컴퓨터 프로그래밍
(Computer Programming)

이 선 순

11. 문자열과 메소드

목차

1. 문자열

2. 메소드

3. 지역변수와 전역변수

4. 메소드의 반환값과 매개변수

01

문자열

01 문자열

■ 문자열 메소드의 개념

- JAVA 는 문자열을 저장하는 String 클래스를 제공
- String 클래스에 다양한 메소드를 내장시켜 편리하게 문자열을 처리할 수 있음
- JAVA는 문자열을 쉽게 사용할 수 있도록 도와주는 메소드를 여러 개 제공하고 있음

```
String 문자열변수;
문자열변수.문자열메소드( );
```

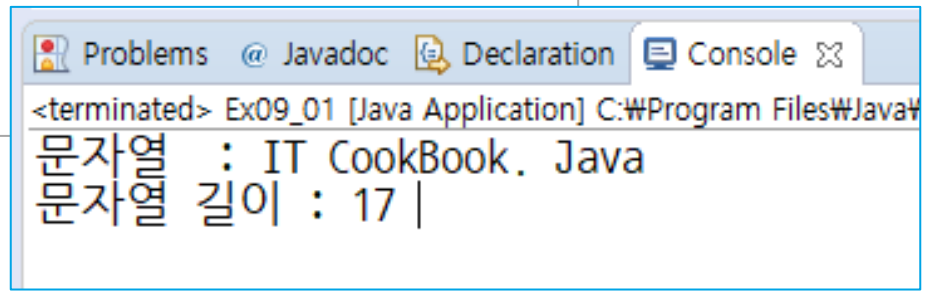
- TIP : 앞서 언급했듯이 메소드(method) 는 미리 만들어진 특정한 기능을 하며 '메소드이름( ) '의 형식임.
  다른 언어에서는 메소드를 함수(function) 또는 멤버 함수(member function) 라 함.
  또한 필드(field) 는 객체의 상태를 나타내며 '변수이름.필드'의 형식. 필드는 다른 말로 속성이라 함

01 문자열

■ 문자열 메소드의 개념

▪ [실습9-1] length( )메소드 사용 예1

```
1  public class Ex09_01 {
2      public static void main(String[] args) {
3
4          String str = "IT CookBook. Java";  //문자열을 선언하고 대입
5
6          int len;    //문자열 배열과 길이를 저장할 변수를 선언
7
8          len = str.length();
9
10         System.out.printf("문자열  : %s \n", str);    //문자열을 출력함
11         System.out.printf("문자열 길이 : %d ", len);  //문자열 길이를 출력함
12
13
14     }
15 }
```



- 6행 : length() 메소드를 사용하여 문자열의 길이를 구함

■ 문자열 메소드의 개념

- [실습9-2] length( )메소드 사용 예2
- 문자열의 길이 활용, 문자열을 입력받은 후 알파벳 O를 $로 변경하여 출력하는 프로그램

```
1  import java.util.Scanner;
2
3  public class Ex09_02 {
4      public static void main(String[] args) {
5          Scanner s = new Scanner(System.in);
6          String str;
7
8          System.out.print("문자열 입력 ==> ");
9
10         str = s.nextLine();    //문자열을 키보드로 입력받는다
11
12         System.out.print("출력 문자열 ==> ");
13         for (int i = 0; i < str.length(); i++) {
14             if (str.charAt(i) == 'o')
15                 System.out.printf("%c", '$');
16             else
17                 System.out.printf("%c", str.charAt(i));
18         }
19
20         s.close();
21     }
22 }
```

문자열의 길이만큼 반복한다

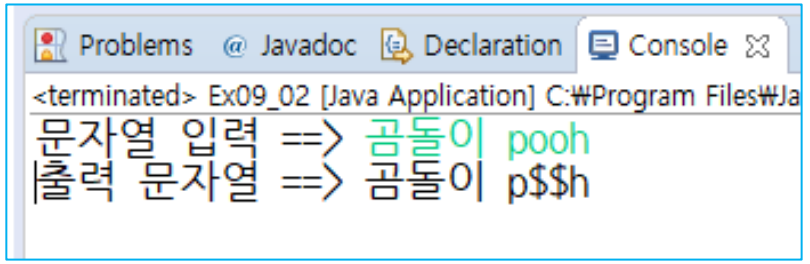
문자가 O이면 $를 출력하고, 아니면 원래 문자를 출력한다

9행에서 문자열을 입력받아 str변수에 저장

12행의 str.length()는 입력한 문자열의 길이를 반환

12행~17행 문자열의 길이만큼 반복

14행의 charAt(위치) 메소드는 문자열의 '위치'에 있는 문자 하나를 반환함. 즉 i가 문자열의 개수만큼 반복되므로 문자열의 모든 문자를 한번씩 꺼내온다. 이 문자가 o 라면 $를 출력하고 그 외에는 원래 문자 출력함



01 문자열

■ 문자열의 처음 또는 끝이 특정 문자열인지 확인하는 startsWith( ), endsWith( )

- 돌려주는 값은 논리형의 true와 false

```
String str = "IT CookBook";
boolean yn = str.startsWith("A") ;
```

■ 문자열의 처음 또는 끝이 특정 문자열인지 확인하는 startsWith( ), endsWith( )

- [실습9-3] startsWith(), endsWith() 사용 예
- 입력한 문자열의 처음과 끝이 괄호인지 확인하고, 괄호가 아니면 처음과 끝에 괄호를 넣는 코드를 작성

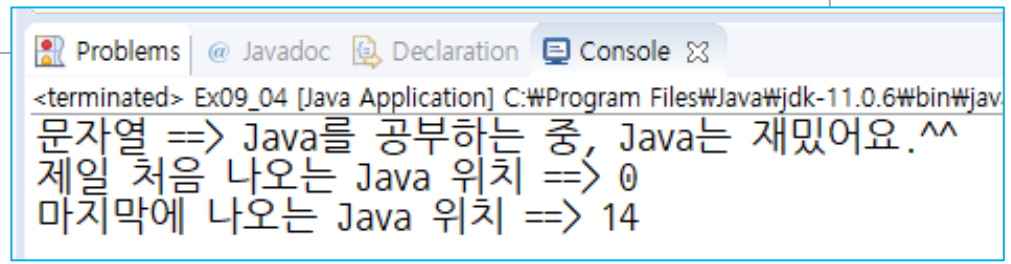
```java
import java.util.Scanner;

public class Ex09_03 {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String str;

        System.out.print("문자열 입력 ==> ");
        str = s.nextLine();

        System.out.print("출력 문자열 ==> ");

        if (!str.startsWith("("))     //문자열의 시작이 (가 아니면 (를 출력한다.
            System.out.printf("(");

        for (int i = 0; i < str.length(); i++)   //입력한 문자를 모두 출력한다.
            System.out.printf("%c", str.charAt(i));

        if (!str.endsWith(")"))       //문자열의 마지막이 )가 아니면)를 출력한다.
            System.out.printf(")");

        s.close();
    }
}
```

```
Problems  Javadoc  Declaration  Console
<terminated> Ex09_03 [Java Application] C:₩Program Files₩Java₩
문자열 입력 ==> 즐거운 java 생활
출력 문자열 ==> (즐거운 java 생활)
```

■ 특정 문자열의 위치를 찾는 indexOf( ), lastIndexOf( )

- [실습9-4] indexOf(), lastindexOf() 사용 예
- indexOf() 메소드 : 찾고자 하는 문자열이 맨 처음 나오는 위치를 돌려줌
- lastIndexOf( ) : 찾고자 하는 문자열이 여러 개 나올 경우 마지막에 나오는 위치를 알려줌
- 찾고자 하는 문자열이 없으면 -1 반환

```
1  public class Ex09_04 {
2      public static void main(String[] args) {
3          String str = "Java를 공부하는 중, Java는 재밌어요.^^";
4
5          System.out.println("문자열 ==> " + str);
6
7          System.out.print("제일 처음 나오는 Java 위치 ==> ");
8
9          System.out.println(str.indexOf("Java"));   //"Java" 글자가 처음 나오는 위치를 출력한다.
10
11         System.out.print("마지막에 나오는 Java 위치 ==> ");
12
13         System.out.println(str.lastIndexOf("Java")); //"Java" 글자가 마지막 나오는 위치를 출력한다.
14     }
15 }
```



01 문자열

■ 문자열을 바꿔주는 replace( )

- "Java" 를 "자바" 로 변경

```
String str1 = "Java를 공부 중... Java는 즐겁습니다. ^^";
String str2 = str1.replace("Java", "자바");
```

■ 일부 문자열을 추출하는 substring( )

- 0번째(맨 앞)에서 네 글자 추출하므로 "Java" 추출

```
String str1 = "Java를 공부 중... Java는 즐겁습니다. ^^";
String str2 = str1.substring(0, 4);
```

■ 문자열을 분리하는 split( )

- 문자열을 특정문자로 분리함, 지정된 문자에 의해 문자열이 분리되기 때문에 결과는 배열로 저장됨
- 다음은 문자열이 쉼표(,)로 분리되어 3개의 배열로 분리됨

```
String str1 = "IT,CookBook,Java";
String str2[] = str1.split(",");
```

01 문자열

```
String str1 = "Java를 공부하는 중...Java는 즐겁습니다.^^";
String str2 = str1.replace("Java", "자바");
System.out.println(str2);
```

```
자바를 공부하는 중...자바는 즐겁습니다.^^
```

```
String str1 = "Java를 공부하는 중...Java는 즐겁습니다.^^";
String str2 = str1.substring(0,4);
System.out.println(str2);
```

```
Java
```

```
import java.util.Arrays;
String str1 = "IT,CookBook,Java";
String str2[] = str1.split(",");
System.out.println(Arrays.toString(str2));
```

```
[IT, CookBook, Java]
```

01 문자열

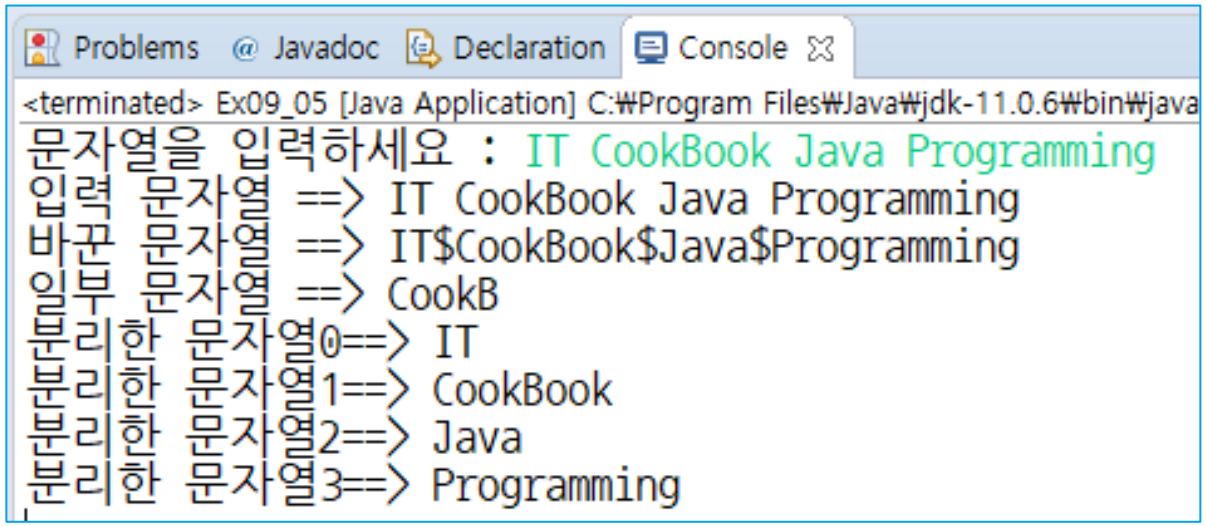
- **[실습9-5] 문자열 처리 메소드 활용 예**

```java
import java.util.Scanner;

public class Ex09_05 {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        String str, strRep, strSub, strAry[]; //입력받을 문자열, 바꿀 문자열, 일부 문자열,
                                              //분리한 문자열 배열을 선언한다.

        System.out.print("문자열을 입력하세요 : ");
        str = s.nextLine();

        strRep = str.replace(" ", "$");   //입력 문자열의 공백을  $로 바꾼다
        strSub = str.substring(3, 8);     //입력 문자열의 세번째부터 여덟번때 문자를 추출한다
        strAry = str.split(" ");          //입력 문자열을 공백으로 분리한다.

        System.out.println("입력 문자열 ==> " + str);
        System.out.println("바꾼 문자열 ==> " + strRep);
        System.out.println("일부 문자열 ==> " + strSub);
        for (int i = 0; i < strAry.length; i++)
            System.out.println("분리한 문자열" + i + "==> " + strAry[i]);

        s.close();
    }
}
```

분리한 문자열 배열을 하나씩 출력한다

01 문자열

- **실습9-5 문자열 처리 메소드 활용 예**



■ 대·소문자로 전환하는 toUpperCase( ), toLowerCase( ), 공백 제거하는 trim( )

- 영문인 경우 대문자 전환 : toUpperCase( ), 소문자 전환 : toLowerCase( ),
- 앞뒤의 공백 문자 모두 제거 : trim( )
- [실습9-6]

```
1  public class Ex09_06 {
2      public static void main(String[] args) {
3
4          String str = "   한글   ABCD  efgh  "; //앞뒤와 중간에 공백이 있으며,
5                                                  //한글과 영문 대,소문자 혼용되어 있는 문자열
6
7          System.out.println("원 문자열 ==> [" + str + "]");
8          System.out.println("대문자로  ==> [" + str.toUpperCase() + "]"); //대문자로 변경
9          System.out.println("소문자로  ==> [" + str.toLowerCase() + "]"); //소문자로 변경
10         System.out.println("공백제거  ==> [" + str.trim() + "]");        //공백 제거
11     }
12 }
```

Problems @ Javadoc Declaration Console

<terminated> Ex09_06 [Java Application] C:₩Program Files₩Java₩j

```
원 문자열 ==> [   한글   ABCD  efgh  ]
대문자로  ==> [   한글   ABCD  EFGH  ]
소문자로  ==> [   한글   abcd  efgh  ]
공백제거  ==> [한글   ABCD  efgh]
```

- 8행과 9행에서 각각 대,소문자로 변경 →한글은 영향받지 않음
- trim( ) : 중간의 공백은 없어지지 않음

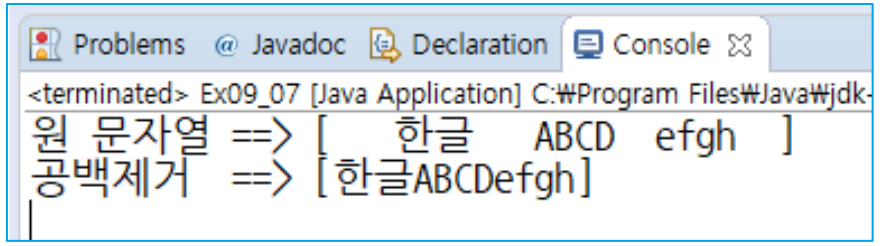
01 문자열

- **[실습9-7] 모든 공백 없애기**

```
1  public class Ex09_07 {
2      public static void main(String[] args) {
3
4          String str = "   한글   ABCD  efgh  ";
5
6          String result = "";   // 결과를 저장할 문자열 변수
7
8          for (int i = 0; i < str.length(); i++) {
9
10             if (str.charAt(i) != ' ')
11                 result += str.substring(i, i + 1);
12
13         }
14
15         System.out.println("원 문자열 ==> [" + str + "]");
16         System.out.println("공백제거  ==> [" + result + "]");
17     }
18 }
```

문자열의 문자개수만큼 반복한다

chatAt(위치)는 해당 위치의 문자를 추출한다.
이 문자가 공백이 아니면 **8**행에서 **result**에 덧붙인다.



- 8행~13행을 보면 문자열의 모든 문자를 체크하여 공백이 아닌 경우에는 result 변수에 추가하고, 공백인 경우에는 그냥 넘어감.
- 결국 공백문자가 제거되는 결과가 나타남

■ 두 문자열 비교하는 compareTo( ), 문자열 포함을 확인하는 contains( )

- **[실습9-8] compareTo( ), contains( ) 사용 예**

```
1  public class Ex09_08 {
2      public static void main(String[] args) {
3          String str1 = "Java Programming";
4          String str2 = "Java IT CookBook";
5
6          System.out.println("원 문자열1 ==> [" + str1 + "]");
7          System.out.println("원 문자열2 ==> [" + str2 + "]");
8
9          System.out.println(str1.compareTo(str2));
10         System.out.println(str1.contains("Java"));
11     }
12 }
```

문자열 변수 2개를 초기화한다.

두 문자열을 비교한다.

"Java"글자가 포함되었는지 확인한다.

```
Problems  Javadoc  Declaration  Console
<terminated> Ex09_08 [Java Application] C:₩Program Files₩Java₩jdk
원 문자열1 ==> [Java Programming]
원 문자열2 ==> [Java IT CookBook]
7
true
```

- 9행의 str1.compareTo(str2)는 str1-str2의 결과를 반환함.
- 결과가 0이 나오면 두 문자열은 완전히 동일한 문자열이고 그외의 숫자는 두 문자열이 다르다는 것을 의미함
- P(아스키값 : 80)와 I(아스키값 : 73)가 다르므로 80-73=7이라는 결과가 나옴
- 10행에서 str1에 "Java"라는 글자가 들어 있으므로 true를 반환함.

■ 두 문자열이 같은지 확인하는 ==과 equals( )

- [실습9-9] ==와 equals( )의 비교
- 두 문자열이 같은지 확인할 때는 ==와 equals()를 사용함
- ==와 equals( ) 둘 다 문자열이 같으면 true, 다르면 false를 반환함

```
1  public class Ex09_09 {
2      public static void main(String[] args) {
3
4          String str1 = "Java Programming";
5          String str2 = "Java Programming";
6          String str3 = new String("Java Programming");
7
8          System.out.println("원 문자열1 ==> [" + str1 + "]");
9          System.out.println("원 문자열2 ==> [" + str2 + "]");
10         System.out.println("원 문자열3 ==> [" + str3 + "]\n");
11
12         System.out.println("문자열1==문자열2 결과 :\t " + (str1 == str2));
13         System.out.println("문자열1.equals(문자열2) 결과 : " + str1.equals(str2));
14         System.out.println("문자열1==문자열3 결과 :\t " + (str1 == str3));
15         System.out.println("문자열1.equals(문자열3) 결과 : " + str1.equals(str3));
16     }
17 }
```

■ 두 문자열이 같은지 확인하는 ==과 equals( )

- **[실습9-9]==와 equals( )의 비교**

```
1 public class Ex09_09 {
2     public static void main(String[] args) {
3
4         String str1 = "Java Programming";
5         String str2 = "Java Programming";
6         String str3 = new String("Java Programming");
```

str1, str2, str3에 동일한 문자열을 저장한다.
str3의 경우 new 연산자를 이용하여 문자열을 초기화하였는데 3, 4행과 동일한 결과를 나타낸다.

- **String 데이터 형식은 4행이나 5행 처럼 직접 문자열을 대입 또는 JAVA에서 제공되는 클래스이므로 6행처럼 new연산자를 사용하여 생성함**
- **4행~6행**
  - 4행에서 생성한 문자열은 메모리공간에 저장됨
  - 5행과 같이 완전히 동일한 문자열은 실제 데이터 하나를 공유하는 개념임
  - 6행은 new 연산자를 사용해서 데이터이 내용은 같지만 별도의 저장공간에 저장됨

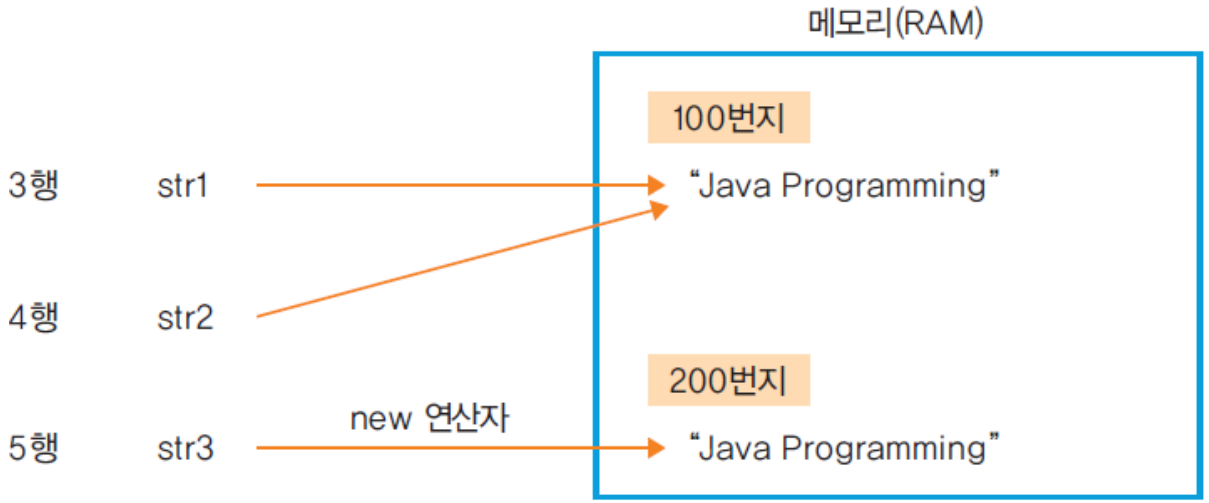


그림 9-10 문자열이 생성되는 구조

■ 두 문자열이 같은지 확인하는 ==과 equals( )

- **[실습9-9]==와 equals( )의 비교**

```
8         System.out.println("원 문자열1 ==> [" + str1 + "]");
9         System.out.println("원 문자열2 ==> [" + str2 + "]");
10        System.out.println("원 문자열3 ==> [" + str3 + "]\n");
11
12        System.out.println("문자열1==문자열2 결과 :\t " + (str1 == str2));
13        System.out.println("문자열1.equals(문자열2) 결과 : " + str1.equals(str2));
14        System.out.println("문자열1==문자열3 결과 :\t " + (str1 == str3));
15        System.out.println("문자열1.equals(문자열3) 결과 : " + str1.equals(str3));
16    }
17 }
```

동일한 문자열이 출력된다.

str1과 **str2**는 완전히 동일하므로
==나 **equals( )** 모두 **true**가 나왔으나,
str1과 **str3**의 비교에서 ==는 **false**가 나왔다.

- equals()는 문자열의 값을 가지고 비교하므로 13행, 15행에서는 true가 나옴
- 즉, str1, str2, str3을 equals() 로 비교하면 모두 "Java Programming" 값이므로 true가 나옴.
- 12행에서는 str1, str2는 완전히 동일하므로 true 나옴
- 14행의 str1, str3은 저장위치가 다르므로 동일하지 않은 데이터로 취급되어 false가 나옴

Problems | Javadoc | Declaration | Console

<terminated> Ex09_09 [Java Application] C:₩Program Files₩Java₩jdk-11.0.6₩bin

```
원 문자열1 ==> [Java Programming]
원 문자열2 ==> [Java Programming]
원 문자열3 ==> [Java Programming]

문자열1==문자열2 결과 :  true
문자열1.equals(문자열2) 결과 : true
문자열1==문자열3 결과 :  false
문자열1.equals(문자열3) 결과 : true
```

- **Tip. 문자열을 비교하는 경우는 대부분 문자열의 내용을 비교, 문자열을 비교할때에는 되도록 == 보다는 equal() 메소드를 사용한다.**

감사합니다