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 → false

evaluation by passed

2. b1 || b2 → True by passed.

Conjunction (&&)

b1	b2	(b1) && (b2)

disjunction (||)

b1	b2	b1 || b2

Short-Circuit

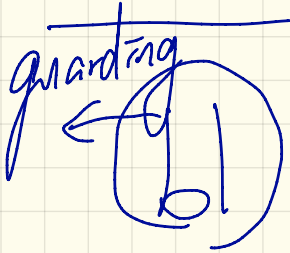
$p \wedge true \equiv p$

$p \vee false \equiv p$

$p \wedge false \equiv false$

$p \vee true \equiv true$

guarding



&& b2

① false — bypassed

② true — evaluate b2

✓ b1 || b2

① true — bypassed

② false — evaluate b2

```
int x = 0;
int y = 10;
```

guard condition

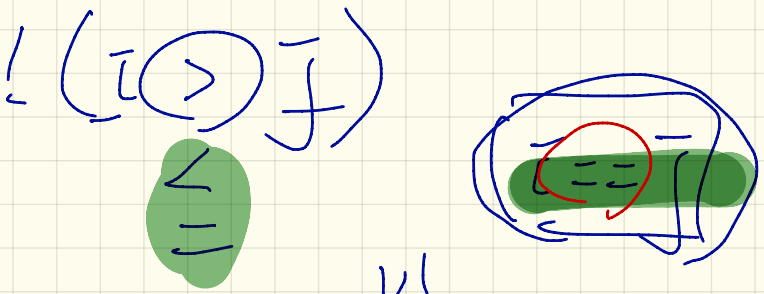
① boolean b1 = x != 0 && (y / x > 2) — false

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.

if (i < j) {

Act1 ←

} else { // !(i < j)

i >= j

Act2

}

V2.

if (i <= j) {

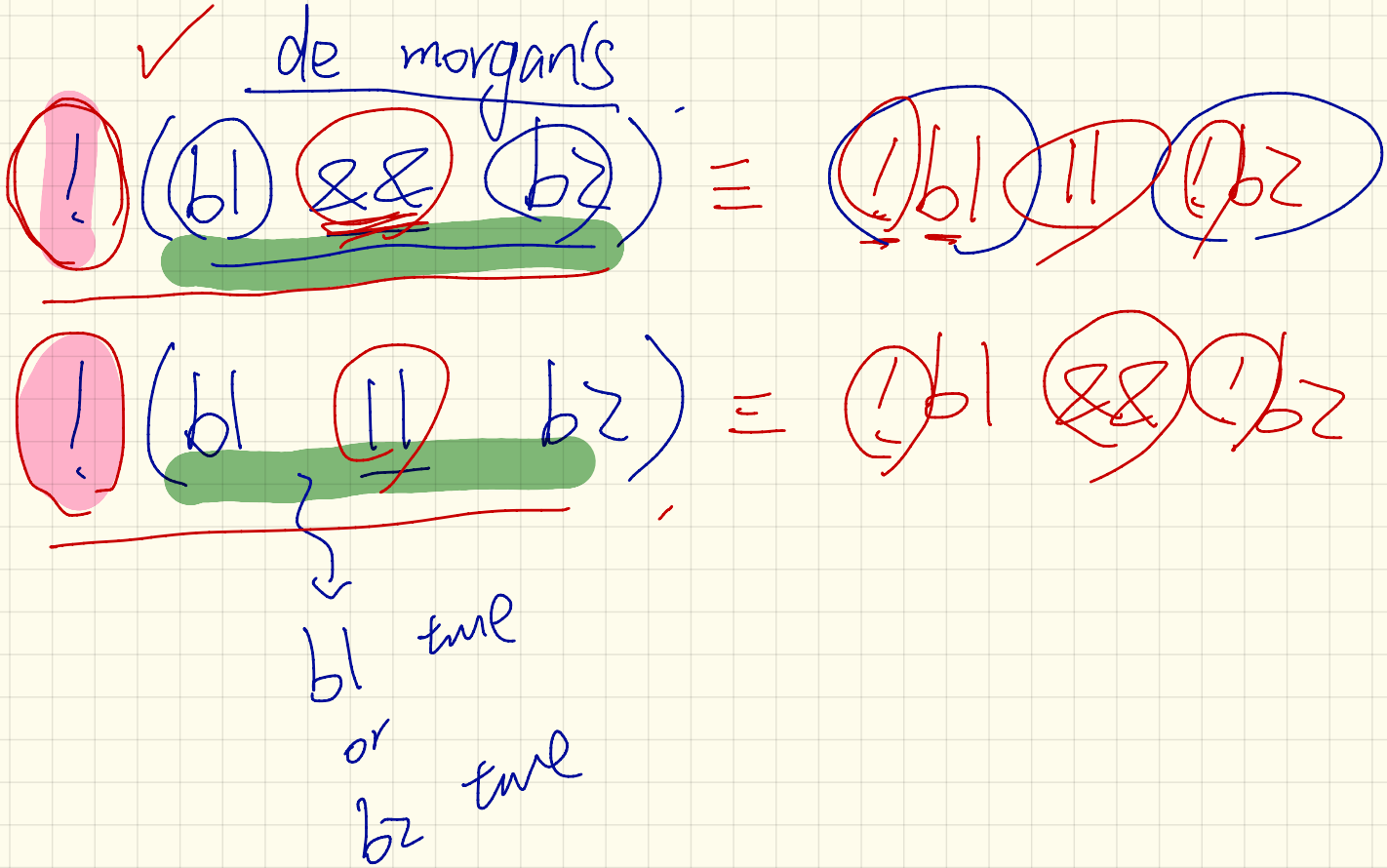
Act1

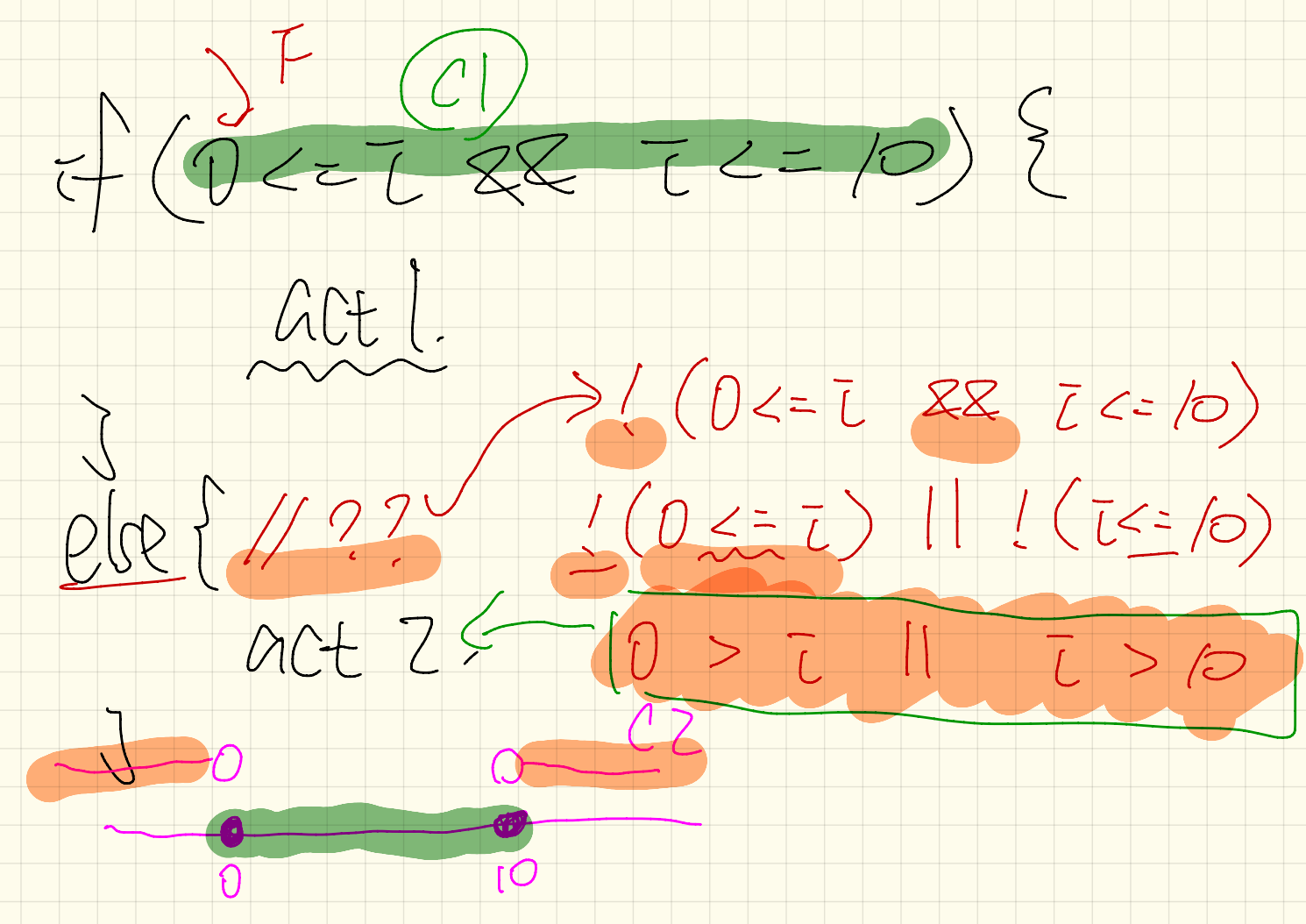
} else { // !(i < j)

i > j

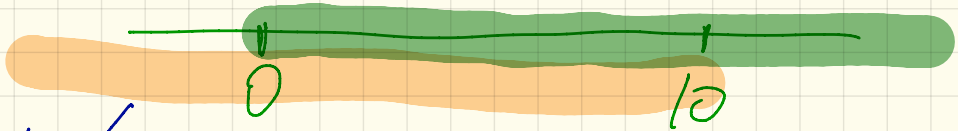
Act2

}


de morgan's
!(b1 && b2) ≡ !b1 || !b2
!(b1 || b2) ≡ !b1 && !b2
b1 true
or
b2 true


F
C1
if (0 <= i && i <= 10) {
act 1.
} else {
// ??
act 2;
!(0 <= i && i <= 10)
!(0 <= i) || !(i <= 10)
0 > i || i > 10
C2
0
10
0
10

if(t < 0 && t > 10)



else: !(t < 0 && t > 10)

!(t < 0) || !(t > 10)

t >= 0 || t <= 10

if(i < 10 || i >= 10)

else : ! (i < 10 || i >= 10)

X i >= 10 && i < 10

not possible

||| false.

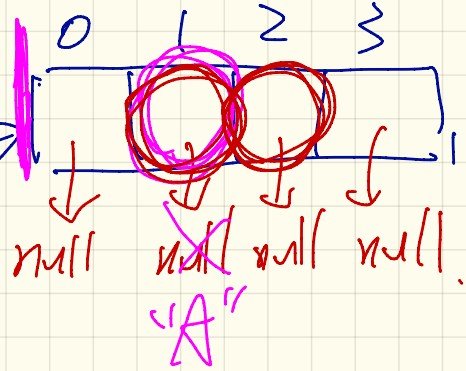
1. NullPointerException.

Primitive Type
int, double, float, char, boolean

① sa[1].equals("A")

② sa[2].equals("A")

null



2. IndexOutOfBoundsException

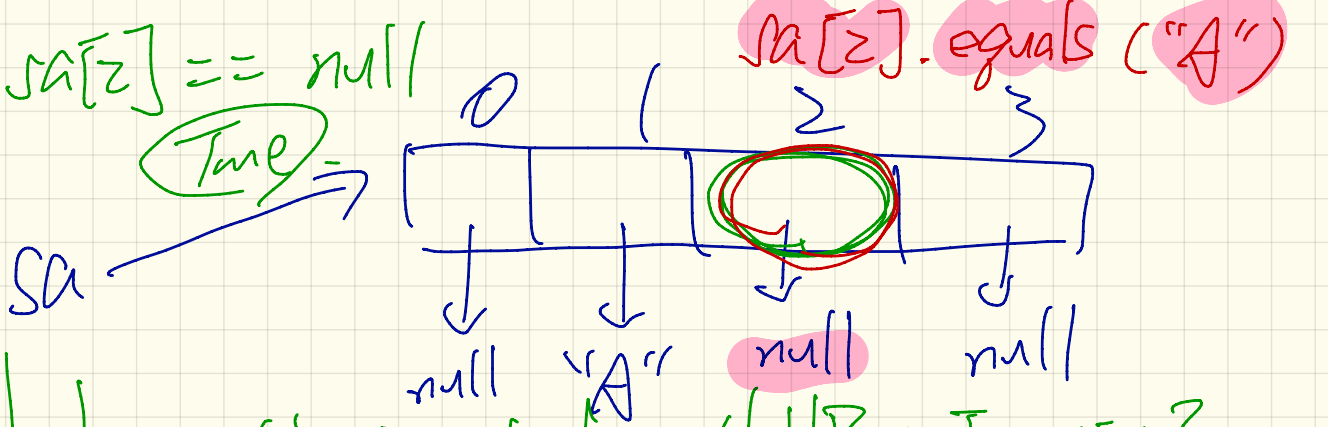
String[] sa = new String[4];

reference type

default value: null.

sa[-1]
sa[4]

sa[1] = "A";
sa[sa.length-1]



Which one(s) prevent from NullPointerException?

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

② `sa[2] == null && sa[2].equals("A")` — null; 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 / (double) numbers.length;

78.0

5

① 15 or ② 15.6

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

15

```
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("\n");

Alan Mark Tom.

i < names.length - 1
↳ i is not the last index.

names.length size

names.length - 1 last valid index

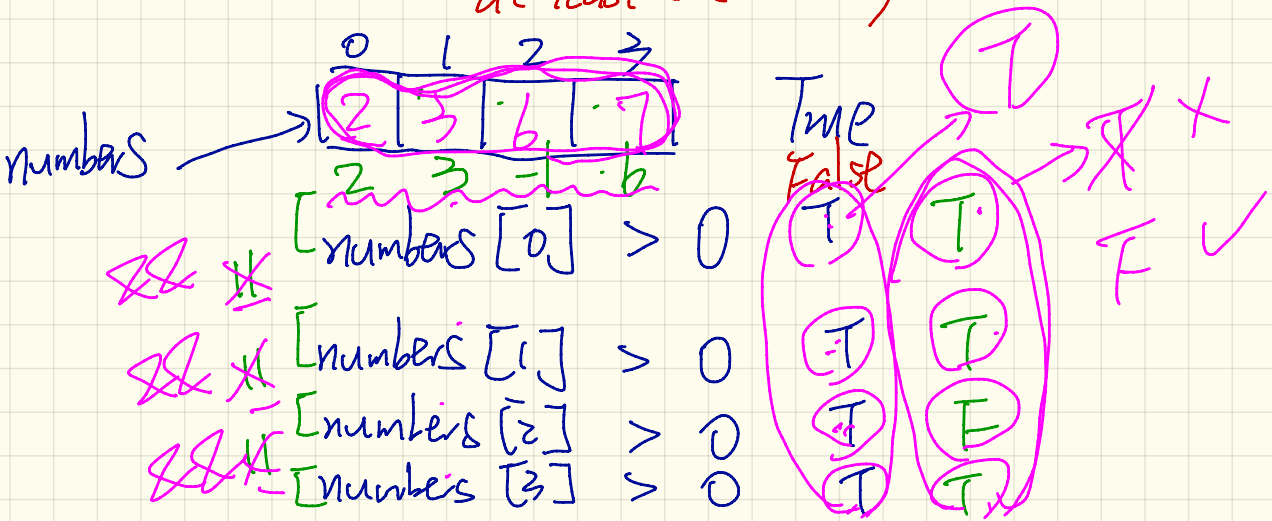
Determine if all numbers are positive.

```
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

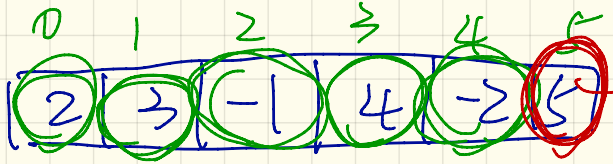
at least one not positive.



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

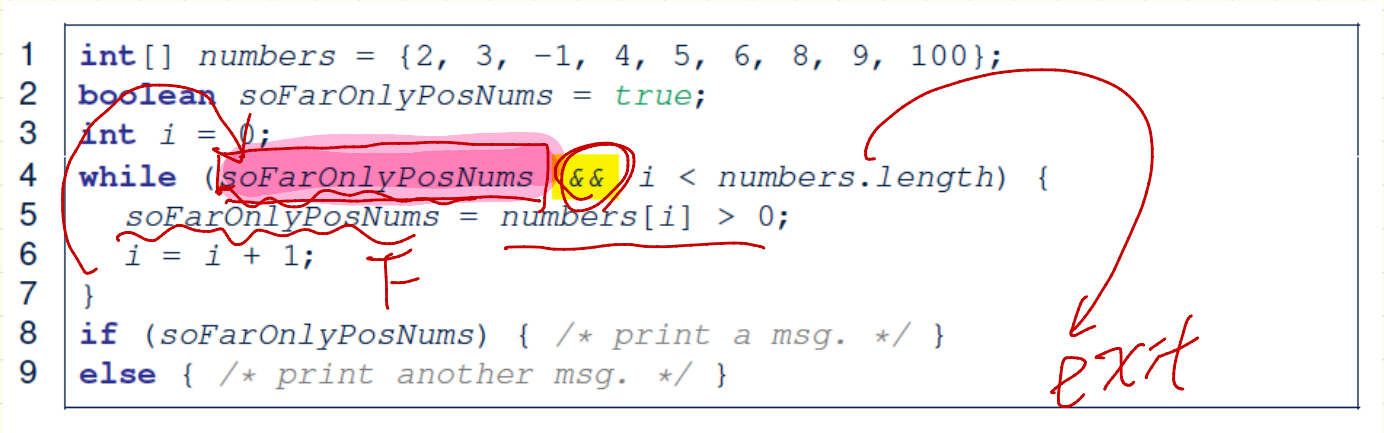


ns

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

we neither accumulate nor exit.

As soon a violation is found, as witness


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. */ }
F
exit