[Java] String, StringBuffer, StringBuilder 차이 및 장단점

Java 에서 문자열을 다루를 대표적인 클래스로 **String , StringBuffer , StringBuilder 가 있습니다.** 연산이 많지 않을때는 위에 나열된 어떤 클래스를 사용하더라도 이슈가 발생할 가능성은 거의 없습니다. 그러나 연산횟수가 많아지거나 멀티쓰레드, Race condition 등의 상황이 자주 발생 한다면 **각 클래스의 특징을 이해하고 상황에 맞는 적절한 클래스를 사용 해 주셔야 합니다!**

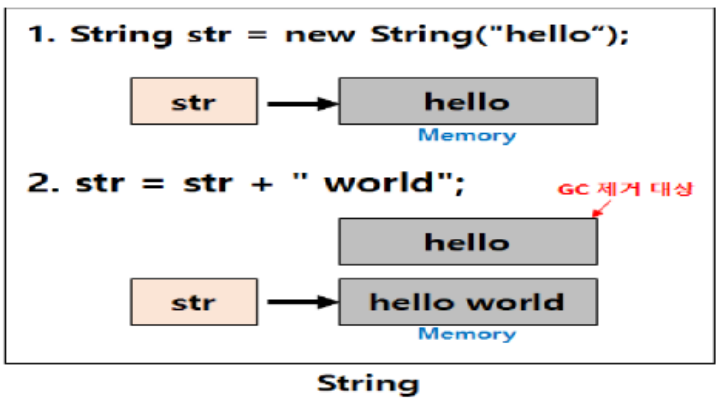
String vs StringBuffer/StringBuilder

String과 StringBuffer/StringBuilder 클래스의 가장 큰 차이점은 **String**은 **불변(immutable)**의 속성을 갖는다는 점입니다.

```
String str = "hello";   // String str = new String("hello");
str = str + " world";  // [ hello world ]
```

직관적이어서 가장 많이 사용할 듯한 위의 예제에서 "hello" 값을 가지고 있던 String 클래스의 참조변수 str이 가리키는 곳에 저장된 "hello"에 "world" 문자열을 더해 "hello world"로 변경한 것으로 착각할 수 있습니다.

하지만 기존에 "hello" 값이 들어가있던 String 클래스의 참조변수 str이 "hello world"라는 값을 가지고 있는 **새로운 메모리영역을 가리키게 변경**되고 **처음 선언했던 "hello"로 값이 할당되어 있던 메모리 영역은 Garbage로 남아있다가 GC(garbage collection)에 의해 사라지게 되는 것** 입니다. String 클래스는 불변하기 때문에 문자열을 수정하는 시점에 새로운 **String 인스턴스가 생성된 것**이지요.

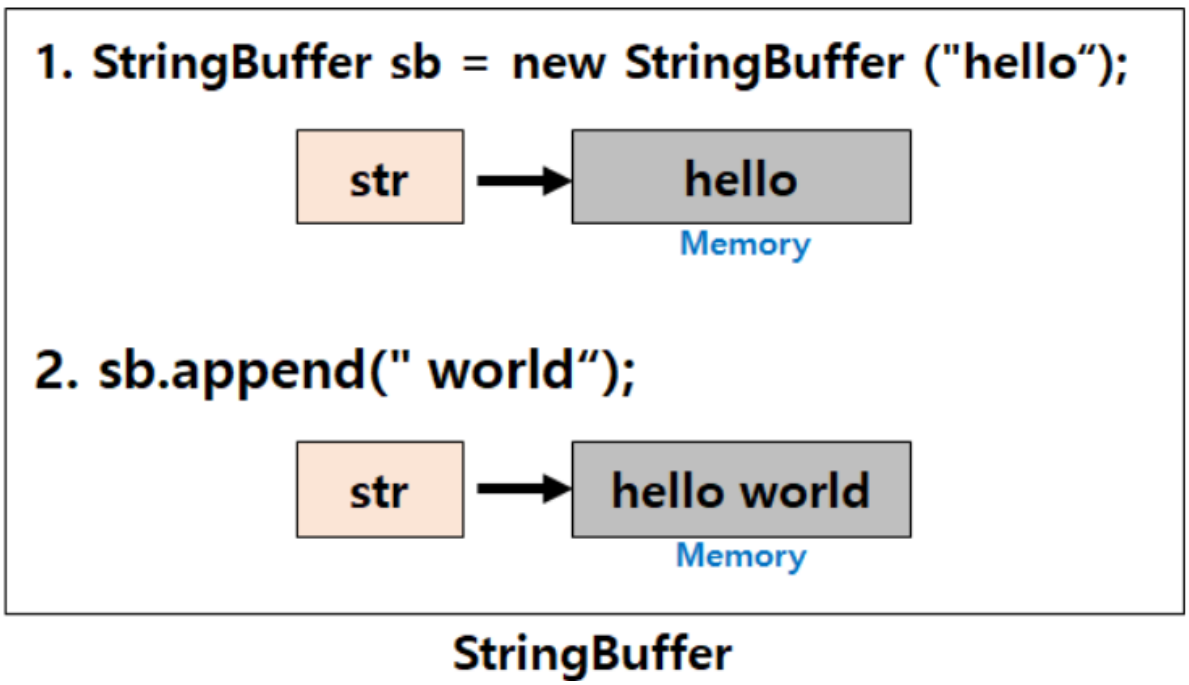


String

위와 같이 String은 불변성을 가지기 때문에 **변하지 않는 문자열을 자주 읽어들이는 경우 String을 사용** 해 주시면 좋은 성능을 기대할 수 있습니다. 그러나 문자열 추가,수정,삭제 등의 연산이 빈번하게 발생 하는 알고리즘에 String 클래스를 사용하면 힙 메모리(Heap)에 많은 임시 가비지(Garbage)가 생성 되어 힙메모리가 부족으로 어플리케이션 성능에 치명적인 영향을 끼치게 됩니다.

이를 해결하기 위해 Java에서는 **가변(mutable)성**을 가지는 **StringBuffer** / **StringBuilder** 클래스를 도입했습니다.
String 과는 반대로 StringBuffer/StringBuilder 는 가변성 가지기 때문에 .append() .delete() 등의 API를 이용하여 **동일 객체내에서 문자열을 변경**하는 것이 가능합니다. 따라서 **문자열의 추가,수정,삭제가 빈번하게 발생할 경우**라면 String 클래스가 아닌 **StringBuffer/StringBuilder를 사용**하셔야 합니다.

```
StringBuffer sb= new StringBuffer("hello");
sb.append(" world");
```



StringBuffer

StringBuffer vs StringBuilder

그렇다면 동일한 API를 가지고 있는 **StringBuffer**, **StringBuilder**의 **차이점**은 무엇일까요?
가장 큰 차이점은 **동기화의 유무**로써 **StringBuffer**는 동기화 키워드를 지원하여 **멀티쓰레드 환경에서 안전하다는 점(thread-safe)** 입니다. 참고로 **String**도 불변성을 가지기때문에 마찬가지로 **멀티쓰레드 환경에서의 안정성(thread-safe)**을 가지고 있습니다.

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

정리

마지막으로 각 클래스별 특징을 정리해 보겠습니다. 컴파일러에서 분석 할때 최적화에 따라 다른 성능이 나올 수도 있지만 **일반적인 경우**에는 아래와 같은 경우에 맞게 사용하시면 될 것 같네요.

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

	String	StringBuffer	StringBuilder
Storage	String pool	Heap	Heap
Modifiable	No(immutable)	Yes (mutable)	Yes (mutable)
Thread safe	Yes	Yes	No
Synchronized	Yes	Yes	No
Performance	Fast	Slow	Fast

String, StringBuffer, StringBuilder 비교