

Monday March 5

Lecture 8

Input : array of elements

0	1	2	3	4
3	2	-1	2	6

witness of violation

A

positive

int[]

String[]

equal to "A"

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

witness of satisfaction

(2)

Is there at least an element in A satisfied with a property.

Input : Empty array .

(return wt. of type if a sat. is found)

- (1) true ('; no wt. of vio.)
- (2) false ('; no wt. of sat.).

$$(1) \exists x \mid x \in \emptyset \cdot P(x)$$

False

tmp

$$(2) \neg \exists x \mid x \in \emptyset \cdot P(x)$$

False

(1) all numbers positive

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 so far =  both

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

accu. $\left[\text{soFar} = \text{soFar} \& \text{a}[i] > 0 \right]$

i	$a[i] > 0$	$a[i] > 0$	accu.
0	false	$a[0] > 0$	
1	false	$a[1] > 0$	
2	false	$a[2] > 0$	
3	true	$a[3] > 0$	false

`mat = [5, 0, -1, 2, 3]` ; * array contains all negative

```

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);
    
```



max

3x

4

i	$a[i] > \text{max}$	
0	$2 > 2$	false
1	$1 > 2$	false
2	$3 > 2$	true
3	$4 > 3$	true

-2 -3 -1 ✓

1

0

1

arrays sort(a)

(non)-decreasing: $!(_{-} \rightarrow -)$

Sorting $\leq \rightarrow [!(_{-} \rightarrow -)]$

[increasing]
ASCending order $1 < 3 \leq 5 < 6 < 7 < 8 \checkmark$

order $1 [3 \quad 3] [4 \quad 4] 5 X$

Non-ascending order $8 \geq 7 \geq 6 \geq 5 \geq 3 \geq 1 \checkmark$

[decreasing]
DESCending order $5 \geq 4 \geq 4 \geq 3 \geq 3 \geq 1 \checkmark$

order $5 \geq 4 \geq 4 \geq 3 \geq 3 \geq 1 X$

$\checkmark 1 \leq 3 \leq 5 \leq 6 \leq 7 \leq 8$

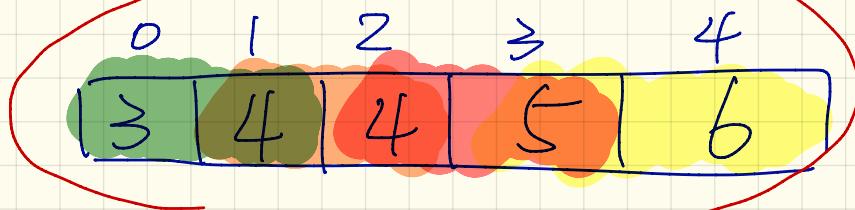
$\checkmark 1 \leq 3 \leq 3 \leq 4 \leq 4 \leq 5$

Version 1: Scan entire array → what if: $i < a.length$.

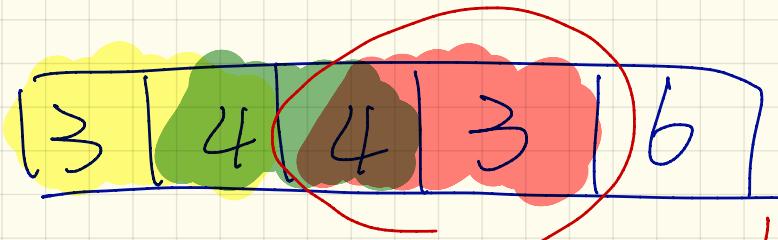
```

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

i from 0 to $a.length - 1$



$$\begin{aligned}
 a[0] &\leq a[1] \\
 a[1] &\leq a[2] \\
 a[2] &\leq a[3] \\
 a[3] &\leq a[4] \\
 a[4] &\leq a[5]
 \end{aligned}$$



X violation :: 4 \leq false

Version 2) → as soon as rotation width of Δ length found is π if
 Δ length $\approx \pi$.

```

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

```

Annotations:

- Red circles highlight the condition `i < a.length - 1` and the assignment `isSorted = a[i] <= a[i + 1];`
- A green oval highlights the loop invariant `a[i] <= a[i + 1]`.
- Handwritten notes on the right side of the code:
 - $\neg (a[i] > a[i+1])$
 - $\neg \text{isSorted} = \text{false}$
 - \neg
 - $a.length$ is 5
 - 5

Below the code, there is a diagram showing an array `a` with indices 0, 1, 2, 3, 4. The elements are 3, 4, 4, 3, 6. The element at index 2 is circled in red, and the element at index 3 is highlighted in green.

Handwritten notes below the array:

- $a[0] \leq a[1]$
- $a[1] \leq a[2]$
- $a[2] \leq a[3]$
- $a[3] \leq a[4]$

A yellow box labeled "Version" points to the first element of the array.

Handwritten note at the bottom left: $\neg \text{isSorted} = \text{false}$

boolean isSorted = true;

```
for (int i = 0; i < a.length - 1; i++) {
```

$$\text{sorted} = \text{sorted} \quad \text{if } a[i] \leq a[i+1];$$

4 permutations [0, 1, 2, 3] : $a[0] \leq a[1]$
 $a[1] \leq a[2]$
 $a[2] \leq a[3]$

```

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.print("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 }

```

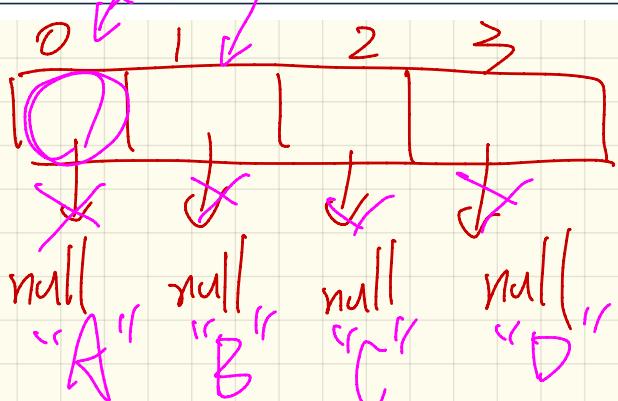
String result = "";

"A" "B" "C" "D"

0 1 2 3
"A" "B" "C" "D"

$\bar{c} = \emptyset$

① strings



V2. Person Q5 → null

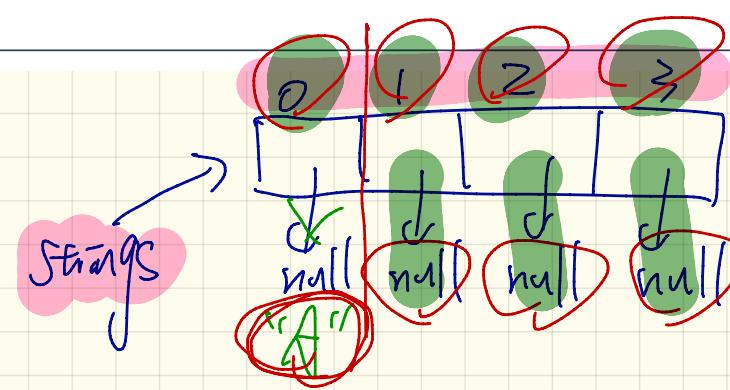
```
1 Scanner input = new Scanner(System.in);
2 System.out.println("How many strings?");
3 int howMany = input.nextInt();
4 boolean userWantsToContinue = true;
5 String[] strings = new String[howMany];
6 for(int i = 0; i < howMany && userWantsToContinue; i++) {
7     System.out.println("Enter a string:");
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 }
```

MAX CAPACITY
1. `s.equals("exit")`

"A"
"exit"

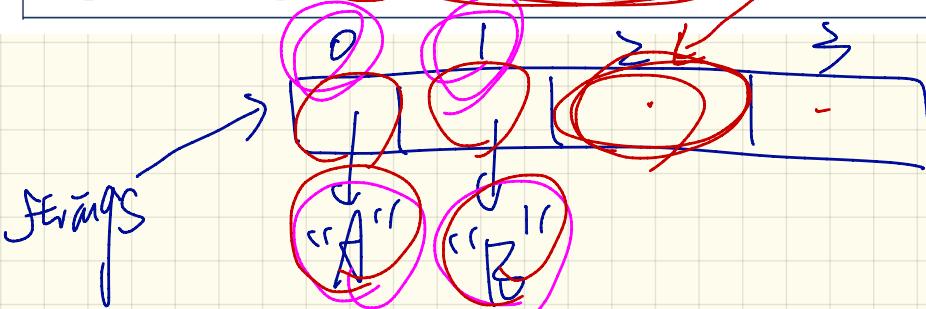
C = V

4
A
exit



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; X 2
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    }
15    System.out.println("You entered: ");
16    for(int i = 0; i < numberOfStringsRead; i++) {
17        System.out.print(strings[i] + " ");
18    }
```



2 "A" "B"
strings[0] = "A"
strings[1] = "B"

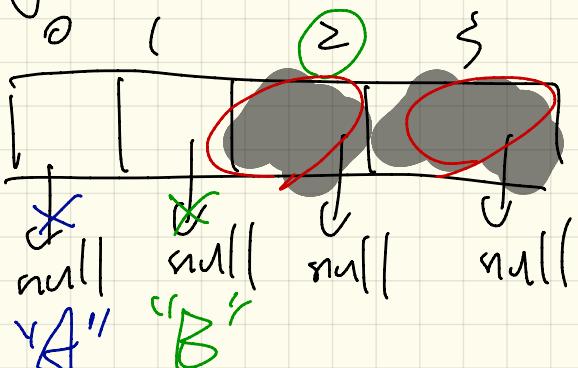
howMany

4

for(int i=0; i<nos; i++)

 cout << strings[i];

~~strings~~ →



number of strings

→ X X →

- 1 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    }
15    System.out.println("You entered: ");
16    for(int i = 0; i < numberOfStringsRead; i++) {
17        System.out.print(strings[i] + " ");
18    }
```

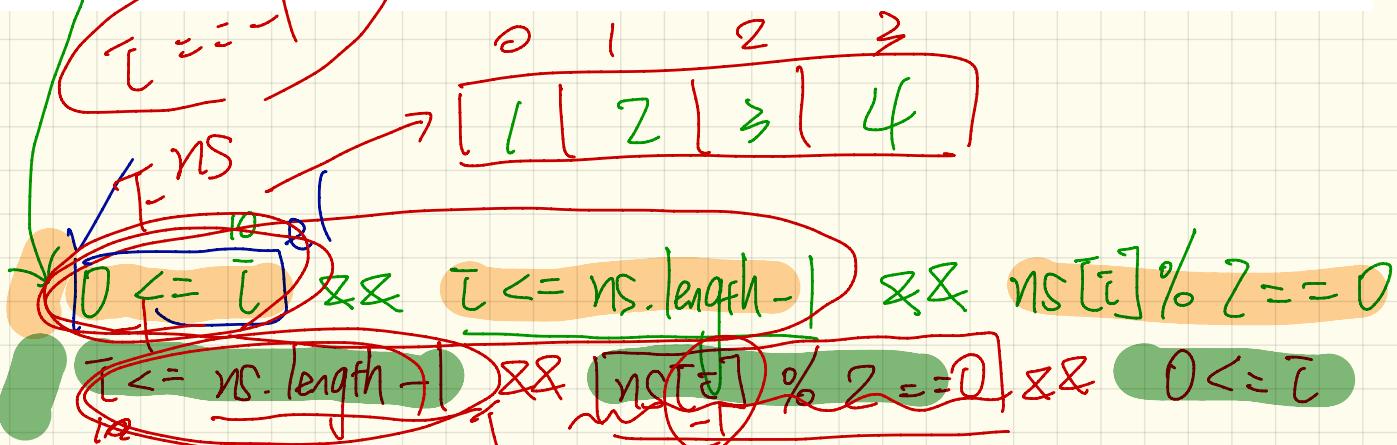
user does not want to exit

4
A
exit

```

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

