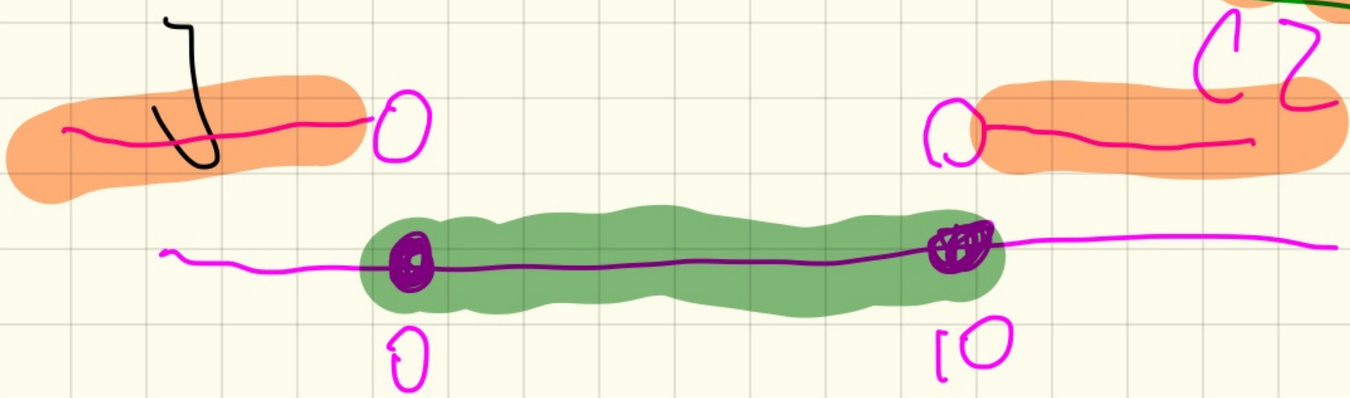
act 2;

!(0 <= i && i <= 10)

!(0 <= i) || !(i <= 10)

0 > i || i > 10

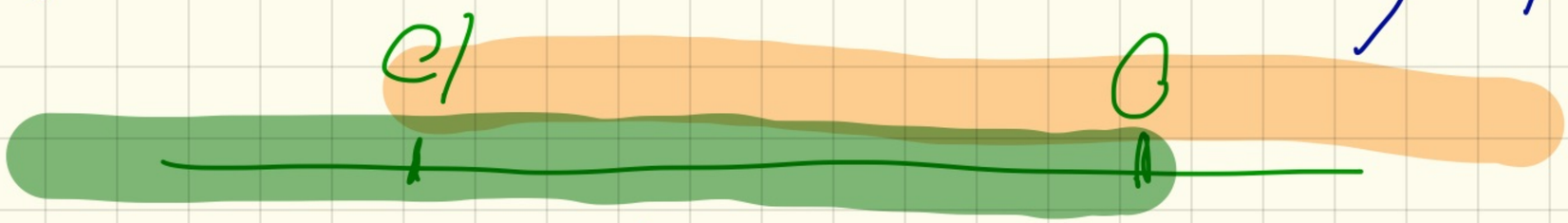
C2



$a1 \geq 1 \quad || \quad a < 1$

$(a1 < 1)$; $||$ $(a > 1)$;

$(a1 < 1 \quad \&\& \quad a > 1)$; : else



$(a1 < 1 \quad \&\& \quad a > 1)$;

false.

||

possible not p

a1 > 1 && a1 =< 1 X

(a1 =< 1 || a1 > 1) ! : asa

(a1 =< 1 || a1 > 1) ?

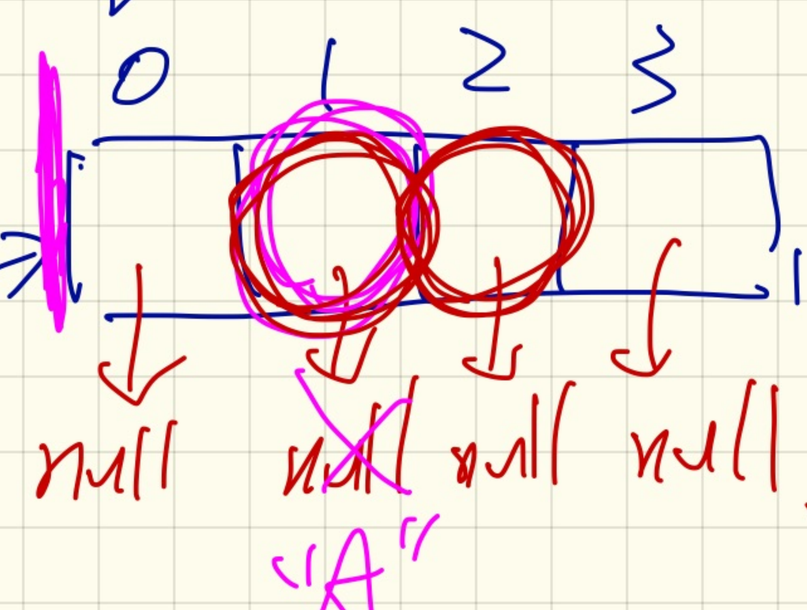
1. NullPointerException.

Primitive Type
int, double, float, char, boolean

① sa[1].equals("A")

② sa[2].equals("A")

null



2. IndexOutOfBoundsException

String[] sa = new String[4];

reference type

default value: null

sa[-1]

sa[4]

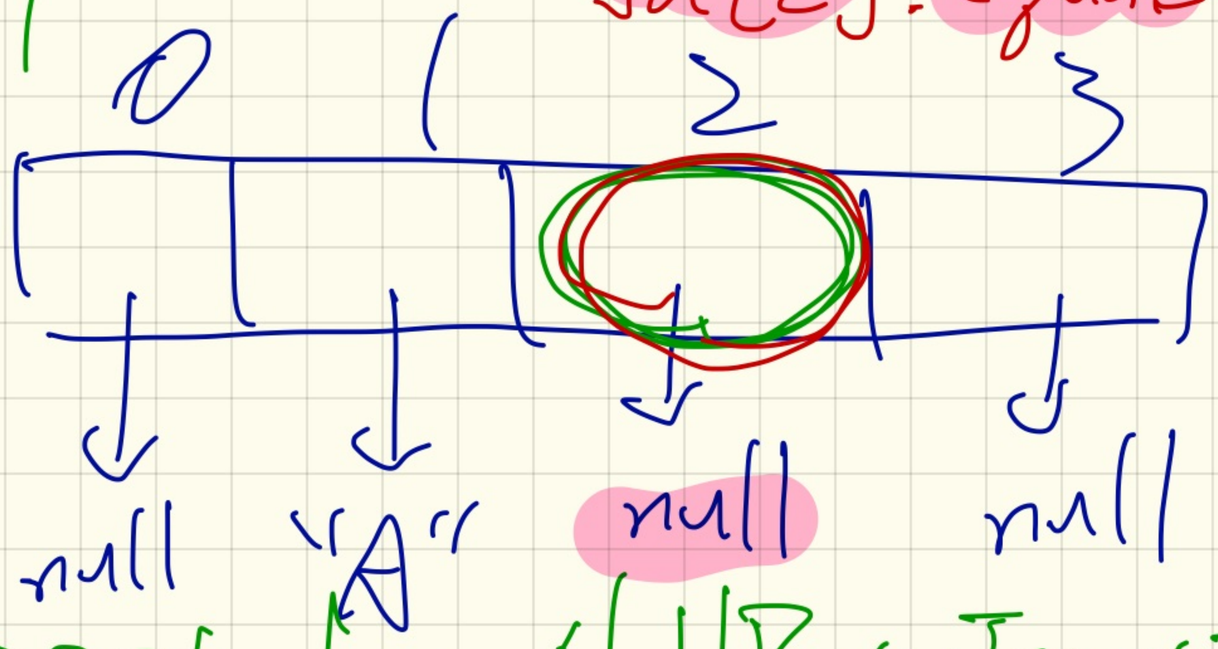
sa[1] = "A";

sa[sa.length-1]

sa[2] == null

True

sa



sa[2].equals("A")

Which one(s) prevent from NullPointerException?

① sa[2] != null && sa[2].equals("A")
null

② sa[2] == null && sa[2].equals("A")
crashing

③ sa[2] == null || sa[2].equals("A")

④ sa[2] != null || sa[2].equals("A")
crashing

```
int sum = 0;
for(int i = 0; i < numbers.length; i ++) {
  sum += numbers[i];
}
double average = (double) sum / numbers.length;
System.out.println("Average is " + average);
```

double

double average = (double) sum / numbers.length;

78.0

average = (double) (sum / numbers.length);

(double)

(double)

5

① 15

or

② 15.6

```
System.out.print("Names:")
for(int i = 0; i < names.length; i ++) {
  System.out.print(names[i]);
  if (i < names.length - 1) {
    System.out.print(", ");
  }
}
System.out.println(".");
```

println(" ")

Alan Mark Tom .

names.length size

names.length - 1 last valid index

i < names.length - 1

↳ i is not the last index

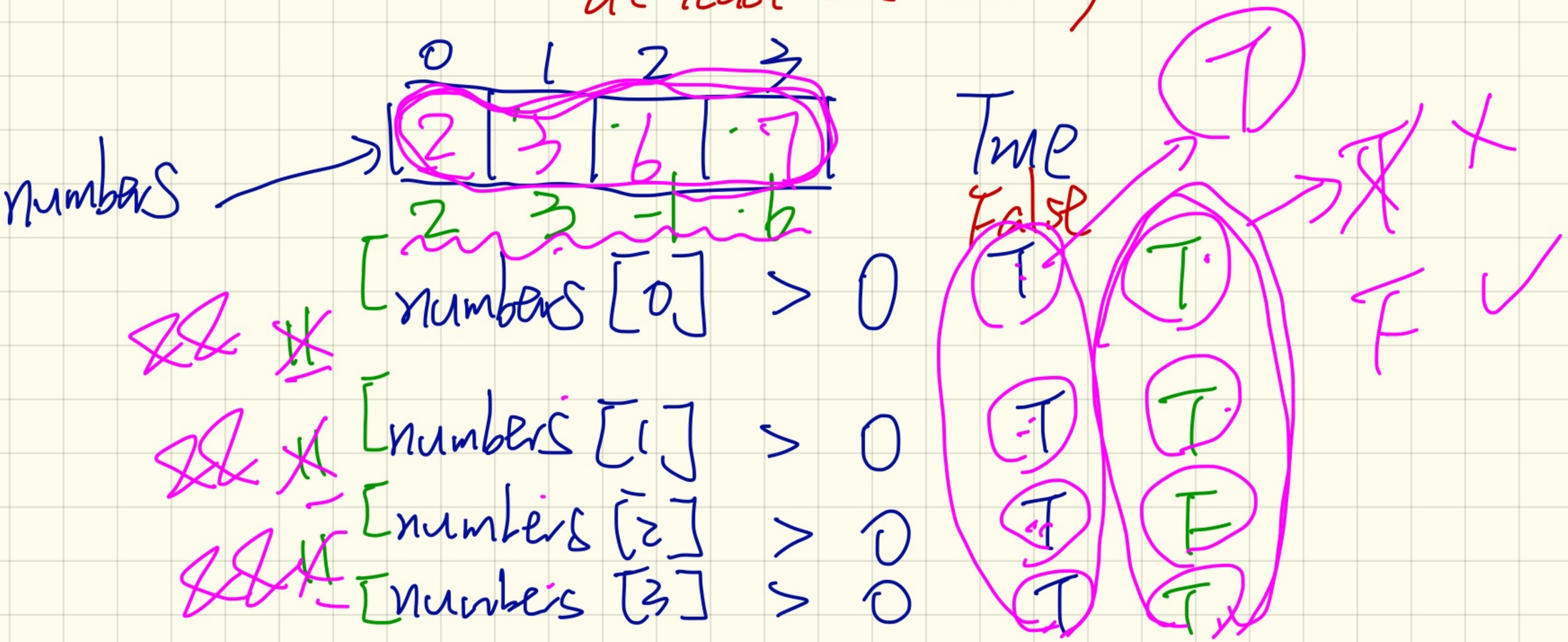
Determine if all numbers are positive.

```
1  int[] numbers = {2, 3, -1, 4, 5, 6, 8, 9, 100};
2  boolean soFarOnlyPosNums = true;
3  int i = 0;
4  while (i < numbers.length) {
5    soFarOnlyPosNums = soFarOnlyPosNums && (numbers[i] > 0);
6    i = i + 1;
7  }
8  if (soFarOnlyPosNums) { /* print a msg. */ }
9  else { /* print another msg. */ }
```

false?

all positive

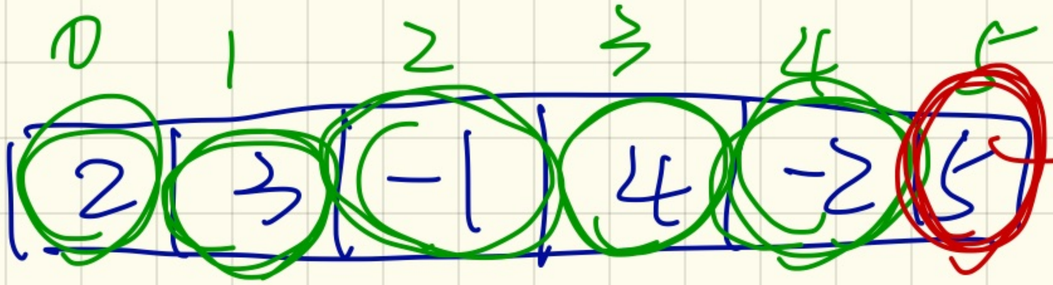
at least one not positive.



Version 2: Wrong.

```
1  int[] ns = {2, 3, -1, 4, 5};
2  boolean soFarOnlyPosNums = true;
3  int i = 0;
4  while (i < ns.length) {
5    soFarOnlyPosNums = ns[i] > 0; /* wrong */
6    i = i + 1;
7  }
```

no accumulation

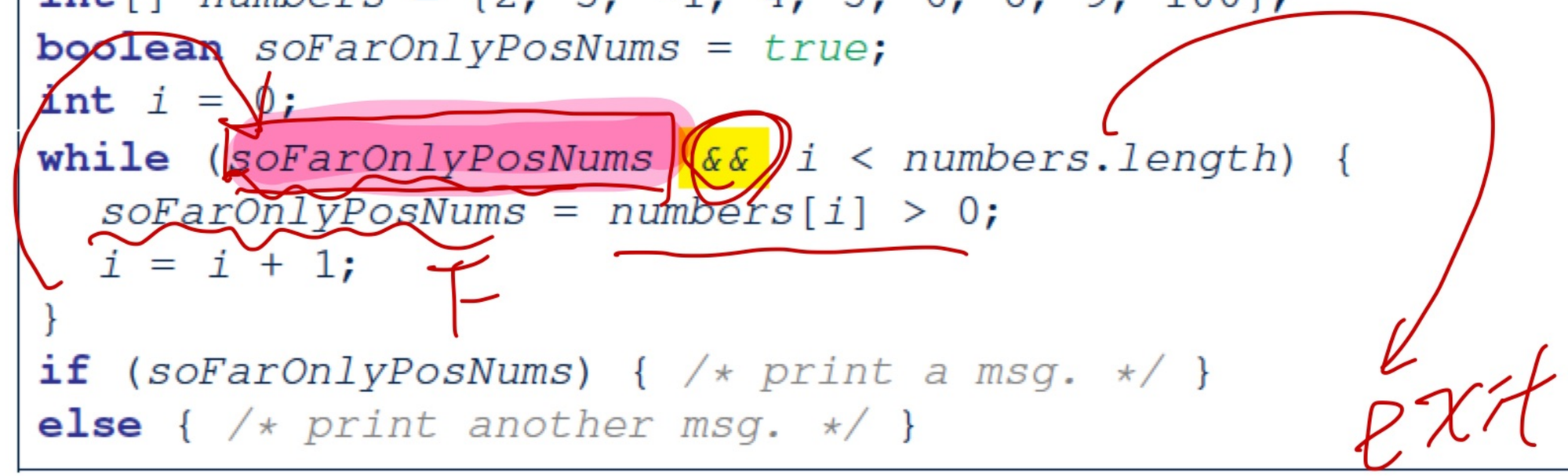


ns

i	ns[i] > 0	soFarOnlyPosNums
0	True	True
1	True	True
2	False	False
3	True	False
4	False	False

as soon as a violation is found, we neither accumulate nor exit. as we should

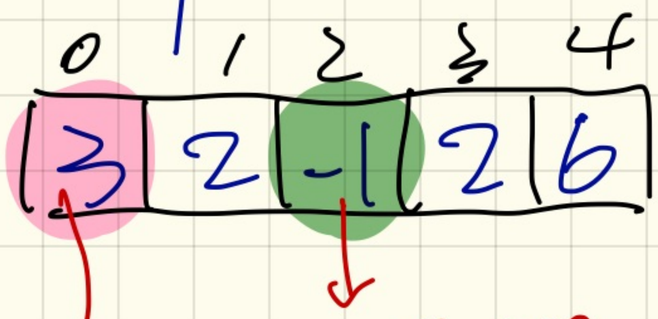
```
1  int[] numbers = {2, 3, -1, 4, 5, 6, 8, 9, 100};
2  boolean soFarOnlyPosNums = true;
3  int i = 0;
4  while (soFarOnlyPosNums && i < numbers.length) {
5     soFarOnlyPosNums = numbers[i] > 0;
6     i = i + 1;
7  }
8  if (soFarOnlyPosNums) { /* print a msg. */ }
9  else { /* print another msg. */ }
```



Monday March 5

Lecture 8

Input : array of elements



positive int[]

String[]

equal to "A"

witness of violation

Problem : (1) Are all elements in a satisfied with a property. (return false if a wit. of viol. is found)

witness of satisfaction

(2) Is there at least an element in a satisfied with a property. (return true if a wit. of satisf. is found)

Input : empty array.

(1) true (:(no wit. of vio.)

(2) false (:(no wit. of sat.).

(1) $\forall x \mid x \in \phi \cdot P(x)$] tmp

False

(2) $\exists x \mid x \in \phi \cdot P(x)$] false

False

(1) all numbers positive

false

true

```
boolean soFar = true;
for(int i=0; i< a.length; i++){
    soFar = soFar && a[i] > 0;
}
```

||

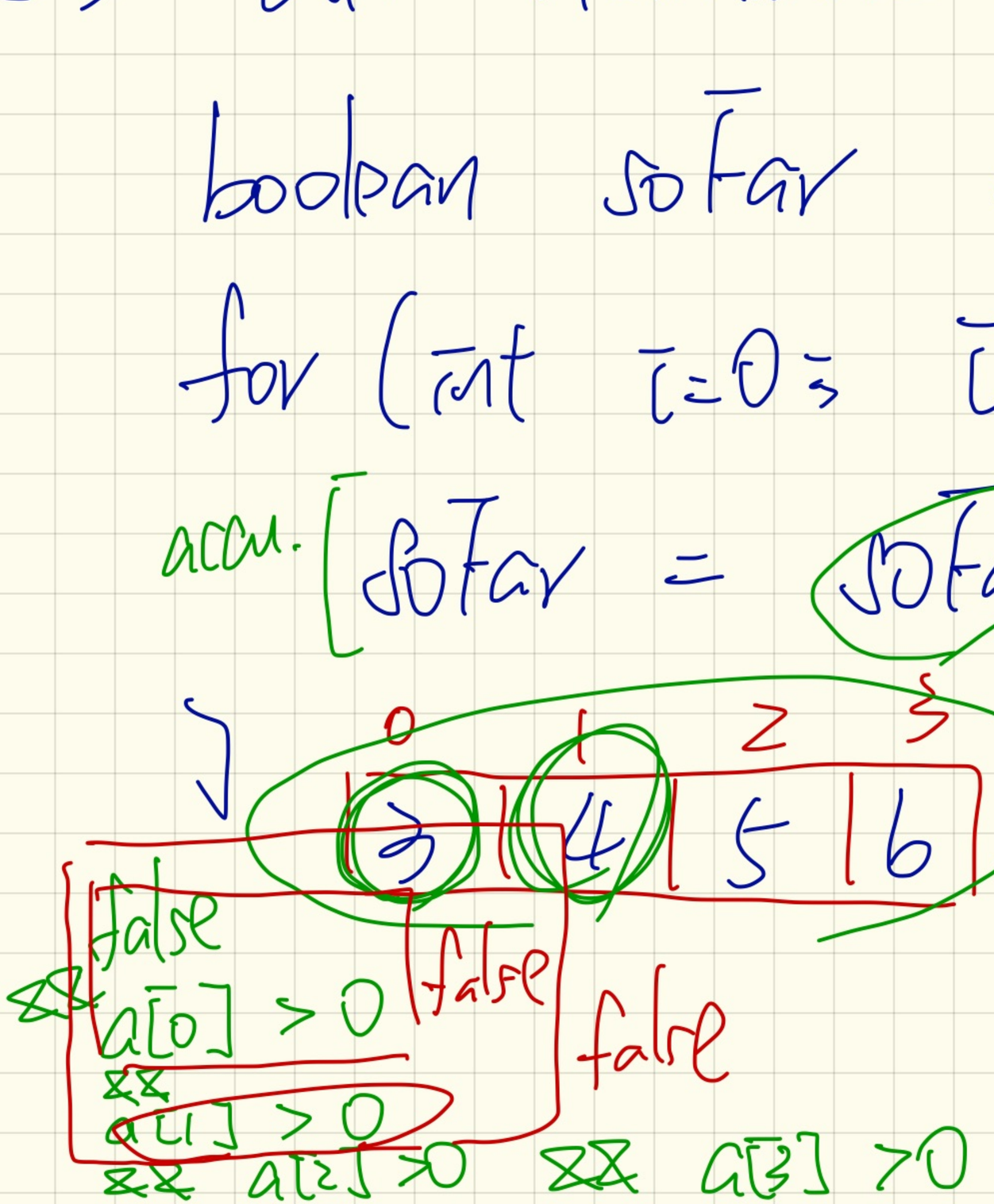
(2) at least one number is positive

(1) all numbers positive

boolean soFar = false; fml.

for (int i=0; i< a.length; i++){

accm. [soFar = soFar && a[i] > 0;]



i	soFar	a[i] > 0	accm.
0	false	3 > 0 True	false
1	false	4 > 0 True	false

max = a[0]

0 X : array contains all negatives

Exercise: find min.

```
1  int max = a[0];
2  for(int i = 0; i < a.length; i ++) {
3     if (a[i] > max) { max = a[i]; } }
4  System.out.println("Maximum is " + max);
```

b

Arrays.sort(a)

0	1	2	3	4	5
2	1	3	4	-4	10

a →

max 2 ~~2~~ 4

i	a[i] > max	
0	2 > 2	false
1	1 > 2	false
2	3 > 2	true
3	4 > 3	~~true~~ true

-2	-3	-1

non-decreasing : !(_ > _)

"" _ ≤ _ → [!(_ < _) "" _ ≥ _]

Sorting

ascending order (increasing) 1 < 3 < 5 < 6 < 7 < 8 ✓

1 3 ≤ 3 4 4 5 ✗

non-ascending order 8 ≥ 7 ≥ 6 ≥ 5 ≥ 3 ≥ 1 ✓

5 ≥ 4 ≥ 4 ≥ 3 ≥ 3 ≥ 1 ✓

descending order (decreasing) 8 > 7 > 6 > 5 > 3 > 1 ✓

5 4 4 3 3 1 ✗

non-descending order ✓ 1 ≤ 3 ≤ 5 ≤ 6 ≤ 7 ≤ 8

✓ 1 ≤ 3 ≤ 3 ≤ 4 ≤ 4 ≤ 5

Version 1: scan entire array

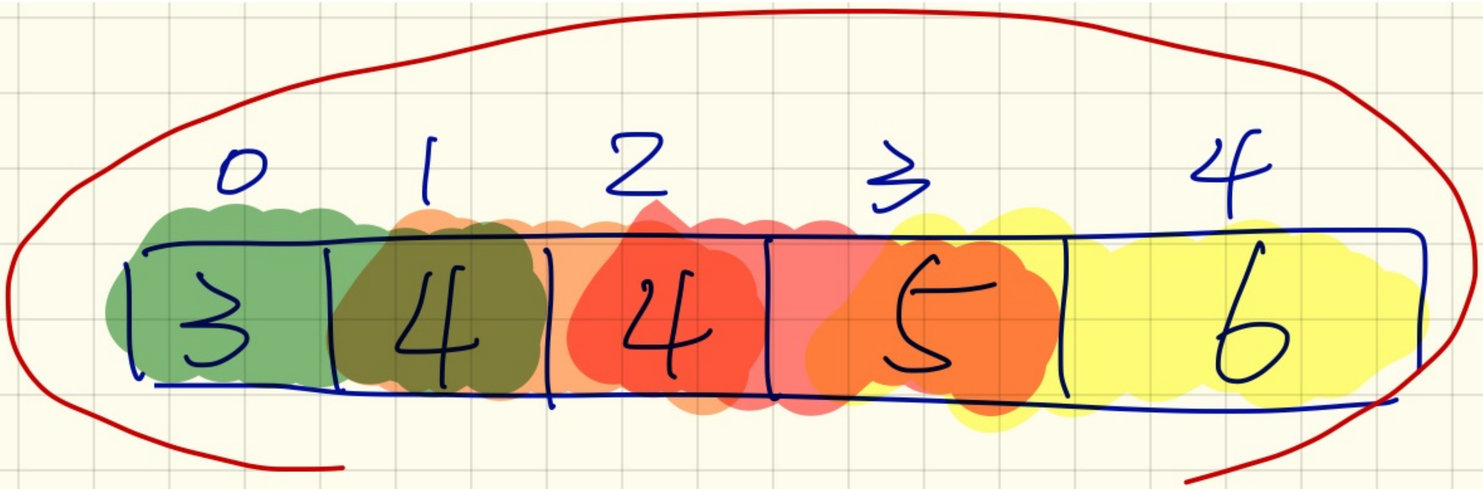
```
1  boolean isSorted = true;
2  for(int i = 0; i < a.length - 1; i ++) {
3    isSorted = isSorted && (a[i] <= a[i + 1]);
4  }
```

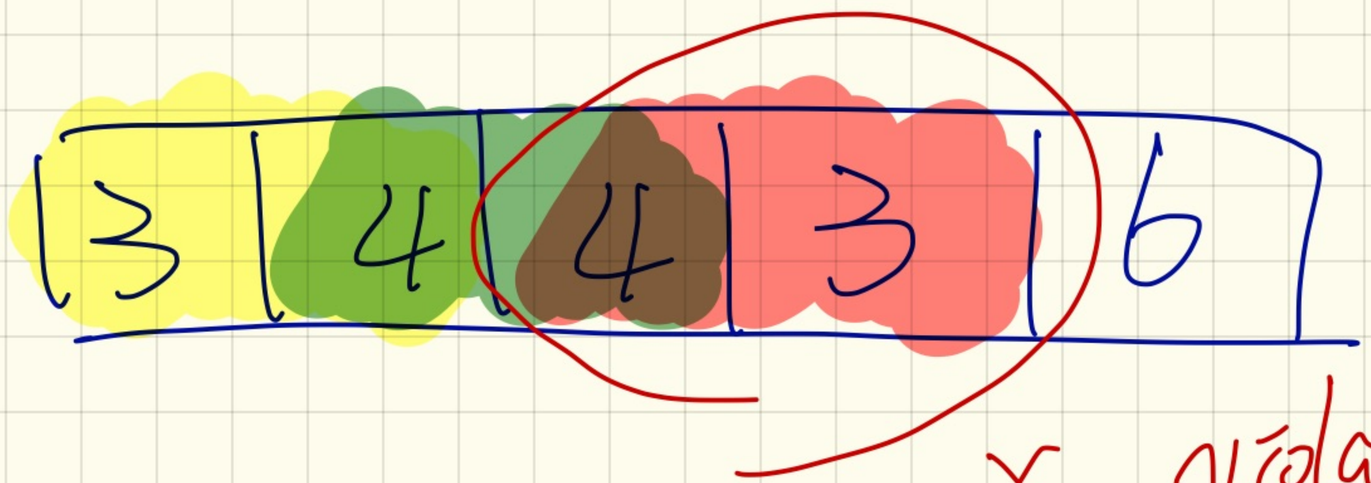
what if:
i < a.length.

i: 0 1 2 4

i from 0 to a.length - 2

i + 1





x violation ∵ 4 ≤ 3 false

a[0] ≤ a[1]
a[1] ≤ a[2]
a[2] ≤ a[3]
a[3] ≤ a[4]
a[4] ≤ a[5]

Version 2 → as soon as wit. of validation is false exit. a.length. found ; exit.

```
1  boolean isSorted = true;
2  for(int i = 0; isSorted && i < a.length - 1; i ++) {
3    isSorted = a[i] <= a[i + 1];
4  }
```

if (a[i] > a[i+1]) isSorted = false

a.length is 5

i	
0	a[0] ≤ a[1]
1	a[1] ≤ a[2]
2	a[2] ≤ a[3]

0	1	2	3	4
3	4	4	3	6

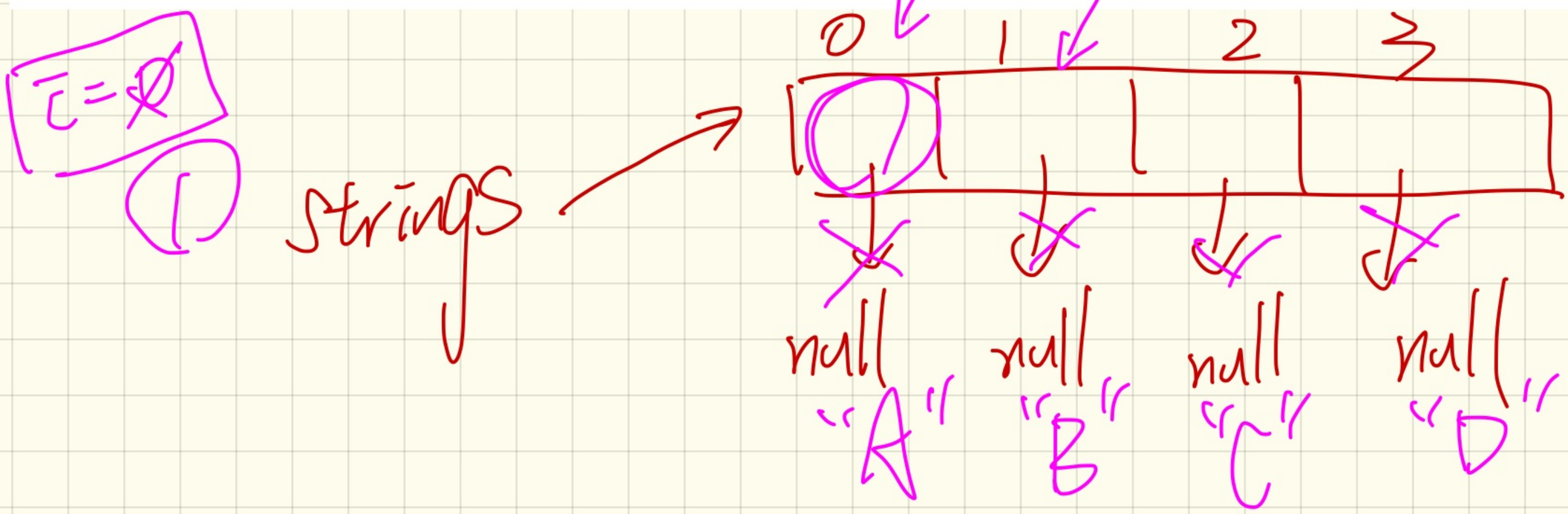
Version 1.

```
boolean isSorted = true;
for (int i = 0; i < a.length - 1; i++){
    isSorted = isSorted && a[i] <= a[i+1];
```

4 iterations

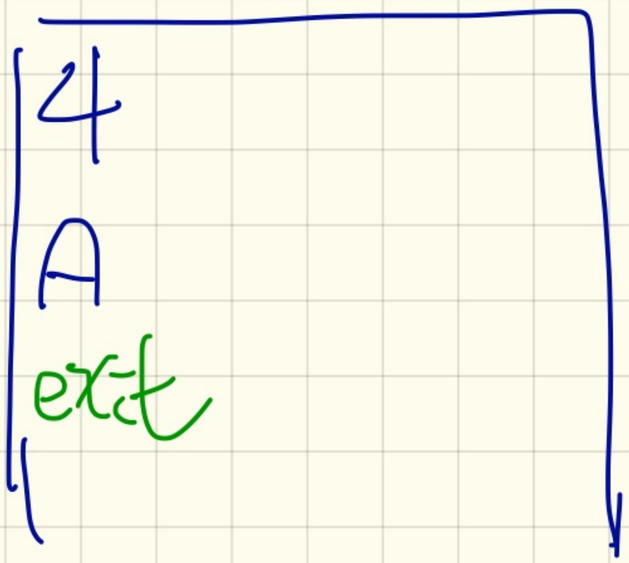
i	
0	a[0] <= a[1]
1	a[1] <= a[2]
2	a[2] <= a[3]

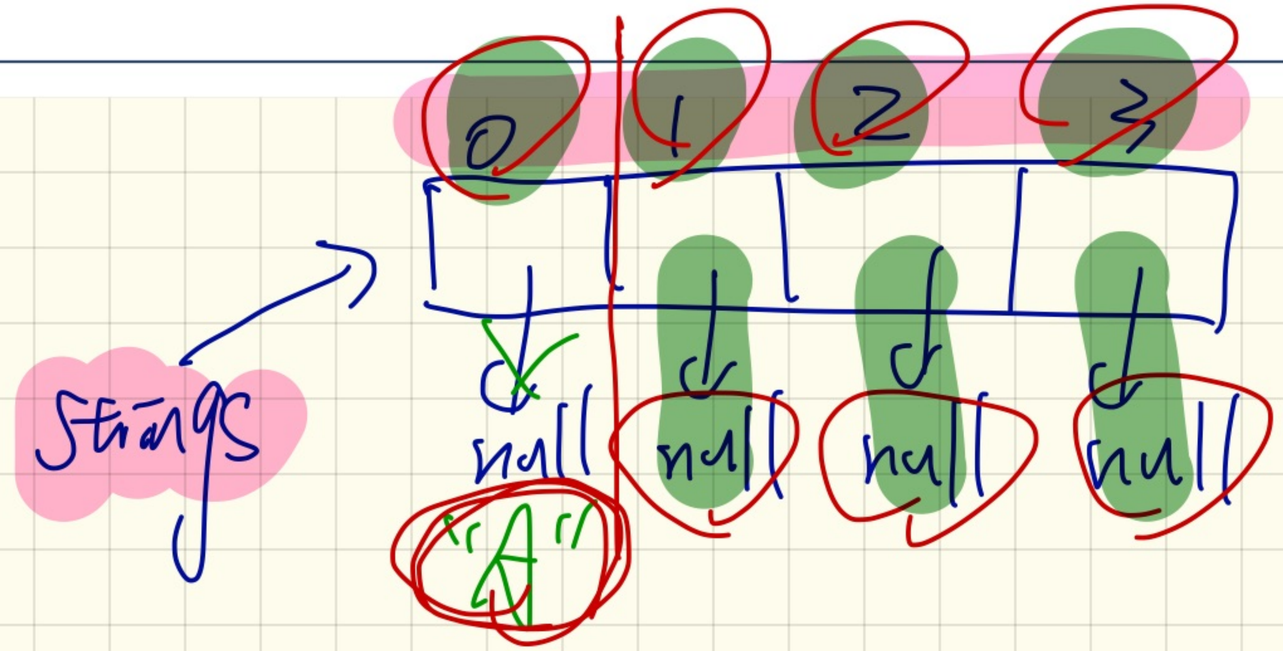
```
1  Scanner input = new Scanner(System.in);
2  System.out.println("How many strings?");
3  int howMany = input.nextInt();
4  String[] strings = new String[howMany];
5  for(int i = 0; i < howMany; i ++) {
6    System.out.println("Enter a string:");
7    String s = input.nextLine();
8    strings[i] = s;
9  }
10 System.out.println("You entered: ");
11 for(int i = 0; i < strings.length; i ++) {
12   System.out.print(strings[i] + " ");
13 }
```



V2. Person

```
1  Scanner input = new Scanner(System.in);
2  System.out.println("How many strings?");
3  int howMany = input.nextInt();
4  boolean userWantsToContinue = true;
5  String[] strings = new String[howMany];
6  for(int i = 0; i < howMany && userWantsToContinue; i ++) {
7    System.out.println("Enter a string:");
8    String s = input.nextLine();
9    userWantsToContinue = !s.equals("exit");
10   if(userWantsToContinue) { strings[i] = s; }
11 }
12 System.out.println("You entered: ");
13 for(int i = 0; i < strings.length; i ++) {
14   System.out.print(strings[i] + " ");
15 }
```





Version 3: Extended from V2.

```
1  Scanner input = new Scanner(System.in);
2  System.out.println("How many strings?");
3  int howMany = input.nextInt();
4  boolean userWantsToContinue = true;
5  String[] strings = new String[howMany];
6  int numberOfStringsRead = 0;
7  for(int i = 0; i < howMany && userWantsToContinue; i ++) {
8    System.out.println("Enter a string:");
9    String s = input.nextLine();
10   userWantsToContinue = !s.equals("exit");
11   if(userWantsToContinue) {
12     strings[i] = s;
13     numberOfStringsRead ++; } }
14 System.out.println("You entered: ");
15 for(int i = 0; i < numberOfStringsRead; i ++) {
16   System.out.print(strings[i] + " "); }
```

howMany = 4

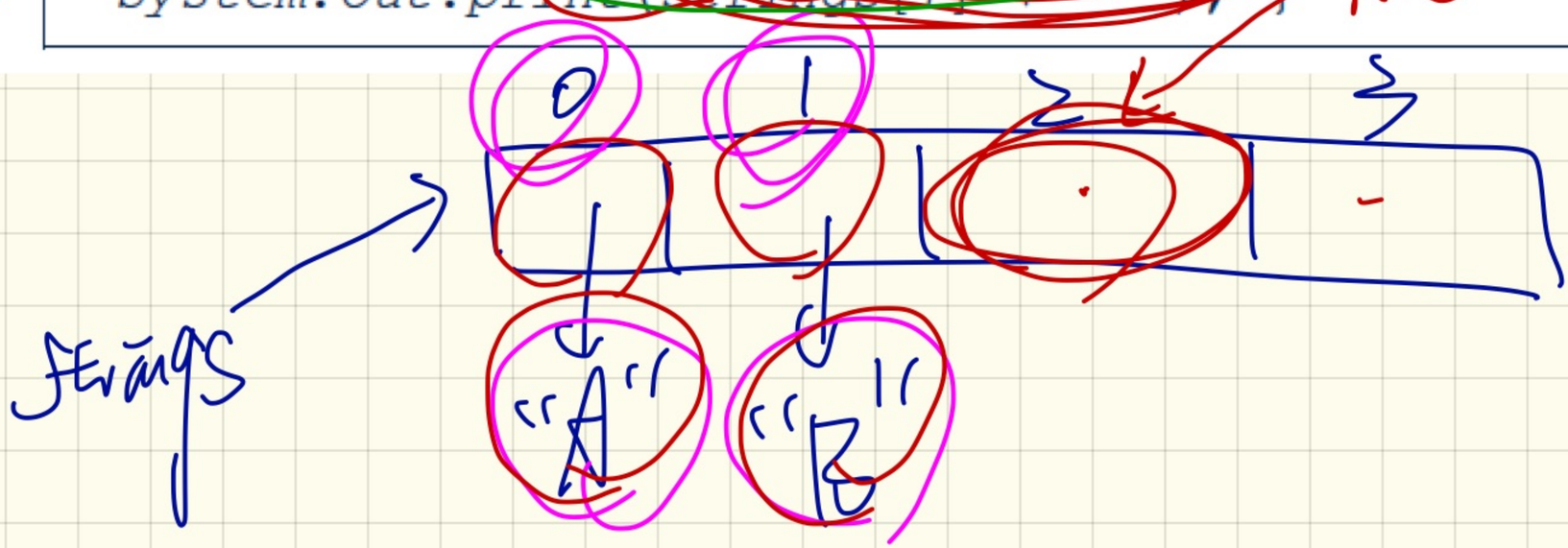
numberOfStringsRead = 0 × 2

i < 2 "A" "B"

strings[0] = "A"

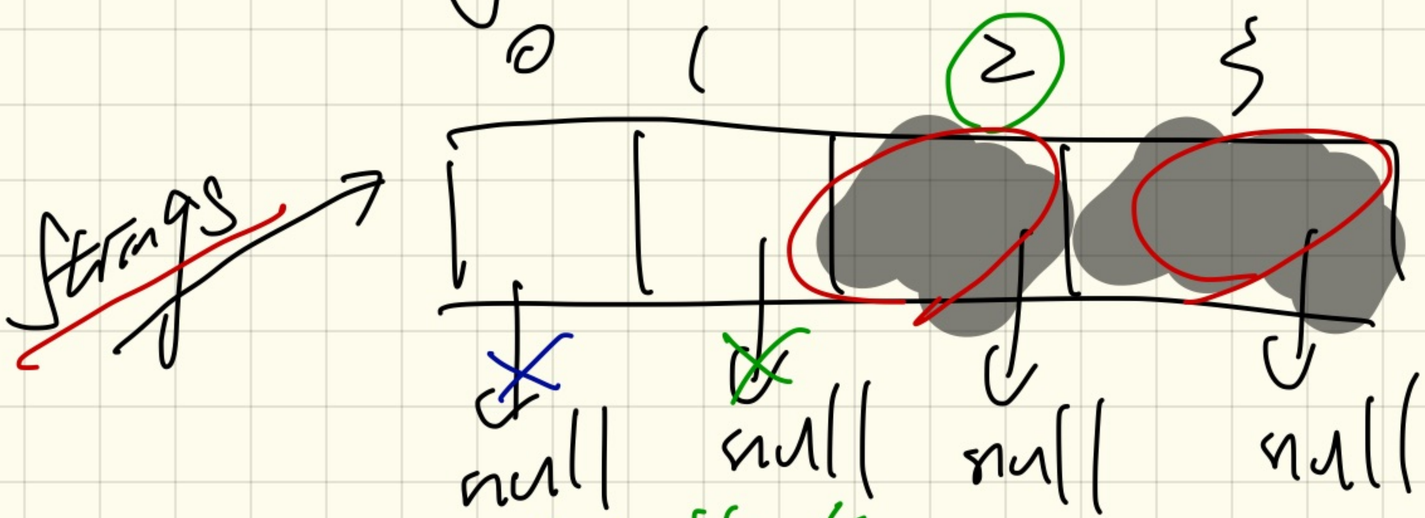
strings[1] = "B"

nos



howMany 4

```
for(int i=0; i<nos; i++){
    println(strings[i]);
}
```

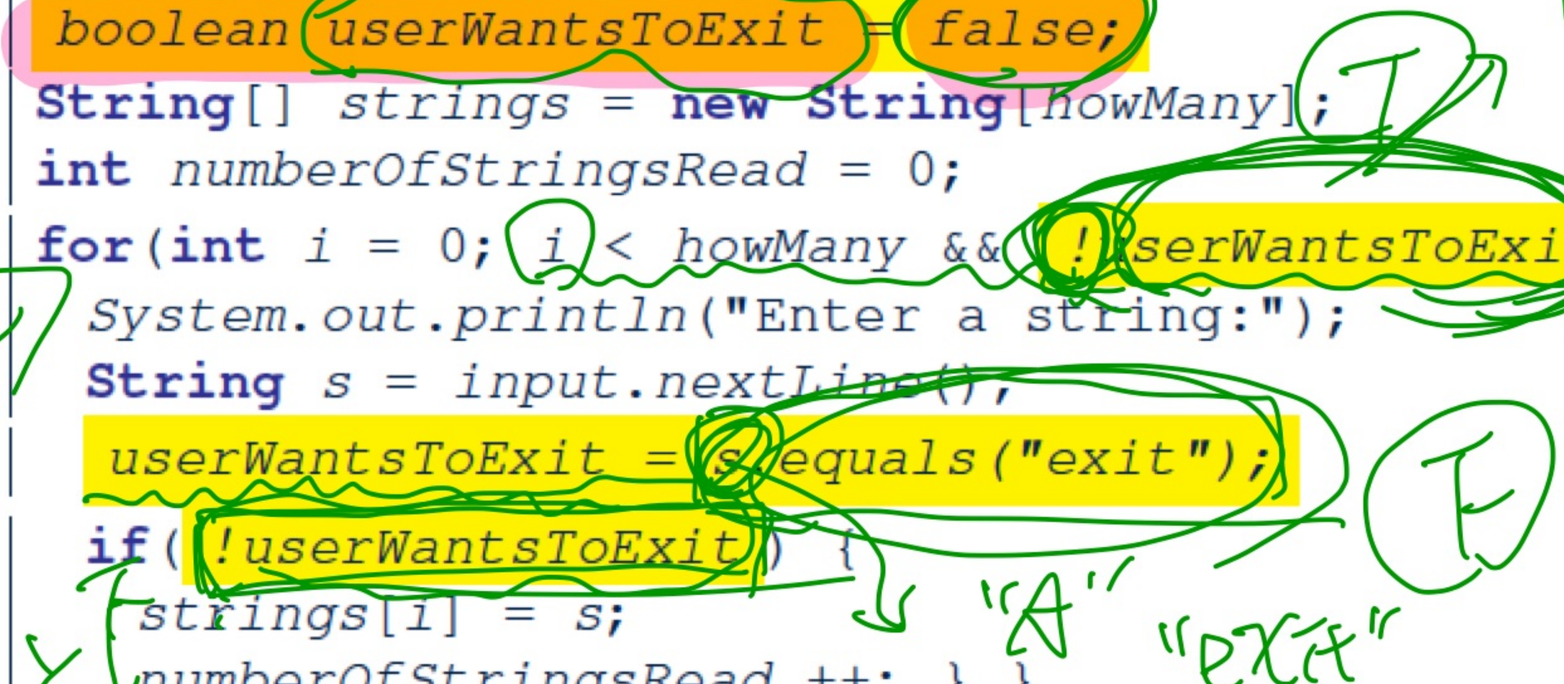


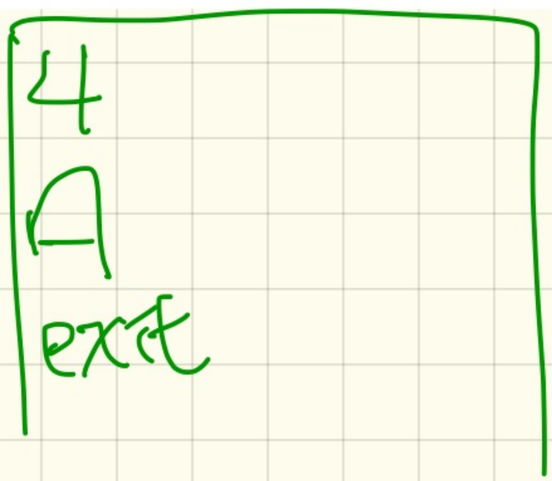
numberOfStrings

- I string read so far
- store next string at index 1

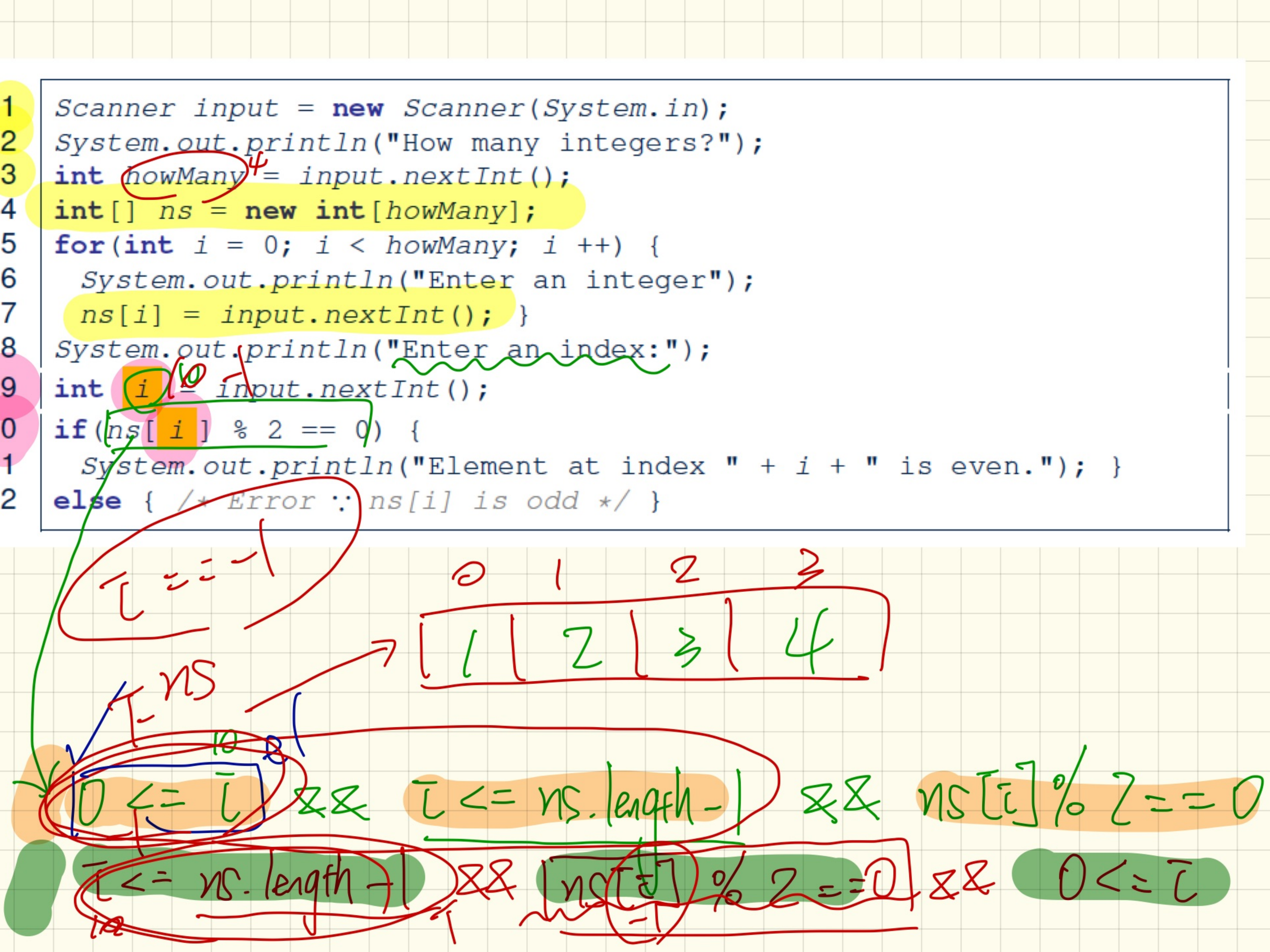
2 → 2 strings read
next string : 2

```
1  Scanner input = new Scanner(System.in);
2  System.out.println("How many strings?");
3  int howMany = input.nextInt();
4  boolean userWantsToExit = false;
5  String[] strings = new String[howMany];
6  int numberOfStringsRead = 0;
7  for(int i = 0; i < howMany && !userWantsToExit; i ++) {
8    System.out.println("Enter a string:");
9    String s = input.nextLine();
10   userWantsToExit = s.equals("exit");
11   if(!userWantsToExit) {
12     strings[i] = s;
13     numberOfStringsRead ++; } }
14 System.out.println("You entered: ");
15 for(int i = 0; i < numberOfStringsRead; i ++) {
16   System.out.print(strings[i] + " "); }
```





```
1  Scanner input = new Scanner(System.in);
2  System.out.println("How many integers?");
3  int howMany = input.nextInt();
4  int[] ns = new int[howMany];
5  for(int i = 0; i < howMany; i ++) {
6    System.out.println("Enter an integer");
7   ns[i] = input.nextInt(); }
8  System.out.println("Enter an index:");
9  int i = input.nextInt();
10 if(ns[ i ] % 2 == 0) {
11   System.out.println("Element at index " + i + " is even."); }
12 else { /* Error ∵ ns[i] is odd */ }
```



Short Circuit

e1 && e2

↓ false

Left to right

e1 || e2

↓ true

Monday March 12

Lecture 9

Draw the flow chart for a nested loop:

```
for (int i = 0; i < 4; i++) {
    print(i);
    for (int j = 0; j < 3; j++) {
        print(j);
    }
}
```

outer → 1

2 → `print(i);` → executed 4 times

3 → `for (int j = 0; j < 3; j++) {`

4 → `print(j);` → executed 12 times

4 * 3

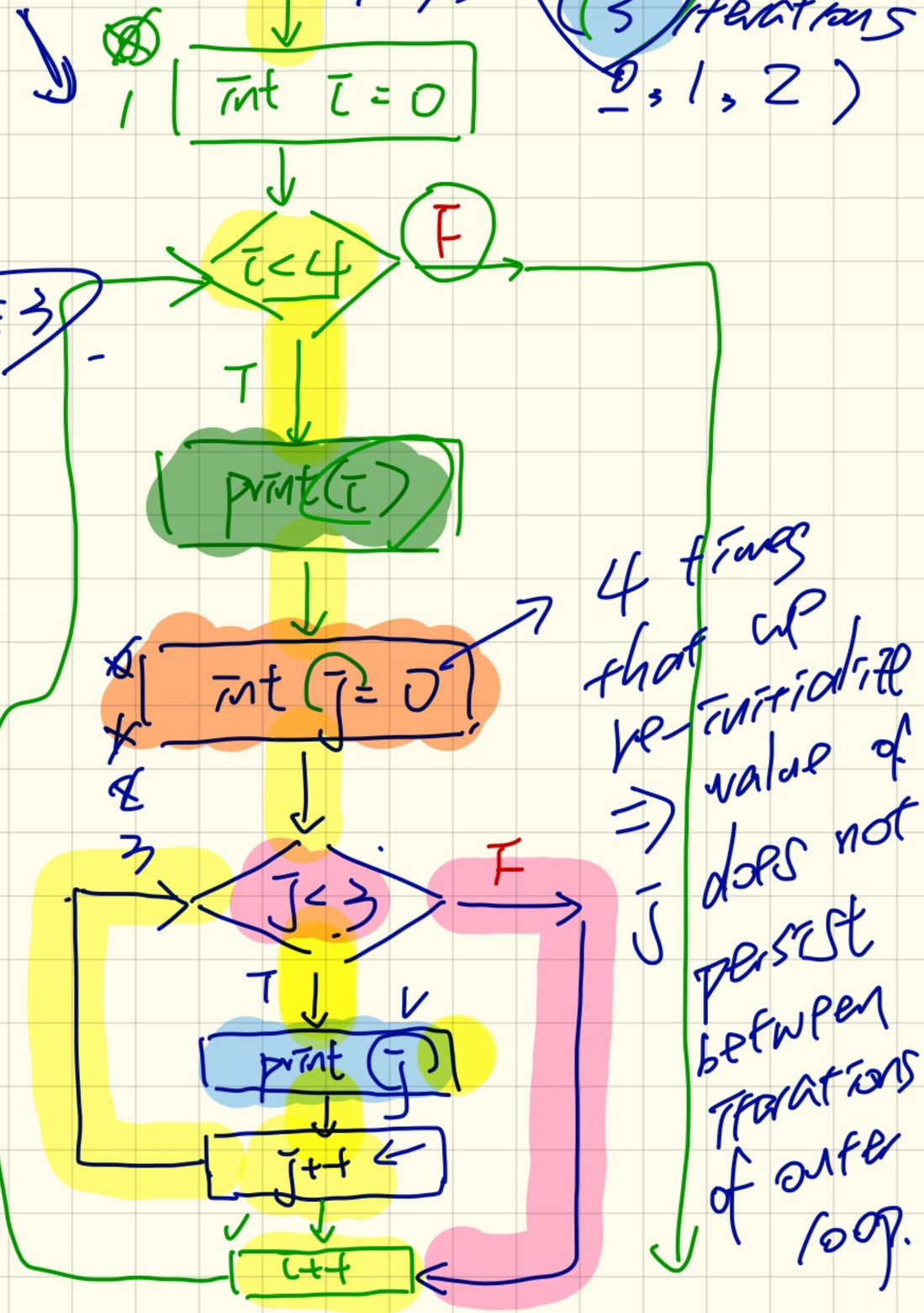
outputs

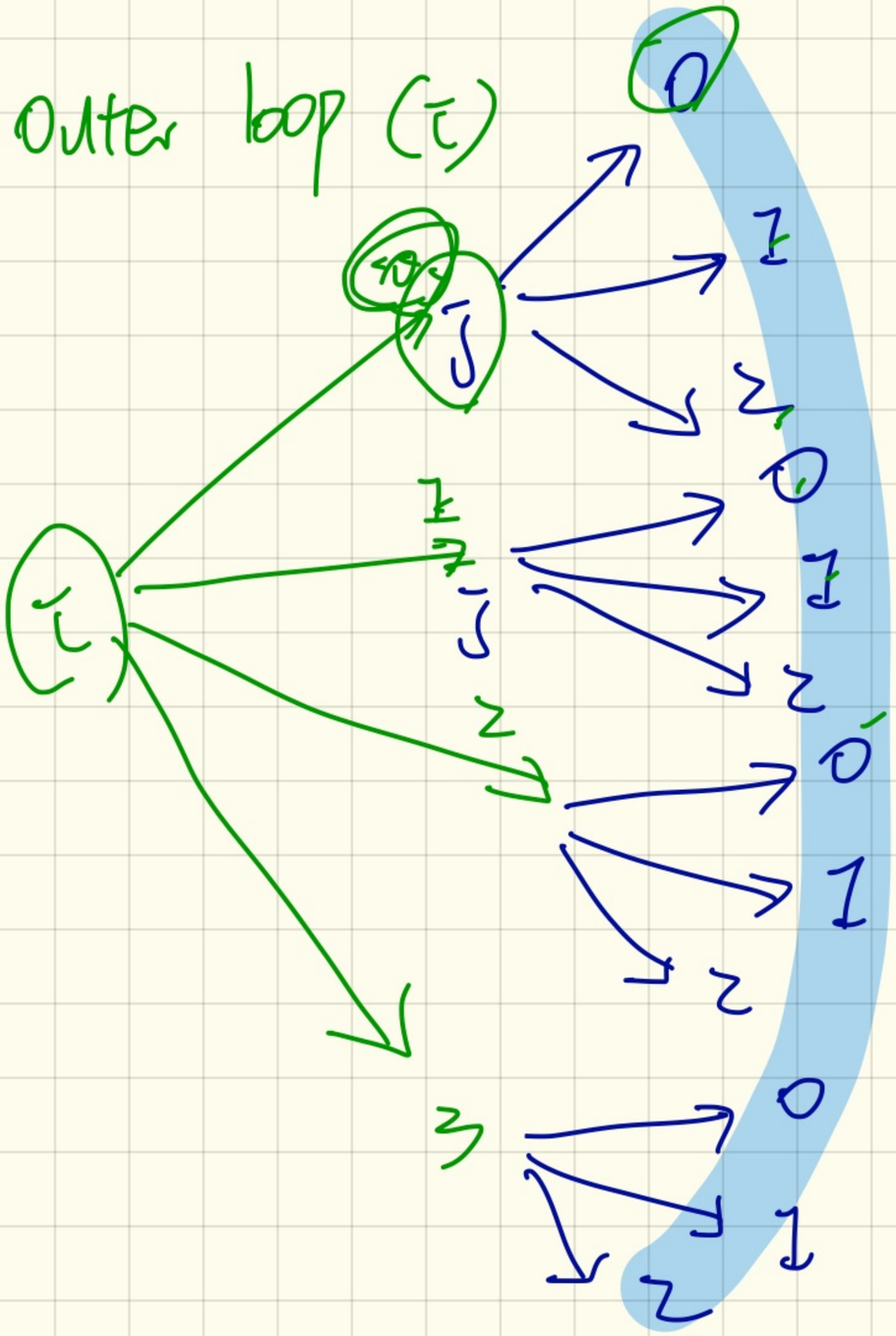
0	1	2	3
0	0	0	0
1	1	1	1
2	2	2	2
1 ←	3		
0			
1			
2			

outer loop

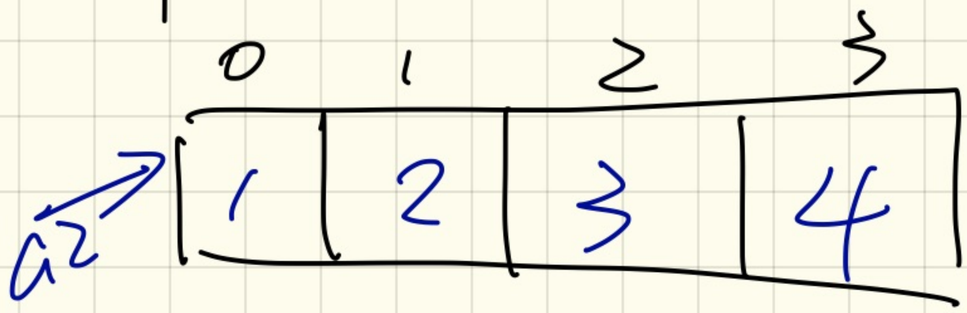
4 iterations (0, 1, 2, 3)

3 iterations (0, 1, 2)

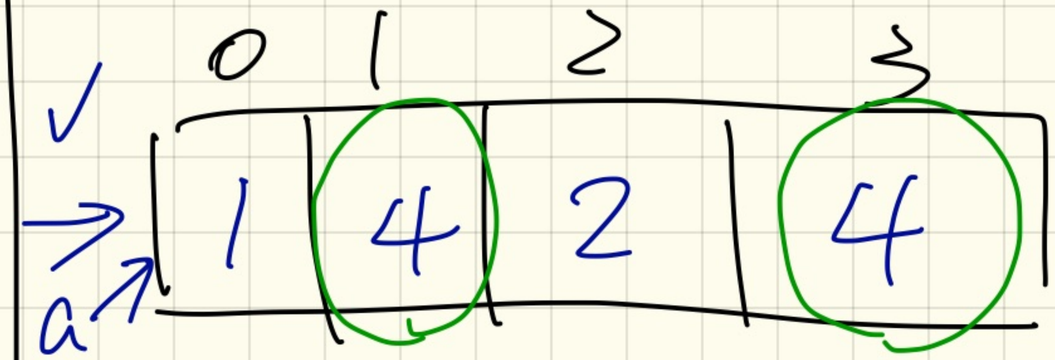




has duplicate ?

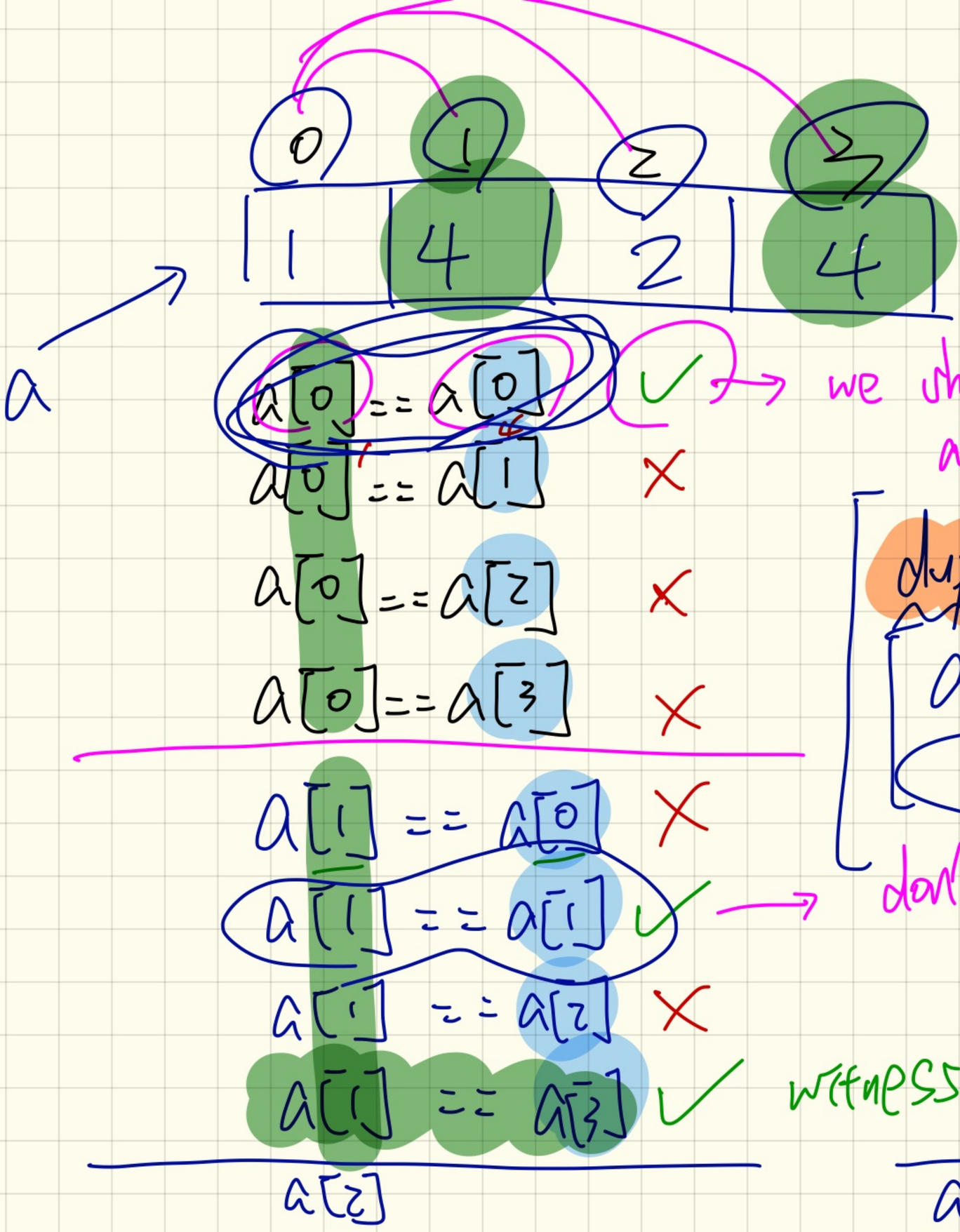


False



true

a[1] != a[3] → witness



4 elements → 4 * 4 → 16 comparisons

↓

4 cases not to be considered.

→ we should not consider this as a witness of duplicate

duplicates:

$a[i] == a[j]$ && $i \neq j$

→ don't consider as a witness

witness!

Correct But Redundant Scan

```
1  boolean hasDup = false;
2  for(int i = 0; i < a.length; i ++) {
3    for(int j = 0; j < a.length; j ++) {
4      hasDup = hasDup || (i != j && a[i] == a[j]);
5    } /* end inner for */ } /* end outer for */
6  System.out.println(hasDup);
```

true

L4 executed

i: 0 → a.length − 1

j: 0 → a.length − 1

a.length * a.length times

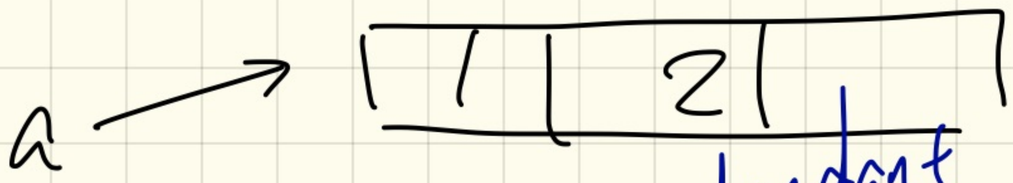
a.length

i	0	1	...	a.length − 1
j	0	1	...	a.length − 1

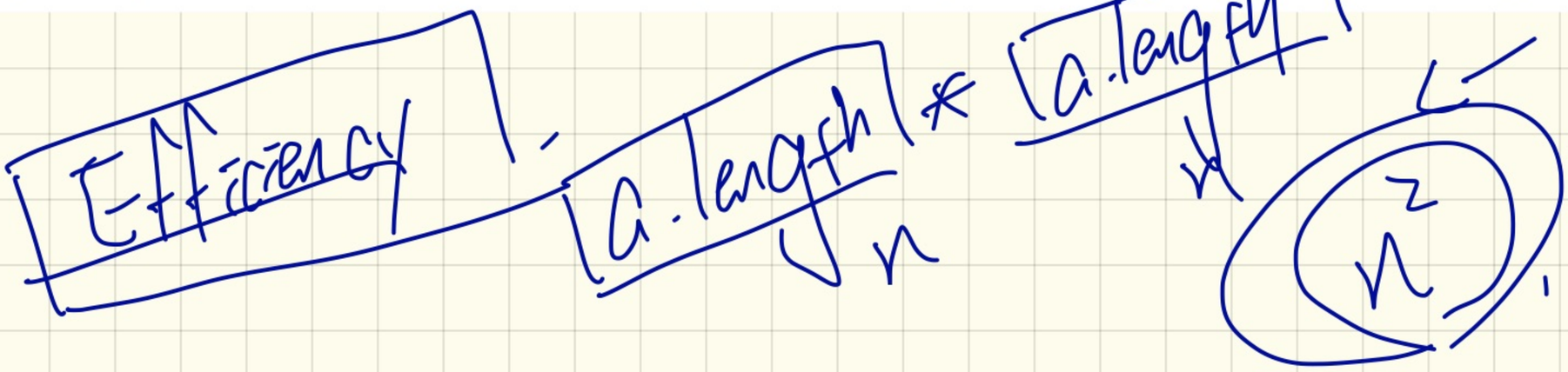
a →	1	4	2	4
0	1	2	3

Before the change : True
(i; a[i] == a[3])

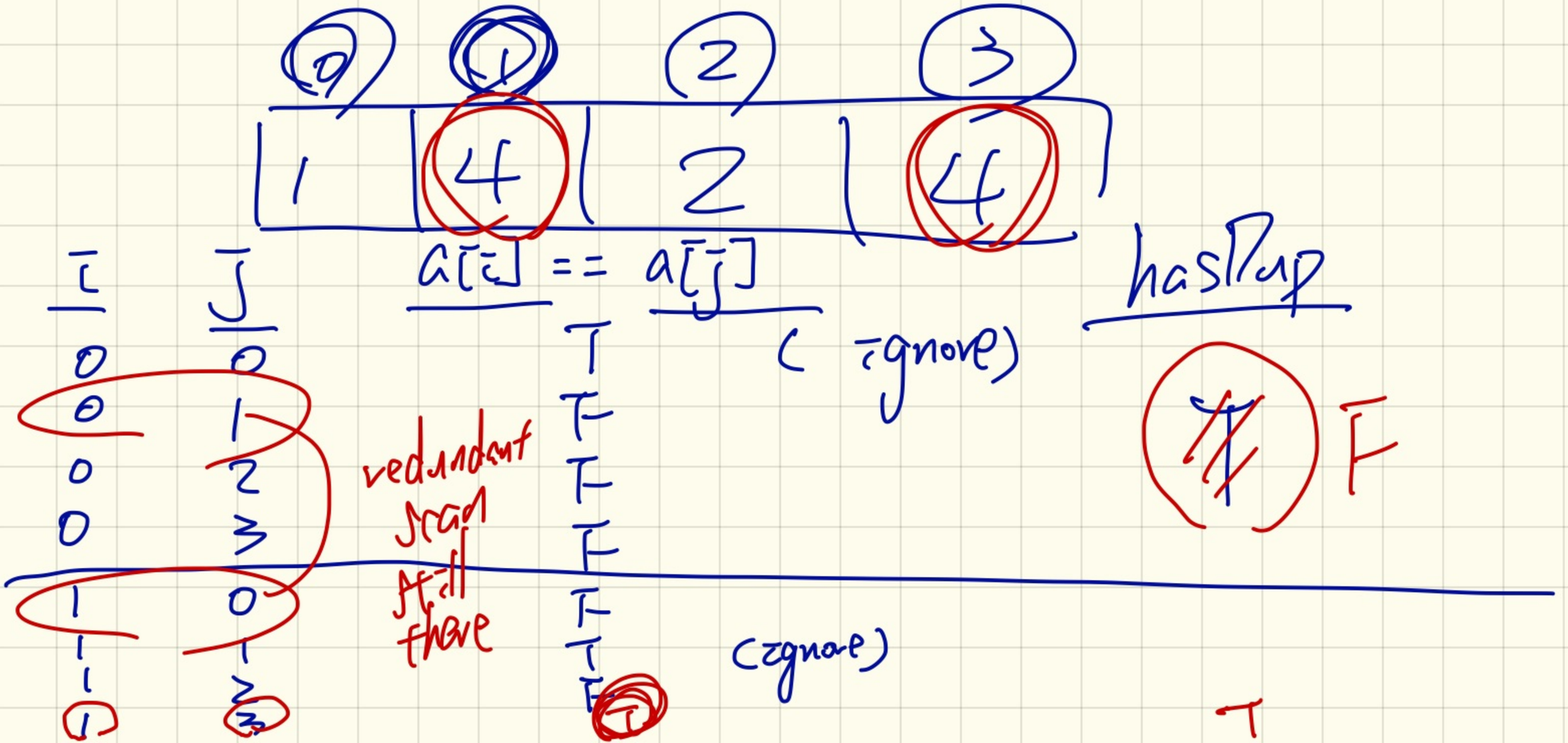
after the change : False
(i; a[0] == a[i] is false)



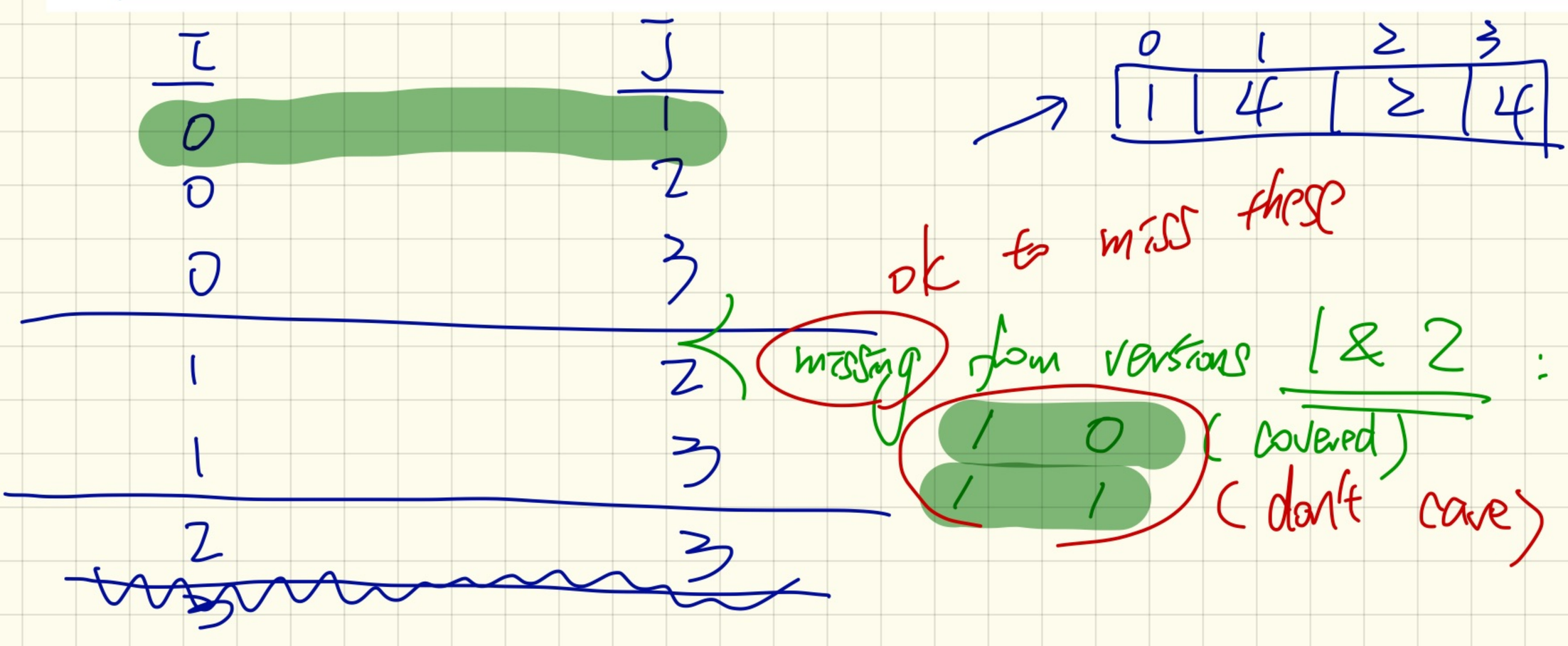
i	j	i != j	a[i]	a[j]	a[i] == a[j]	hasDup
0	0	false	1	1	true	false
0	1	true	1	2	false	false
0	2	true	1	3	false	false
1	0	true	2	1	false	false
1	1	false	2	2	true	false
1	2	true	2	3	false	false
2	0	true	3	1	false	false
2	1	true	3	2	false	false
2	2	false	3	3	true	false



```
1  /* Version 2 with redundant scan */
2  int[] a = {1, 2, 3}; /* no duplicates */
3  boolean hasDup = false;
4  for(int i = 0; i < a.length && !hasDup; i ++) {
5    for(int j = 0; j < a.length && !hasDup; j ++) {
6      hasDup = i != j && a[i] == a[j];
7    } /* end inner for */ } /* end outer for */
8  System.out.println(hasDup);
```



```java
/* Version 3 with no redundant scan:
 * array with duplicates causes early exit
 */
int[] a = {1, 2, 3, 2}; /* duplicates: a[1] and a[3] */
boolean hasDup = false;
for(int i = 0; i < a.length && !hasDup; i++) {
  for(int j = i + 1; j < a.length && !hasDup; j ++) {
    hasDup = a[i] == a[j];
  } /* end inner for */ } /* end outer for */
System.out.println(hasDup);
```

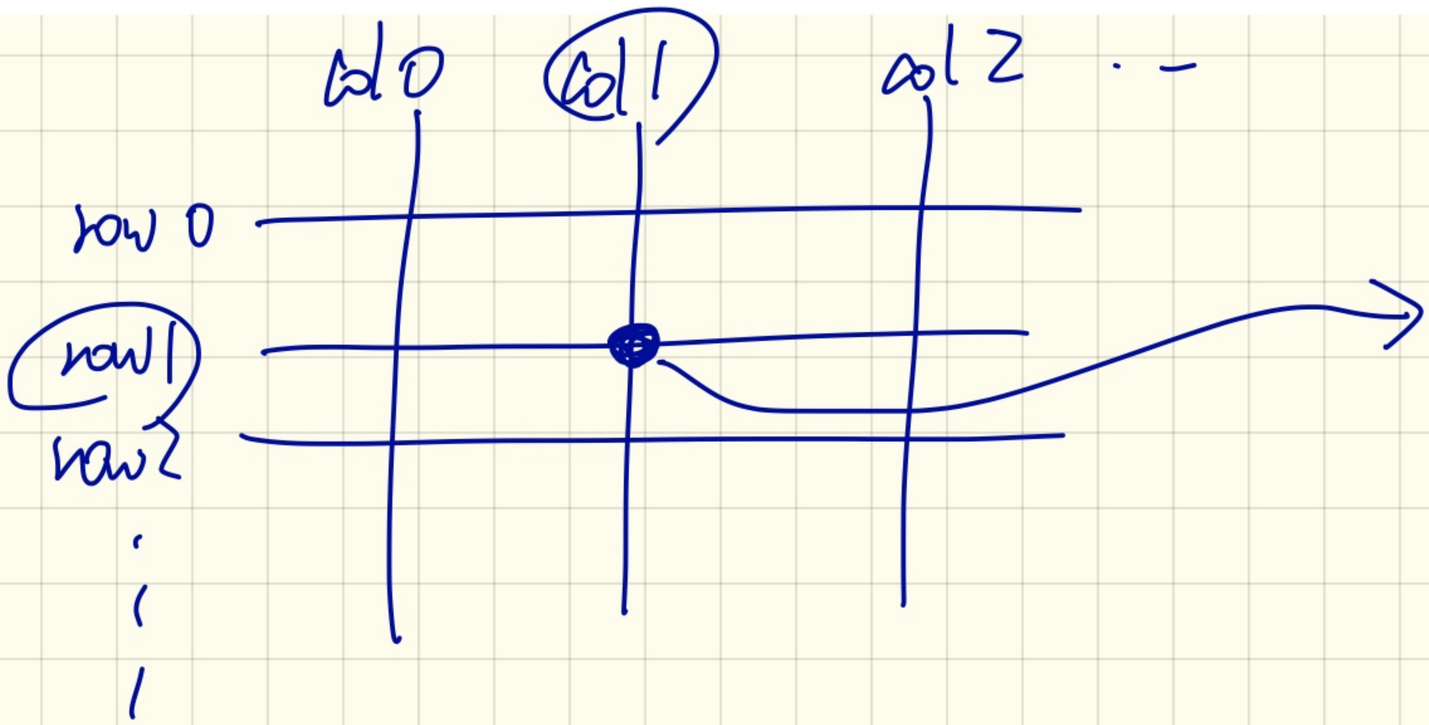


Column → { Bos, Chi, Mia, Hou }

0 983 1375 1187

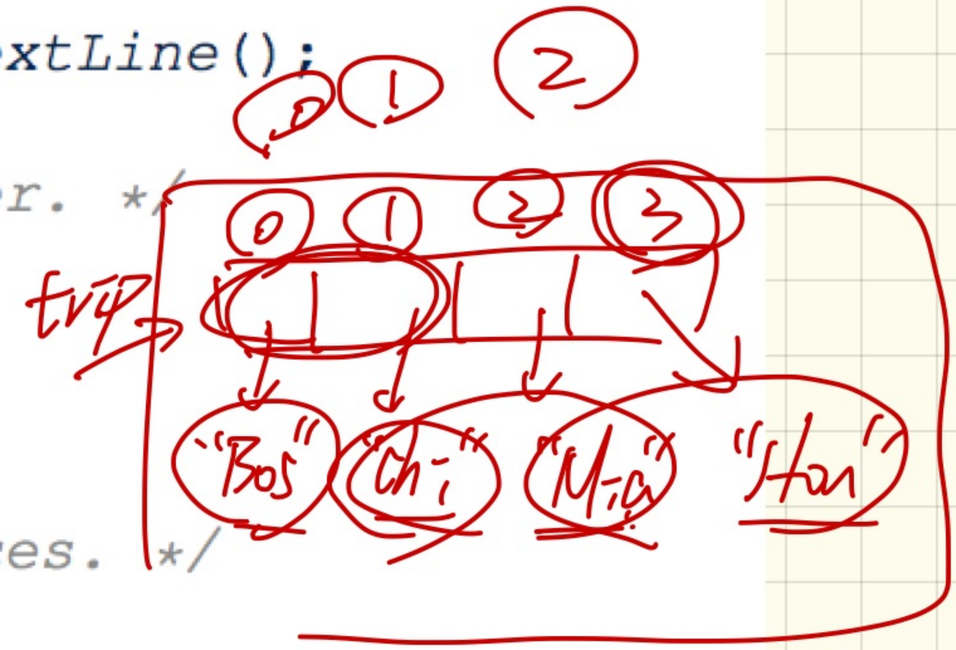
	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	**1375**	967	1087
Boston	**983**	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	**1187**
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

row



→ intersection of row 1 and col 1

```
1  Scanner input = new Scanner(System.in);
2  System.out.println("How many cities?");
3  int howMany = input.nextInt(); input.nextLine();
4  String[] trip = new String[howMany];
5  /* Read cities in the trip from the user. */
6  for(int i = 0; i < howMany; i ++) {
7    System.out.println("Enter a city:");
8    trip[i] = input.nextLine();
9  }
10 /* Add up source-to-destination distances. */
11 int dist = 0;
12 for(int i = 0; i < howMany - 1; i ++) {
13   String src = trip[i];
14   String dst = trip[i + 1];
15   /* How to accumulate the distance between src and dst? */
16 }
```



i=0

src: trip[0]
dst: trip[1]

"Boston"
"Chicago"

```
if ( src.equals("Boston") ){
    if (dst.equals("Chicago")){
        fromBoston[CHICAGO] ;
    }
}
```

fromBoston ["Chicago"]

depending on value of src, determine which array.

depending on val. of dst, figure out index

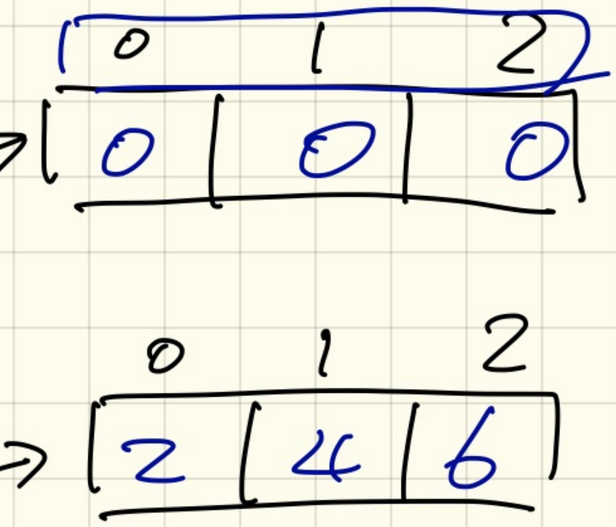
```
13  String src = trip[i];
14  String dst = trip[i + 1];
15  if(src.equals("Chicago")) {
16    if(dst.equals("Boston")) {dist += fromChicago[BOSTON];}
17    else if(dst.equals("New York")) {dist += fromChicago[NY];}
18    ...
19  }
20  else if(src.equals("Boston")) {
21    if(dst.equals("Chicago")) {dist += fromBoston[CHICAGO];}
22    else if(dst.equals("NEW YORK")) {dist += fromBoston[NY];}
23    ...
24  }
25  ...
```

1D - array

index

```
int[] a1 = new int[3];
int[] a2 = {2, 4, 6};
```

size



2D - array

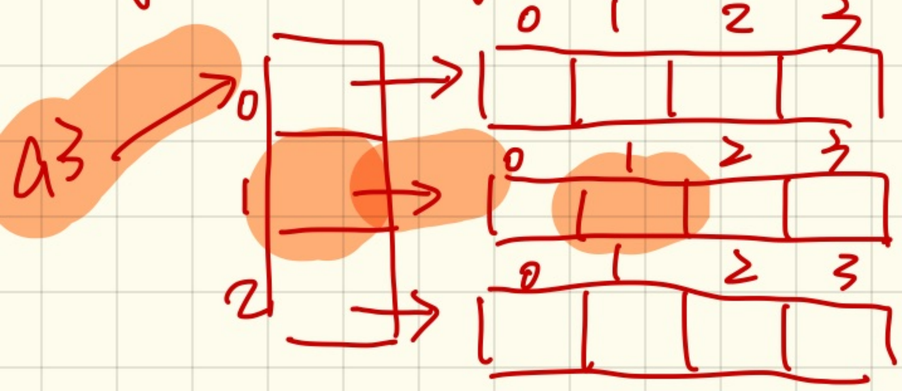
```
int[][] a3 = new int[3][4];
```

points to an array of 3 elements, each of which being an array of 4 elements.

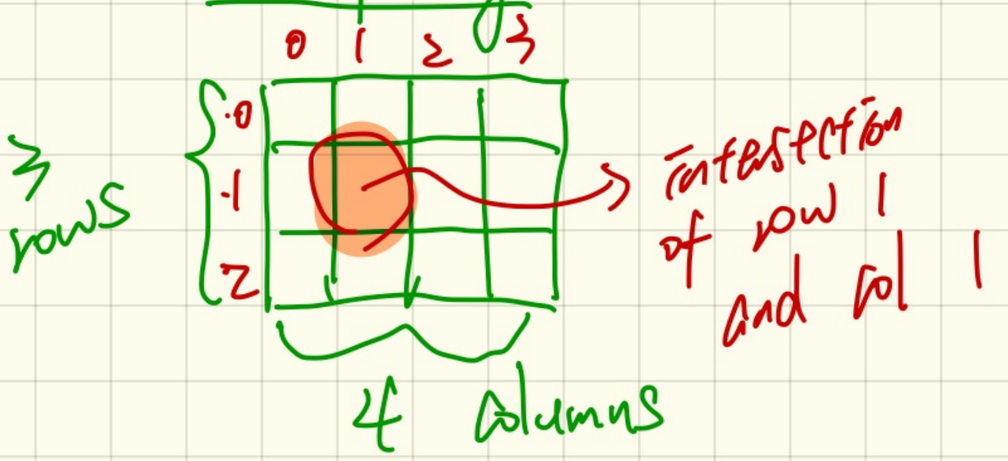
# of rows

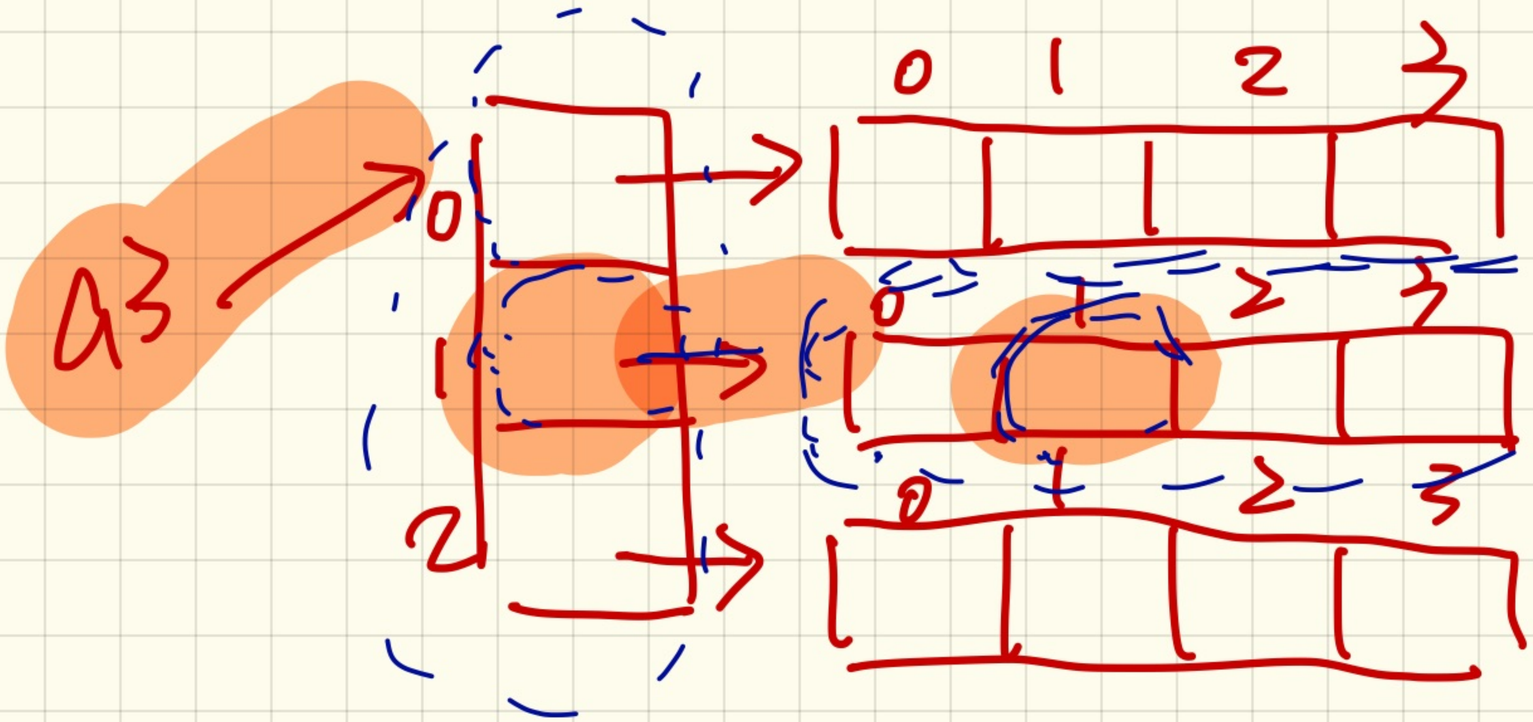
# of columns.

programmatically



Conceptually





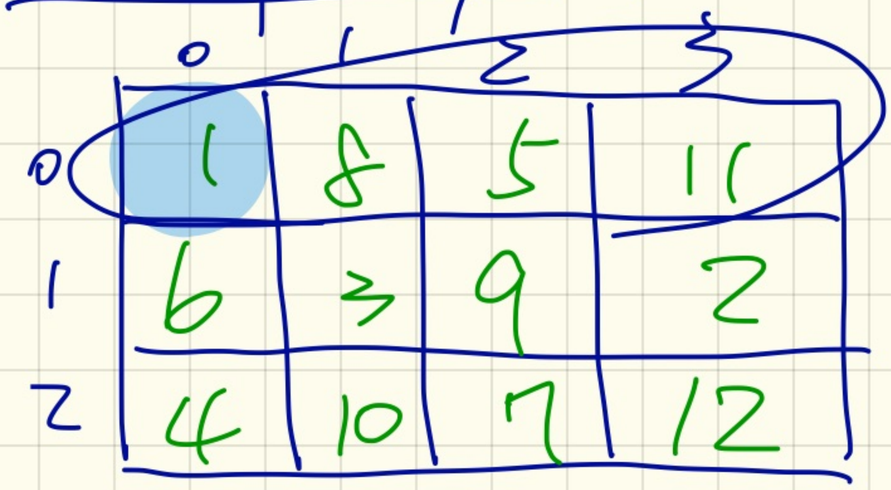

```
int[][][][]...[]
a =
new int[2][3][4]...[10]
```

a3 : an array of arrays of integers.

a3[1] : an array of integers

a3[1][1] : an integer.

Conceptually



Approach 1

```
int[][] a1 = new int[3][4];
a1[0][0] = 1;
    :
    :
```

Approach 2

```
int[][] a2 = {
    {1, 8, 5, 11},
    {6, 3, 9, 2},
    {4, 10, 7, 12}
};
```

```java
double[][] distances = {
    {0, 983, 787, 714, 1375, 967, 1087},
    {983, 0, 214, 1102, 1763, 1723, 1842},
    {787, 214, 0, 888, 1549, 1548, 1627},
    {714, 1102, 888, 0, 661, 781, 810},
    {1375, 1763, 1549, 661, 0, 1426, 1187},
    {967, 1723, 1548, 781, 1426, 0, 239},
    {1087, 1842, 1627, 810, 1187, 239, 0},
};
```

Q: print(distances[4][2]) ? → 1549

distances[4][2] = 2549;

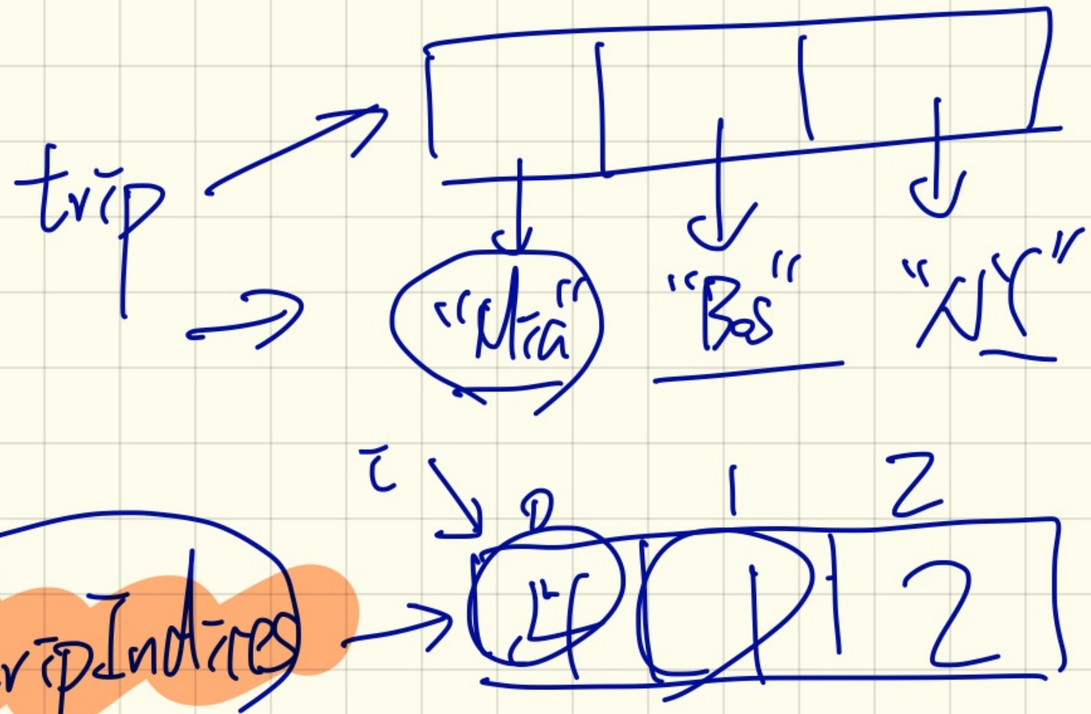
```
final int CHICAGO = 0;
final int BOSTON = 1;
...
final int HOUSTON = 6;

int MiamiToBoston = distances[MIAMI][BOSTON];
int BostonToNewYork = distances[BOSTON][NEWYORK];
int MiamiToNewYork = MiamiToBoston + BostonToNewYork;
```

t {Mia, Bos, NY}

int src = tripIndices[0];

int dst = tripIndices[1];

distances[src][dst]



Monday March 19

Lecture 10

Motivating Example Solution : Part I

```
final int CHICAGO   = 0;
final int BOSTON    = 1;
final int NY        = 2;
final int ATLANTA   = 3;
final int MIAMI     = 4;
final int DALLAS    = 5;
final int HOUSTON   = 6;

int[][] distances = {
    {0, 0    , 983  , 787  , 714  , 1375 , 967  , 1087 },  /* row for Chicago */
    {1, 983  , 0    , 214  , 1102 , 1763 , 1723 , 1842 },  /* row for Boston  */
    {2, 787  , 214  , 0    , 888  , 1549 , 1548 , 1627 },  /* row for NY      */
    {3, 714  , 1102 , 888  , 0    , 661  , 781  , 810  },  /* row for Atlanta */
    {4, 1375 , 1763 , 1549 , 661  , 0    , 1426 , 1187 },  /* row for Miami   */
    {5, 967  , 1723 , 1548 , 781  , 1426 , 0    , 239  },  /* row for Dallas  */
    {6, 1087 , 1842 , 1627 , 810  , 1187 , 239  , 0    }   /* row for Houston */
};
```

arriving at Boston

departing from Miami

Q1. How to look up distance between "Miami" to "Boston"? 1763

Q2. How to calculate distances for itinerary {"Miami", "Boston", "NY"}?

Motivating Example Solution: Part 2

```
Scanner input = new Scanner(System.in);
System.out.println("How many cities?");
int howMany = input.nextInt(); input.nextLine();
String[] trip = new String[howMany];
int[] tripPos = new int[howMany];
boolean someCityIsInvalid = false;
String[] undefinedCities = new String[howMany];
int numberOfUndefinedCities = 0;
/* Read cities in the trip from the user. */
for(int i = 0; i < howMany; i ++) {
    System.out.println("Enter a city:");
    String city = input.nextLine();
    trip[i] = city;
    if(city.equals("Chicago")) {
        tripPos[i] = CHICAGO;
    }
    else if(city.equals("Boston")) {
        tripPos[i] = BOSTON;
    }
    else if(city.equals("NY")) {
        tripPos[i] = NY;
    }
    else if(city.equals("Atlanta")) {
        tripPos[i] = ATLANTA;
    }
    else if(city.equals("Miami")) {
        tripPos[i] = MIAMI;
    }
    else if(city.equals("Dallas")) {
        tripPos[i] = DALLAS;
    }
    else if(city.equals("Houston")) {
        tripPos[i] = HOUSTON;
    }
    else {
        undefinedCities[numberOfUndefinedCities] = city;
        numberOfUndefinedCities++;
        someCityIsInvalid = true;
    }
}
```

trip.length == howMany

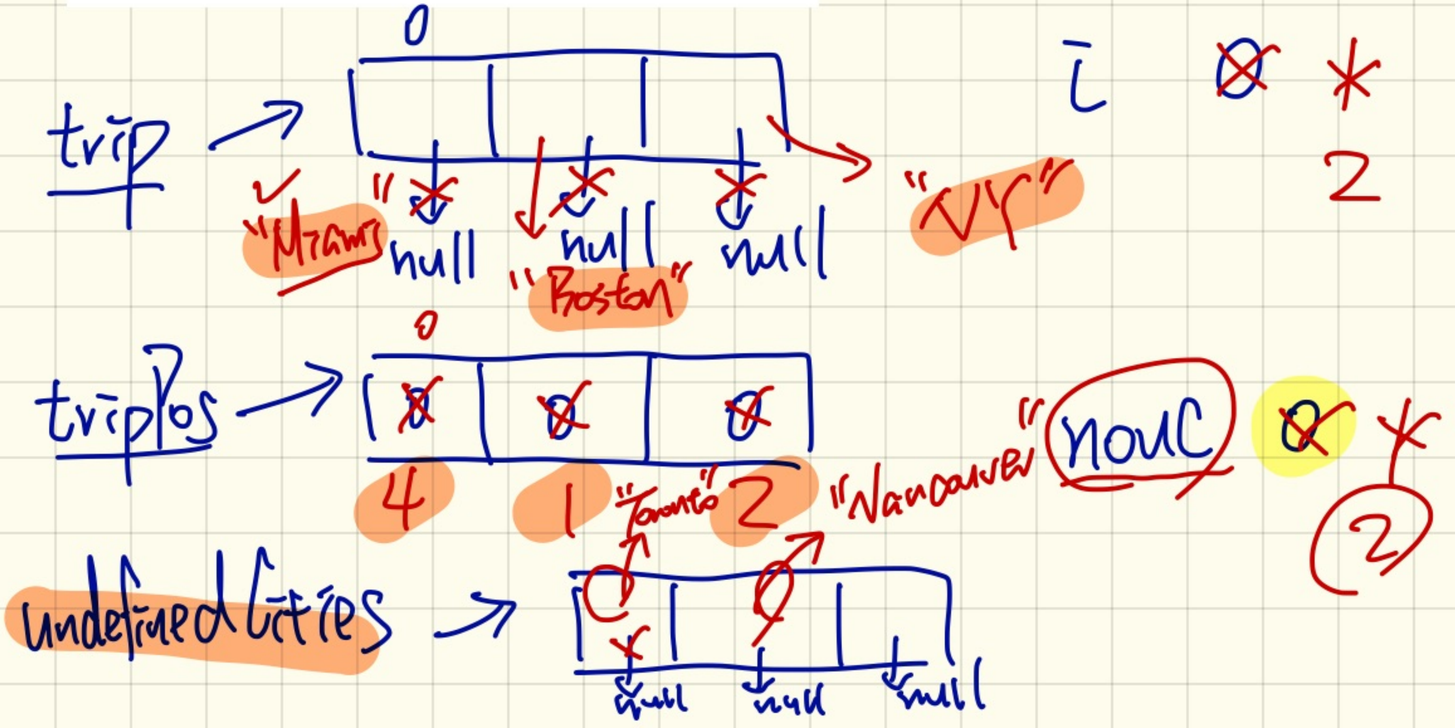
"Miami" MIAMI

indicates if any errors

Console:

```
How many cities?
3
Enter a city:
Miami
Enter a city:
Boston
Enter a city:
NY
From Miami to Boston: 1763
From Boston to NY: 214
Distance: 1977
Bye!
```

Toronto ✗ Vancouver ✗ NY



Motivating Example Solution : Part 3

isSorted

a[i] <= a[i+1]

i < a.length - 1

```
if(someCityIsInvalid) {
    System.out.print("Error: ");
    for(int i = 0; i < numberOfUndefinedCities; i ++) {
        System.out.print(undefinedCities[i]);
        if(i < numberOfUndefinedCities - 1) {
            System.out.print(", ");
        }
    }
    System.out.println(" are undefined.");
}
else {
    /* Add up source-to-destination distances. */
    int dist = 0;
    for(int i = 0; i < howMany - 1; i ++) {
        String srcCity = trip[i];
        String dstCity = trip[i + 1];
        int src = tripPos[i];
        int dst = tripPos[i + 1];
        int currentDist = 0;
        currentDist = distances[src][dst];
        dist += currentDist;
        System.out.print("From " + srcCity + " to " );
        System.out.println(dstCity + ": " + currentDist);
    }
    System.out.println("Distance: " + dist);
}
System.out.println("Bye!");
input.close();
```

howMany

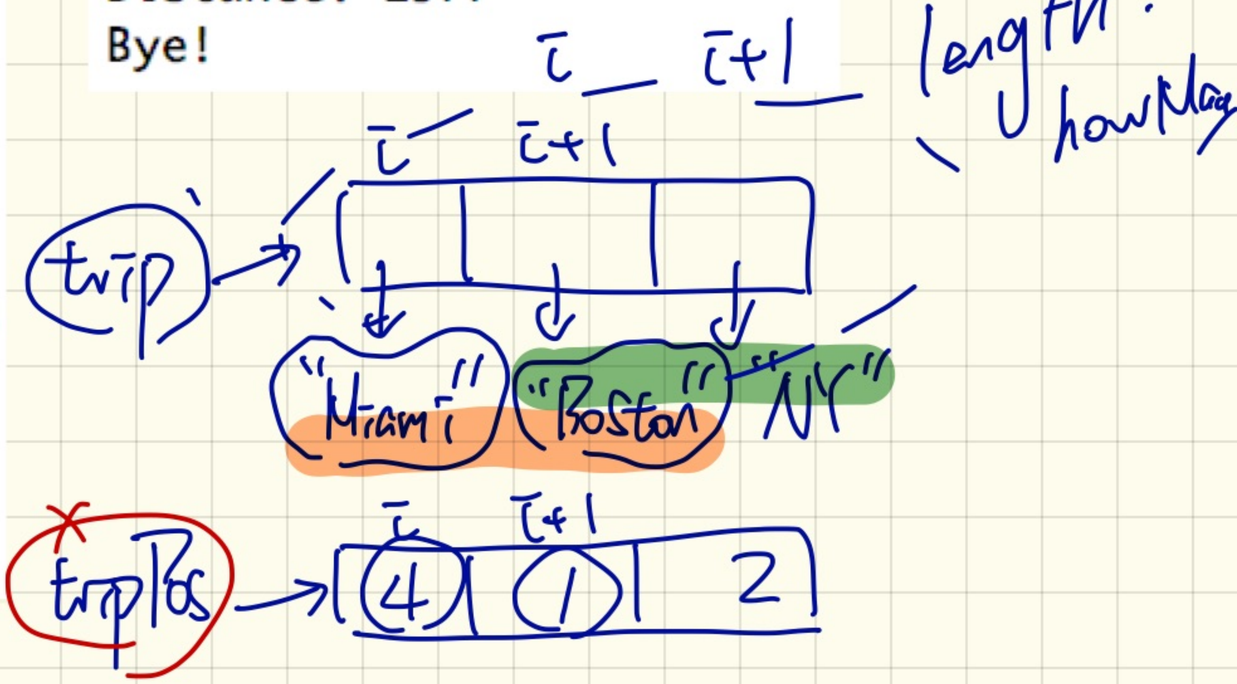
distances[srcCity][dstCity]

i	srcCity	dstCity	src	dst
0	"Miami"	"Boston"	4	1

1763

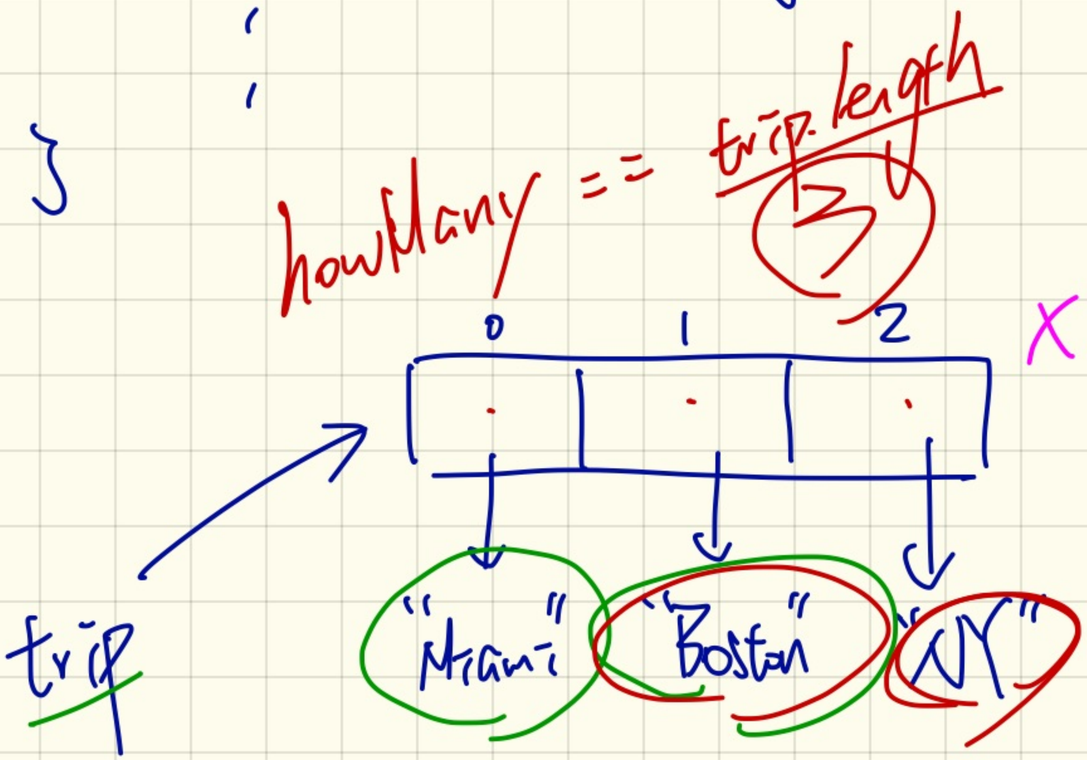
Console:

```
How many cities?
3
Enter a city:
Miami
Enter a city:
Boston
Enter a city:
NY
From Miami to Boston: 1763
From Boston to NY: 214
Distance: 1977
Bye!
```



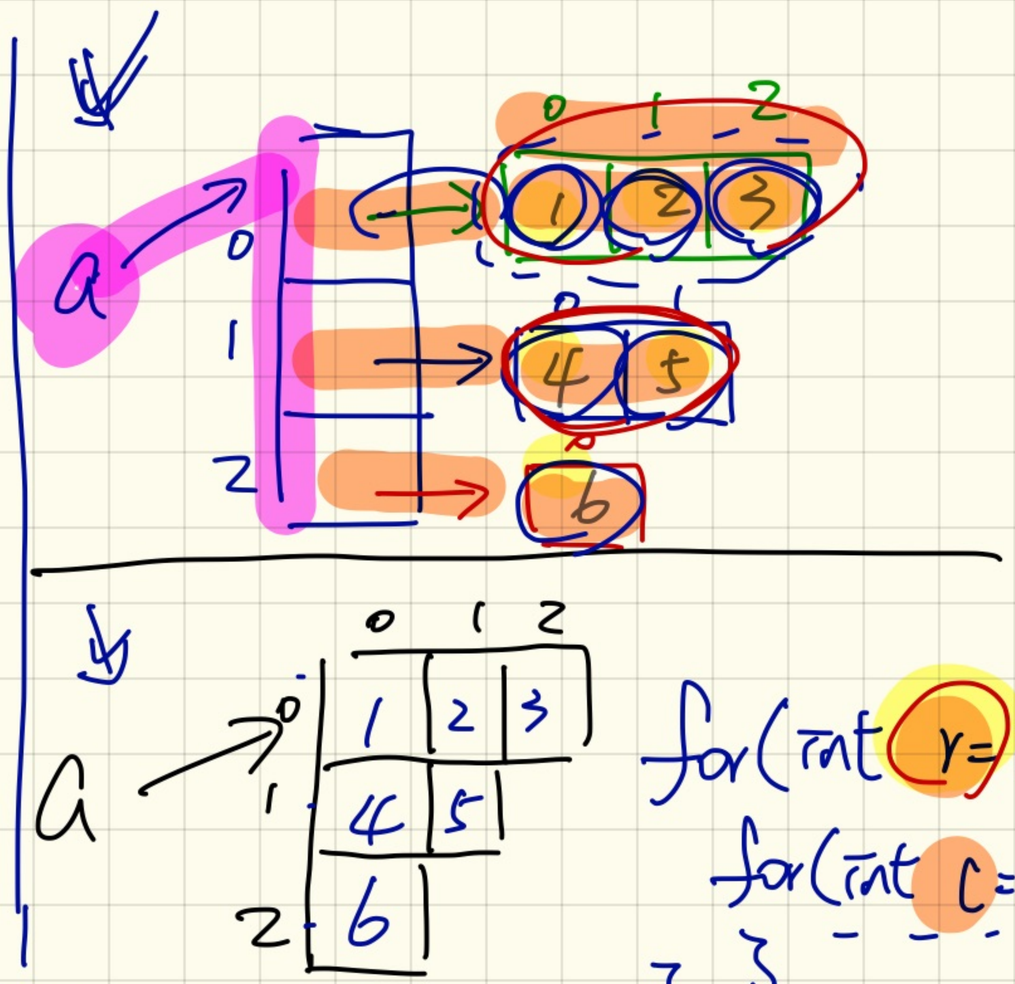
```
String[]  trip = new String[ howMany ];
   :
   :

for( int i = 0;   i < howMany;   i++ ) {
     String srcCity = trip[ i ];
     String dstCity = trip[ i+1 ];
     :
     :
}
```

howMany − 1



i	i+1	srcCity	dstCity
0	1	"Miami"	"Boston"
1	2	"Boston"	"NY"
2	3	"NY"	trip[3] → IndexOutOfB.

```
int[][] a = {
    {1, 2, 3},
    {4, 5},
    {6}
};
```



	0	1	2
0	1	2	3
1	4	5	
2	6		

```
for(int r=0; r<a.length; r++){
    for(int c=0; c<a[r].length; c++){
    }
}
```

# of rows: a.length → 3

# of columns: a[0].length → 3, a[1].length → 2, a[2].length → 1

a[0][0] a[0][1] a[0][2]

a[1][0] a[1][1]

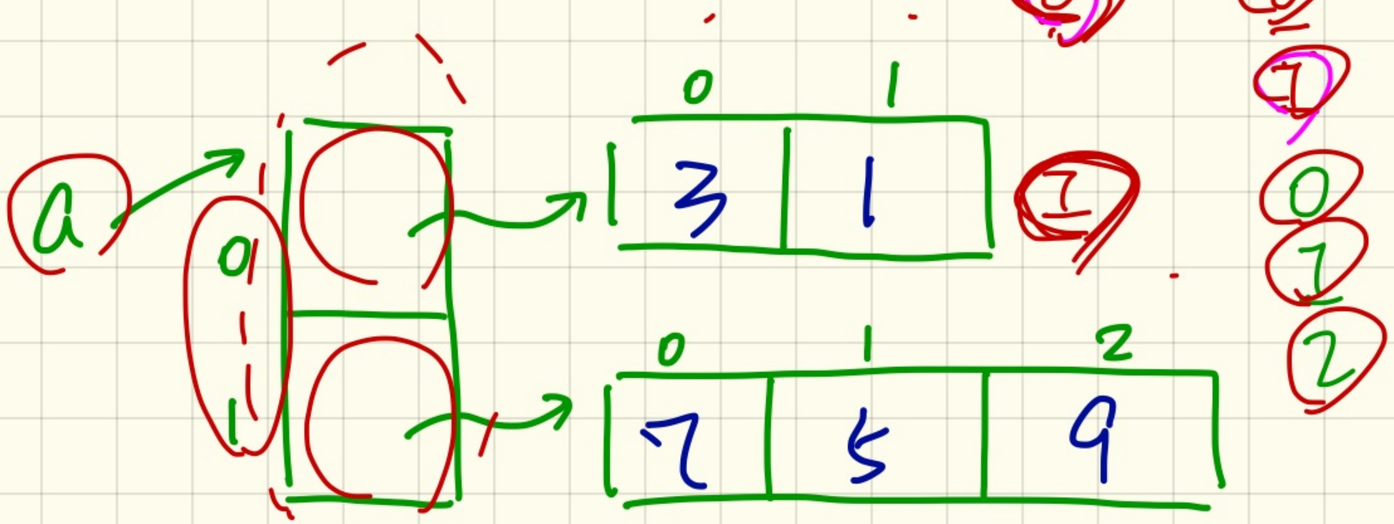
a[2][0]

Example 1: Print 2D Array

```
1  for(int row = 0; row < a.length; row ++) {
2    System.out.print("Row" + row);
3    for(int col = 0; col < a[row].length; col ++) {
4      System.out.print(a[row][col]);
5    }
6    System.out.println(); }
```

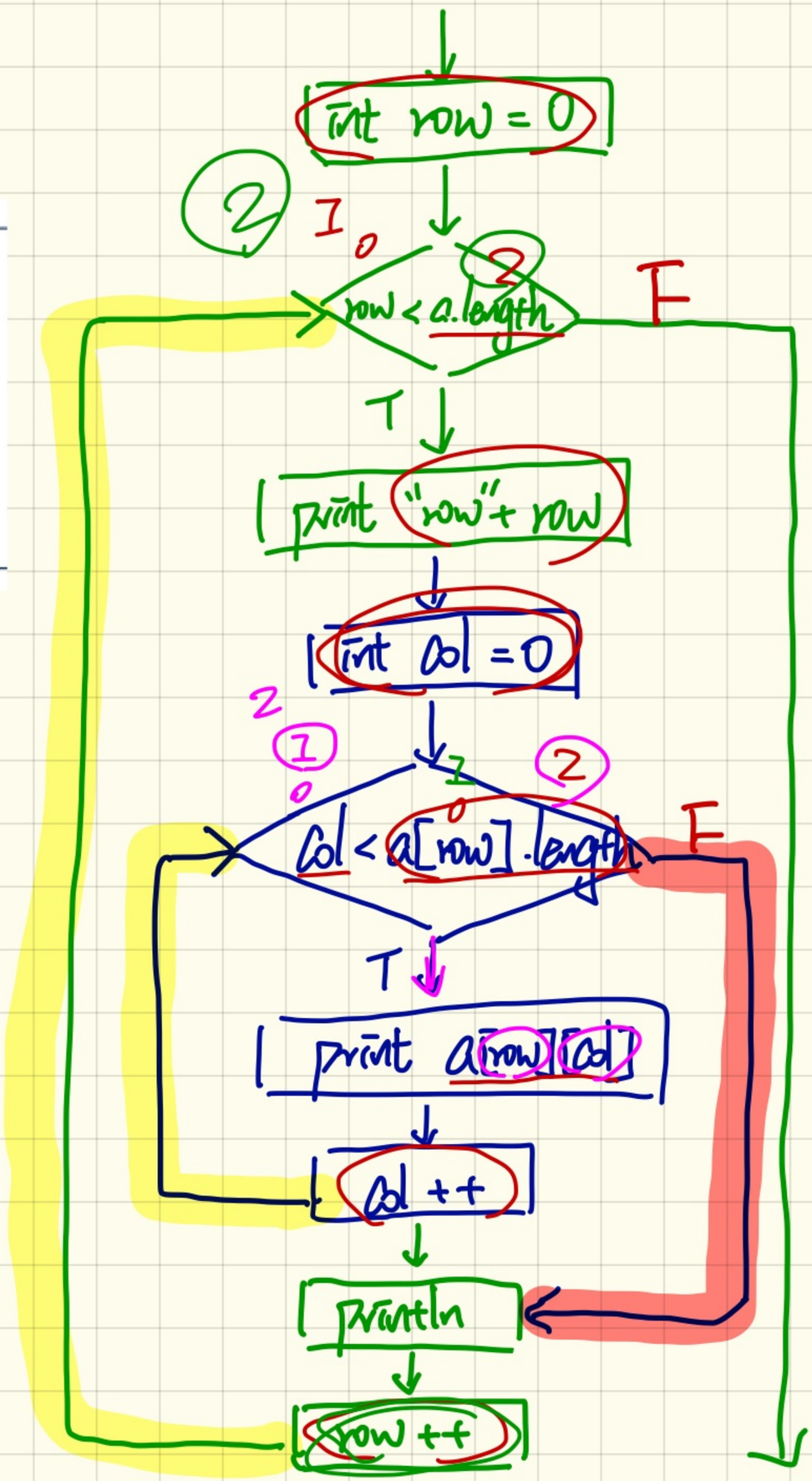
row 0

row	col
0	0
	1
1	0
	1
	2



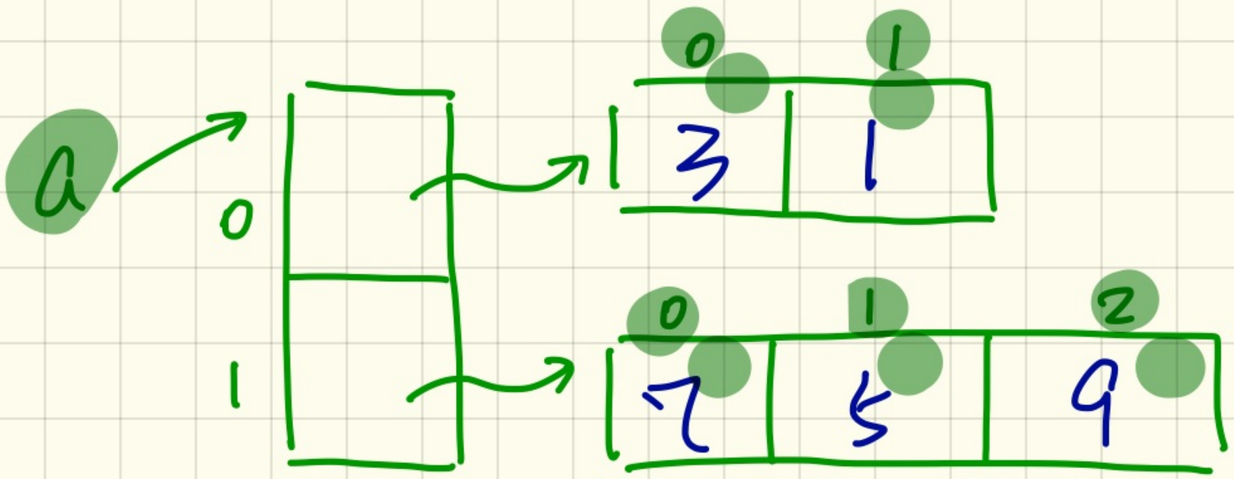
a[0].length 2

a[1].length 3



Example 2: Calculate Average

```
int total = 0;
int numOfElements = 0;
for(int row = 0; row < a.length; row ++) {
 for(int col = 0; col < a[row].length; col ++) {
   total += a[row][col];
   numOfElements ++;
  }
}
double average = ((double) total) / numOfElements;
System.out.println("Average is " + average);
```



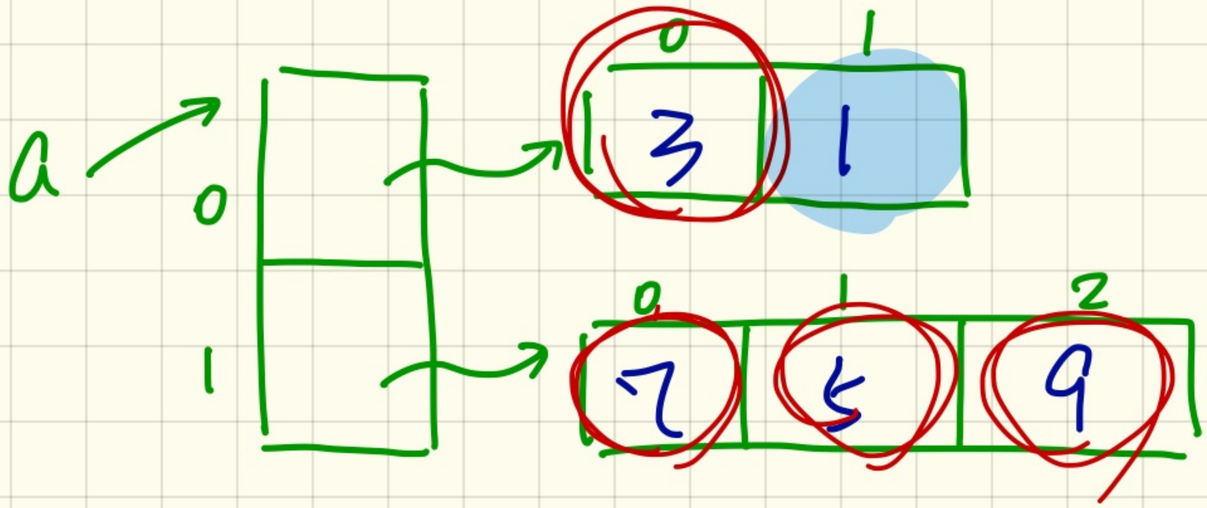
total += a[0][0]

total += a[0][1]

___ += a[1][0]

___ += a[1][1]

___ += a[1][2]



a[0][0] 3

max

min

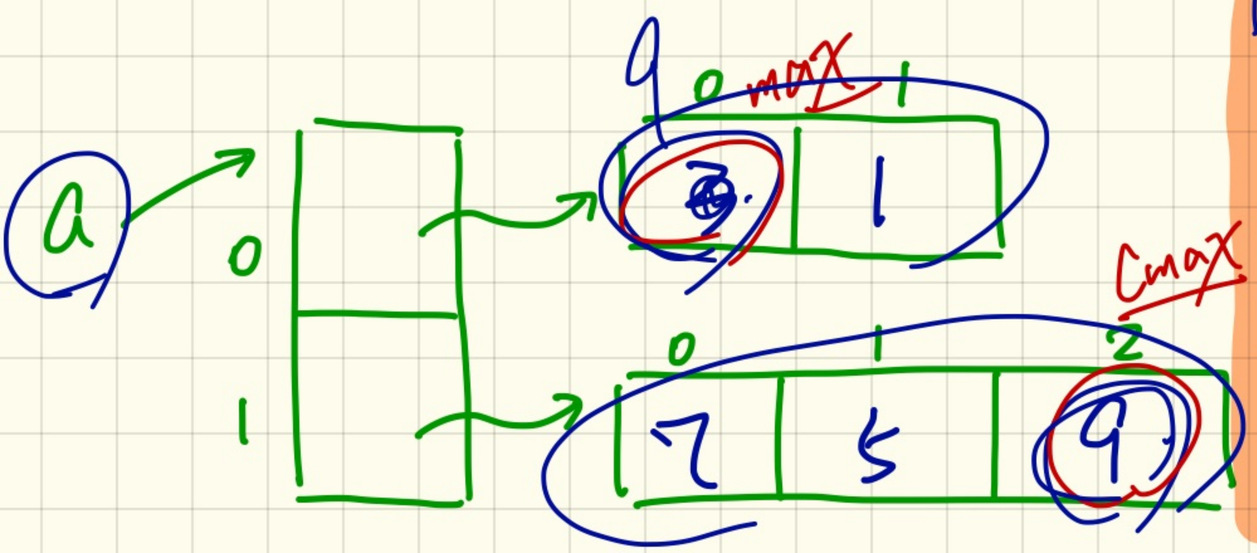
9

1

Example 3: Calculat Max and Min

```
int max = a[0][0];
int min = a[0][0];
for(int row = 0; row < a.length; row ++) {
  for(int col = 0; col < a[row].length; col ++) {
    if (a[row][col] > max) {
      max = a[row][col];
    }
    if (a[row][col] < min) {
      min = a[row][col];
    }
  }
}
System.out.println("Maximum is " + max);
System.out.println("Minimum is " + min);
```

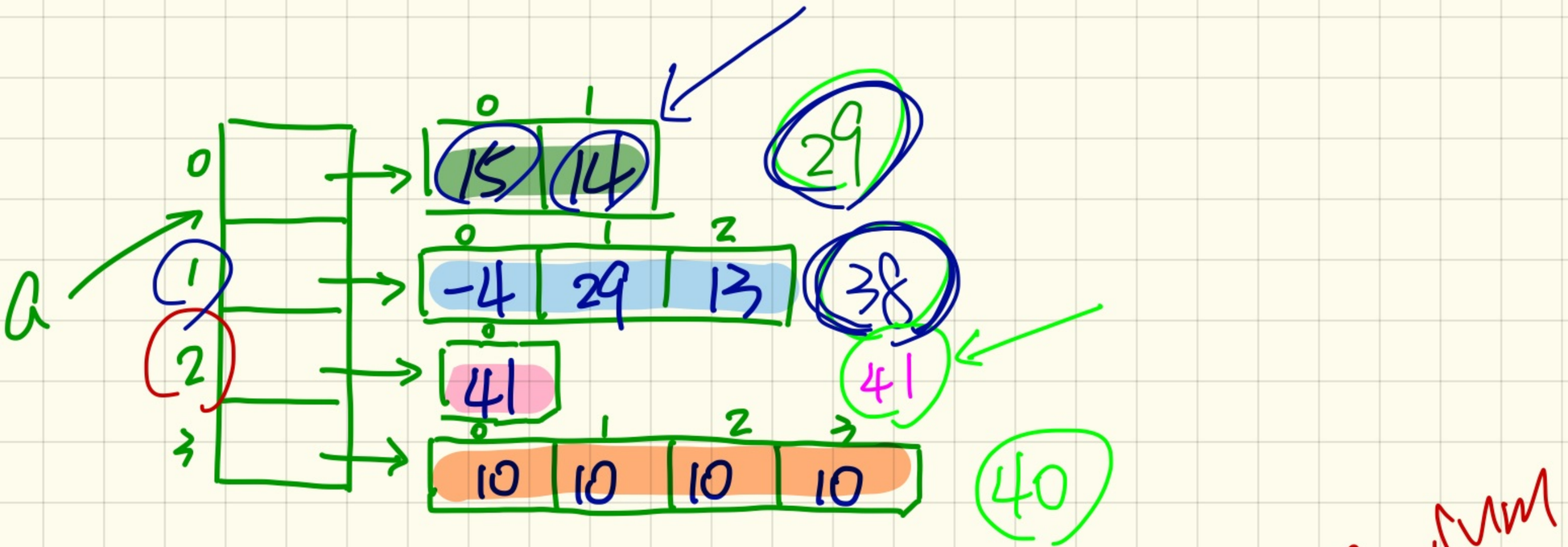
```
class Utilities {
  int maxOf(int[] a){
    :
  }

  int maxOf(int[][] a){
    :
  }
}
```




```
Assume: int maxOf(int[] a)
int max = maxOf(a[0]);
for(int r = 1; r < a.length; r++){
  int cmax = maxOf(a[r]);
  if(cmax > max) { max = cmax; }
}
```

int maxSum = ?;

int maxRow = ?;



Row 2 has max sum 41

Example 4: Calculat Row with Max Sum

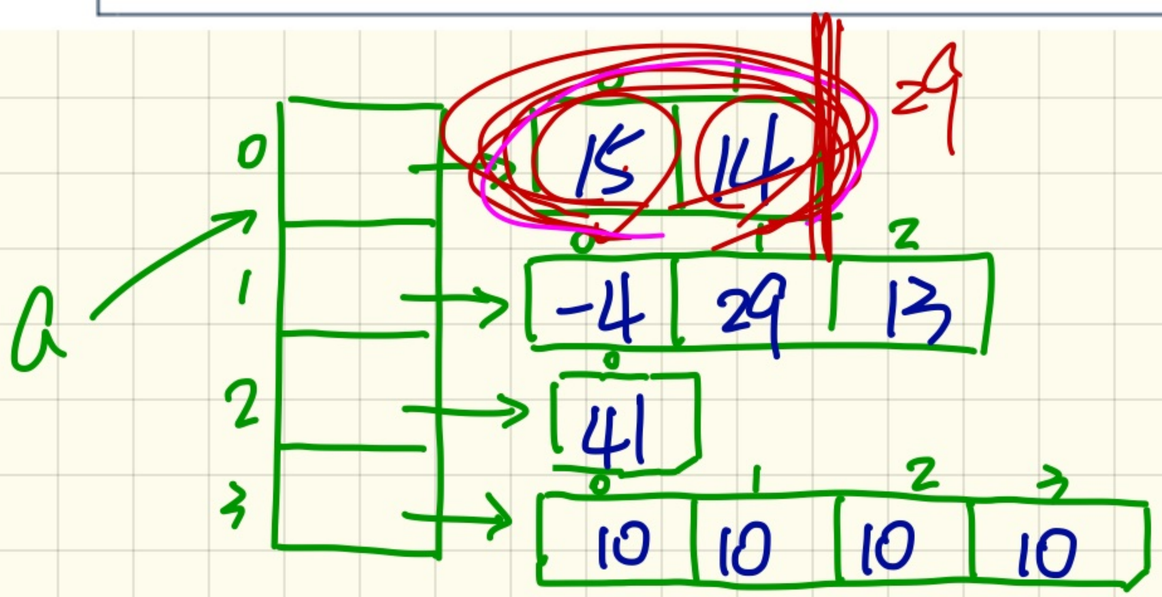
```
1  int maxRow = 0; int maxSum = 0;
2  for(int col=0; col < a[0].length; col ++){maxSum += a[0][col];}
3  for(int row = 1; row < a.length; row ++) {
4    int sum = 0;
5    for(int col = 0; col < a[row].length; col ++) {
6      sum += a[row][col];
7    }
8    if (sum > maxSum) {
9      maxRow = row;
10     maxSum = sum;
11   }
12 }
13 System.out.print("Row at index " + maxRow);
14 System.out.println(" has the maximum sum " + maxSum);
```

treating Row 0 to be containing the max sum.

sum for row "row"

*Q: Move L4 to between L2 and L3?

No! sum will be summing up all elements of 2D array



	maxRow	maxSum
L1	0	0
L2	0	29

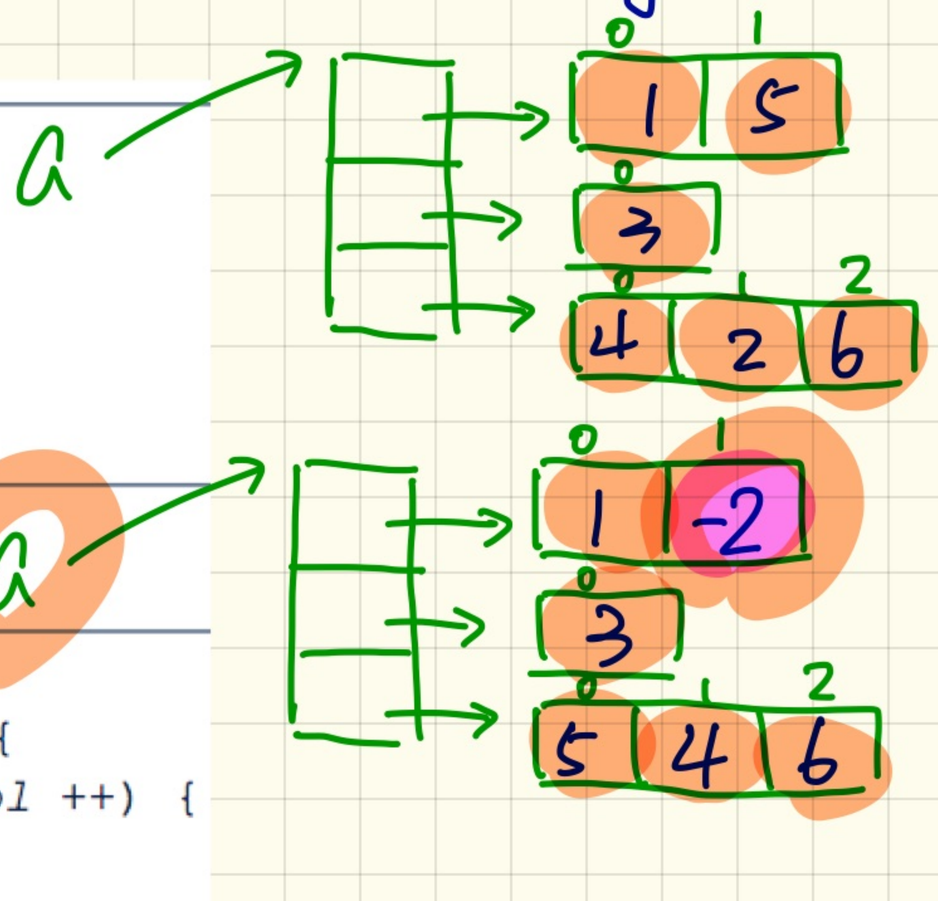
Example 5: all positive?

```
boolean allPos = true;
for(int row = 0; row < a.length; row ++) {
  for(int col = 0; col < a[row].length; col ++) {
   allPos = allPos && a[row][col] > 0;
  } }
if (allPos) { /* print */ } else { /* print */ }
```

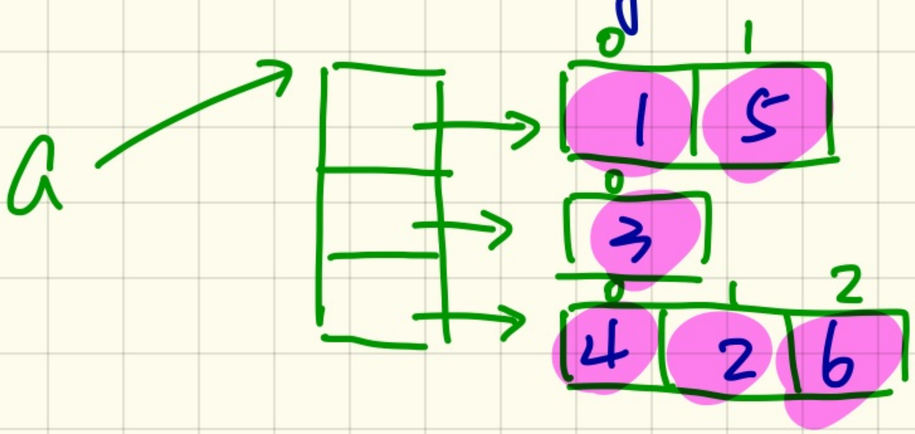
Alternatively (with *early exit*):

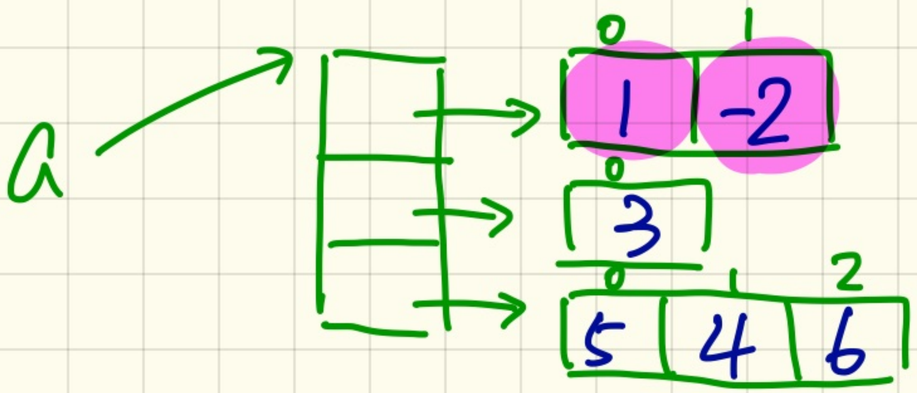
```
boolean allPos = true;
for(int row = 0; allPos && row < a.length; row ++) {
  for(int col = 0; allPos && col < a[row].length; col ++) {
   allPos = a[row][col] > 0;
  } }
if (allPos) { /* print */ } else { /* print */ }
```

Version of No Early Exit



Version with Early Exit





Monday March 26

Lecture 11

Schedule After Strike

Week #1	Week #2	Week #3	Revised Exam Date
LAB TEST 2 (10%)	Grading of Lab 5 & 6 (5% 3%)	LAB TEST 3 (10%)	EXAM (52 %)

END OF STRIKE

What's Expected during Strike

- LAB TEST 2 prep.
- LAB 5 Background Study
  - ~ PDF
  - ~ Tutorial Videos
- LAB 5 & 6
- Lectures

To Be Released

- LAB TEST 3 guide
- Exam Guide

To Study "everything"

1. slides, code examples, iPad notes, exercises

2. Tutorial videos (iPad notes). → debugger.
   - Spinner X
   - BMI calculator (MVC, OO, methods)
   - Lab 3: share object among multiple Listeners (v1 vs. v2)
   - Lab 4 ~~Lab 5~~
     reference att. array ref. attributes.

3. PDF document: Point vs. PointTester (Lab 5)

   PDF : == vs. equals.

Format

~~1.~~ Coding Questions ($\leq 15\%$)

- Implement manually a method (e.g. matrix ops)

50 ~ 100

2. Multiple Choices ($\geq 85\%$).

- Definitions (e.g. aliasing, data type, class, object)
- Compilation (syntax v.s. type error).
- Tracing "correct" code
  - ~ nested if ~ nested for
  - ~ 1D 2D arrays
  - ~ aliasing, ArrayList HashTable
- Tracing "incorrect" code
  (NullPointerEx., IndexOutOfBoundsEx.)

When is the exam?

Wait for moodle announ.

Tutoring Hours

Tuesdays
Wednesdays
Thursdays

} 14:20 ~ 15:20 Prism Lab (LAS 1006)

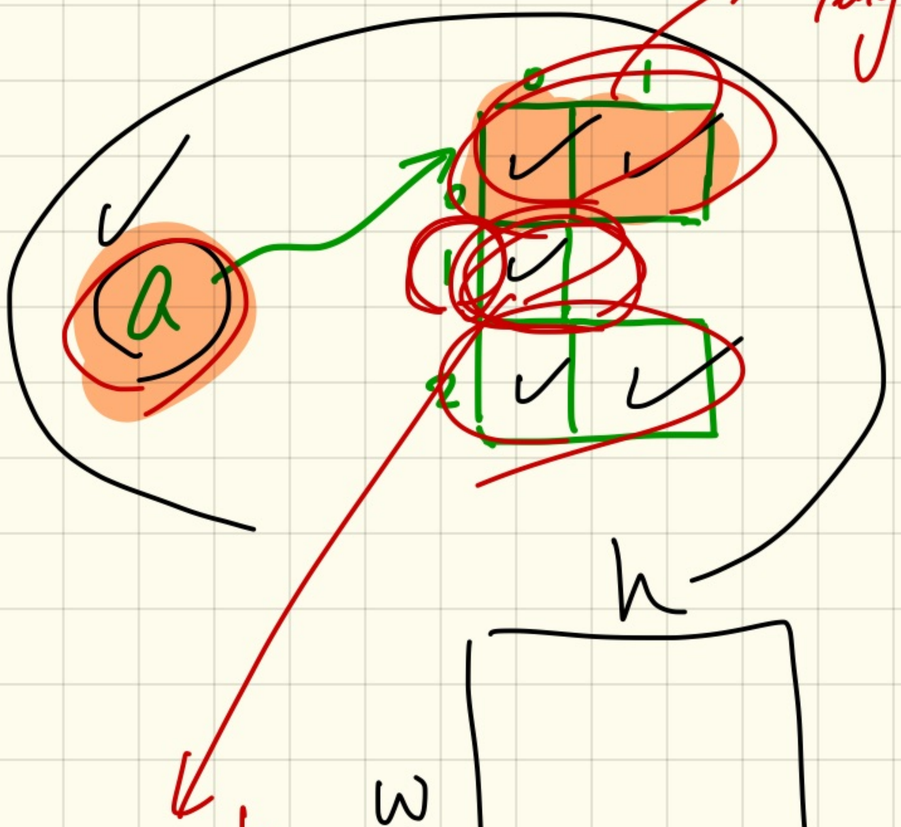
Or, Zoom appointments!

WSC Drop-In Sessions (no TA supervision)

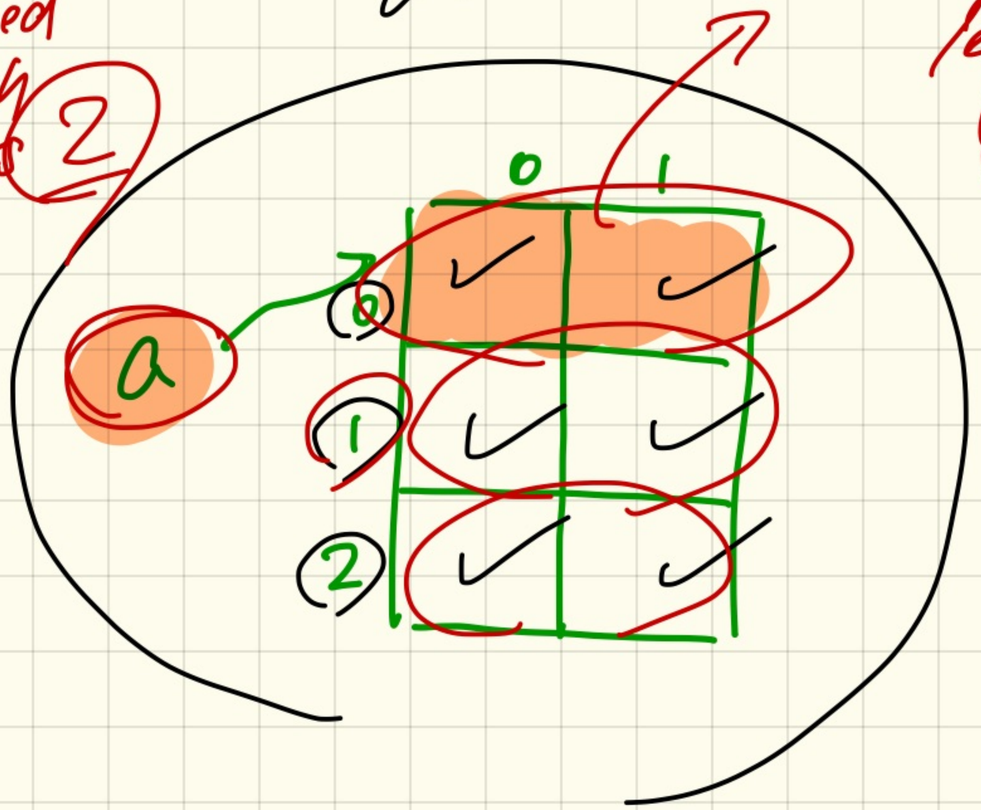
Tuesdays	14:00 ~ 16:00
Wednesdays	17:00 ~ 19:00
Fridays	11:00 ~ 13:00

isRectangle?

False



✓ true



Example 6: isRectangle?

a = { }

assume a is not empty

```
if(a.length == 0) { /* empty array can't be a rectangle */ }
else { /* a.length > 0 */
  int assumedLength = a[0].length;
  boolean isRectangle = true;
  for(int row = 0; row < a.length; row ++) {
    isRectangle =
      isRectangle && a[row].length == assumedLength;
  }
  if (isRectangle) { /* print */ } else { /* print */ }
}
```

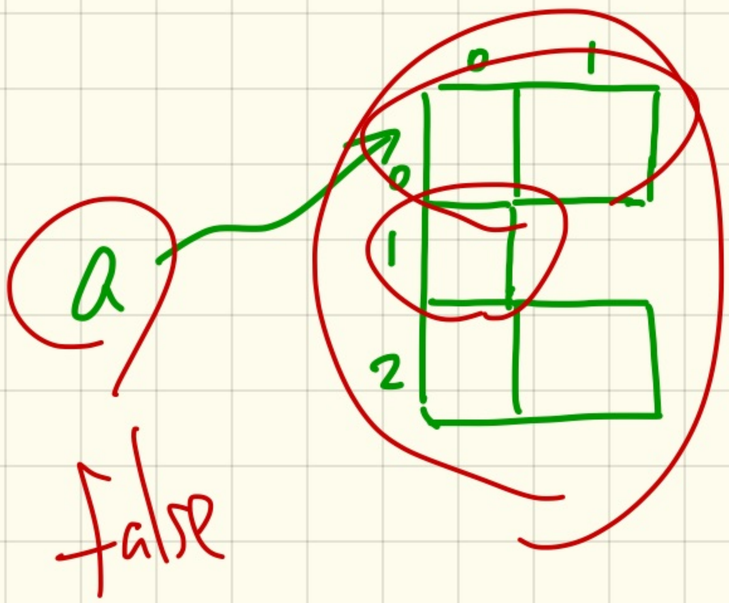
isRectangle && row < a.length.

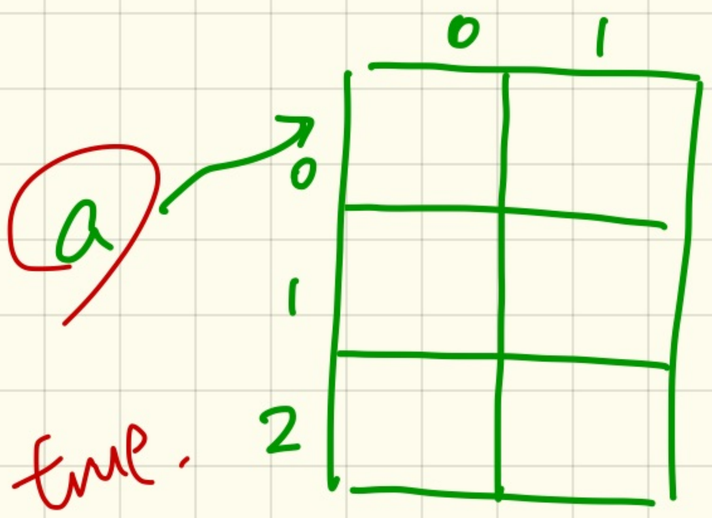
replace by 1.

isRectangle && (row == 0)

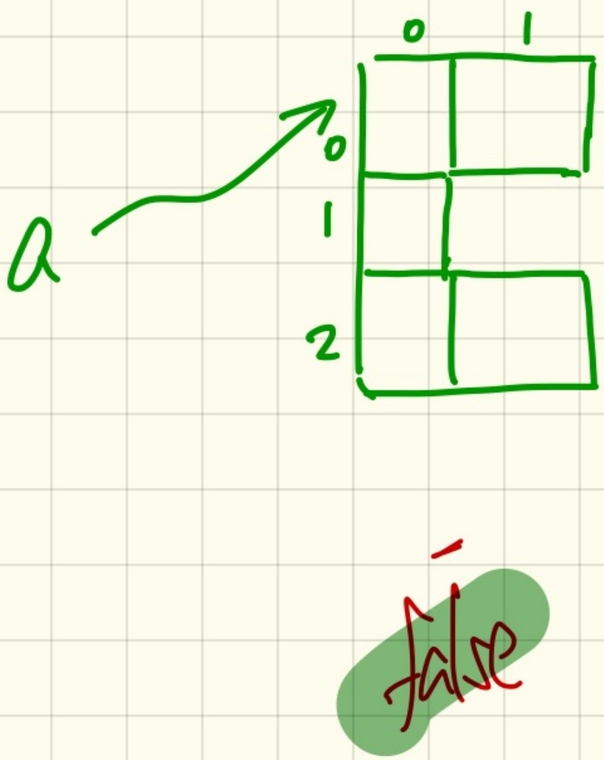
1st iteration:

isRectangle = isRectangle && a[0].length == assumedLength;

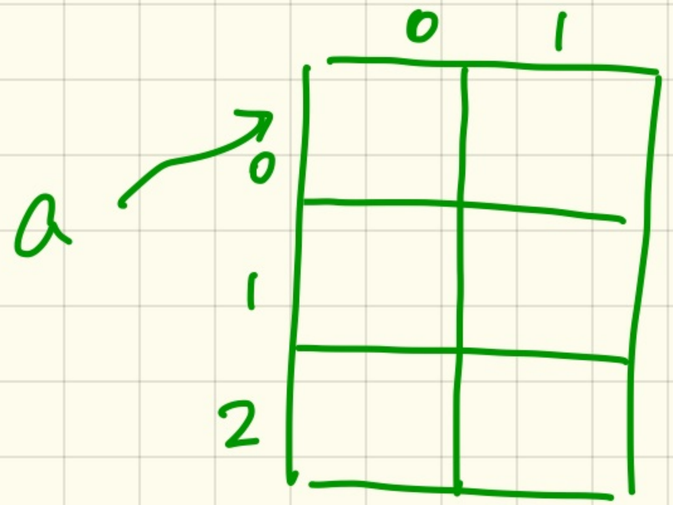




isSquare?

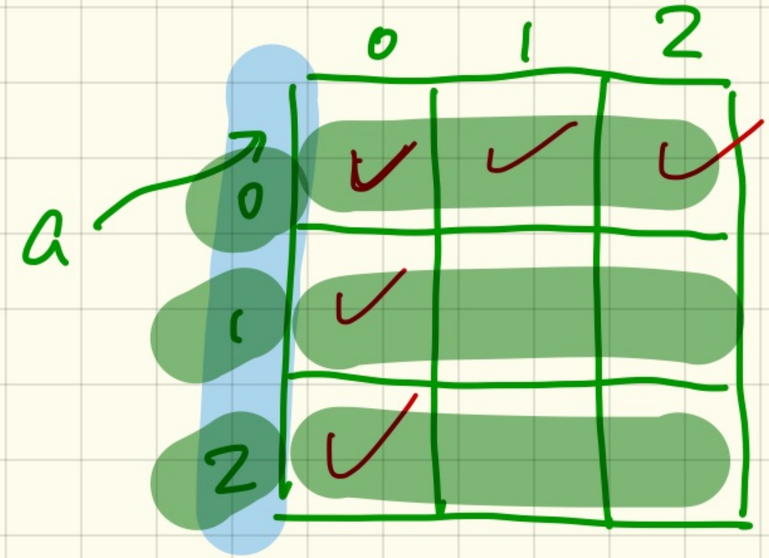


false



false

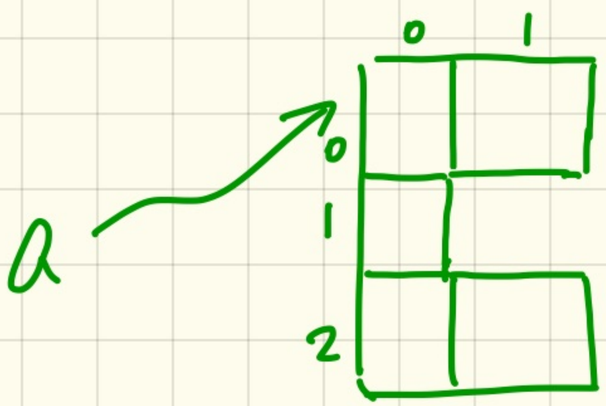
a.length == 3

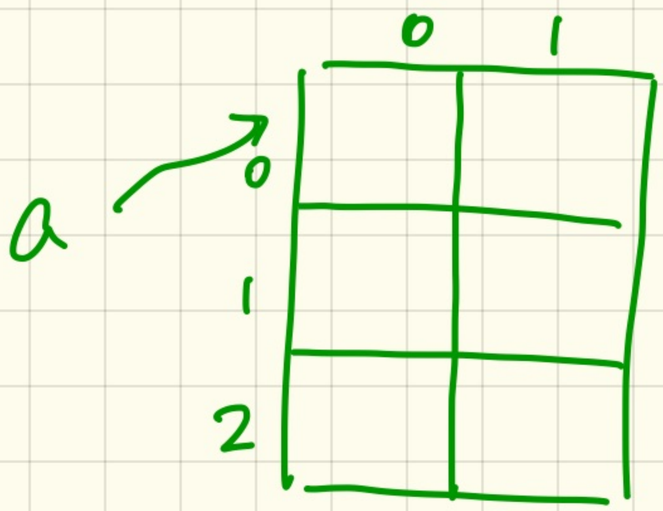


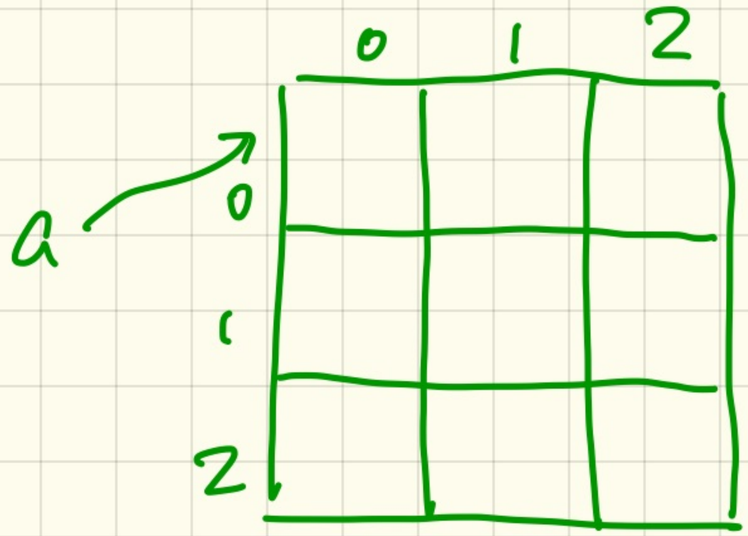
true

Example 7: isSquare

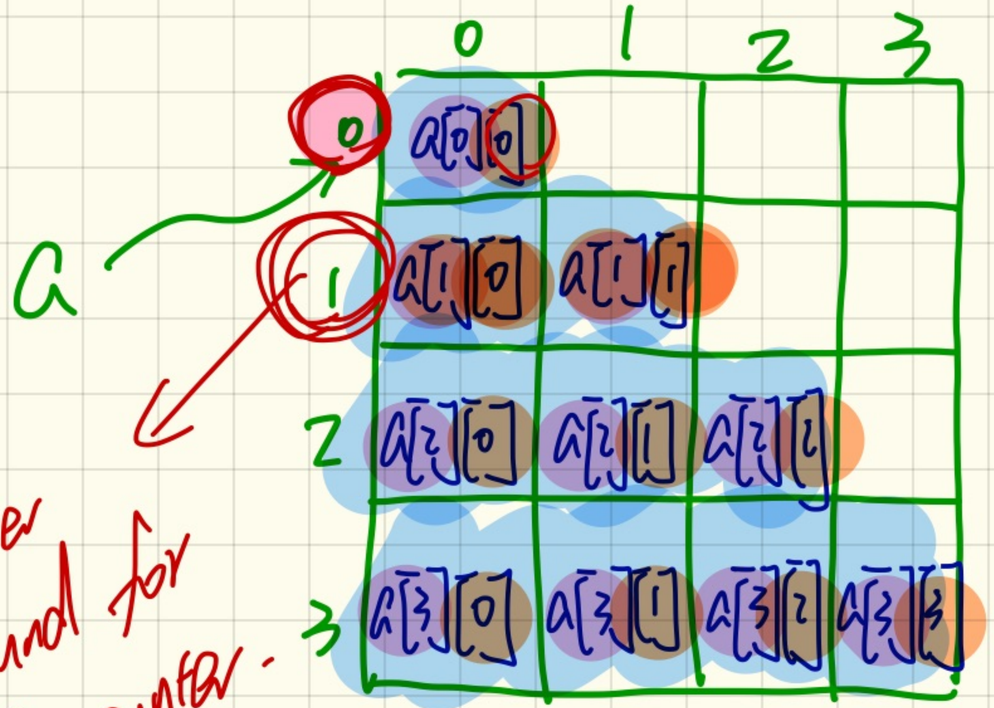
```
if(a.length == 0) { /* empty array can't be a square */ }
else { /* a.length > 0 */
  int assumedLength = a.length;
  boolean isSquare = a[0].length == assumedLength;
  for(int row = 0; row < a.length; row ++) {
    isSquare =
      isSquare && a[row].length == assumedLength;
  }
  if (isSquare) { /* print */ } else { /* print */ }
}
```

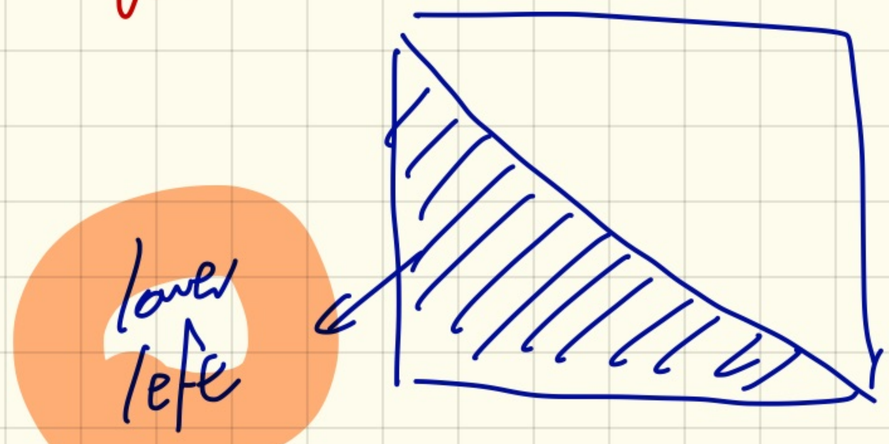




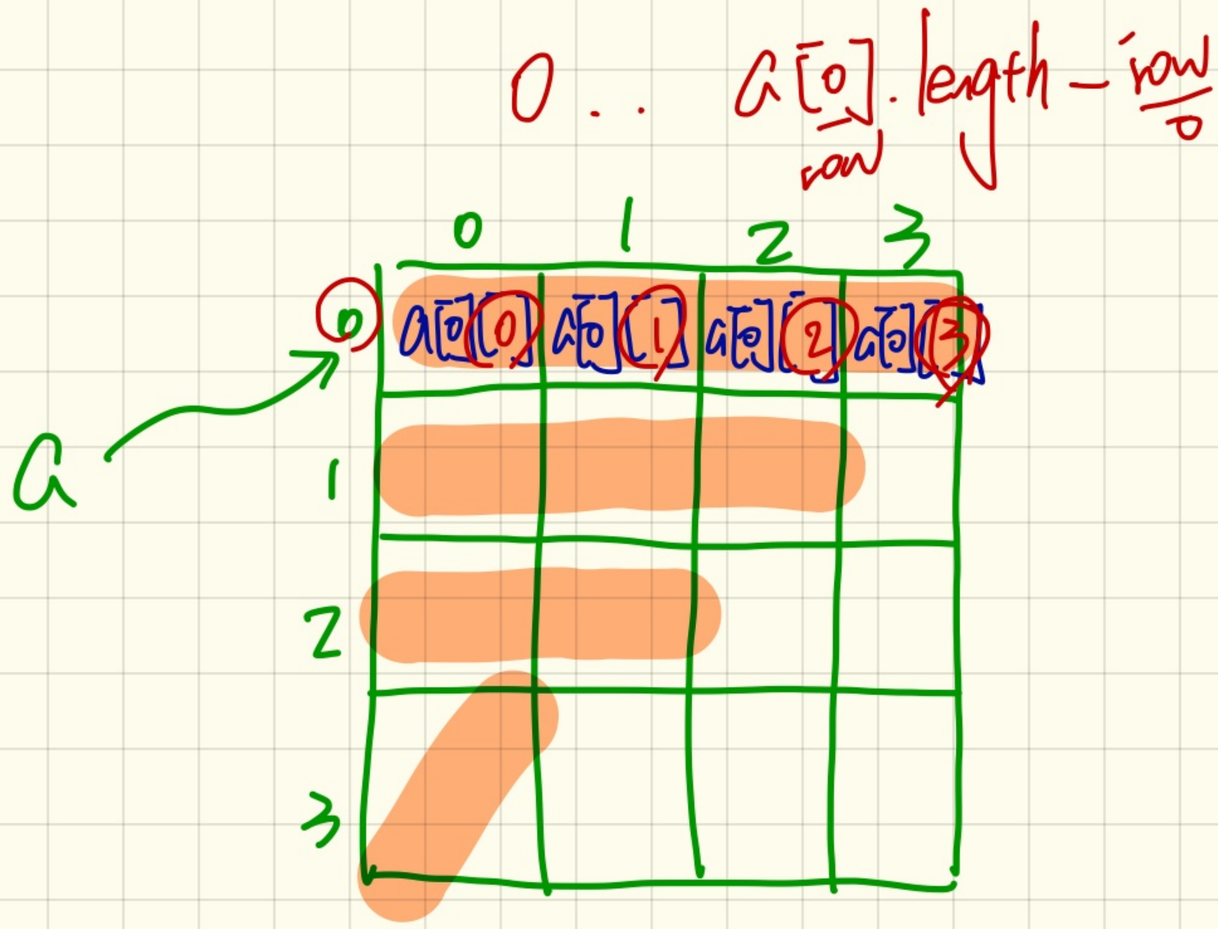


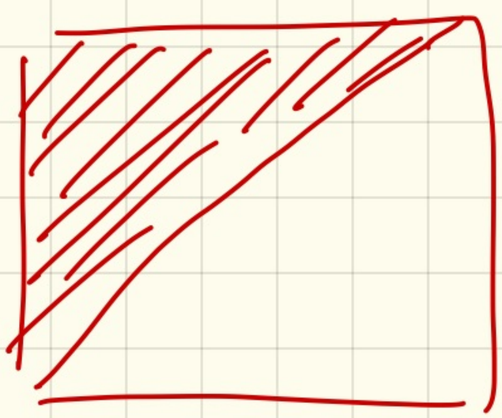
print lower left?





print upper left?

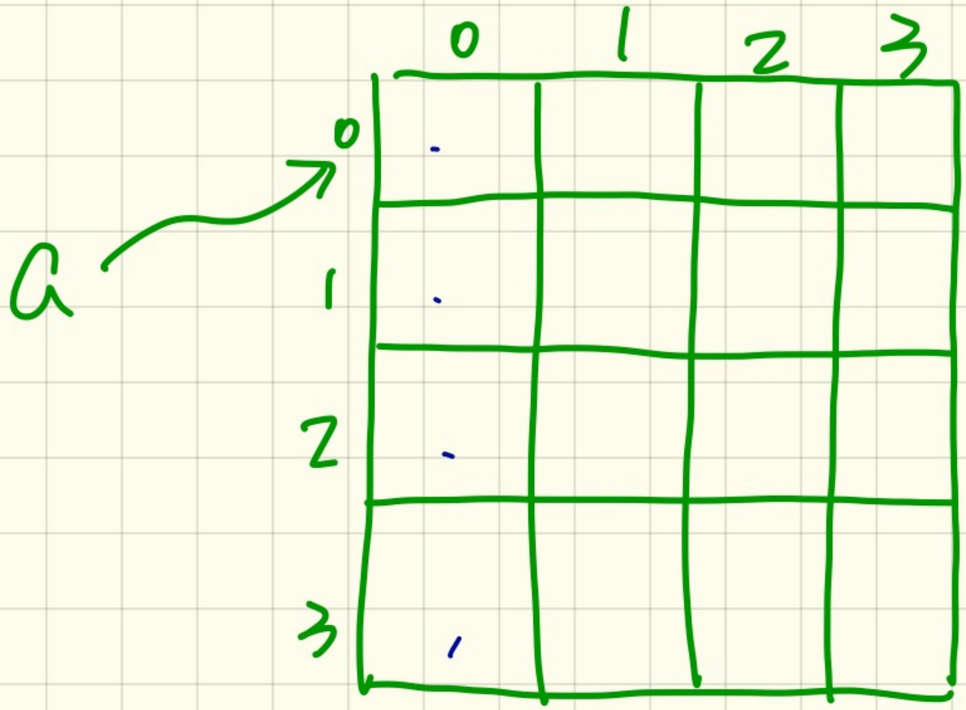




Example 8: Lower Left

4

```
for(int row = 0; row < a.length; row ++) {
  for(int col = 0; col <= row; col ++) {
    System.out.print(a[row][col]); }
  System.out.println(); }
```



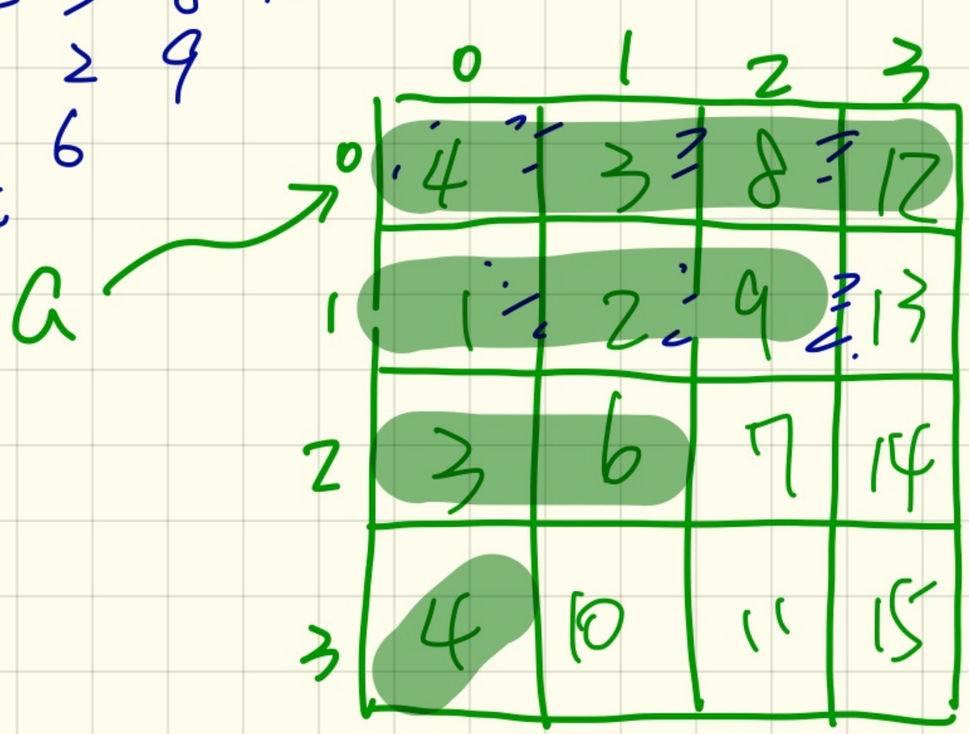
row	col
0	0
1	0
	1
2	0
	1
	2
3	0
	1
	2
	3

Example 9: Upper Left

```
for(int row = 0; row < a.length; row ++) {
  for(int col = 0; col < a[row].length - row; col ++) {
    System.out.print(a[row][col]); }
  System.out.println(); }
```

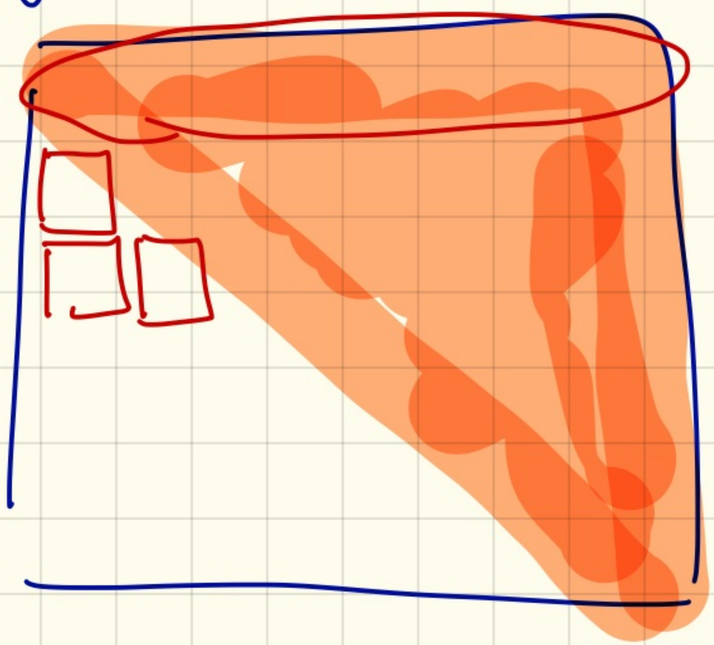
Output

4 3 8 12
1 2 9
3 6
4

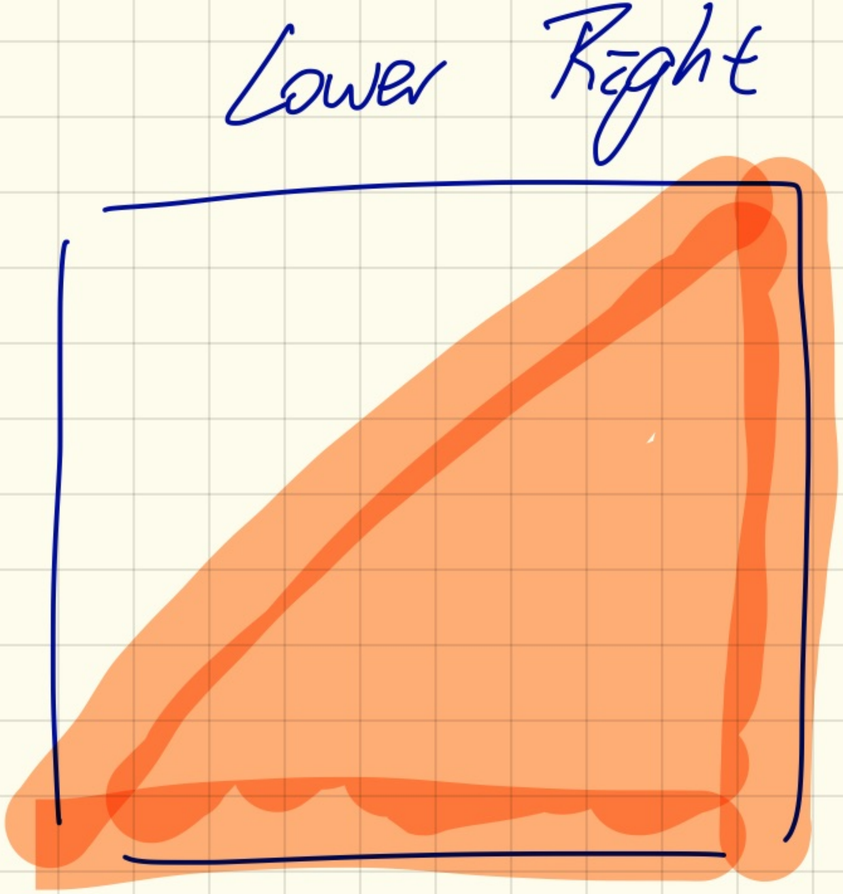


row	a[row].length - row	col
0	4-0	0 1 2 3
1	4-1	0 1 2
2	4-2	0 1
3	4-3	0

Upper Right



Lower Right



Selections: Missing Brackets

```
final double PI = 3.1415926;
Scanner input = new Scanner(System.in);
double radius = input.nextDouble();
double area = 0;
if (radius >= 0)
  area = radius * radius * PI;
System.out.println("Area is " + area);
```

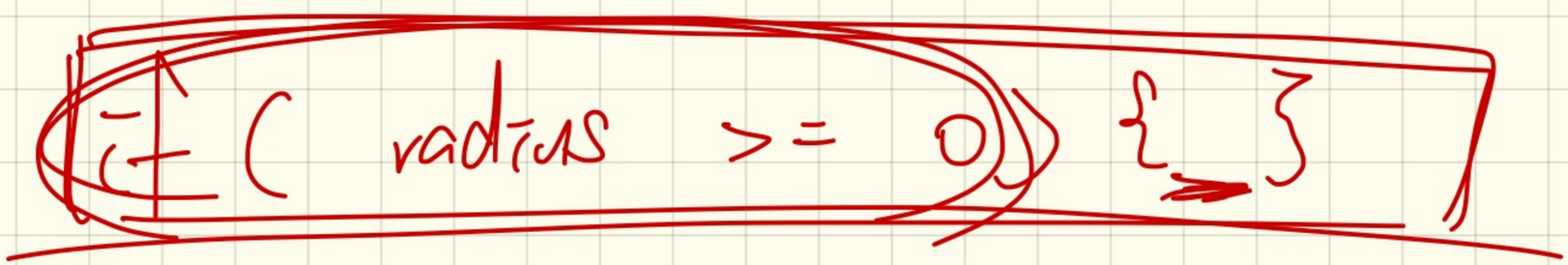
=

if (radius >= 0) {
  area = - -
}
println (Area - -)

Selections: Misplaced Semicolon

3 * 3 * PI vs. -2 → -2 * -2 * PI

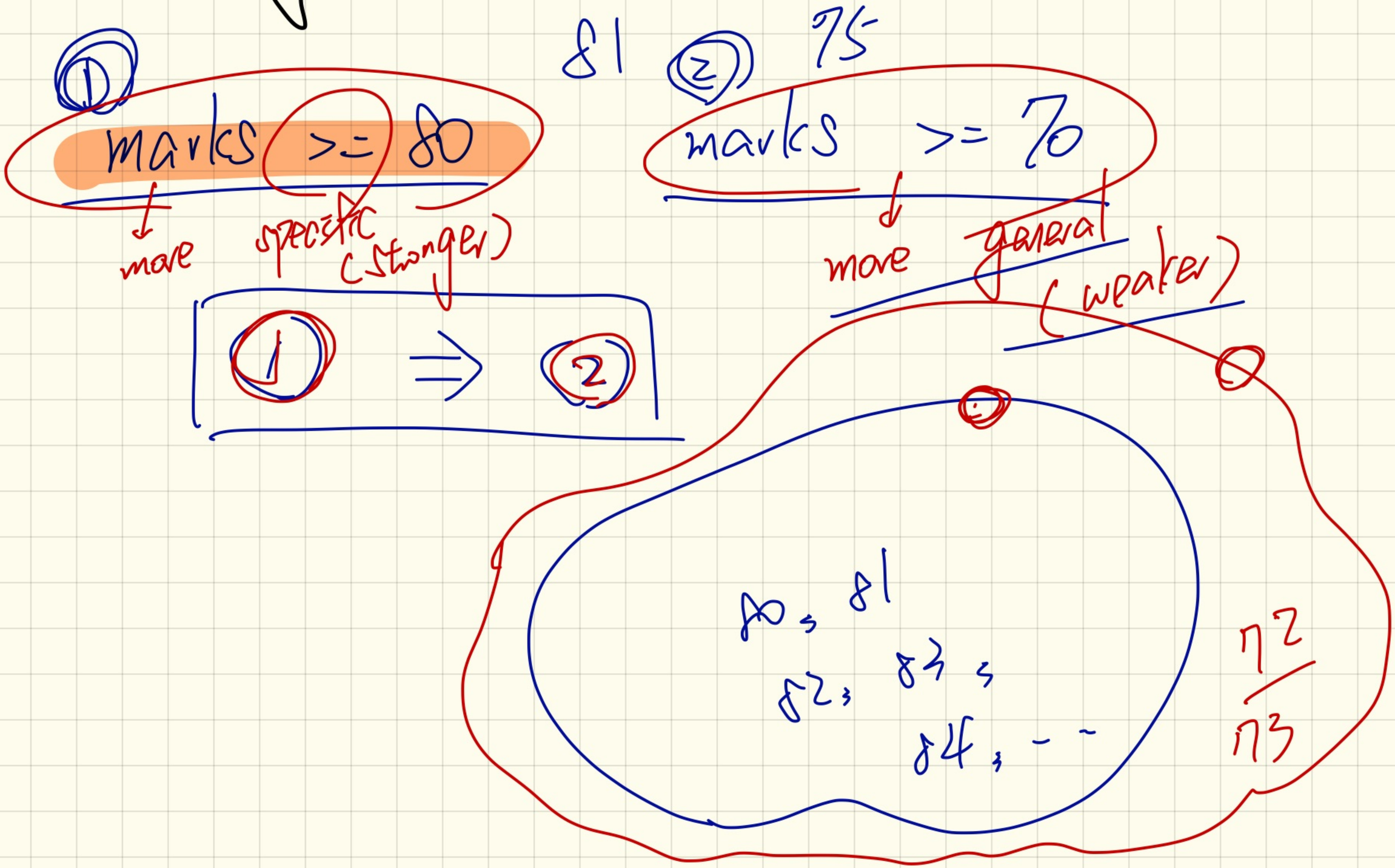
```
if (radius >= 0);{
  area = radius * radius * PI;
  System.out.println("Area is " + area);
}
```



area = . . . ;

println(area)

Overlapping Boolean Conditions.



Common Errors of Selections :

Independent if-statements with overlapping conditions.

```
if (marks >= 80) {
  System.out.println("A");
}
if (marks >= 70) {
  System.out.println("B");
}
if (marks >= 60) {
  System.out.println("C");
}
else {
  System.out.println("F");
}
/* Consider marks = 84 */
```

marks : 89

A

B

C

Common Errors of Selections:

If-conditions sorted the wrong way

```
if (gpa >= 2.5) {
  graduateWith = "Pass";
}
else if (gpa >= 3.5) {
  graduateWith = "Credit";
}
else if (gpa >= 4) {
  graduateWith = "Distinction";
}
else if (gpa >= 4.5) {
  graduateWith = "High Distinction" ;
}
```

Common Errors of Selections:

Ambiguous else-statement

```
if (x >= 0)
    if (x > 100) {
        System.out.println("x is larger than 100");
    }
else {
    System.out.println("x is negative");
}
```

What if x is 20

```
if (x >= 0) {
    if (x > 100) {
    }
} else {
}
```

parsing rule

Fix

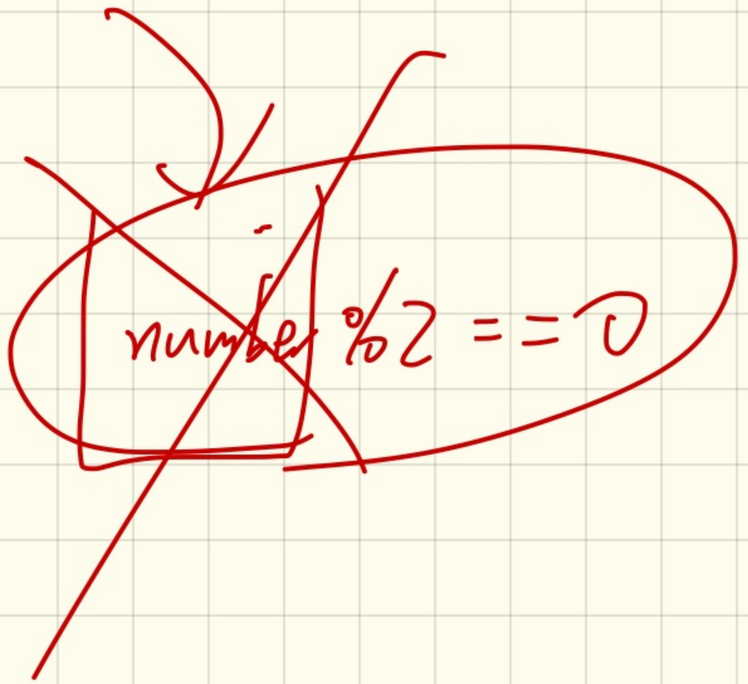
```
if (x >= 0) {
    if (x > 100) {
    }
} else {
    println( - - neg. );
}
```

Common Pitfalls of Selections:

Redundant if

V1

```
boolean isEven;
if (number % 2 == 0) {
  isEven = true;
}
else {
  isEven = false;
}
```

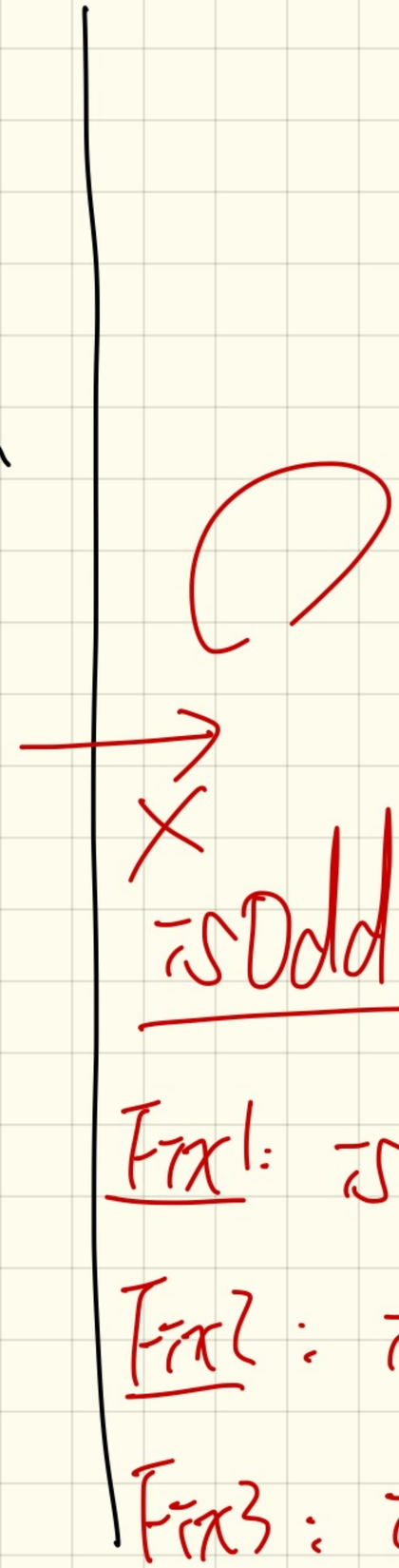


V2

```
isEven = number % 2 == 0;
```

VL

```
boolean
  isOdd ;
if( number % 2 == 0){
      isOdd = false;
}
else {
      isOdd = true ;
}
```



```
isOdd = number % 2 == 0;
```

Fix1: `isOdd = num % 2 == 1`

Fix2: `isOdd = !(num % 2 == 0)`

Fix3: `isOdd = num % 2 != 0`

Primitive

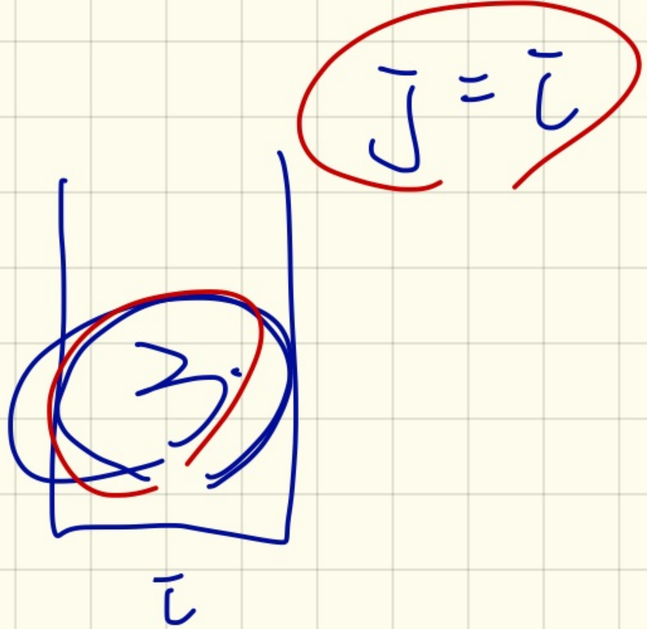
int
short
long
float
double
boolean
char

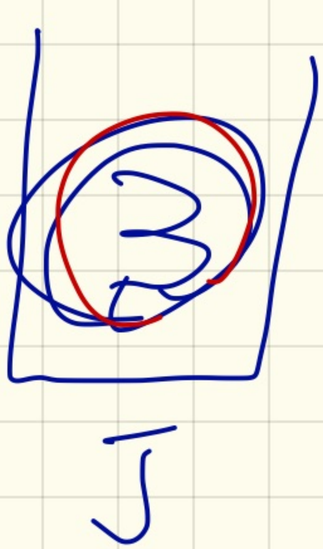
Reference

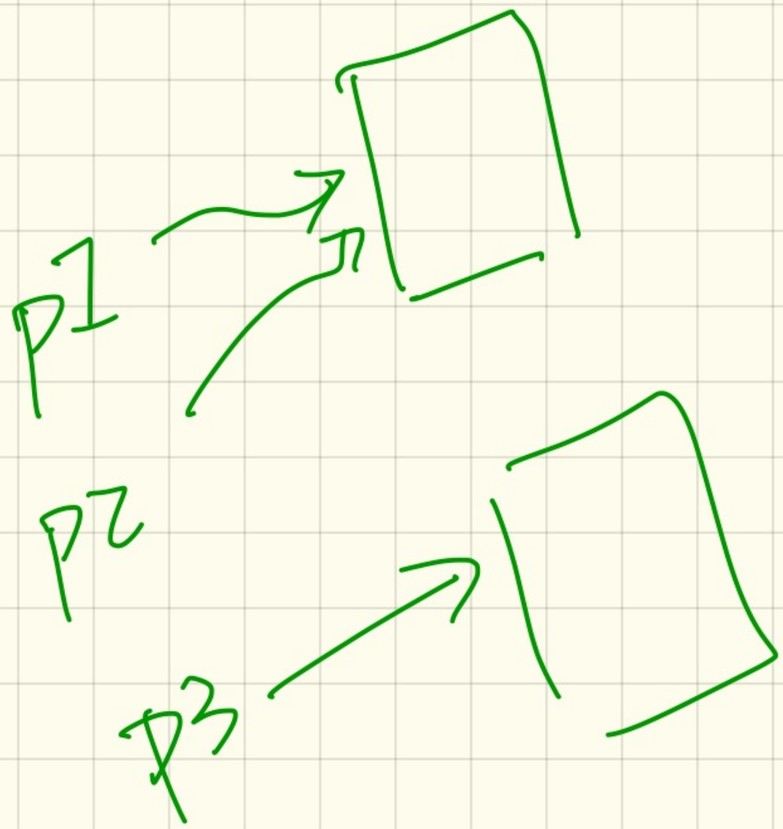
String

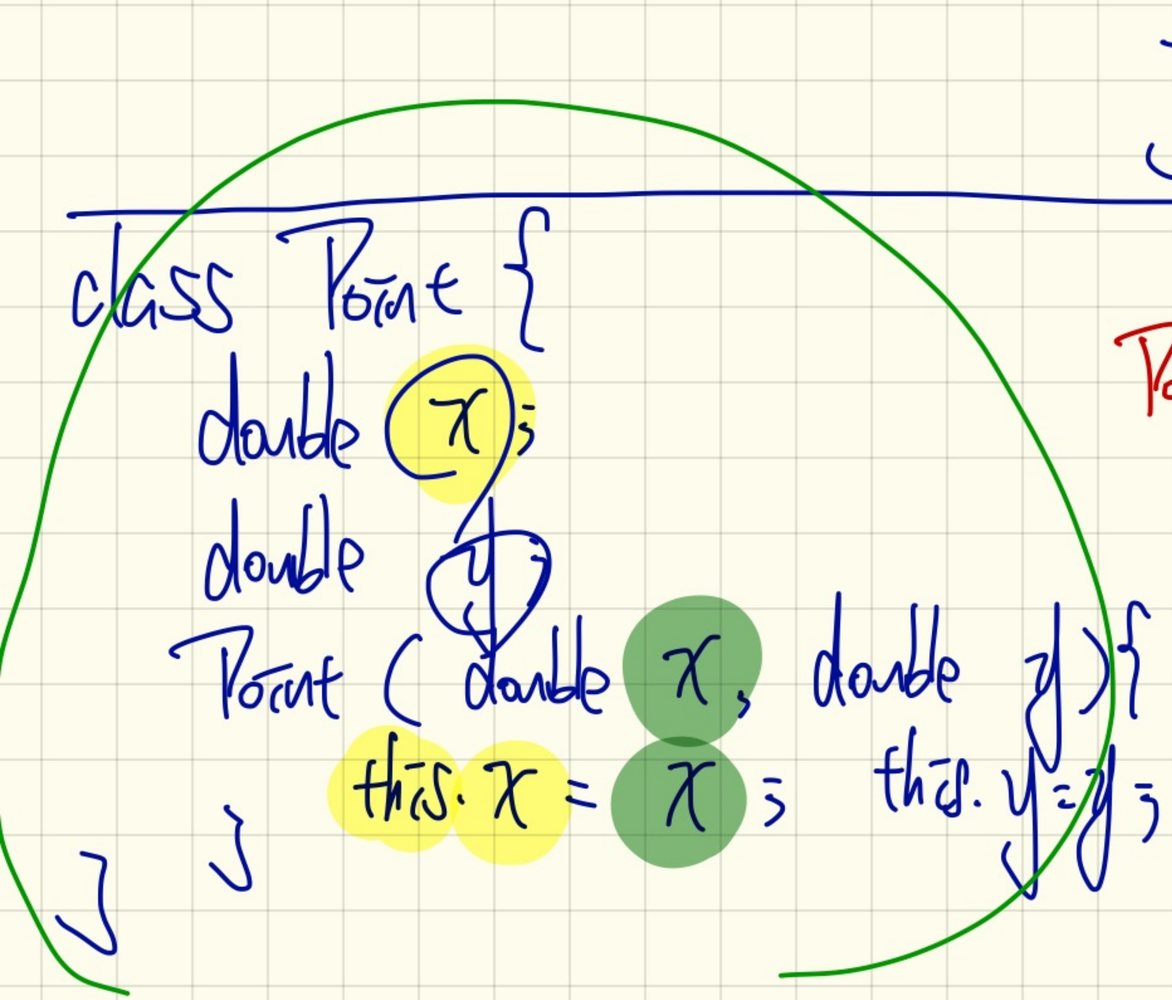
Point

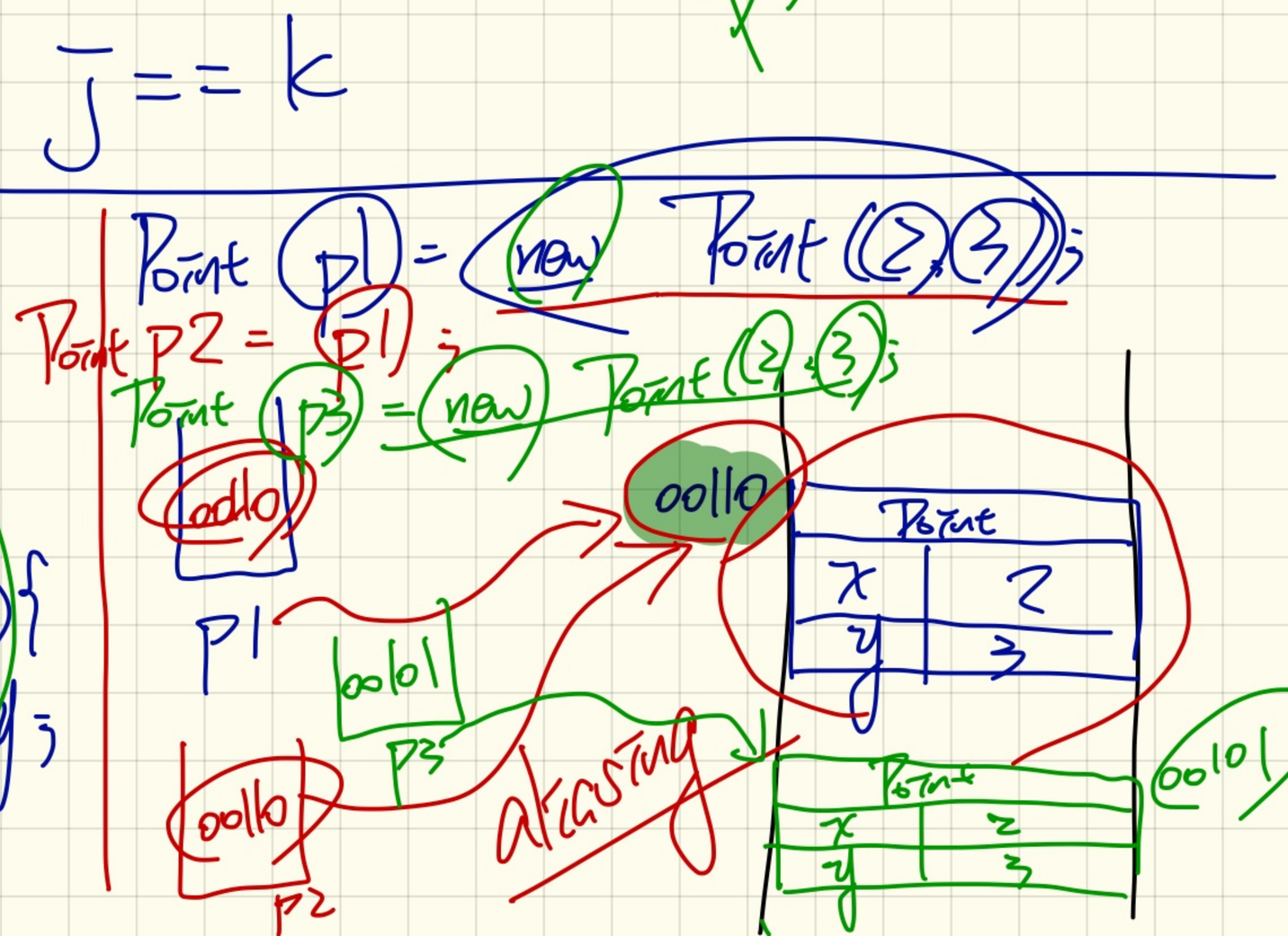
Person


i == j
i
j


j
k


p1
p2
p3


class Point {
double x;
double y;
Point (double x, double y){
this.x = x; this.y = y;
}
}

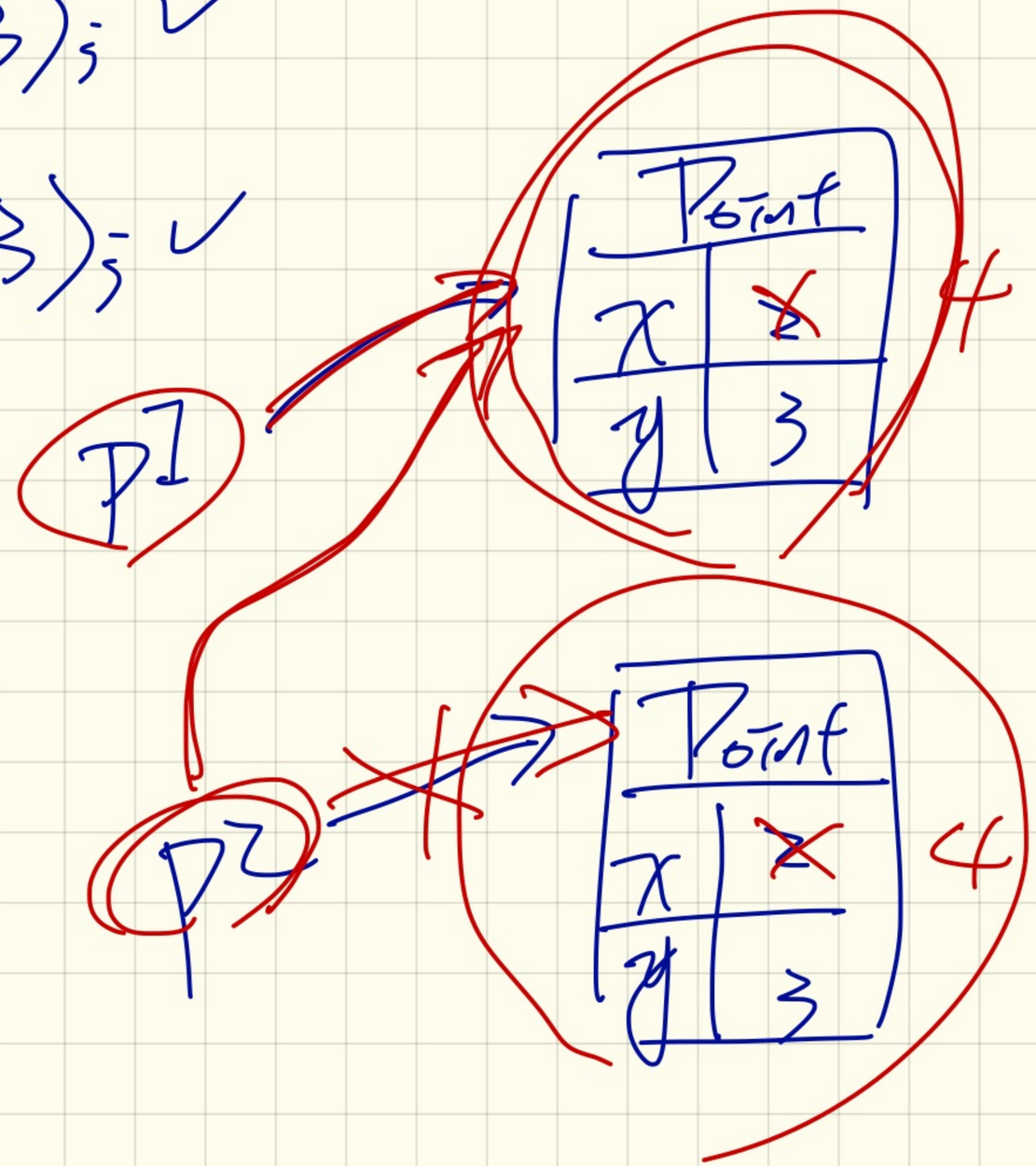

Point p1 = new Point(2,3);
Point p2 = p1;
Point p3 = new Point(2,3);
00110
00101
00110
p1
p2
p3
Point
x
z
y
3
allocating
00101

```
Point p1 = new Point(2, 3);
Point p2 = new Point(2, 3);
println( p1 == p2);
println( p1.x); /* 2 */
```

```
p2.x = 4;
println(p1.x); /* 2 */
```

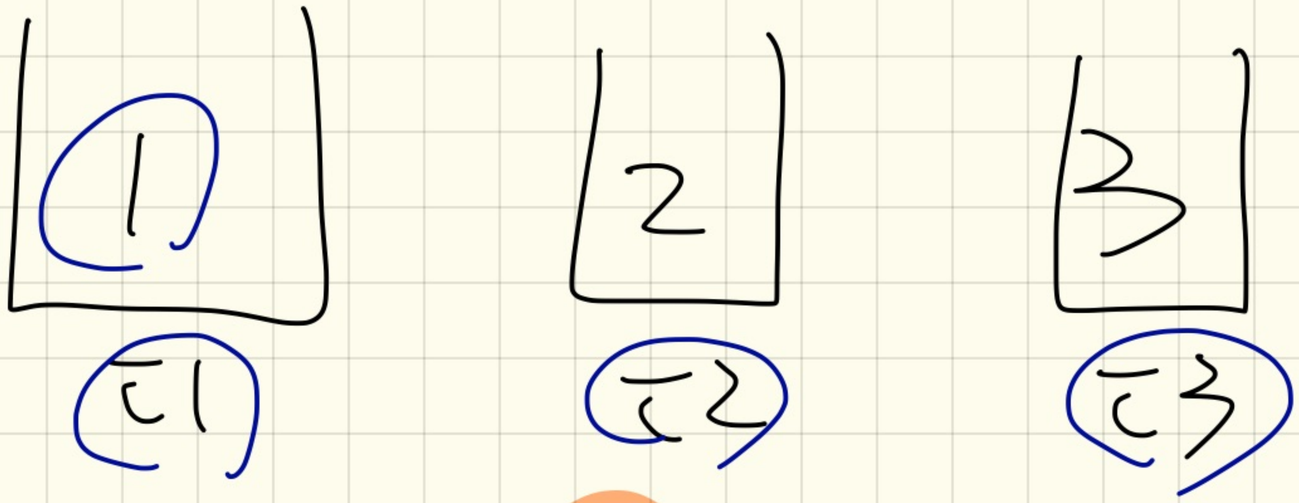
```
p2 = p1;
```

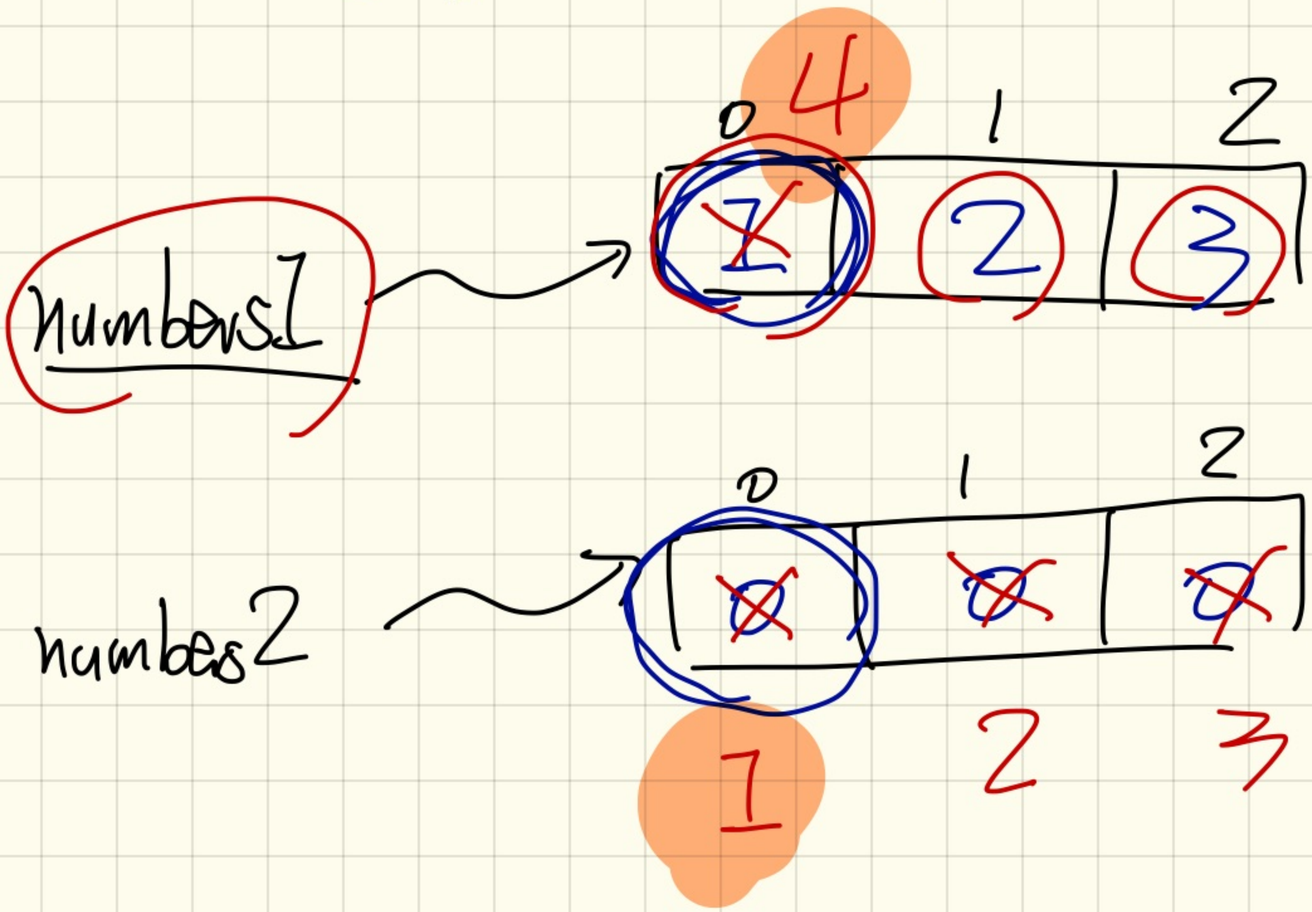
```
p2.x = 4;
println(p1.x); /* 4 */
```



Monday April 2

Lecture 12





1st iteration

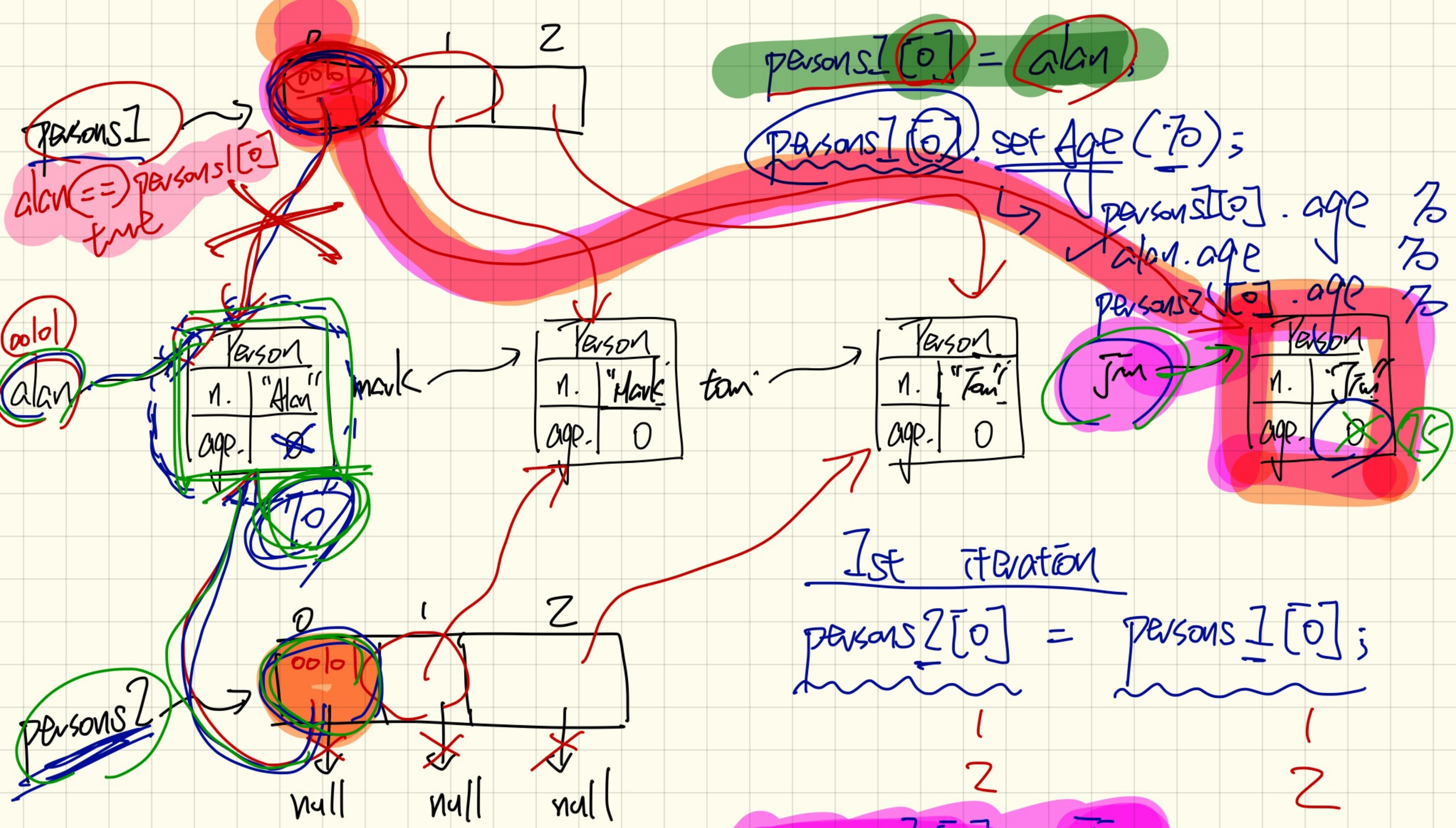
numbers2[0] = numbers1[0];

1

2

1

2


persons1[0] = alan;
persons1[0].setAge(70);
persons1
alan == persons1[0]
true
0 1 2
0010
alan
Person
n. "Alan"
age. 0
mark
Person
n. "Mark"
age. 0
tom
Person
n. "Tom"
age. 0
persons1[0].age 70
alan.age 70
persons2[0].age 70
jim
Person
n. "Jim"
age. 0 75
persons2
0 1 2
0010
null null null
1st iteration
persons2[0] = persons1[0];
1 1
2 2
persons1[0] = jim;
persons1[0] == jim
persons1[0].setAge(75);
alan == persons1[0] true
persons1[0] == persons2[0] true

static (modifier)

int i;

local variable
↳ to a specific context object

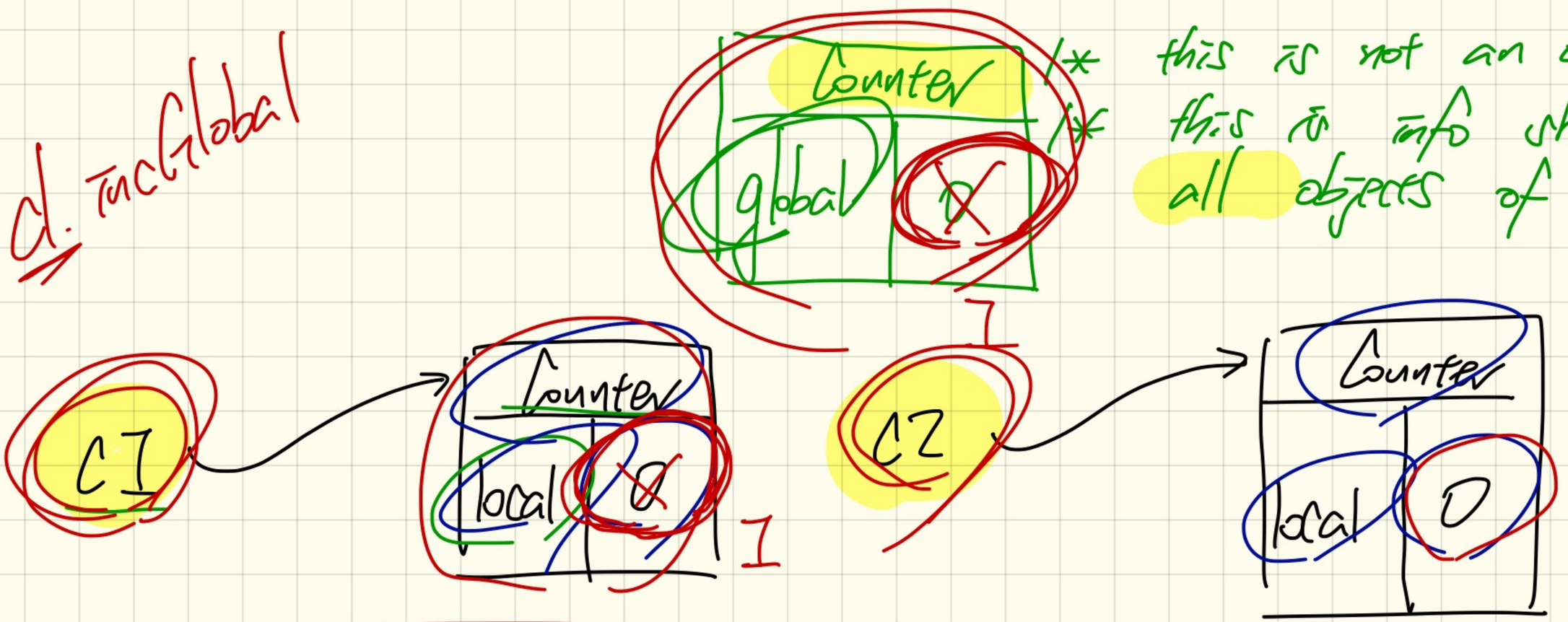
non-static variable

static int j;

static variable

global variable
↳ to all context objects.

cl. incGlobal



C1. global	✓	C1. local	✓
C2. global	✓	C2. local	✓
Counter. global	✓	Counter. local	✗

Parameters vs. Arguments

Developer / Supplier

```
class MyClass {
    int c;
    int getValue() {
        ...
    }
    void setValue(int i) {
        c = i;
    }
}
```

signature

input / parameter (variable)

Math.abs(-2.5);

abs ↓ static method

User / Client

```
class MyClassUser {
    main(...) {
        MyClass o1 = new MyClass();
        o1.setValue(3);
    }
}
```

argument (value).

Math m = new Math();
m.abs(-2.5);

overloading

```
abs ( double a )
abs ( float  a )
abs ( int    a )
abs ( long   a )
```

When multiple methods have the same name, then:

1. they have different numbers of parameters
2. same # of parameters, but different types.

```
class SMS {

    void addStudent(Student s) { -- }

    void addStudent(String name, int marks) { -- }

    void addStudent(String s);

}
```

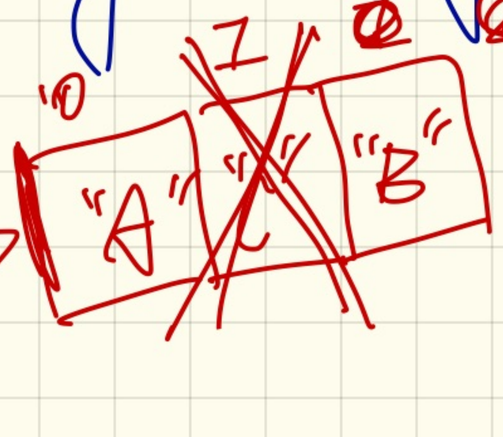
1 parameter

2 parameters.

Tester of ArrayList

```java
import java.util.ArrayList;

public class ArrayListTester {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<String>();
        System.out.println("List size: " + list.size());
        System.out.println("A exists: " + list.contains("A"));
        System.out.println("Index of A: " + list.indexOf("A"));
        list.add("A");
        list.add("B");
        System.out.println("A exists: " + list.contains("A"));
        System.out.println("B exists: " + list.contains("B"));
        System.out.println("C exists: " + list.contains("C"));
        System.out.println("Index of A: " + list.indexOf("A"));
        System.out.println("Index of B: " + list.indexOf("B"));
        System.out.println("Index of C: " + list.indexOf("C"));
        list.add(1, "C");
        System.out.println("A exists: " + list.contains("A"));
        System.out.println("B exists: " + list.contains("B"));
        System.out.println("C exists: " + list.contains("C"));
        System.out.println("Index of A: " + list.indexOf("A"));
        System.out.println("Index of B: " + list.indexOf("B"));
        System.out.println("Index of C: " + list.indexOf("C"));
        list.remove("C");
        System.out.println("A exists: " + list.contains("A"));
        System.out.println("B exists: " + list.contains("B"));
        System.out.println("C exists: " + list.contains("C"));
        System.out.println("Index of A: " + list.indexOf("A"));
        System.out.println("Index of B: " + list.indexOf("B"));
        System.out.println("Index of C: " + list.indexOf("C"));

        for(int i = 0; i < list.size(); i ++) {
            System.out.println(list.get(i));
        }
    }
}
```

ArrayList<String> list = new ArrayList<String>();



list.remove("B");

ArrayList<Person>

Tester of HashTable

```java
import java.util.Hashtable;

public class HashTableTester {
    public static void main(String[] args) {
        Hashtable<String, String> grades = new Hashtable<String, String>();
        System.out.println("Size of table: " + grades.size());
        System.out.println("Key Alan exists: " + grades.containsKey("Alan"));
        System.out.println("Value B+ exists: " + grades.containsValue("B+"));
        grades.put("Alan", "A");
        grades.put("Mark", "B+");
        grades.put("Tom", "C");
        System.out.println("Size of table: " + grades.size());
        System.out.println("Key Alan exists: " + grades.containsKey("Alan"));
        System.out.println("Key Mark exists: " + grades.containsKey("Mark"));
        System.out.println("Key Tom exists: " + grades.containsKey("Tom"));
        System.out.println("Key Simon exists: " + grades.containsKey("Simon"));
        System.out.println("Value A exists: " + grades.containsValue("A"));
        System.out.println("Value B+ exists: " + grades.containsValue("B+"));
        System.out.println("Value C exists: " + grades.containsValue("C"));
        System.out.println("Value A+ exists: " + grades.containsValue("A+"));
        System.out.println("Value of existing key Alan: " + grades.get("Alan"));
        System.out.println("Value of existing key Mark: " + grades.get("Mark"));
        System.out.println("Value of existing key Tom: " + grades.get("Tom"));
        System.out.println("Value of non-existing key Simon: " + grades.get("Simon"));
        grades.put("Mark", "F");
        System.out.println("Value of existing key Mark: " + grades.get("Mark"));
        grades.remove("Alan");
        System.out.println("Key Alan exists: " + grades.containsKey("Alan"));
        System.out.println("Value of non-existing key Alan: " + grades.get("Alan"));
    }
}
```

empty table

Hashtable<String, String> grades

- String: type for keys
- String: type for values.

grades → | key | value |

grades.get("Alan").charAt(2); → null

"One key corresponds to one value"

Review Lecture

Thursday August 23

MaxSum -

maxSum -

0 | 2 | -4 | 3 | 7 | 6 | -2 | -4 |

1 | 20 | 1 | -2 | 4 |

2 | 7 | 1 | 1 | 1 | 1 | 2 | 2 | 3 |

8

23

Analogy - find max in a 1D array -

| -4 | 6 | -2 | 20 | 30 | 100 | -200 |

maxOfA

```
int max(int[] a) {
    int maxOfA = ;
    return maxOfA;
}
```

-> int value ?

maxOfA

-∞ 0 20 ∞

-∞

ArrayList<String> list = new ArrayList<String>();

[empty list]

boolean b = list.add("alan"); → true

[{"alan"}]

boolean c = list.add("alan"); → ~~false~~ true

[{"alan", "alan"}]

Short Circuit

- evaluate L to R | Logic
- In case &&:
  - if L is F, overall is F anyway
  - skip evaluating R

boolean b1
boolean b2

conjunction ("and")

b1	b2	b1 and b2
T	T	T
T	F	F
F	T	F
F	F	F

guarding condition

b1 && b2

"conjunction"

e.g. x != 0 && y/x > 2

crash

0 F

say: y is 4, x is 0

y/x > 2 && x != 0

say: x is 0, y is 4

`x == 0` || `y / x > 2`

`x == 0` → T

word evaluating this

`y / x > 2` → `4 / 0` → ?

`x == 0` || `y / x > 2`

b1	b2	b1 or b2
T	T	
T	F	
F	T	
F	F	

int i = ?? -1

int[] a = ?? {2, 3, 4, 5, 6}

① F

-1

0 <= i && i < a.length && a[i] > 3

② i < a.length && a[i] > 3 && 0 <= i

guarding cond.

False if i ≥ a.length (too large)

crash when i is too small.

False if 0 > i (too small)

array indexing

a[i] > 3

P. m

Person p = ??

store address of Person object. p = null

p.getName();

p → context object

method

X ① p == null && p.gN() NPE.

② p == null || p.gN()

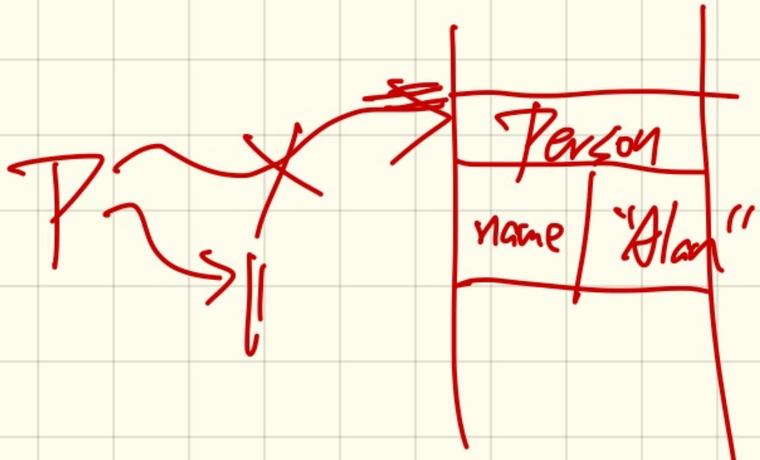
③ p != null && p.gN()

X ④ p != null || p.gN() NPE

Person p = new Person("Alan"); ≠ null

null | p.gN() || p == null

NPE



Which one(s) can crash at runtime?

Short Circuit

1. division `x != 0 && y/x > 2`
2. array indexing `i >= 0 && i < a.length && a[i] > 2`
3. method call `p != null && p.g()`

```
class Person {
    String name;
    Person spouse;
}
```

reference type

jim.spouse.spouse.name

① `Person jim = new Person( );`

`jim.setName("Jim");`

`jim.spouse.spouse.name`

context object — attribute

for name

`jim.spouse.spouse` → null

NPE.

② `Person jim = new Person( );`

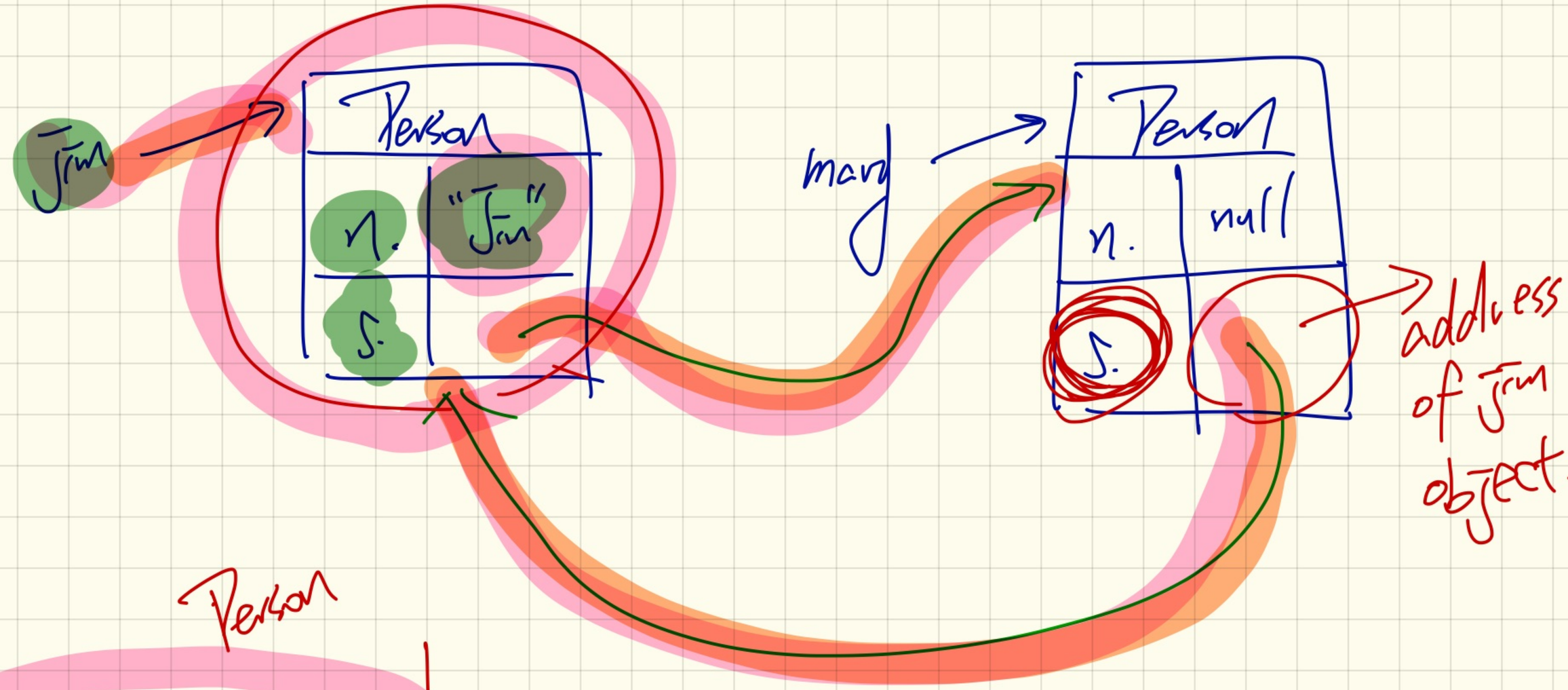
`jim.setName("Jim");`

`Person mary = new Person( );`

`jim.spouse = mary;`

`mary.spouse = jim;`

`jim.spouse.spouse`



Person

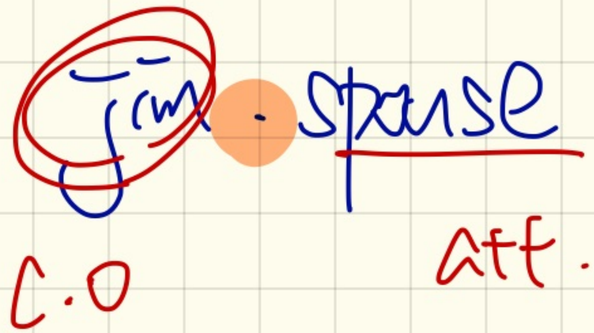
① Jim . spouse . spouse

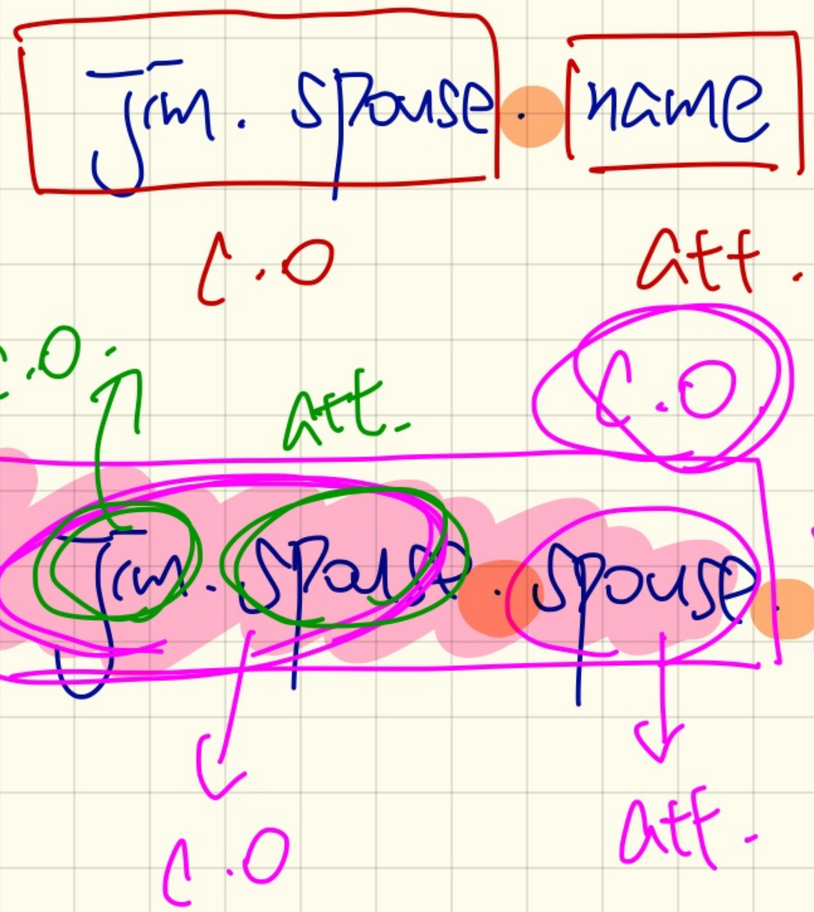
② Jim . spouse . spouse . name

Jim . spouse . spouse . spouse . name

Jim . spouse

Jim . spouse . spouse


Jim
spouse
C.O
Aff.


Jim. spouse. name
C.O
Aff.
C.O
Aff.
Jim. spouse. spouse
name
Aff
C.O
Aff.

String answer = nextLine();

boolean continue = answer.equals("Y");

while (continue) {

"N"

answer = nextLine();

continue = answer.equals("Y");

}

"N"

NPE : when the context object is null

p.q.r.s.m

null

AIOBE : a[i]

i < 0 || i >= a.length.

Person[] persons = { p1, p2 };

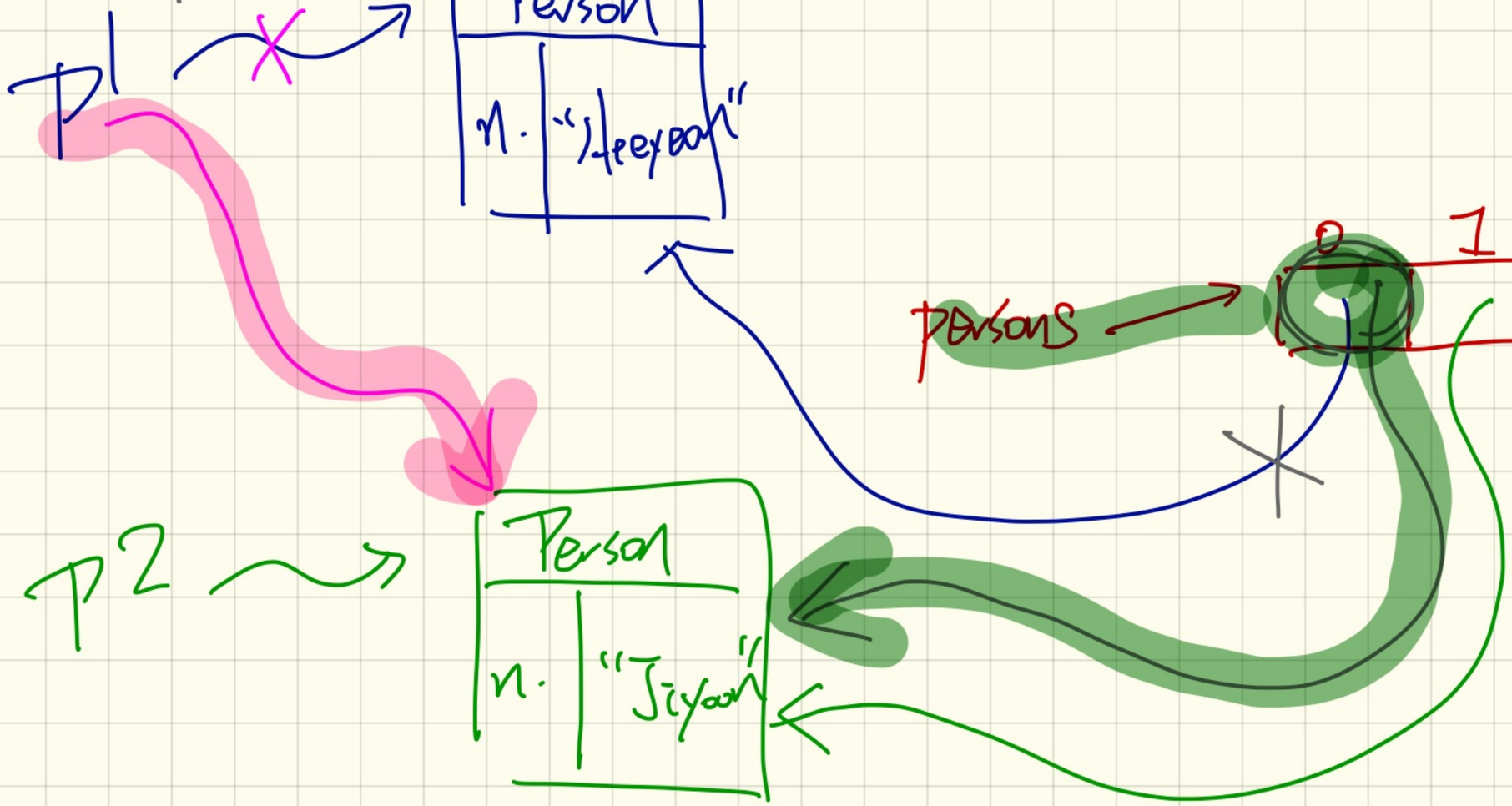
Person[] persons = new Person[2];

persons[0] = p1;

persons[1] = p2;

p1 = p2;

persons[0] = p2;



persons[0] == p1 true

∴ p1 and p2 point to the same object

∴ change done via p2 is visible to p1

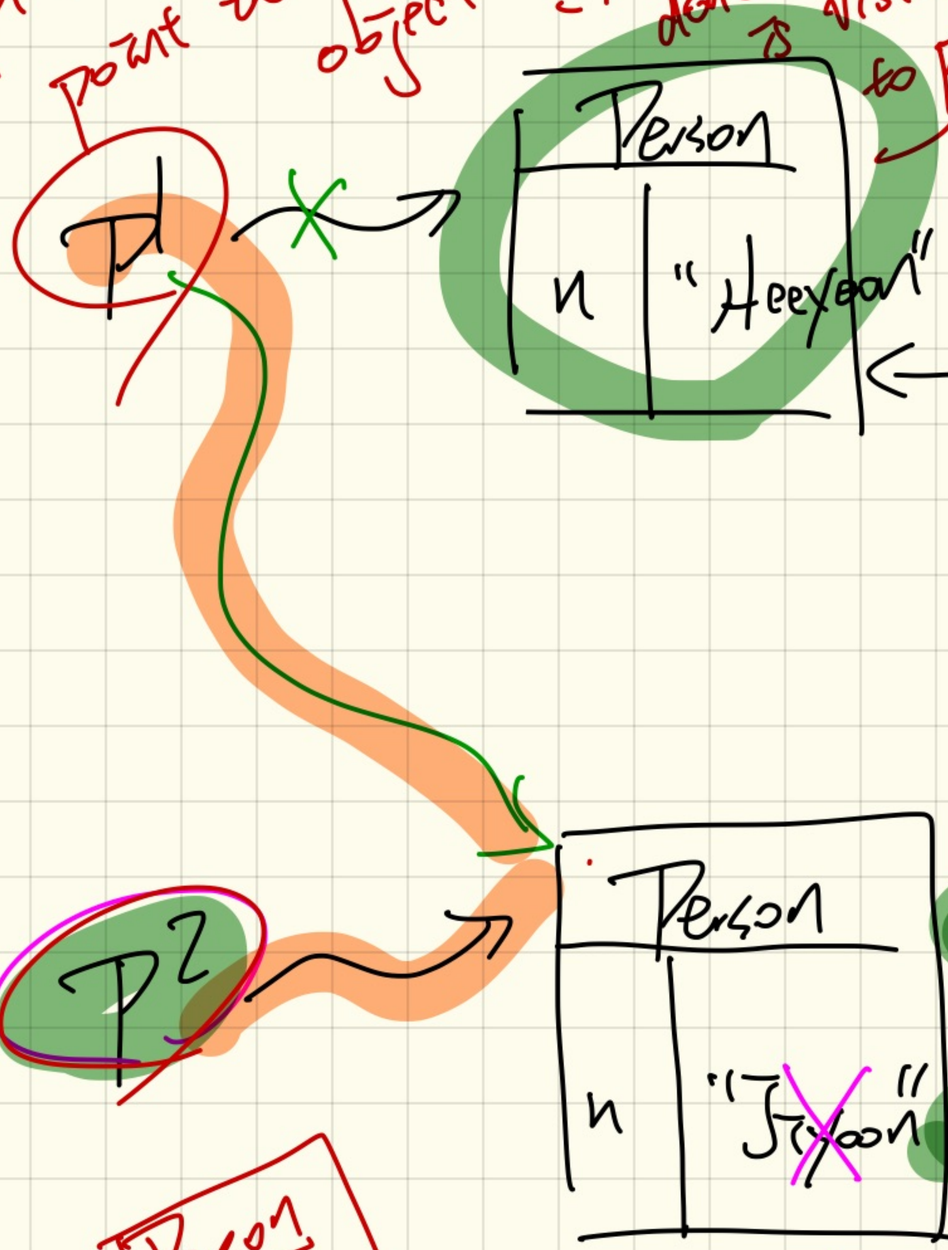
p1 = persons[1];

persons[0] = p2;

x = 3;

p2.setName("Jihye")

p1.name



change1:

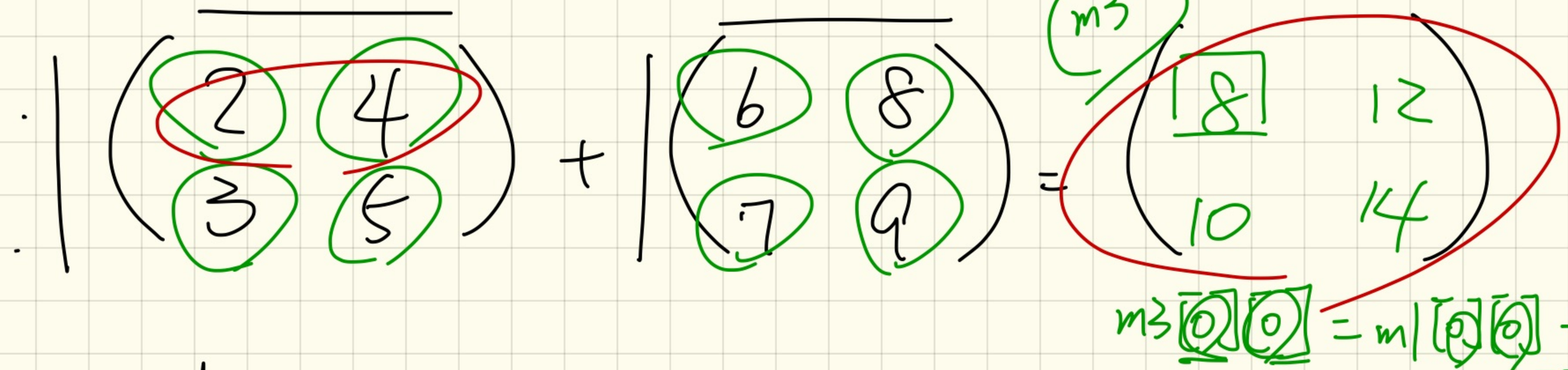
persons[1] = p1;

↳ no change

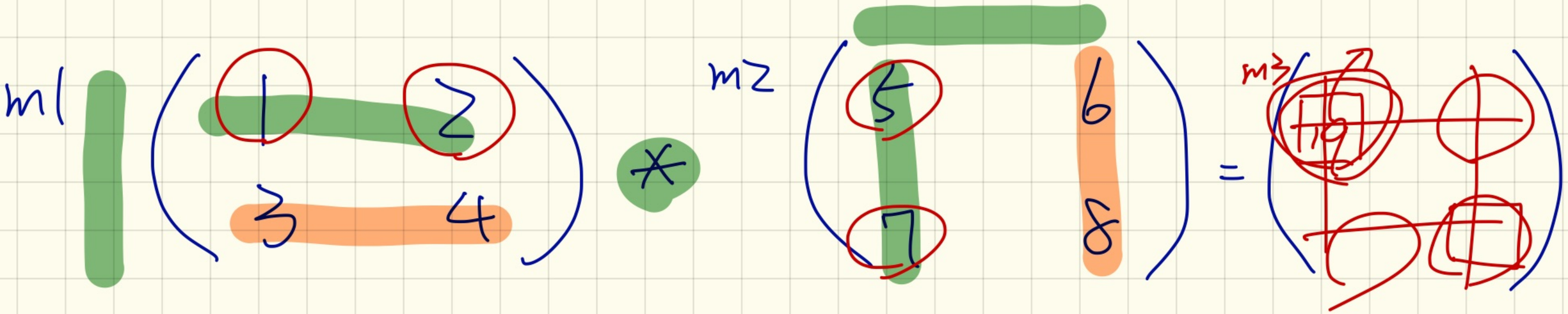
change 2:

Person p = new Person("A");

persons[1] = p;




```
void add(int[][] m1, int[][] m2){
    int[][] m3 = new int[m1.length][m1[0].length];
    for(int r=0; ---){
        for(int c=0; ---){
            m3[r][c] = m1[r][c] + m2[r][c];
        }
    }
    return m3;
}
```



$1*5 + 2*7 = 19$

$3*6 + 4*8 = 50$

m3[0][0]

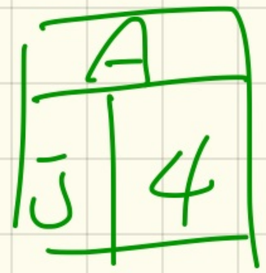
row 0 of m1 — column 0 of m2

m3[1][1]

```
class A {
    int i;
    static int j;
    A() { };
}
```

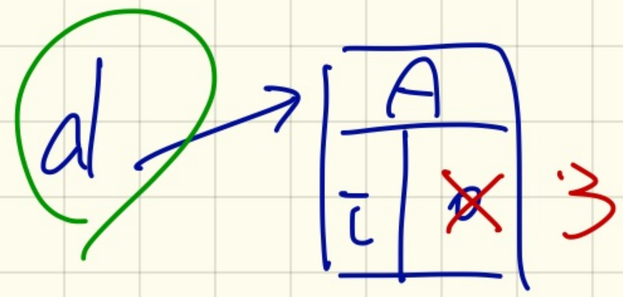
non-static

```
A a1 = new A();
A a2 = new A();
```

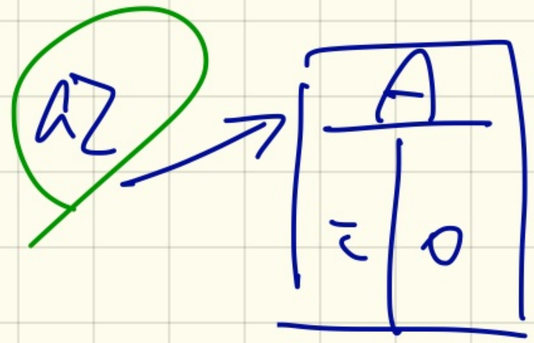



```
a1.i = 3;
```

① println(a1.i); 3



② println(a2.i); 0




```
A.j
a1.j = 4;
```

println(a1.j) 4

println(a2.j) 4