Interned String in Java

자바(Java)의 문자열(String)은 불변(immutable)하다. String의 함수를 호출을 하면 해당 객체를 직접 수정하는 것이 아니라, 함수의 결과로 해당 객체가 아닌 다른 객체를 반환한다. 그러나 항상 그런 것은 아니다. 아래 예를 보자.

```
public void func() {
    String haribo1st = new String("HARIBO");
    String copiedHaribo1st = haribo1st.toUpperCase();

    System.out.println(haribo1st == copiedHaribo1st);
}
```

"HARIBO"라는 문자열을 선언한 후, toUpperCase()를 호출하고 있다. 앞서 말대로 불변 객체이기 때문에 toUpperCase()를 호출하면 기존 객체와 다른 객체가 나와야 한다. 그러나 ==으로 비교를 해보면 true로 서로 같은 값이다. 그 이유는 toUpperCase() 함수의 로직 때문이다. 해당 함수는 lower case의 문자가 발견되지 않으면 기존의 객체를 반환한다.

그렇다면 생성자(new String("HARIBO"))를 이용해서 문자열을 생성하면 "HARIBO"으로 선언한 객체와 같은 객체일까? 아니다. 생성자를 통해 선언하게 되면 같은 문자열을 가진 새로운 객체가 생성된다. 즉, 힙(heap)에 새로운 메모리를 할당하는 것이다.

```
public void func() {
    String haribo1st = new String("HARIBO");
    String haribo3rd = "HARIBO";

    System.out.println(haribo1st == haribo3rd);
    System.out.println(haribo1st.equals(haribo3rd));
}
```

위의 예제를 보면 == 비교의 결과는 false이지만 equals()의 결과는 true이다. 두 개의 문자열은 같은 값을 가지지만 실제로는 다른 객체이다. 두 객체의 hash 값을 비교해보면 확실하게 알 수 있다.

```
public void func() {
    String haribo3rd = "HARIBO";
    String haribo4th = String.valueOf("HARIBO");

    System.out.println(haribo3rd == haribo4th);
    System.out.println(haribo3rd.equals(haribo4th));
}
```

이번에는 리터럴(literal)로 선언한 객체와 String.valueOf()로 가져온 객체를 한번 살펴보자. valueOf()함수를 들어가보면 알겠지만, 주어진 매개 변수가 null인지 확인한 후 null이 아니면 매개 변수의 toString()을 호

출한다. 여기서 `String.toString()`은 `this`를 반환한다. 즉, 두 구문 모두 `"HARIBO"`처럼 리터럴 선언이다. 그렇다면 리터럴로 선언한 객체는 왜 같은 객체일까?

바로 JVM에서 constant pool을 통해 문자열을 관리하고 있기 때문이다. 리터럴로 선언한 문자열이 constant pool에 있으면 해당 객체를 바로 가져온다. 만약 pool에 없다면 새로 객체를 생성한 후, pool에 등록하고 가져온다. 이러한 플로우를 거치기 때문에 `"HARIBO"`로 선언한 문자열은 같은 객체로 나오는 것이다.
`String.intern()` 함수를 참고해보자.

References

- https://www.latera.kr/blog/2019-02-09-java-string-intern/
- https://blog.naver.com/adamdoha/222817943149