

Java.lang.String (immutable) 주요 메소드

| 메서드 | 설명 | 예시 |
|---|---------------------|--|
| 문자열 생성 및 길이 | | |
| String(String original) | 새로운 String 객체를 생성 | String str = new String("Hello"); |
| length() | 문자열의 길이 반환 | str.length(); // 5 |
| 문자열 비교 | | |
| equals(Object anObject) | 내용 비교 (대소문자 구분) | str.equals("hello"); // false |
| equalsIgnoreCase(String anotherString) | 내용 비교 (대소문자 무시) | str.equalsIgnoreCase("hello"); // true |
| compareTo(String anotherString) | 사전순 비교 (양수/0/음수 반환) | str.compareTo("Hello"); // 0 |
| startsWith(String prefix) | 지정된 접두사로 시작하는지 확인 | str.startsWith("He"); // true |
| endsWith(String suffix) | 지정된 접미사로 끝나는지 확인 | str.endsWith("lo"); // true |
| contains(CharSequence s) | 지정된 문자열 포함 여부 확인 | str.contains("el"); // true |
| 문자열 검색 및 추출 | | |
| indexOf(String str) | 지정된 문자열의 첫 번째 위치 반환 | str.indexOf("el"); // 1 |
| lastIndexOf(String str) | 지정된 문자열의 마지막 위치 반환 | str.lastIndexOf("l"); // 3 |
| charAt(int index) | 지정된 위치의 문자 반환 | str.charAt(1); // 'e' |
| substring(int beginIndex) | 지정된 위치부터 끝까지 문자열 추출 | str.substring(2); // "llo" |
| substring(int beginIndex, int endIndex) | 지정된 범위의 문자열 추출 | str.substring(1, 4); // "ell" |

| | | |
|--|-----------------------------|--|
| 문자열 변환 | | |
| toLowerCase() | 소문자로 변환 | str.toLowerCase(); // "hello" |
| toUpperCase() | 대문자로 변환 | str.toUpperCase(); // "HELLO" |
| trim() | 양쪽 공백 제거 | " Hello ".trim(); // "Hello" |
| replace(char oldChar, char newChar) | 문자 치환 | str.replace('l', 'x'); // "Hexxo" |
| replace(CharSequence target, CharSequence replacement) | 문자열 치환 | str.replace("ll", "xx"); // "Hexxo" |
| split(String regex) | 지정된 정규표현식으로 문자열 분리 (배열 반환) | "apple,banana".split(","); // ["apple", "banana"] |
| join(CharSequence delimiter, CharSequence... elements) | 지정된 구분자로 문자열 연결 | String.join("-", "apple", "banana"); // "apple-banana" |
| valueOf(Object obj) | 객체를 문자열로 변환 | String.valueOf(123); // "123" |
| toCharArray() | 문자열을 문자 배열로 변환 | str.toCharArray(); // ['H', 'e', 'l', 'l', 'o'] |
| strip() | 문자열 앞뒤 공백 제거(유니코드 공백 지원) | " Hello ".strip(); // "Hello" |
| stripLeading() | 문자열 앞 공백 제거(유니코드 공백 지원) | " Hello ".stripLeading(); // "Hello " |
| stripTrailing() | 문자열 뒤 공백 제거(유니코드 공백 지원) | " Hello ".stripTrailing(); // " Hello" |
| lines() | 문자열을 줄 단위로 분리하여 Stream으로 반환 | "Hello\\nWorld".lines().forEach(System.out::println); |
| repeat(int count) | 문자열을 주어진 횟수만큼 반복 | str.repeat(3); // "HelloHelloHello" |
| indent(int n) | 문자열에 주어진 개수만큼 공백을 추가 | str.indent(4); // " Hello" |

Q1. 다음 코드의 실행 결과는 무엇인가?

```
String s1 = "Hello";  
String s2 = "World";  
String s3 = s1 + s2;  
System.out.println(s3 == "HelloWorld");
```

- (1) true (2) false (3) 컴파일 오류 (4) 예외 발생

Q2. 다음 코드의 실행 결과는 무엇인가?

```
String multiLine = ""  
    Hello  
    World  
    """,  
System.out.println(multiLine.stripIndent());
```

- (1) HelloWorld
(2) HelloWorld
(3) HelloWorld
(4) HelloWorld World

Q3. translateEscapes() 메서드는 어떤 기능을 수행하는가?

- (1) 문자열 내부의 escape 문자를 실제 문자로 변환한다.
(2) 문자열을 소문자로 변환한다.
(3) 문자열의 줄바꿈을 자동으로 제거한다.
(4) 문자열을 JSON 형식으로 변환한다.

Q4. 다음 코드의 실행 결과는 무엇인가?

```
String input = "Java 21 is awesome!";
```

```
System.out.println(input.replaceAll("wwd+", "XX"));
```

- (1) JavaXXis awesome! (2) JavaXX is awesome! (3) Java 21 is awesome! (4) 예외 발생

Q5. 다음 코드의 실행 결과는?

```
String str = "Java";
```

```
System.out.println(str.indent(4));
```

- (1) "Java" (변화 없음) (2) " Java" (앞에 공백 4칸 추가)
(3) "Java " (뒤에 공백 4칸 추가) (4) 예외 발생

Q6. 다음 코드의 실행 결과는?

```
String str = " Hello World ";
```

```
System.out.println(str.strip());
```

- (1) " Hello World " (2) "Hello World" (3) "HelloWorld" (4) 예외 발생

Q7. 다음 코드의 실행 결과는?

```
String str = "Java";
```

```
System.out.println(str.repeat(3));
```

- (1) Java (2) JavaJavaJava (3) Java3 (4) 예외 발생

Q8. String.lines() 메서드의 역할은 무엇인가?

- (1) 문자열을 한 줄씩 리스트로 반환한다.
(2) 문자열을 정렬한다.

- (3) 문자열을 대문자로 변환한다.
- (4) 문자열을 JSON 형식으로 변환한다.

Q9. 다음 코드의 실행 결과는?

```
String text = "Java\n21\nRocks";
```

```
text.lines().forEach(System.out::println);
```

(1) Java 21 Rocks (2) Java\n21\nRocks

(3) Java

21

Rocks

(4) 예외 발생

Q10. 다음 코드의 실행 결과는?

```
String str = "abcdef";
```

```
System.out.println(str.substring(2, 5));
```

(1) "abc" (2) "bcd" (3) "cde" (4) "def"

Q11. 다음 코드의 실행 결과는?

```
String str = " Java 21 ";
```

```
System.out.println(str.trim());
```

(1) " Java 21 " (2) "Java 21" (3) "Java21" (4) 예외 발생

Q12. 다음 코드의 실행 결과는?

```
String str = "Hello";  
  
System.out.println(str.charAt(1));
```

- (1) H (2) e (3) l (4) o

Q13. 다음 코드의 실행 결과는?

```
String str = "Hello,World";  
  
String[] arr = str.split(",");  
  
System.out.println(arr[1]);
```

- (1) Hello (2) , (3) World (4) 예외 발생

Q14. 다음 코드의 실행 결과는?

```
String str = "Hello";  
  
System.out.println(str.replace("l", "X"));
```

- (1) HeXXo (2) HXIXo (3) Hello (4) 예외 발생

Q15. 다음 코드의 실행 결과는?

```
String str = "Hello";  
  
System.out.println(str.contains("el"));
```

- (1) true (2) false (3) 예외 발생 (4) 컴파일 오류
(2)

Q16. String.join() 메서드의 역할은?

- (1) 문자열을 하나의 리스트로 분할한다.
(2) 주어진 문자열들을 특정 구분자로 연결한다.
(3) 문자열을 대문자로 변환한다.

(4) 문자열을 JSON 형식으로 변환한다.

Java 문자열 관련 클래스/인터페이스 비교

Web

| 특징 | String | StringBuilder | StringBuffer | CharSequence |
|---------|--|--|---|---|
| 타입 | 클래스 | 클래스 | 클래스 | 인터페이스 |
| 특징 | 일반적인 문자열 | 빠른 문자열 수정 | 멀티스레드 환경에서 안전 | 문자열을 표현하는 인터페이스 |
| 불변 여부 | 불변 (Immutable) | 가변 (Mutable) | 가변 (Mutable) | 구현체에 따라 다름 |
| 성능 | 수정이 적은 경우 효율적 | 문자열 수정이 빈번한 경우 효율적 | 멀티스레드 환경에서 문자열 수정 시 효율적 | 구현체에 따라 다름 |
| 스레드 안전성 | 안전 | Web WAS 안전하지 않음 | 안전 | 구현체에 따라 다름 |
| 주요 용도 | 문자열 리터럴, 읽기 전용 문자열 | 문자열 빌더, 문자열 버퍼 | 멀티스레드 환경 에서의 문자열 빌더, 문자열 버퍼 | 문자열 관련 클래스들의 공통 인터페이스 |
| 주요 메서드 | length(), equals(), substring(), concat(), replace() 등 | append(), insert(), delete(), replace(), reverse() 등 | append(), insert(), delete(), replace(), reverse() 등 (StringBuilder와 유사, 스레드 안전성 보장) | length(), charAt(), subSequence(), toString() |
| 구현체 | - | - | - | String, StringBuilder, StringBuffer 등 |