■ String 클래스의 인스턴스 생성

- JAVA는 큰 따옴표로 묶여서 표현되는 문자열을 모두 인스턴스화 한다.
- 문자열은 String 이라는 이름의 클래스로 표현된다.

```
String str1 = "String Instance";
String str2 = "My String";
```

두 개의 String 인스턴스 생성,
그리고 참조변수 str1과 str2로 참조

```
System.out.println("Hello JAVA!");
System.out.println("My Coffee");
```

println 메소드의 매개변수형이 String이
기 때문에 이러한 문장의 구성이 가능하다.

```
class StringInstance
{
    public static void main(String[] args)
    {
        java.lang.String str="My name is Sunny";
        int strLen1=str.length();
        System.out.println("길이 1 : "+strLen1);
        int strLen2="한글의 길이는 어떻게?".length();
        System.out.println("길이 2 : "+strLen2);
    }
}
```

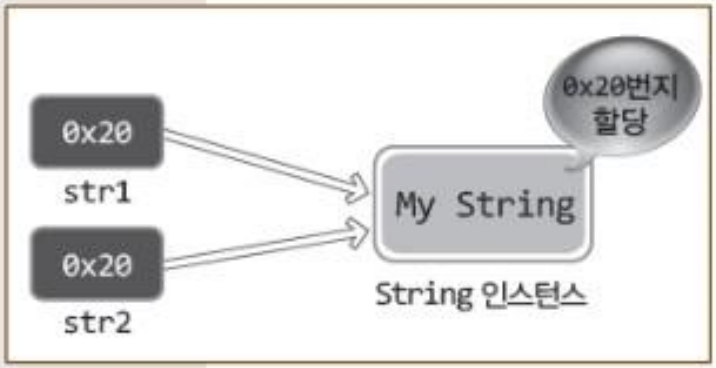
문자열의 선언은 인스턴스의 생성으로
이어짐을 보이는 문장

■ String 클래스의 특징

- String 인스턴스에 저장된 문자열의 내용은 변경이 불가능하다.
- 이는 동일한 문자열의 인스턴스를 하나만 생성해서 공유하기 위함이다.

```
public static void main(String[] args)
{
    String str1="My String";
    String str2="My String";
    String str3="Your String";

    if(str1==str2)
        System.out.println("동일 인스턴스 참조");
    else
        System.out.println("다른 인스턴스 참조");
```



String 인스턴스의 문자열 변경이 불가능하기 때문에 둘 이상의 참조변수가 동시에 참조를 해도 문제가 발생하지 않는다!

■ String 클래스의 메소드

메소드	기능	메소드	기능
length	**문자열의 길이**	**equals**	**동일내용 비교**
indexOf	**문자열의 위치**	lastIndexOf	문자열의 위치
substring	**문자 자르기**	**trim**	**공백 제거**
valueOf	문자타입 변환	toLowerCase	소문자 변환
toUpperCase	대문자 변환	startsWith	지정문자 시작여부
endsWith	지정문자 끝여부	equalsIgnoreCase	대소문자 무시 비교
charAt	지정위치의 문자	concat	문자열 합치기
contains	문자열 포함여부	**replace**	**문자열 치환**
replaceAll	**문자열 치환**	**split**	**문자열을 배열로 변환**

■ 문자열 비교 – equals()

● 문자열을 비교할때는 반드시 equals() 사용

```
public class Equals {
    public static void main(String args[]) {
        String str1 = "AA";
        String str2 = "AA";
        String str3 = new String("AA");

        if(str1 == str2) {
            System.out.println("1 2 같음");
        } else {
            System.out.println("1 2 다름");
        }
        if(str1 == str3) {
            System.out.println("1 3 같음");
        } else {
            System.out.println("1 3 다름");
        }

        if(str1.equals(str3)) {
            System.out.println("1 3 같음");
        } else {
            System.out.println("1 3 다름");
        }
    }
}
```

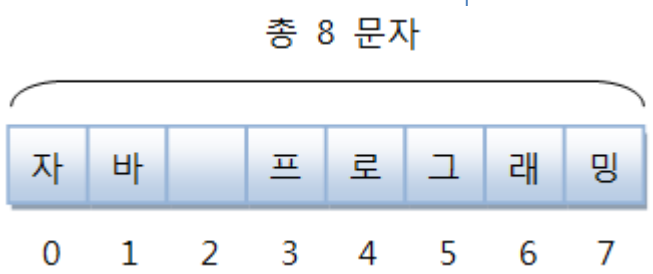
■ 문자열 길이 – length()

● 공백도 문자에 포함

```
public class Length {
    public static void main(String[] args) {
        String str1 = "Java Programming";
        int str1Len = str1.length();
        System.out.println("str1Len : " + str1Len);

        String str2 = "자바 프로그래밍";
        int str2Len = str2.length();
        System.out.println("str2Len : " + str2Len);

        int str3Len = "이렇게도 가능".length();
        System.out.println("str3Len : " + str3Len);
    }
}
```



■ 문자 추출 – charAt()

● 해당 인덱스의 문자 반환

```
public class CharAt {
    public static void main(String[] args) {
        String str = "Java Secure Coding";
        int len = str.length();
        for(int i = 0; i < len; i++) {
            char c = str.charAt(i);
            System.out.println(c);
        }
    }
}
```

■ 문자열 찾기 – indexOf()

- 매개값으로 주어진 문자열이 시작되는 인덱스 리턴
- 주어진 문자열이 포함되어 있지 않으면 -1 리턴

```
class IndexOf {
    public static void main(String args[]) {
        String str = "Java 개발자 양성을 통한 취업연계과정!!";

        int idx = str.indexOf(" ");
        System.out.println(idx);

        int idx2 = str.lastIndexOf(" ");
        System.out.println(idx2);

        idx = str.indexOf(" ", idx + 1);
        System.out.println(idx);

        idx2 = str.lastIndexOf(" ", idx2 - 1);
        System.out.println(idx2);
    }
}
```

■ 문자열 치환 – replace()

● 첫번째 매개값으로 문자열을 찾은 후 두번째 매개값으로 변환

```
public class Replace {
    public static void main(String[] args) {
        String str1 = "자바 프로그래밍";
        String str2 = null;

        str2 = str1.replace("자바", "Java"); // str1의 값은 그대로 유지

        System.out.println(str2);
    }
}
```

■ 문자열 치환 – replaceAll()

- 첫번째 매개값으로 문자열을 찾은 후 두번째 매개값으로 변환
- 정규식 지원

```
public class ReplaceAll {
    public static void main(String[] args) {
        String str = "aba bbb abb ddd abc fff";

        // ab로 시작하는 문자 변경
        String str1 = str.replaceAll("ab.", "ABC");
        System.out.println(str1);    // ABC bbb ABC ddd ABC fff

        // b가 두개인 문자 변경
        String str2 = str.replaceAll("b{2}", "Z");
        System.out.println(str2);    // aba Zb aZ ddd abc fff

        // d 또는 f 인 문자 변경
        String str3 = str.replaceAll("[df]", "Z");
        System.out.println(str3);    // aba bbb abb ZZZ abc ZZZ
    }
}
```

■ 문자열 공백 제거 – trim()

- 문자열의 앞뒤 공백 제거
- 문자열 중간 공백은 제거 불가

```
public class Trim {
    public static void main(String args[]) {
        String str = "  Java Programming  ";
        System.out.println("@" + str + "@");

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

■ 문자열을 배열로 변환 – split()

- 첫번째 매개값을 기준으로 배열 변환

```
public class Split {
    public static void main(String[] args) {
        String str = "Java Secure Coding";
        String[] strs = str.split(" ");
        for(String s : strs) {
            System.out.println(s);
        }
    }
}
```

■ 문자열 잘라내기 – substring()

- 첫번째 매개값(인덱스) 부터 두번째 매개값(인덱스) 앞까지 문자열 추출
- 첫번째 매개값(인덱스) 부터 문자열의 끝까지 추출

```
public class Substring {
    public static void main(String args[]) {
        String str = "Java 개발자 양성을 통한 취업연계과정!!";

        String str1 = str.substring(5);
        System.out.println(str1);

        String str2 = str.substring(0, 6);
        System.out.println(str2);

        String str3 = str.substring(12, 15);
        System.out.println(str3);
    }
}
```

■ 문자열을 처리할 수 있는 클래스 – StringBuffer

```
public class StrBuffer {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer();
        sb.append(25);
        sb.append('Y').append(true);
        System.out.println(sb.toString());

        sb.insert(2, false);
        System.out.println(sb.toString());

        sb.insert(sb.length() - 1, 'Z');
        System.out.println(sb.toString());
    }
}
```

■ 문자열을 처리할 수 있는 클래스 – StringBuilder

```
public class StrBuilder {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder();
        sb.append(25);
        sb.append('Y').append(true);
        System.out.println(sb.toString());

        sb.insert(2, false);
        System.out.println(sb.toString());

        sb.insert(sb.length() - 1, 'Z');
        System.out.println(sb.toString());
    }
}
```

■ String / StringBuffer / StringBuilder 속도 비교

```
public class Test {
    public static void main(String[] args) {
        long start = 0L;
        long end = 0L;

        String s = "";
        start = System.currentTimeMillis();
        for(int i = 0; i < 50000; i++) {
            s += i;
        }
        end = System.currentTimeMillis();
        System.out.println(
                "String : " + (end - start));
        StringBuffer sb = new StringBuffer();
        start = System.currentTimeMillis();
        for(int i = 0; i < 500000; i++) {
            sb.append(i);
        }
        end = System.currentTimeMillis();
        System.out.println(
                "StringBuffer : " + (end - start));

        StringBuilder bb = new StringBuilder();
        start = System.currentTimeMillis();
        for(int i = 0; i < 500000; i++) {
            bb.append(i);
        }
        end = System.currentTimeMillis();
        System.out.println(
                "StringBuilder : " + (end - start));
    }
}
```