

NAME _____

DATE _____

PERIOD _____

UNIT 2 · TOPIC 2.10 · WORKSHEET

Letter by Letter

Walk a string one character at a time, count what you find, flip it backward, and catch the flag that forgot its job.

About 35 minutes

10 questions

Cumulative through 2.10

TOPIC

2.10

**DIRECTIONS**

Work each puzzle in pencil. For code, write exactly what prints. You may use the Java Quick Reference. Some puzzles include pieces you won't need.

PART A · 2 QUESTIONS

A

Warm up your brain

1**PENCIL CHECK**

A loop looks at pieces of "sunflower" with `word.substring(i, i + n)`, where `n` is the size of each piece. Fill in each value.

Question	Value
<code>"sunflower".length()</code>	
The last index <code>i</code> when the pieces are single characters (<code>n</code> is 1)	
The last starting index <code>i</code> for 2-character pieces	
The last starting index <code>i</code> for 3-character pieces	
How many 3-character pieces there are	
<code>"sunflower".substring(6, 9)</code> , the last 3-character piece	

**THINK FIRST**

How many times does "ee" appear in "beeeep"? Count carefully, guess in the margin, and question 8 settles it.

2 YOU'RE THE JVM

You are the Java Virtual Machine. This loop hops through the word instead of walking. Run it in your head. What prints?

```
String w = "kangaroo";
String out = "";
for (int i = 1; i < w.length(); i += 3) {
    out += w.substring(i, i + 1);
}
System.out.println(out);
```

- A** aao
- B** kgo
- C** aa
- D** Nothing prints; it throws a `StringIndexOutOfBoundsException`.

**FAIR QUESTION****Why write `substring(i, i + 1)` instead of just `substring(i)`?**

Because they ask for different things. `substring(i)` hands you everything from index `i` to the end: in "tadpole", `substring(3)` is "pole". `substring(i, i + 1)` stops just before `i + 1`, so you get exactly one character, which is what a traversal needs on each pass.

PART B · 4 PUZZLES

B Puzzles**3** CODE TILES

This program should spell "stressed" backward. Someone knocked it off the fridge. The first line stayed put. Put the tiles back in order under it so it prints the target output. One tile is a decoy and isn't used.

```
String word = "stressed";
```

```
backward += word.substring(i, i + 1);
```

```
System.out.println(backward);
```

```
for (int i = word.length(); i > 0; i--) {
```

```
String backward = "";
```

```
}
```

```
for (int i = word.length() - 1; i >= 0; i--) {
```

Target output

desserts

YOUR PROGRAM

4 SNIPPET POOL

Fill each numbered hole with a snippet from the pool so the program counts every "La" in the song and prints `la appears 3 times`. Each snippet is used at most once.

LaCount.java

```
String song = "la la la";
int count = 0;
for (int i = 0; (1); i++) {
    if (song.substring((2)).equals("la")) {
        (3);
    }
}
System.out.println("la appears " + count + " times");
```

THE POOL

Each snippet can be used only once, and not every snippet is needed.

<code>i < song.length()</code>	<code>i, i + 2</code>	<code>count = 1</code>
<code>i < song.length() - 1</code>	<code>i, i + 1</code>	<code>count++</code>
<code>i < song.length() - 2</code>		

5 FACE-OFF

Tonight: two boolean flags argue about who's the better detective. The Any Flag answers "does any character have it?" The Every Flag answers "does every character have it?" Fill in the blanks.

ANY FLAG Before the loop starts, I'm set to _____. Nothing has been found yet.

EVERY FLAG I start at _____. I assume every character is fine until one proves me wrong.

ANY FLAG The moment one character has the property, I switch. After that, nothing ever switches me _____.

EVERY FLAG Same for me, the other way around. One character that _____ the test, and I'm false for good.

ANY FLAG Someone once gave me an `else` that reset me on every pass. After that I only described the _____ character.

EVERY FLAG Rookie mistake. And nobody should read either of us until _____ the loop ends.



WATCH OUT

Declare a counter **before** the loop. If you write `int count = 0;` inside the loop's braces, it only exists inside them, so a `println(count)` after the loop won't compile (cannot find symbol). Even if it could, the counter would restart at 0 on every pass.

6 MATCH-UP

Each loop below goes in the gap in this program. Loop 1 is the original, and each other loop changes one thing. Match each loop to what it prints. Each output is used once.

```
String word = "pond";
String result = "";
// the loop goes here
System.out.println(result);
```

Loop	Code	Output
1	<pre>for (int i = 0; i < word.length(); i++) { result += word.substring(i, i + 1); }</pre>	
2	<pre>for (int i = 0; i < word.length(); i++) { result = word.substring(i, i + 1) + result; }</pre>	
3	<pre>for (int i = 0; i < word.length(); i += 2) { result += word.substring(i, i + 1); }</pre>	
4	<pre>for (int i = 1; i < word.length(); i++) { result += word.substring(i, i + 1); }</pre>	

- **P** dnop
- **Q** ond
- **R** pond
- **S** pn

PART C · 4 QUESTIONS

C Catch the mistake

7 YOU'RE THE COMPILER

You are the compiler. For each snippet, decide whether it compiles. If it does, write what happens when it runs. If it doesn't, say why in a few words.

Snippet	Compiles?	What happens, or why not
<pre>1 String s = "frog"; String r; for (int i = 0; i < s.length(); i++) { r += s.substring(i, i + 1); }</pre>		
<pre>2 String s = "cat"; for (int i = 0; i <= s.length(); i++) { System.out.print(s.substring(i, i + 1)); }</pre>		
<pre>3 String s = "hop"; String r = ""; for (int i = s.length() - 1; i >= 0; i--) { r += s.substring(i, i + 1); } System.out.println(r);</pre>		
<pre>4 String s = "moss"; int count = 0; for (int i = 0; i < s.length(); i++) { count += s.substring(i, i + 1); }</pre>		

8 TWO TRUTHS AND A LIE

Two of these claims about the code are true and one is a lie. Circle the lie, then rewrite it so it's true.

```
String sound = "beeeep";
int count = 0;
for (int i = 0; i < sound.length() - 1; i++) {
    if (sound.substring(i, i + 2).equals("ee")) {
        count++;
    }
}
```

- **1** The last value of `i` that the loop body runs with is 4.
- **2** `count` ends at 2, because the four e letters make exactly two pairs.
- **3** After the loop, `sound` still holds "beeeep".

THE LIE, REWRITTEN

9 ONE-TOKEN FIX

A music app checks whether every letter in a tune is a note from A to G. For "BAGGAGE" it prints `Playable: false`, but every letter is a note, so it should print `Playable: true`. Change exactly one token so the program gives the right answer for **any** tune.

```
1 String tune = "BAGGAGE";
2 boolean allNotes = false;
3 for (int i = 0; i < tune.length(); i++) {
4     String note = tune.substring(i, i + 1);
5     if ("ABCDEFGH".indexOf(note) == -1) {
6         allNotes = false;
7     }
8 }
9 System.out.println("Playable: " + allNotes);
```

LINE, OLD TOKEN, NEW TOKEN, AND WHY

10 WRAP-UP**REFLECTION**

In one sentence: why does the `println` for a flag or a counter go after the loop instead of inside it?

YOUR ANSWER