■ 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);
    }
}
```

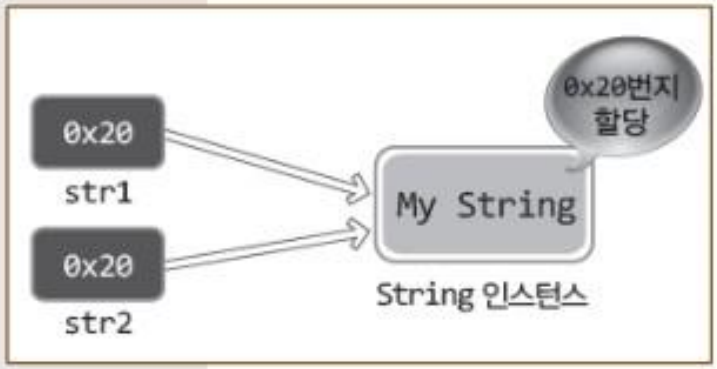
문자열의 선언은 인스턴스의 생성으로
이어짐을 보이는 문장

■ 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

ch12.Equals

```
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");
}
```

```
1 2 같음
1 3 다름
1 3 같음
```

■ 문자열 길이 - length()

● 공백도 문자에 포함

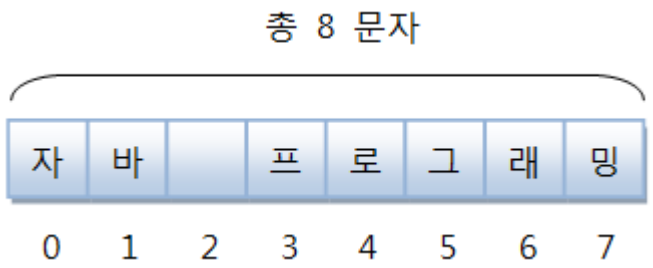
ch12.Length

```
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);
```

```
str1Len : 16
str2Len : 8
str3Len : 7
```



■ 문자 추출 - charAt()

● 해당 인덱스의 문자 반환

ch12.CharAt

```
String str = "Java Secure Coding";
int len = str.length();
for (int i = 0; i < len; i++) {
  char c = str.charAt(i);
  if(c == ' ') {
    System.out.println();
  } else {
    System.out.print(c);
  }
}
```

```
Java
Secure
Coding
```

■ 문자열 찾기 - indexOf()

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

ch12.IndexOf

```
String str = "Lorem ipsum dolor sit amet, ... aliqua.";

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);
```

```
5
116
11
110
```

■ 문자열 치환 - replace()

● 첫번째 매개값으로 문자열을 찾은 후 두번째 매개값으로 변환 **ch12.Replace**

```
String str1 = "Java Programming 01~12";
String str2 = null;
String str3 = null;
String str4 = null;

str2 = str1.replace("Java", "자바");
str3 = str1.replaceAll("[a-g]", "☆");
str4 = str1.replaceAll("\\d", "X");
System.out.println(str2);
System.out.println(str3);
System.out.println(str4);
```

```
자바 Programming 01~12
J☆v☆ Pro☆r☆mmin☆ 01~12
Java Programming XX~XX
```

■ 문자열 공백 제거 - trim()

- 문자열의 앞뒤 공백 제거
- 문자열 중간 공백은 제거 불가

ch12.Trim

```
String str = "  Java Programming  ";
System.out.println("@" + str + "@");

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

```
@  Java Programming  @
Java Programming
```

■ 문자열을 배열로 변환 - split()

● 첫번째 매개값을 기준으로 배열 변환

ch12.Split

```
String str = "Java Secure Coding";

String[] strs = str.split(" ");

for (String s : strs) {
  System.out.println(s);
}
```

```
Java
Secure
Coding
```

■ 문자열 잘라내기 - substring()

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

ch12.Substring

```
//              0         10        20        30
//              012345678901234567890123456789012
String str = "Life is too short, You need Java?";

String str1 = str.substring(19);
System.out.println(str1);

String str2 = str.substring(12, 17);
System.out.println(str2);

String str3 = str.substring(28, 32);
System.out.println(str3);
```

```
You need Java?
short
Java
```

■ 연습문제 (ch12.연습문제01)

● 아스키코드 11이 증가되어 암호화된 문자열 text를 복호화 하기

```
String text = "spwwz";

for(int i = 0; i < text.length(); i++) {

  char c = (   ①   ); // 문자 1개 추출

  char decrypt = (   ②   ); // 아스키코드 11 감소

  System.out.print(decrypt);
}
```

결과

```
hello
```

■ 연습문제 (ch12.연습문제02)

● 애국가의 한 소절을 결과와 같이 출력하기

```
String text = "동해물과 백두산이 ... 우리나라 만세";

String[] texts = (   ①   );

String temp = "";

for(String t : texts) {
  (   ②   );
}
```

결과

```
동해물과
동해물과 백두산이
동해물과 백두산이 마르고
동해물과 백두산이 마르고 닳도록
동해물과 백두산이 마르고 닳도록 하느님이
동해물과 백두산이 마르고 닳도록 하느님이 보우하사
동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라
동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세
```

■ 연습문제 (ch12.연습문제03)

● "소나기" 소설의 내용 중 "칡"으로 시작하는 문장을 찾은 후 출력하기

```
/* 소나기.txt 문자열 읽어오기 */
BufferedReader reader = new BufferedReader(new FileReader("data/소나기.txt"));
String text = "";
while(true) {
  String data = reader.readLine();
  if(data == null) break;
  text += data + "\n";
}
reader.close();

int startIdx = 0;
int endIdx = 0;
while(true) {
  startIdx = (  ①  );
  if(startIdx == -1) break;
  endIdx = (  ②  );
  System.out.println(text.substring(startIdx, endIdx + 1));
}
```

결과

```
칡덩굴이 엉키어 꽃을 달고 있었다.
칡덩굴을 그러쥐었다.
칡덩굴 있는 데로 내려가, 꽃 많이 달린 몇 줄기를 이빨로 끊어 가지고 올라온다.
```

■ 문자열을 처리할 수 있는 클래스 - StringBuffer

ch12.StrBuffer

```
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());
```

```
25Ytrue
25falseYtrue
25falseYtruZe
```

■ 문자열을 처리할 수 있는 클래스 - StringBuilder

ch12.StrBuilder

```
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());
```

```
25Ytrue
25falseYtrue
25falseYtruZe
```

■ String / StringBuffer / StringBuilder 속도 비교

```
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));
```

```
String : 1643ms
StringBuffer : 24ms
StringBuilder : 16ms
```