# 4.3

## Looping over Strings

10/14/2022

# Refresher: String Methods!

| | | |
|---|---|---|
| **`int length()`** | Returns the number of characters in a String object. | `str.length()` |
| **`int indexOf(String str)`**<br><br>**`int indexOf(String str, int fromIndex)`** | Returns the index of the first occurrence of `str` [starting at `fromIndex`, if provided].<br><br>Returns -1 if not found. | `str.indexOf("ing")`<br>`str.indexOf("ch", 9)` |
| **`String substring(int from, int to)`**<br><br>**`String substring(int from)`** | Returns substring beginning at index `from` and ending at (`to - 1`) [or `length()-1`, if `to` isn't provided]. | `str.substring(7, 10)`<br>`str.substring(3)`<br>`str.substring(i, i+1)` |
| **`char charAt(int index)`** | Returns the character in the string at `index`. | `str.charAt(2)` |

# String Transformations Using Loops

Many loops over strings are to **transform** a string into another string, e.g., remove spaces, reverse it. This can be approached in multiple ways. Here are two approaches:

**Approach #1: Transform the same String variable repeatedly until you achieve the desired result.**

```
s = "Let us remove all spaces"
s ← "Letus remove all spaces"
s ← "Letusremove all spaces"
s ← "Letusremoveall spaces"
s ← "Letusremoveallspaces"
```

(Remember, Java Strings are immutable, meaning they cannot be modified. When `s` changes, you aren't modifying the same String instance… you are repeatedly changing what the String reference variable `s` points to.)

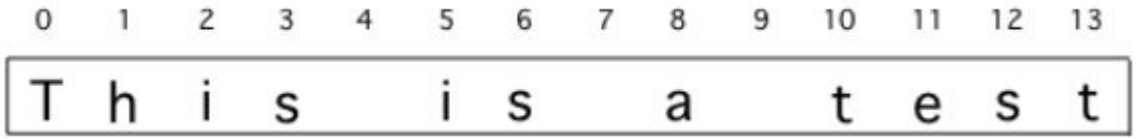
**Approach #2: Loop over the source String, leaving it unchanged, to build up a new result String.**

```
s = "Let us remove all spaces"
result ← "Let"
result ← "Letus"
result ← "Letusremove"
result ← "Letusremoveall"
result ← "Letusremoveallspaces"
```

Neither approach is always better. It depends on the problem you're solving. You may need to **benchmark** the code both ways to find what works better.

# Manipulating Strings with `while` Loops

# Example: Use a `while` loop to remove letters from a String




How can we use a `while` loop to remove all spaces from this String?

# Fill in the code

```
public static String removeSpaces(String s) {
  int i = s.indexOf(" ");

  // while there is a " " in the string
  while (i >= 0) {
    ...
    i = s.indexOf(" ");
  }

  return s;
}
```

# Fill in the code

Here's one way you can do it (there are more than one ways to complete this code! Can you think of another way?)

```
public static String removeSpaces(String s) {
  int i = s.indexOf(" ");

  // while there is a " " in the string
  while (i >= 0) {
    // Remove the " " at index by concatenating
    // substring up to index and then rest of the string.
    s = s.substring(0, i) + s.substring(i+1);
    i = s.indexOf(" ");
  }

  return s;
}
```

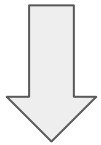
# Try it yourself: Work on **Exercise 4.3.6 Replace Letter** in CodeHS

If you finish ahead of time, try **Exercise 4.3.7**

# When to use `while` vs `for` with strings?

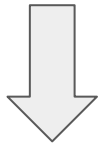
Looking for a certain character or substring?

Don’t know how many times the loop needs to run?



`while`

Want to visit every character (e.g. reversing a string, checking if palindrome)?



`for`

# Manipulating Strings with `for` Loops

# Iterating through a String with a `for` Loop

Note here that the `for` loop starts at 0 and use the string's `length()` for the ending condition.

```
String s = "example";

// loop through the string from 0 to length
for (int i = 0; i < s.length(); i++) {
   String ithLetter = s.substring(i,i+1);
   // Process the string at that index
   ...
}
```

# Reverse String Using `for` Loops

```
class Main {
  public static String reverseString(String s) {
    String result = "";
    for (int i = s.length() - 1; i >= 0; i--) {
      result += s.charAt(i);
    }
    return result;
  }
  public static void main(String[] args) {
    System.out.println(reverseString("Hello world!"));
  }
}
```

How else can we do this?

# StringBuilder

Java actually has a mutable String class, `StringBuilder`, which can be much more performant for string loops.

It's not on the AP curriculum, so don't plan to use it on the exam.

```
> java -classpath .:target/dependency/* Main
Concatenation technique: 2002601111 ns
StringBuilder technique: 425486973 ns
>
```

With StringBuilder, reversing strings by swapping chars was 5X faster than concatenation!

(StringBuilder actually has a built-in reverse method, too. String doesn't for some reason.)

Still, StringBuilder is for *building* Strings… don't use it as a general replacement for String!

```
class Main {
  public static String reverseStringBuilder(String s) {
    StringBuilder result = new StringBuilder(s);
    for (int i=0, j=s.length()-1; i<j; i++, j--) {
      char ch = result.charAt(i);
      result.setCharAt(i, result.charAt(j));
      result.setCharAt(j, ch);
    }
    return result.toString();
  }

  public static String reverseString(String s) {
    String result = "";
    for (int i=s.length(); --i>=0; ) {
      result += s.charAt(i);
    }
    return result;
  }

  public static void main(String[] args) {
    long startTime = System.nanoTime();
    for (int i=0; i<1000000; i++) {
      reverseString("Hello world!");
    }
    long elapsed = System.nanoTime() - startTime;
    System.out.println("Concatenation technique: " + elapsed + " ns");

    startTime = System.nanoTime();
    for (int i=0; i<1000000; i++) {
      reverseStringBuilder("Hello world!");
    }
    elapsed = System.nanoTime() - startTime;
    System.out.println("StringBuilder technique: " + elapsed + " ns");
  }
}
```

Try it yourself: Work on **Exercise 4.3.8 Finding Palindromes** in CodeHS

Practice!

Replit: Palindrome