

# 2.10 Implementing String Algorithms

## Lesson Plan

**Pacing:** 3 class periods of 45 minutes

**Learning objectives:** 2.10.A

**Essential knowledge:** 2.10.A.1

### Materials

- Projector and the Letter by Letter slide deck
- Whiteboard and markers
- Mini whiteboards or scrap paper
- IDE on every computer (Days 1 and 3)
- Printed Letter by Letter puzzle worksheet (answer key for you) and exit ticket

## Agenda

| Minutes | Activity                                                                                                   |
|---------|------------------------------------------------------------------------------------------------------------|
| 5       | Day 1 warm-up — does it have a digit? (slide 2)                                                            |
| 15      | Day 1 direct instruction — traversal with <code>substring(i, i + 1)</code> and the loop bound (slides 3–6) |
| 5       | Day 1 pause and try — hop, don't walk (slide 7)                                                            |
| 17      | Day 1 guided practice — walk strings in the IDE                                                            |
| 3       | Day 1 closure — which index is last?                                                                       |
| 5       | Day 2 warm-up — settle the digit question with a flag (slide 2)                                            |
| 15      | Day 2 direct instruction — any or every flags (slides 8–11)                                                |
| 12      | Day 2 direct instruction — longer pieces and counting, overlaps (slides 12–13)                             |
| 5       | Day 2 pause and try — double letters (slide 14)                                                            |
| 8       | Day 2 guided practice — count overlapping matches in the IDE                                               |
| 5       | Day 3 warm-up — what does <code>+=</code> do to a string?                                                  |
| 15      | Day 3 direct instruction — building a new string and reversing (slides 15–18)                              |
| 5       | Day 3 pause and try — same backward? (slide 19)                                                            |
| 10      | Day 3 guided practice — build the palindrome check in the IDE                                              |
| 5       | Day 3 closure — flag, counter, or builder? (slides 20–21)                                                  |
| 5       | Day 3 exit ticket                                                                                          |

## Overview

This topic combines tools students already have: a loop and `substring` (Topics 1.15 and 2.8), a counter (2.9), and `if` with boolean expressions (2.3, 2.5). Day 1 is traversal: visit every character with `substring(i, i + 1)` and a correct bound. Day 2 is flags (does any character have a property,

does every character) and looking at pieces longer than one character, including counting overlapping matches. Day 3 is building a new string with `+=` and reversing a string, then choosing between a flag, a counter, and a builder (2.10.A.1).

Stay inside the boundary: no `String` methods beyond `length`, `substring`, `indexOf`, and `equals`, no arrays, no `char`. Compare strings with `equals`, never `==`.

#### TEACHER NOTE

The slides are the Letter by Letter deck (21 slides); slide numbers refer to it. Day 1 uses slides 2–7, Day 2 slides 2 and 8–14, Day 3 slides 15–21. The worksheet (10 questions, about 35 minutes) is homework split by day: questions 2, 3, and 6 after Day 1; questions 1, 4, and 5 after Day 2; questions 7, 8, 9, and 10 after Day 3.

Demonstrations are in the topic's `examples/` folder (`.snippet.java` files hold only the statements for `main`): `EachLetter`, `EveryOther`, `OneTooFar` (Day 1); `HasDigit`, `AllDigits`, `FlagReset`, `CountAa`, `CountAna`, `PairTooFar` (Day 2); `RemoveSpaces`, `ReversePrepend`, `ReverseBackward`, `BackwardTooFar`, `Palindrome` (Day 3).

## Day 1 warm-up: does it have a digit?

Show slide 2 with the three passwords. Partners describe, step by step, how a program decides. Collect two ideas: look at one character at a time, and "yes" can be said as soon as one digit is found, while "no" needs every character checked. Keep the ideas on the board; Day 2 returns to them.

## Day 1 direct instruction: visit every character

Show slides 3 and 5. A loop from `i = 0` while `i < word.length()` with `word.substring(i, i + 1)` visits each character once. Run `EachLetter` and point at the index row on slide 5. Then slide 6, the one step too far: `<=` makes one extra pass that asks for an index that does not exist.

#### MISCONCEPTION Looping up to `length()` with `<=`

The length is 5, so the last index "should" be 5. Indexes start at 0, so the last index is `length() - 1`. Run `OneTooFar`: it prints `robin` and then throws `StringIndexOutOfBoundsException`. Have students say the last index of "robin" aloud before they type a bound.

## Day 1 pause and try

Slide 7: students write the output of the loop that steps by 2 and the indexes it visits (p, n, a, e; indexes 0, 2, 4, 6 of "pancake", printing `pnæ`). Reveal and check indexes, not just the output.

## Day 1 guided practice

In the IDE, students write a traversal that prints each character on its own line, then change it to visit every other character (`EveryOther`), then deliberately write `<=` and read the exception message. Pairs explain which line of the message names the bad index.

## Day 1 independent practice

Puzzle worksheet questions 2, 3, and 6 are homework: the hopping loop (JVM), the tile reverse of "stressed", and the loop match-up.

## Day 1 closure

Cold-call: "A string has length 8. What are the first and last index?" (0 and 7.)

## Day 2 warm-up

Return to slide 2. Which passwords have a digit? (`maple3leaf` and `R2D2`; `sunnyday` does not.) Ask when the program can stop being unsure for "yes" versus "no".

## Day 2 direct instruction: flags

Slide 9: a boolean `hasDigit` starts `false`, switches to `true` when a character matches, and is read only after the loop. Slide 10: for "every", the flag starts `true` and switches to `false` on the first character that fails. Run `HasDigit` and `AllDigits`. Slide 11 shows the `else` that resets the flag.

### **MISCONCEPTION** An else that resets the flag

"If digit, true; otherwise, false" looks tidy, but each pass overwrites the answer, so the flag describes only the last character. Run `FlagReset` with `tiger7x`: it prints `Has a digit: false`. Rule to teach: a flag changes once and never changes back.

## Day 2 direct instruction: longer pieces and counting

Slide 13: for pieces of length `n`, the loop runs while `i <= length() - n`, because `substring(i, i + n)` must stay inside the string. Checking every start counts overlapping matches: `ahahaha` has three `aha`. Run `CountAna` on "banana" (2 matches, starting at 1 and 3).

**MISCONCEPTION** Keeping the one-character bound

Copying the Day 1 header for two- or three-character pieces crashes at the last starts. Ask: "What is the last start index for 3-character pieces in a string of length 7?" (4.) Run `PairTooFar`.

## Day 2 pause and try

Slide 14: "bookkeeper" has three double letters (oo, kk, ee), starting at indexes 1, 3, and 5. Ask for the start indexes, and why the loop stops at `length() - 1`.

## Day 2 guided practice

In the IDE, students count "aa" in several strings and then count "ana" in "banana" to confirm overlaps are found. Declare the counter before the loop (worksheet WATCH OUT after question 5).

## Day 2 independent practice

Puzzle worksheet questions 1, 4, and 5 (the substring bounds table, the snippet pool for counting "la", and the any/every flag face-off). Question 1 supports the `length() - n` rule; question 5 settles the flag rules.

## Day 3 warm-up

On a whiteboard, students write what `String s = ""; s += "a"; s += "b";` leaves in `s`, and whether an earlier string changed (it did not; the variable now refers to a new string).

## Day 3 direct instruction: building and reversing

Slide 16: start the result at "" before the loop, add pieces with `+=`, and the original never changes. Run `RemoveSpaces`. Slide 17: reverse by adding each character to the front while walking forward, or to the end while walking backward; run `ReversePrepend` and `ReverseBackward`. Slide 18: the backward loop runs from `length() - 1` down to 0.

**MISCONCEPTION** Starting the backward loop at `length()`

The length feels like "the end", but index `length()` does not exist. Run `BackwardTooFar` to see the exception. Also point out `i > 0` stops one early and drops the first character.

**MISCONCEPTION** Not starting the result at an empty string

A `String r;` with no value cannot be used with `+=` (the compiler reports the variable might not have been initialized). Worksheet question 7 snippet 1 shows this.

## Day 3 pause and try

Slide 19: "parts" reversed is `strap` and `equals` gives `false`. Ask which word would print `true` and why (a palindrome such as "level").

## Day 3 guided practice

In the IDE, students write the palindrome check from `Palindrome`: remove the spaces with a builder, reverse with a second builder, then compare with `equals`. Anyone who writes `==` runs `PalindromeBug` to see `false` for a word that reads the same backward.

## Day 3 independent practice

Puzzle worksheet questions 7, 8, 9, and 10 are homework (you're the compiler, two truths and a lie, one-token fix, wrap-up).

## Day 3 closure and exit ticket

Slide 20 (vocabulary), then slide 21: partners say which tool (flag, counter, or builder) each job needs. Give the exit ticket (about five minutes). Question 1 checks counting overlapping substrings, question 2 flag logic, and question 3 the backward loop bound (all 2.10.A.1). Re-teach the weakest at the start of Topic 2.11.