StringBuilder and Design problems

4/20/26

Running times from lab

(courtesy of Hissan Tahir; Spring ‘25)

	ArrayList			LinkedList			TreeSet		HashSet	
size	add	get	remove	add	get	remove	add	contains	add	contains
25K	103	2	41	77	1,162	0	3,169	9	438	0
50K	208	6	104	129	2,251	0	6,975	2	1,008	0
100K	281	0	243	243	4,687	4	13,006	7	3,320	0
200K	575	3	468	531	8,568	0	24,175	16	6,675	3
400K	2,519	0	957	1,358	17,319	0	51,599	32	12,641	6
800K	5,207	2	2,881	4145	35,589	0	108,739	22	24,961	3

Recall: String

- Sequence of chars, indexed from 0
- Immutable: Operations can return a new String, but not change this one

- Common operations:
  - equals()
  - length()
  - +
  - substring(int start), substring(int start, int end)
  - charAt(int index)
  - indexOf(char c), indexOf(String s)

A String problem from Codingbat

https://codingbat.com/prob/p165312

String-2.doubleChar: Given a string, return a string where, for every char in the original, there are two chars.

doubleChar("The") → "TThhee"
doubleChar("AAbb") → "AAAAbbbb"
doubleChar("Hi-There") → "HHii--TThheerree"

Sample solution

```
String doubleChar(String s) {
	String retVal = “”;
	for(int i=0; i < s.length(); i++) {
		char c = s.charAt(i);
		retVal = retVal + c + c;
	}
	return retVal;
}
```

Sample solution

```
String doubleChar(String s) {
        String retVal = “”;
        for(int i=0; i < s.length(); i++) {
                char c = s.charAt(i);
                retVal = retVal + c + c;
        }
        return retVal;
}
```

Which of the following best characterizes the running time of this method? (n = s.length())

A. O(1)
B. O(n)
C. O(n^2)
D. O(n^3)
E. None of the above

Sample solution

```
String doubleChar(String s) {
        String retVal = "";
        for(int i=0; i < s.length(); i++) {
                char c = s.charAt(i);
                retVal = retVal + c + c;
        }
        return retVal;
}
```

Which of the following best characterizes the running time of this method? (n = s.length())

A. $O(1)$
B. $O(n)$
C. $O(n^2)$
D. $O(n^3)$
E. None of the above

java.util.StringBuilder

- Designed for operations that change a String
- Sequence of chars with extra space; grows as needed
- “ArrayList for chars”

- Common operations:
  - length()
  - append(stuff[*])
  - substring(int start), substring(int start, int end)
  - charAt(int i)
  - indexOf(String s)
  - insert(int offset, stuff[*])
  - replace(int start, int end, String s)
  - toString()

[*] Technical term for boolean, char, double, int, String, etc

The StringBuilder version

```
String doubleChar(String s) {
	StringBuilder retVal = new StringBuilder();
	for(int i=0; i < s.length(); i++) {
		char c = s.charAt(i);
		retVal.append(c).append(c);
	}
	return retVal.toString();
}
```

The StringBuilder version

```
String doubleChar(String s) {
        StringBuilder retVal = new StringBuilder();
        for(int i=0; i < s.length(); i++) {
                char c = s.charAt(i);
                retVal.append(c).append(c);
        }
        return retVal.toString();
}
```

Which of the following best characterizes the running time of this method? (n = s.length())

A. O(1)
B. O(n)
C. O(n^2)
D. O(n^3)
E. None of the above

The StringBuilder version

```
String doubleChar(String s) {
	StringBuilder retVal = new StringBuilder();
	for(int i=0; i < s.length(); i++) {
		char c = s.charAt(i);
		retVal.append(c).append(c);
	}
	return retVal.toString();
}
```

Which of the following best characterizes the running time of this method? (n = s.length())

A. O(1)
B. O(n)
C. O(n^2)
D. O(n^3)
E. None of the above

Design problems

What ADT(s) would you use for the drink service problem?

A. List w/o Iterator

B. List w/ Iterator

C. Stack

D. Queue

E. Not exactly one of the above

What ADT(s) would you use for the drink service problem?

A. List w/o Iterator

B. List w/ Iterator

C. Stack

D. Queue

E. Not exactly one of the above

What ADT(s) would you use for the airport security problem?

A. List w/o Iterator

B. List w/ Iterator

C. Stack

D. Queue

E. Not exactly one of the above

What ADT(s) would you use for the airport security problem?

A. List w/o Iterator

B. List w/ Iterator

C. Stack

D. <u>Queue</u>

E. Not exactly one of the above

What ADT(s) would you use for the ultimate frisbee simulation problem?

A. List w/o Iterator

B. List w/ Iterator

C. Stack

D. Queue

E. Not exactly one of the above

What ADT(s) would you use for the ultimate frisbee simulation problem?

A. List w/o Iterator

B. List w/ Iterator

C. Stack

D. Queue

E. Not exactly one of the above

What ADT(s) would you use for the room cleaning problem?

A. List w/o Iterator

B. List w/ Iterator

C. Stack

D. Queue

E. Not exactly one of the above

What ADT(s) would you use for the room cleaning problem?

A. List w/o Iterator

B. List w/ Iterator

C. Stack

D. Queue

E. Not exactly one of the above