제 23 강 : String 관련 클래스

※ 학습목표
✓ String 클래스의 특징과 사용법을 설명할 수 있다.
✓ StringBuffer 클래스의 특징과 사용법을 설명할 수 있다.
✓ StringTokenizer 클래스의 특징과 사용법을 설명할 수 있다.

1. String Class
✓ 문자열을 의미하는 클래스
✓ 가장 많이 쓰이는 클래스 중의 하나
✓ 정확히 말하면, 객체 자료형이지만 기본 자료형처럼 사용함

① String 클래스 생성자

생성자	설 명
String()	비어있는 문자열 객체를 생성하고 초기화
String(char[] value)	인자인 char배열 value의 내용을 순차적으로 배정하여 새로운 문자열을 생성함
String(String original)	String형의 original의 문자열을 새롭게 생성된 문자열 객체에 초기화함

② 객체 생성법

생성법	예
암시적 객체 생성	String s1 = “TEST“;
명시적 객체 생성	String s2 = new String(“TEST“);

③ 암시적 객체 생성과 명시적 객체 생성의 차이점
✓ ’==’ 연산자는 객체의 주소 값 비교
✓ 객체의 값 비교는 equals() 메서드를 이용함.

[실습]

```
1  package tommy.java.exam01;
2
3  public class StringExOne {
4          public static void main(String[] args) {
5                  String s1 = “Twinkle“;
6                  String s2 = “Twinkle“;
7                  if (s1 == s2)
8                          System.out.println(“s1과 s2는 같다. “);
9                  else
10                         System.out.println(“s1과 s2는 같지 않다. “);
11                 String s3 = new String(“Little Star“);
12                 String s4 = new String(“Little Star“);
13                 if (s3 == s4)
14                         System.out.println(“s3과 s4는 같다. “);
15                 else
16                         System.out.println(“s3과 s4는 같지 않다. “);
17         }
18 }
```

④ String 객체의 불변성

✓ String 객체는 편집되지 않음

✓ 기존의 String 객체에 새로운 String값을 더하면 기존의 객체가 수정되는 것이 아니고, 새로운 객체를 참조하게 됨

✓ 아래 예제에서 str은 하나의 객체로 보이지만 실제로는 5개의 String 객체가 생성됨

[실습]

```
1  package tommy.java.exam02;
2
3  public class StringExTwo {
4          public static void main(String[] args) {
5                  String str = new String();
6                  str += “Hello“;
7                  str += “ Java“;
8                  System.out.println(“str의 값은 ? : “ + str);
9          }
10 }
```

⑤ String 관련 메서드

✓ String str = “Java” ; 라는 코드를 기반으로 아래의 메서드를 정리하면

✓ char ch = str.charAt(int); // int 가 2 라면 ch = ‘v’

✓ int l = str.length(); // i=4

✓ String so = str.toLowerCase(); 소문자 변환 // 대문자 toUpperCase()

✓ String imsi = String.valueOf(수치); 숫자를 문자열로 변환

✓ String imsi = str.trim(); 공백이나 화이트코드를 제거

✓ String imsi = str.substring(1,3); // av

✓ indexOf(), lastIndexOf() // 문자열의 위치 값을 반환함.

2. StringBuffer 클래스

✓ StringBuffer 클래스는 내부적으로 직접 변경이 가능한 클래스

✓ 문자열의 변경이 자주 사용되는 객체일수록 StringBuffer 클래스를 사용

① StringBuffer 클래스 생성자

생성자	설 명
StringBuffer()	비어있는 StringBuffer객체를 생성하고 초기값으로 문자 16개를 기억하고 있는 용량(buffer의 길이)을 가짐
StringBuffer(CharSequence seq)	인자로 전달된 CharSequence와 같은 문자열을 포함한 StringBuffer 객체를 생성함
StringBuffer(int capacity)	int형 capacity의 길이만큼 buffer의 길이를 초기화하여 StringBuffer객체를 생성
StringBuffer(String str)	str을 초기값으로 하고 buffer의 길이를 str+16로 하여 StringBuffer 객체를 생성함

② Buffer란?

✓ 버퍼(buffer)는 데이터를 한 곳에서 다른 한 곳으로 전송하는 동안 일시적으로 그 데이터를 보관하는 메모리의 영역이다.

✓ 버퍼링(buffering)이라 버퍼를 활용하는 방식 또는 버퍼를 채우는 동작을 말한다.

③ StringBuffer 클래스의 메서드

반환형	메서드명	설 명
int	capacity()	현재 용량(buffer의 길이)를 반환함
StringBuffer	append(String str)	현재 StringBuffer 객체에 인자로 전달된 str을 덧붙임
	delete(int start, int end)	start와 end index 사이의 문자열을 삭제
	insert(int offset, String str)	offset index에 str을 삽입함
	replace(int start, int end, String str)	start와 end index 사이의 문자열을 str로 변환함
void	setLength(int newLength)	newLength만큼 문자열의 길이를 다시 설정함
String	toString()	StringBuffer 객체가 가지고 있는 문자열을 String형으로 변환

[실습]

```
1  package tommy.java.exam03;
2
3  public class StringBufferExOne {
4          public static void main(String[] ar) {
5                  StringBuffer sb = new StringBuffer("JAVA");
6                  sb.append("1.8");
7                  System.out.println(sb);
8                  sb.delete(3, 5);
9                  System.out.println(sb);
10                 sb.insert(3, "A1");
11                 System.out.println(sb);
12                 sb.replace(4, 7, "일점팔");
13                 System.out.println(sb);
14                 sb.reverse();
15                 System.out.println(sb);
16         }
17 }
```

[실습] StringBuffer의 동작원리

```
1  package tommy.java.exam04;
2
3  public class StringBufferExTwo {
4          public static void main(String[] args) {
5                  StringBuffer sb = new StringBuffer("1234567890");
6                  System.out.println("sb의 buffer 용량은 ? " + sb.capacity());
7                  System.out.println("sb 문자열의 길이는 ? " + sb.length());
8                  sb.append("ABCDEFGHIJKLMNOPQ");
9                  System.out.println("sb의 buffer 용량은 ? " + sb.capacity());
10                 System.out.println("sb 문자열의 길이는 ? " + sb.length());
11                 sb.delete(1, 20);
12                 System.out.println("sb의 buffer 용량은 ? " + sb.capacity());
13                 System.out.println("sb 문자열의 길이는 ? " + sb.length());
14         }
15 }
```

3. StringTokenizer 클래스

✓ java.util 패키지에 포함되어 있음

✓ "2022/11/28" 이라는 문자열을 각각 "2022", "11", "28"로 분리할 때 사용

✓ 각각의 문자열을 토큰(Token)이라고 함

① StringTokenizer 클래스의 생성자

생성자	설 명
StringTokenizer(String str)	str을 기본 구분문자인 white space, tab, new line등의 구분문자로 하여 분할할 StringTokenizer 객체를 생성함
StringTokenizer (String str, String delim)	str을 두 번째 인자인 delim으로 구분지어 분할할 StringTokenizer객체를 생성함
StringTokenizer(String str, String delim, boolean returnDelims)	str을 두 번째 인자인 delim으로 구분지어 분할할 StringTokenizer 객체를 생성할 때 delim 역시 token 자원으로 사용할 것인지를 returnDelims로 결정함

② StringTokenizer 클래스의 메서드

반환값	메서드명	설 명
int	countTokens()	token된 자원의 수를 반환함
boolean	hasMoreTokens()	token할 수 있는 자원이 있을 경우 true, 없으면 false를 반환함
String	nextToken()	token된 자원을 반환함

[실습]

```
1  package tommy.java.exam05;
2
3  import java.util.StringTokenizer;
4
5  public class StringTokenEx1 {
6          StringTokenizer st;
7
8          public StringTokenEx1(String str) {
9                  System.out.println("str : " + str);
10                 st = new StringTokenizer(str);
11         }
12         public StringTokenEx1(String str, String delim) {
13                 System.out.println("str : " + str);
14                 st = new StringTokenizer(str, delim);
15         }
16         public void print() {
17                 System.out.println("Token count : " + st.countTokens());
18                 while (st.hasMoreTokens()) {
19                         String token = st.nextToken();
20                         System.out.println(token);
21                 }
22                 System.out.println("----------------------------");
23         }
24         public static void main(String[] args) {
25                 StringTokenEx1 st1 = new StringTokenEx1("Happy day");
26                 st1.print();
27                 StringTokenEx1 st2 = new StringTokenEx1("2022/11/15", "/");
28                 st2.print();
29         }
30 }
```

③ String 클래스의 split() 메서드와 StringTokenizer의 차이

✓ split() 메서드의 경우 무의미한 공백(white Space)도 자리를 차지함

✓ StringTokenizer의 경우 무의미한 공백을 무시하고 제거함

[실습]

```
1  package tommy.java.exam06;
2
3  import java.util.StringTokenizer;
4
5  public class StringTokenEx2 {
6          public static void main(String[] args) {
7                  String str = "학교,집,,게임방";
8                  StringTokenizer tokens = new StringTokenizer(str, ",");
9                  for (int x = 1; tokens.hasMoreTokens(); x++) {
10                         System.out.print("문자" + x + " = " + tokens.nextToken() + "\t");
11                 }
12                 System.out.println("");
13                 System.out.println("============================");
14                 String[] values = str.split(",");
15                 for (int x = 0; x < values.length; x++) {
16                         System.out.print("문자" + (x + 1) + " = " + values[x] + "\t");
17                 }
18                 System.out.println("");
19         }
20 }
```