

Letter by Letter

Implementing string algorithms



Does it have a digit?

PAUSE AND TRY

A sign-up page must check that a password contains at least one digit.

THREE PASSWORDS

maple3leaf

sunnyday

R2D2

With a partner: describe, step by step, how a program could decide. When can it say "yes"? When can it safely say "no"?

By the end of class you can...

- 1 Visit every character of a string with a loop, without going past the end.
- 2 Use a boolean flag to decide whether any character, or every character, has a property.
- 3 Look at substrings longer than one character and pick loop bounds that reach all of them.
- 4 Count the substrings that meet a condition, including overlapping matches.
- 5 Build a new string piece by piece, including the string reversed.

1

PART 1

Visiting every character

A loop walks the string, one index at a time

TRAVERSAL

```
String word = "robin";  
for (int i = 0; i < word.length(); i++) {  
    String letter = word.substring(i, i + 1);  
    System.out.println(i + " " + letter);  
}
```

OUTPUT

```
0 r  
1 o  
2 b  
3 i  
4 n
```

INDEXES 0 TO LENGTH() - 1



Loop **i** from **0** while **i < word.length()**, and use **word.substring(i, i + 1)** to get the character at index **i**. The last index is always **length() - 1**.

One step too far



The length counts from 1; indexes count from 0. Using `<=` runs one extra pass, and that pass asks for an index that isn't there.

BUGGY

```
String word = "robin";  
for (int i = 0; i <= word.length(); i++) {  
    System.out.print(word.substring(i, i + 1));  
}
```

OUTPUT

robin

Then it crashes:

`StringIndexOutOfBoundsException: Range [5, 6) out of bounds for length 5`

Fix: `i < word.length()`. When `i` reaches 5, the condition is false and the body never runs.

Hop, don't walk

PAUSE AND TRY

Write the exact output. Which indexes does the loop visit?

```
String w = "pancake";  
for (int i = 0; i < w.length(); i += 2) {  
    System.out.print(w.substring(i, i + 1));  
}  
System.out.println();
```

0	1	2	3	4	5	6
p	a	n	c	a	k	e

A large, bold, teal-colored number '2' centered within a white rounded square with a slight drop shadow.

2

PART 2

Flags: any or every?

KEY IDEA

Does any character have it?

```
String pass = "maple3leaf";
boolean hasDigit = false;
for (int i = 0; i < pass.length(); i++) {
    String ch = pass.substring(i, i + 1);
    if ("0123456789".indexOf(ch) != -1) {
        hasDigit = true;
    }
}
System.out.println("Has a digit: " + hasDigit);
```

OUTPUT

Has a digit: true

FLAG FLIPS AT INDEX 5

0	1	2	3	4	5	6	7	8	9
m	a	p	l	e	3	l	e	a	f

Flag: a boolean set before the loop and switched inside it when something happens.

Start **false**. Switch to **true** on a match. Read it **after** the loop.

Does every character have it?

```
String zip = "90210";
boolean allDigits = true;
for (int i = 0; i < zip.length(); i++) {
    String ch = zip.substring(i, i + 1);
    if ("0123456789".indexOf(ch) == -1) {
        allDigits = false;
    }
}
System.out.println(allDigits);
```

OUTPUT

false

Question	Start	Switch when a character...
Any?	false	has it
Every?	true	doesn't have it



Either way, the flag changes **at most once** and never switches back.

An else that resets the flag



"If digit, true; otherwise, false" looks tidy, but then the flag only describes the **last** character checked.

PASS IS "MAPLE3LEAF"

```
boolean hasDigit = false;
for (int i = 0; i < pass.length(); i++) {
    String ch = pass.substring(i, i + 1);
    if ("0123456789".indexOf(ch) != -1) {
        hasDigit = true;
    } else {
        hasDigit = false;
    }
}
```

OUTPUT

Has a digit: false

The last character, f, isn't a digit, so the else wins. Delete the else: once switched, a flag stays switched.

3

PART 3

Longer pieces and counting

KEY IDEA

Pieces of length n stop sooner

```
String s = "ahahaha";
int count = 0;
for (int i = 0; i <= s.length() - 3; i++) {
    if (s.substring(i, i + 3).equals("aha")) {
        count++;
    }
}
System.out.println(count);
```

OUTPUT

3

MATCHES START AT 0, 2, 4



For **substring(i, i + n)**, loop while $i \leq \text{length}() - n$.

Checking every start counts **overlapping matches**.

How many double letters?

PAUSE AND TRY

Write the exact output. Where does each match start?

```
String word = "bookkeeper";
int doubles = 0;
for (int i = 0; i < word.length() - 1; i++) {
    String a = word.substring(i, i + 1);
    String b = word.substring(i + 1, i + 2);
    if (a.equals(b)) {
        doubles++;
    }
}
System.out.println(doubles);
```

0	1	2	3	4	5	6	7	8	9
b	o	o	k	k	e	e	p	e	r

A large, white, rounded square containing a bold, dark teal number '4'.

4

PART 4

Building a new string

Start empty, add pieces in order

```
String title = "hop on pop";  
String result = "";  
for (int i = 0; i < title.length(); i++) {  
    String ch = title.substring(i, i + 1);  
    if (ch.equals(" ")) {  
        result += "-";  
    } else {  
        result += ch;  
    }  
}  
System.out.println(result);  
System.out.println(title);
```

OUTPUT

```
hop-on-pop  
hop on pop
```



Start the result at "" before the loop.

Add each piece with +=.

The original string never changes.

Two ways to reverse a string

WAY 1: FORWARD, ADD TO THE FRONT

```
String rev = "";
for (int i = 0; i < word.length(); i++) {
    rev = word.substring(i, i + 1) + rev;
}
```

OUTPUT

evil

WAY 2: BACKWARD, ADD TO THE END

```
String rev = "";
for (int i = word.length() - 1; i >= 0; i--) {
    rev += word.substring(i, i + 1);
}
```

OUTPUT

evil



Both run with **word = "live"**. Change the order characters are **added** or the order they are **visited**, not both, or you get a copy.

The backward loop must reach index 0



A backward loop runs from **length() - 1** down to **0**. Both ends are easy to get one off.

BUGGY

```
String word = "live";
String rev = "";
for (int i = word.length() - 1; i > 0; i--) {
    rev += word.substring(i, i + 1);
}
System.out.println(rev);
```

OUTPUT

evi

Fix: `i >= 0`. **Also wrong:** `int i = word.length()` starts one past the end and crashes.

Same backward?

PAUSE AND TRY

Write the exact output. Which word would print true?

```
String word = "parts";  
String rev = "";  
for (int i = word.length() - 1; i >= 0; i--) {  
    rev += word.substring(i, i + 1);  
}  
System.out.println(rev);  
System.out.println(word.equals(rev));
```

0	1	2	3	4
p	a	r	t	s

Words to know

Traversal

Visiting a string's positions one at a time with a loop.

Flag

A boolean set before a loop and switched inside it when something happens.

Counter

An int set to 0 before a loop and increased by 1 for each match.

Overlapping matches

Matches that share characters, like aha twice inahaha.

Empty string

A string with no characters, "", the start of a built string.

Accumulator

A variable that collects a result across the passes of a loop.

Flag, counter, or builder?



For each job, tell your partner which tool you'd use and what it starts as before the loop.

Does the name contain a digit?

How many vowels are in the name?

Spell the name backward.

Next: the Letter by Letter puzzle worksheet.