Java: StringBuilder? ?? ???

---

1. `StringBuilder`? ???? ???

1.1 ??

`StringBuilder` 클래스는 가변 문자열 처리 를 위해 설계된 클래스이다 . 모든 메서드는 `StringBuilder` 객체 자체를 수정하며 , 새로운 객체를 생성하지 않는다 . 이는 `String`과의 가장 큰 차이점이다 .

```
StringBuilder sb = new StringBuilder("hello");
sb.append(" world");  // 같은 객체 내에서 문자열을 수정
```

1.2 ?? ??? + ?? ??

> 모든 메서드는 시간복잡도가 평균적으로 **O(1)** 또는 **O(n)**이며 , 대부분 **String**보다 빠름 .
> 대부분 메서드가 `return this`를 통해 메서드 체이닝 을 지원한다 .

? `append(String str)`

- 문자열을 뒤에 이어붙임
- 예 : `new StringBuilder("hi").append(" there") → "hi there"`
- 내부 작동 : `char[]` 배열 뒤쪽에 복사 → 공간 부족 시 자동 확장
- 시간복잡도 : 평균 O(1), 확장 시 O(n)

? `insert(int offset, String str)`

- 지정 위치에 문자열 삽입
- 예 : `"abc".insert(1, "X") → "aXbc"`
- 내부 작동 : offset 뒤쪽 문자들을 밀고 `str` 삽입

- 시간복잡도 : O(n)

? `delete(int start, int end)`

- 주어진 범위의 문자를 삭제 (`start <= i < end`)
- 예 : `"abcdef".delete(2, 4) → "abef"`
- 내부 작동 : end 이후 문자들을 앞당김
- 시간복잡도 : O(n)

? `deleteCharAt(int index)`

- 특정 인덱스의 문자 1개 삭제
- 예 : `"abc".deleteCharAt(1) → "ac"`
- 내부 작동 : `delete(index, index+1)` 호출과 동일

? `replace(int start, int end, String str)`

- 주어진 범위를 `str`로 대체
- 예 : `"abcde".replace(1, 4, "X") → "aXe"`
- 내부 작동 : `delete` 후 `insert` 조합
- 시간복잡도 : O(n)

? `reverse()`

- 문자열을 역순으로 뒤집음
- 예 : `"hello".reverse() → "olleh"`
- 내부 작동 : `char[]`를 양 끝에서 swap
- 시간복잡도 : O(n)

? `toString()`

- `StringBuilder` 내용을 `String` 객체로 반환
- 예 : `sb.toString()`
- 내부 작동 : 새로운 `String` 객체 생성 (불변 객체)
- 시간복잡도 : O(n)

? `setCharAt(int index, char c)`

- 지정한 인덱스의 문자를 변경
- 예 : `"java".setCharAt(1, 'o') → "jova"`
- □□ 내부 작동 : 배열 원소 직접 교체
- 시간복잡도 : O(1)

? `charAt(int index)`

- 특정 위치의 문자 반환
- 예 : `"java".charAt(2) → 'v'`
- □□ 설명 : 내부 `char[]` 배열의 직접 접근
- 시간복잡도 : O(1)

? `length()`

- 현재 문자열 길이 반환
- 예 : `"abc".length() → 3`
- □□ 설명 : 버퍼 내 유효 문자 수 (`count` 필드)
- 시간복잡도 : O(1)

? `capacity()`

- 내부 버퍼의 현재 용량 반환 (할당된 char 배열 길이)
- 예 : `new StringBuilder(10).capacity() → 10`
- □□ 동작 : 초기 용량 16, 필요 시 자동 확장 (공식 : `newCapacity = (old * 2) + 2`)
- 시간복잡도 : O(1)

? `ensureCapacity(int minimumCapacity)`

- 최소 버퍼 크기를 보장 (수동 확장)
- 예 : `sb.ensureCapacity(100)`
- □□ 설명 : 대량 조작 전 성능 최적화에 사용

---

2. ??? ??? ??

```
StringBuilder sb = new StringBuilder();
String result = sb.append("Hi")
                  .append(" ")
                  .append("there")
                  .replace(0, 2, "Bye")
                  .reverse()
                  .toString();

System.out.println(result);  // "ereht eyB"
```

3. ?? ??

주의 포인트	설명

□
`null`
값 전달
	`append(null)`
	은 문자열
	`"null"`
	로 처리됨

□ □□ 수정	객체 내부 수정 → 참조가 유지됨
□ □□□□□ 비안전	멀티스레드 환경에서는 `StringBuffer` 사용

? ??? ??

메서드	설명
`append()`	문자열 끝에 추가
`insert()`	중간에 삽입
`delete()`	구간 삭제
`replace()`	문자열 교체
`reverse()`	문자열 뒤집기
`charAt()`	문자 추출
`setCharAt()`	문자 수정
`toString()`	최종 문자열 반환
`capacity()`	내부 버퍼 용량 확인

`ensureCapacity()`	□□ □□ □□

Revision #2
Created 20 May 2025 04:53:09 by Dain
Updated 20 May 2025 06:26:28 by Dain