# String, StringBuffer, StringBuilder 차이와 장단점

Java에서 문자열을 다루는 대표적인 클래스로는 String, StringBuffer, Stringbuilder가 있습니다.
연산이 많지 않을 때는 위에 나열된 어떤 클래스를 사용하더라도 이슈가 발생할 확률은 거의 없습니다만, 연산횟수가 많아지거나 수정/삭제가 많아지고 멀티쓰레드, Race condition 등의 상황이 자주 발생한다면 각 클래스의 특징을 이해하고 사용해야 합니다.

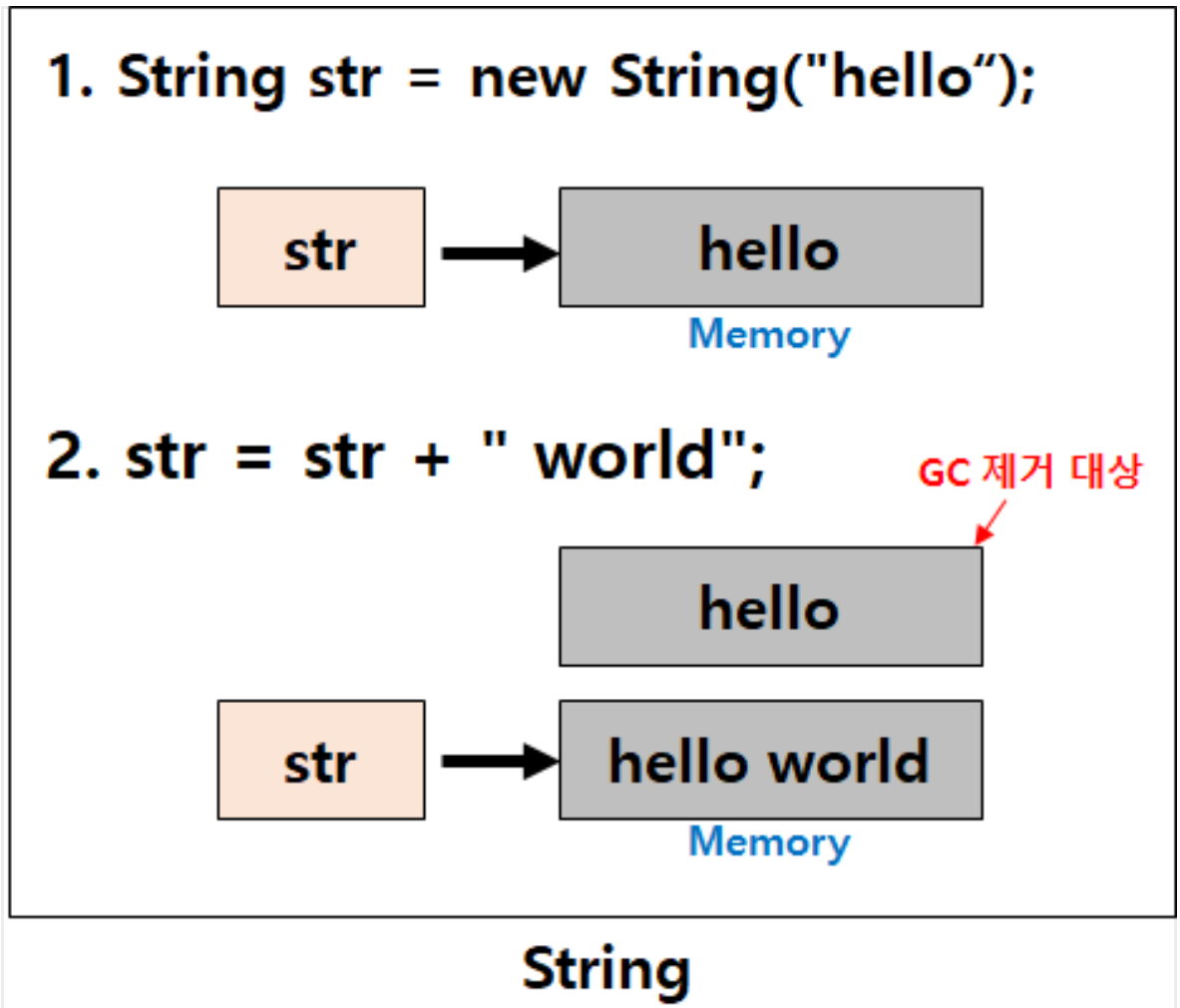
- String

String과 나머지의 가장 큰 차이점은 String은 불변(immutable)의 속성을 갖고 있다는 점입니다.

예를들어
String str = "hello";  == hello
str = str + " world";  == hello world

이런 코드가 있다면 그냥 보기에는 hello 라는 문자열에 world가 추가된 것 처럼 보입니다. 하지만 이것은 틀렸습니다.

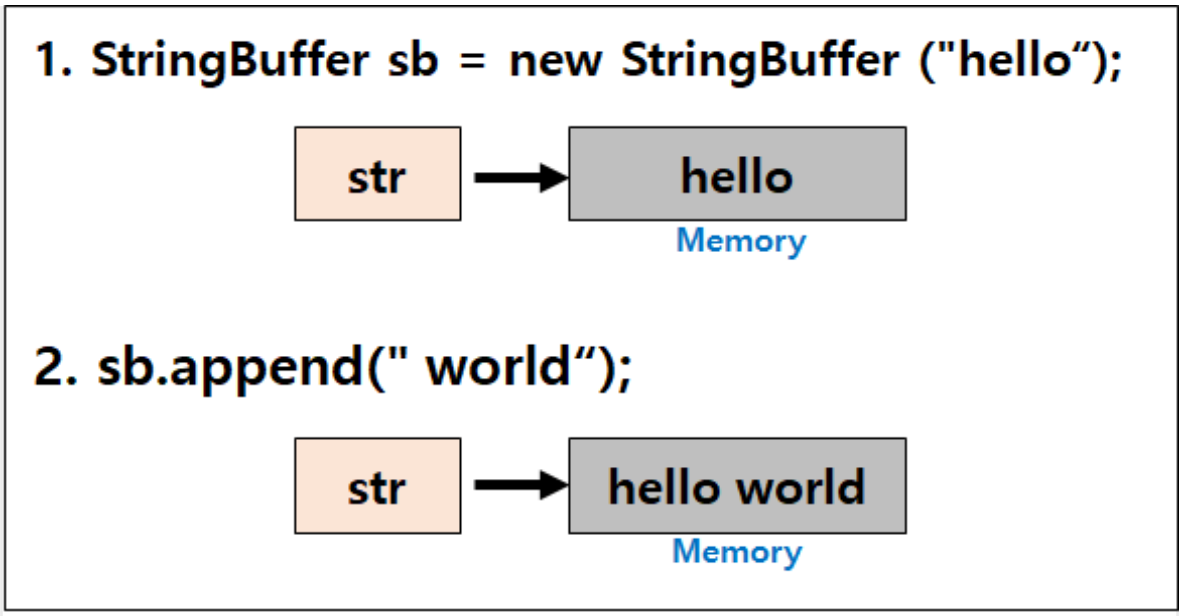
hello를 가지고 있던 str 변수가 hello world로 새로운 메모리영역을 가리키게 변경됩니다. 그리고 처음 선언했던 hello는 garbage로 남아있다가 gc에 의해 사라지게 됩니다.




String

위와 같이 String은 불변성을 가지기 때문에 변하지 않는 문자열을 자주 읽어드리는 경우 String을 사용해주면 좋은 성능을 기대할 수 있습니다.  하지만 중간중간 추가,수정,삭제가 빈번하게 발생하면 힙메모리에 상당히 많은 임시 가비지가 생성되어서 힙 메모리 부족으로 애플리케이션 성능에 치명적인 영향을 끼치게 됩니다.

이를 해결하기 위해서 Java에서는 가변성(mutable)을 가지는 StringBuffer와 StringBuilder 클래스를 도입했습니다.
String 과는 반대로 StringBuffer/StringBuilder 는 가변성을 가지기 때문에 .append() .delete() 등의 API를 이용하여 동일 객체내에서 문자열을 변경하는 것이 가능합니다.




StringBuffer

StringBuffer vs StringBuilder

동일한 API를 가지고 있는 StringBuffer, StringBuilder의 차이점이 있습니다.
가장 큰 차이점은 동기화 유무입니다.
StringBuffer는 동기화 키워드를 지원하여 멀티쓰레드 환경에서 안전하다는 점(thread-safe) 입니다.

반대로 StringBuilder는 동기화를 지원하지 않기 때문에 멀티쓰레드 환경에서 사용하는 것은 적합하지 않지만 동기화를 고려하지 않는 만큼 단일쓰레드에서의 성능은 StringBuffer 보다 뛰어납니다.

정리

String : 문자열 연산이 [적고] 멀티쓰레드 환경일 경우
StringBuffer : 문자열 연산이 [많고] 멀티쓰레드 환경일 경우
StringBuilder : 문자열 연산이 [많고] 단일쓰레드이거나 동기화를 고려하지 않아도 되는 경우