**18회차(1)**

package test01;

public class Test01 {

public static void main(String[] args) {

// 리터럴(literal) 표현

// primitive type(int, double, ..)에서만 사용가능한 표현

// String 타입은 참조형 변수이기 때문에 원래는 new 연산자를 사용해야 하지만

// 자바에서는 String에 한해서 특별취급(-> 리터럴 표현 사용가능)

String str = "Hello Java";

// 사실은 String도 new를 통해 객체를 생성

// but 자바에서 문자열 객체를 생성할 때 new를 사용하는 경우는 거의 없다

String s = new String("Hello Java");

System.out.println(str);

System.out.println(s);

// 문자열도 배열과 같은 형태라면 인덱싱을 통해서 접근이 가능해야 하지만

// 자바는 문자열의 원소에 대한 직접적인 접근을 허용하지 않는다!

// System.out.println(str[0]); -> 오류

// => 그래서 이런 경우에 charAt을 이용해서 원소에 우회참조가 가능!

System.out.println(str.charAt(0));

}

}

**18회차(2)**

package test01;

public class Test02 {

public static void main(String[] args) {

String str = "Hello java";

// str 객체에서 'j' 문자의 위치를 알고 싶다면

System.out.println(str.indexOf('j'));

// str 객체에서 "java" 문자열의 위치를 알고 싶다면

// 문자열인 경우에는 문자열이 시작하는 위치를 돌려주게 된다

System.out.println(str.indexOf("java"));

}

}

**18회차(3)**

package test01;

public class Test03 {

public static void main(String[] args) {

String str1 = "java";

// Java와 java는 다른 문자열이다

System.out.println(str1.equals("Java"));

// 전체 문자열을 다른 문자열로 치환한다

System.out.println(str1.replaceAll("java", "World"));

String str2 = "Hello java";

// "Hello"라는 문자열만 추출

System.out.println(str2.substring(0, 5)); // 인덱스 0부터 인덱스 4까지

// "Java"라는 문자열만 추출

System.out.println(str2.substring(6)); // 인덱스 6부터 끝까지

}

}

**18회차(4)**

package test01;

public class Test04 {

public static void main(String[] args) {

// 공백으로 구분되어 있는 문자열

String str = "Hello java";

// 예제의 문자열은 공백을 기준으로 분할해 볼 수 있다

// split을 하면 구분자를 기준으로 분할된 배열을 돌려준다

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

System.out.println(result);

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

System.out.println(result[i]);

}

// 아래와 같은 경우를 가정

System.out.println("3 is sum of 1 + 2");

// 실제로는 이렇게 출력하고 싶지 않았을까?

// 그러나 문자열 내에서는 연산이 불가능하다

System.out.println("1 + 2 is sum of 1 + 2");

// => 이럴 때 format을 이용하면 매우 유용하게 사용할 수 있다

// 이렇게! -> 자리에 대입된다

System.out.println(String.format("%d is sum of 1 + 2", 1 + 2));

// => 그래서 이렇게도 사용할 수 있다

int a = 1;

int b = 2;

System.out.println(String.format("%d is sum of %d + %d", a + b, a, b));

// System.out.printf를 더 많이 사용한다

// System.out.print + format

System.out.printf("%d is sum of %d + %d", a + b, a, b);

}

}

**18회차(5)**

package test01;

public class Test05 {

public static void main(String[] args) {

/\*

\* 1) 문자열을 숫자로 변환 : Integer.parseInt()

\* String year = "89”;

\* int n = Integer.parseInt(year);

\* System.out.println(n);

\* 2) 숫자를 문자열로 변환 : Integer.toString()

\* String str = Integer.toString(n);

\* System.out.println(n);

\*/

String jumin = "8920101-2012932";

// 문제 1) 나이 출력

// 정답 1) 33세

String age = jumin.substring(0, 2);

System.out.println(age);

// 적당한 형태로 변형 -> 1989

int year = Integer.parseInt(age);

year += 1900;

System.out.println("나이 : " + (2022 - year + 1) + "세");

// 문제 2) 성별 출력

// 정답 2) 여성

char gender = jumin.charAt(8);

if (gender == '1' || gender == '3') {

System.out.println("남성");

}

else if (gender == '2' || gender == '4') {

System.out.println("여성");

}

}

}

**18회차(6)**

package test01;

import java.util.Arrays;

public class Test06 {

public static void main(String[] args) {

String str = "김철수/87,이만수/42,이영희/95";

// 이름은 name 배열에, 성적은 score 배열에 각각 저장하고 출력

/\*

\* 내 풀이방법 String name[] = { str.substring(0, 3), str.substring(7, 10),

\* str.substring(14, 17) }; for (int i = 0; i < name.length; i++) {

\* System.out.println(name[i]); }

\* String score[] = { str.substring(4, 6), str.substring(11, 13),

\* str.substring(18, 20) }; for (int j = 0; j < score.length; j++) {

\* System.out.println(score[j]); }

\*/

String name[] = new String[3];

int[] score = new int[3];

// split을 응용해서 작성

// ,를 기준으로 분리!

String tmp[] = str.split(",");

// if 배열은 반복문 없이 한 번에 출력하고자 한다면

System.out.println(Arrays.toString(tmp));

// 분리된 배열을 이용해서 각각의 원소를 /를 기준으로 분리

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

String[] tmp2 = tmp[i].split("/");

name[i] = tmp2[0];

score[i] = Integer.parseInt(tmp2[1]);

}

System.out.println(Arrays.toString(tmp));

System.out.println(Arrays.toString(name));

System.out.println(Arrays.toString(score));

// 문제) 이름과 성적을 다시 원래의 문자열로 연결

// 정답) "김철수/87,이만수/42,이영희/95"

// 문자열과 문자열은 "+"을 이용해서 하나의 문자열로 만들 수 있다.

// 예를 들면

String exam = "Hello" + "java";

System.out.println(exam);

// 새로 만들 문자열

String newStr = "";

// name과 score를 반복문으로 순회하면서 "+"(추가)해주면 된다

// name과 score의 길이가 동일하므로 어떤걸 기본으로 해도 상관없다

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

newStr += name[i];

newStr += "/";

newStr += score[i];

newStr += ","; // 약간의 문제가 생긴다 -> 이영희/95 뒤에도 쉼표가 생긴다

}

System.out.println(newStr);

}

}

**18회차(7)**

#API : 프로그램에서 자주 사용되는 클래스 및 인터페이스들의 모음

( \*패키지 : 여러 개의 클래스가 들어있는 폴더 )

( \*인터페이스 : 추상 메소드로만 이루어진 클래스 )

- java.lang.\*;

- String

-> String도 객체이다. - 생성자 오버로딩, 4가지 형태의 생성자가 존재한다.

- charAt(int index)

: 문자열 내에서 특정 문자의 위치를 의미 (0부터 시작)

- 주로 문자열을 반복적으로 처리하고자 하는 경우에 주로 사용

- 문자열은 배열과 동일한 형태, 즉 문자들의 집합

- 자바에서는 문자열내의 원소(문자)에 직접 접근할 수 있도록 되어있다.

- indexOf(int ch) / (String str)

: charAt과는 반대로 주어진 문자열에 대해서 해당 문자열이 시작되는 위치를 알려준다

- api 문서를 보면 알겠지만 4가지의 메소드가 오버로딩으로 정의되어 있다

- 단일문자 또는 문자열에 대한 위치정보를 돌려준다.

- equals(Object object)

: 문자열을 비교하는 경우에(같은 문자열인지 아닌지를 판단)

- 주어진 문자열이 같으면 True, 다르면 False

- replace(CharSequence target, CharSequence replacement)

: 특정 문자열로 치환하는 기능

- 메모장에서 바꾸기 기능과 동일한 기능

- replaceAll 형태도 가능

- substring(int beginIndex) / (int beginIndex, int endIndex)

: 특정 문자열만 추출하고 싶은 경우에 사용

- split

\*(문자열을 처리할 때 정말 많이 사용하는 기능)

: 문자열을 특정 구분자에 맞춰서 분할할 때 사용

- format

: 출력하고 싶은 문자열을 꾸밀 때 사용

- "3 is sum of 1 + 2"를 가정해보자

<=> System.out.println(String.format("%d is sum of 1 + 2", 1 + 2));

<=> System.out.printf("%d is sum of %d + %d", a + b, a, b);

(+)

1) 문자열을 숫자로 변환 : Integer.parseInt()

2) 숫자를 문자열로 변환 : Integer.toString()

- java.util.collection;

- List

- Map

- Set