Data Structure

(Java programming)

Chapter 02.

Contents

- Simple Java program
- Exercise1-Factorial
- Exercise2-Fibonacci

- ***Simple Java Program***

• Simple Java Program

- Scanner Keyboard = new Scanner(System.in);

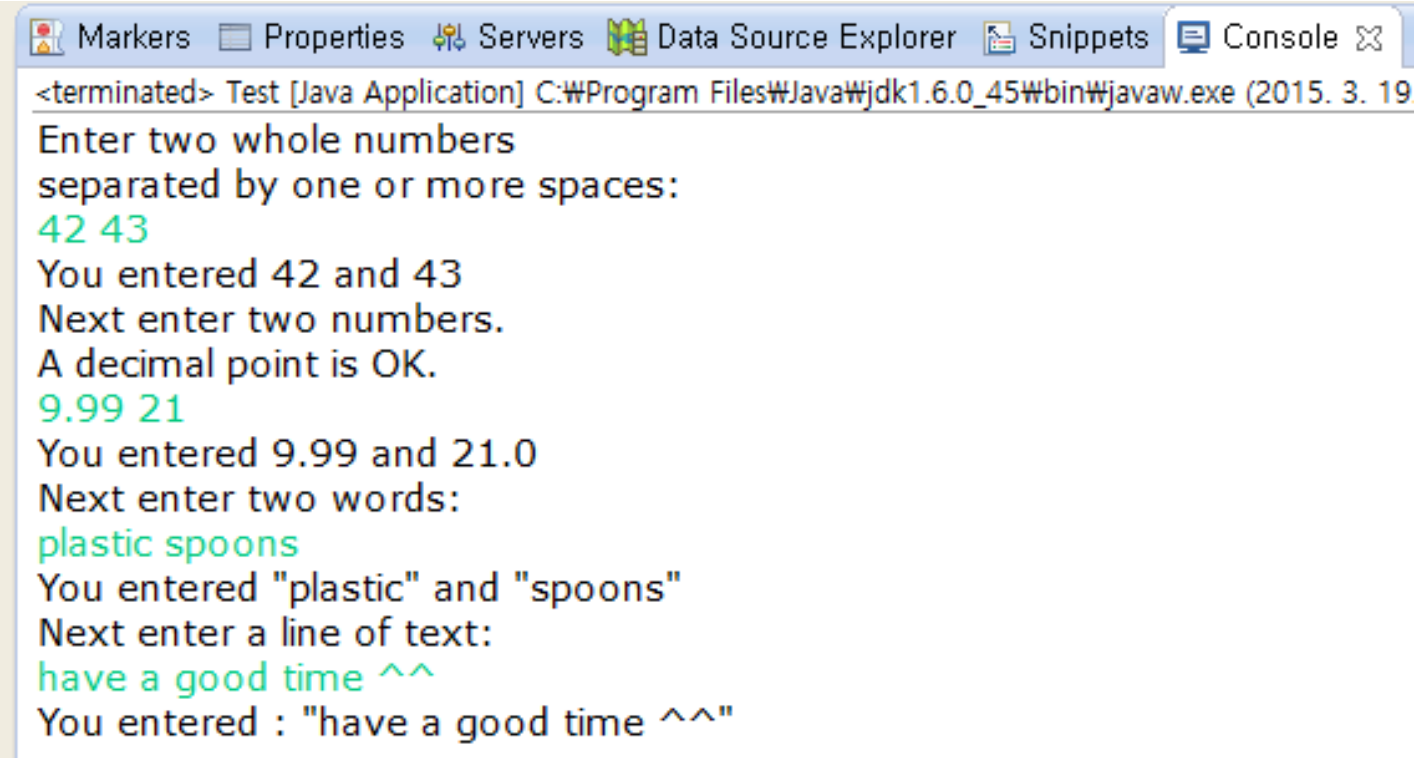
☞ Declaration variable name(keyboard) and new scanner object

☞ int n1 = keyboard.nextInt();
☞ double d1 = keyboard.nextInt();
☞ boolean b1 = keyboard.nextBoolean();
☞ String s1 = keyboard.next();

• Simple Java Program

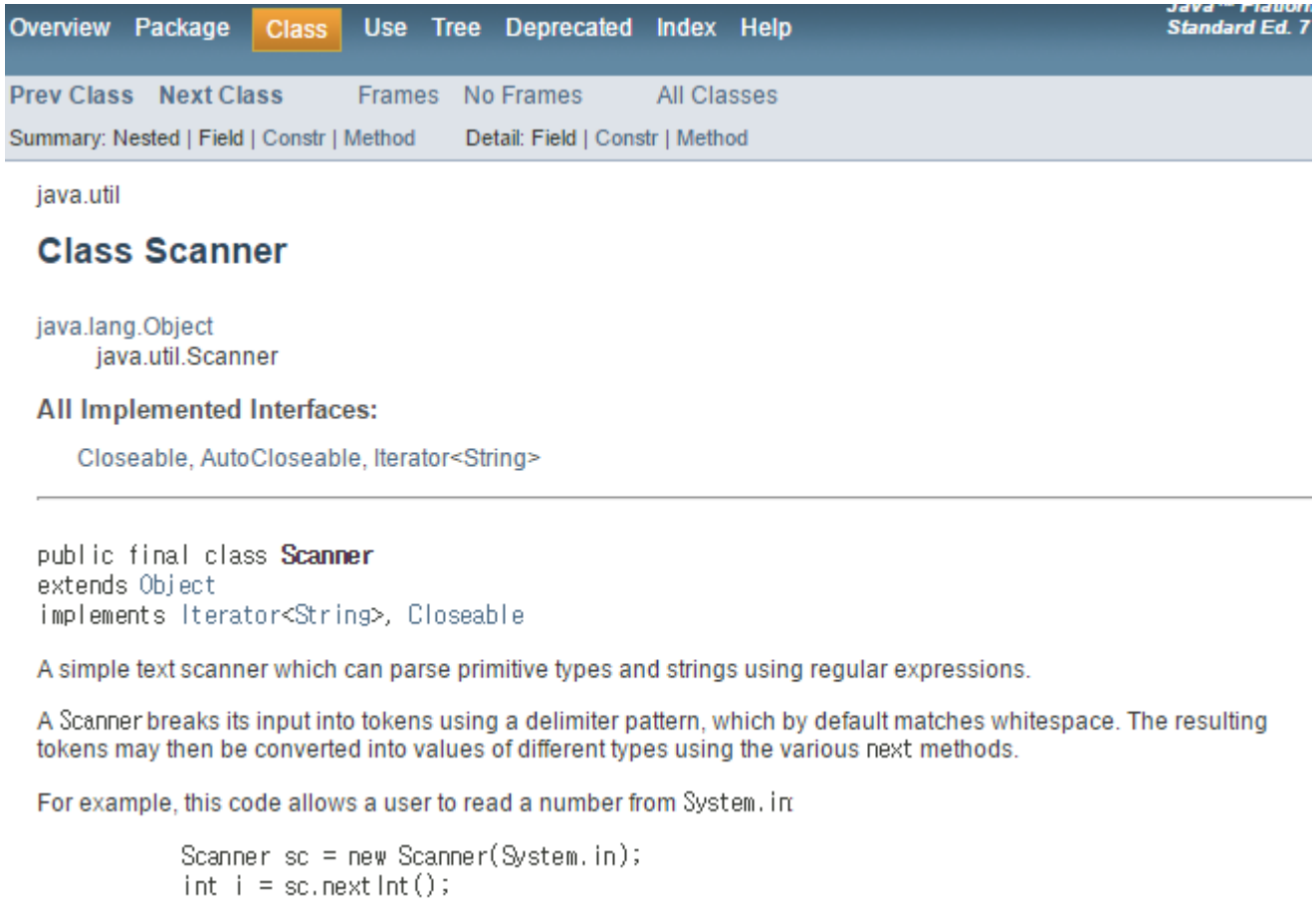
```
import java.util.Scanner;

public class ScanTest
{
        public static void main(String[] args)
        {
                Scanner keyboard = new Scanner(System.in);
                System.out.println("Enter two whole numbers");
                System.out.println("separated by one or more spaces:");
                int n1, n2;
                n1 = keyboard.nextInt();
                n2 = keyboard.nextInt();
                System.out.println("You entered " + n1 + " and " + n2);
                System.out.println("Next enter two numbers.");
                System.out.println("A decimal point is OK.");
                double d1, d2;
                d1 = keyboard.nextDouble();
                d2 = keyboard.nextDouble();
                System.out.println("You entered " + d1 + " and " + d2);
                System.out.println("Next enter two words:");
                String s1, s2;
                s1 = keyboard.next();
                s2 = keyboard.next();
                System.out.println("You entered ₩"" + s1+ "₩" and ₩"" +s2 + "₩"");
                s1 = keyboard.nextLine();
                System.out.println("Next enter a line of text:");
                s1 = keyboard.nextLine();
                System.out.println("You entered : ₩"" + s1 + "₩"");
        }
}
```

• Simple Java Program



• Simple Java Program

http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html



Overview Package Class Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.util

Class Scanner

java.lang.Object
    java.util.Scanner

All Implemented Interfaces:

Closeable, AutoCloseable, Iterator<String>

```
public final class Scanner
extends Object
implements Iterator<String>, Closeable
```

A simple text scanner which can parse primitive types and strings using regular expressions.

A Scanner breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various next methods.

For example, this code allows a user to read a number from System.in:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

• Simple Java Program

http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

boolean	**nextBoolean**() Scans the next token of the input into a boolean value and returns that value.
byte	**nextByte**() Scans the next token of the input as a byte.
byte	**nextByte**(int radix) Scans the next token of the input as a byte.
double	**nextDouble**() Scans the next token of the input as a double.
float	**nextFloat**() Scans the next token of the input as a float.
int	**nextInt**() Scans the next token of the input as an int.
int	**nextInt**(int radix) Scans the next token of the input as an int.
String	**nextLine**() Advances this scanner past the current line and returns the input that was skipped.
long	**nextLong**() Scans the next token of the input as a long.

• Simple Java Program

http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html#nextInt()

nextInt

```
public int nextInt()
```

Scans the next token of the input as an `int`.

An invocation of this method of the form `nextInt()` behaves in exactly the same way as the invocation `nextInt(radix)`, where `radix` is the default radix of this scanner.

Returns:

the `int` scanned from the input

Throws:

`InputMismatchException` - if the next token does not match the *Integer* regular expression, or is out of range

`NoSuchElementException` - if input is exhausted

`IllegalStateException` - if this scanner is closed

• Simple Java Program

- if(condition)

        command;

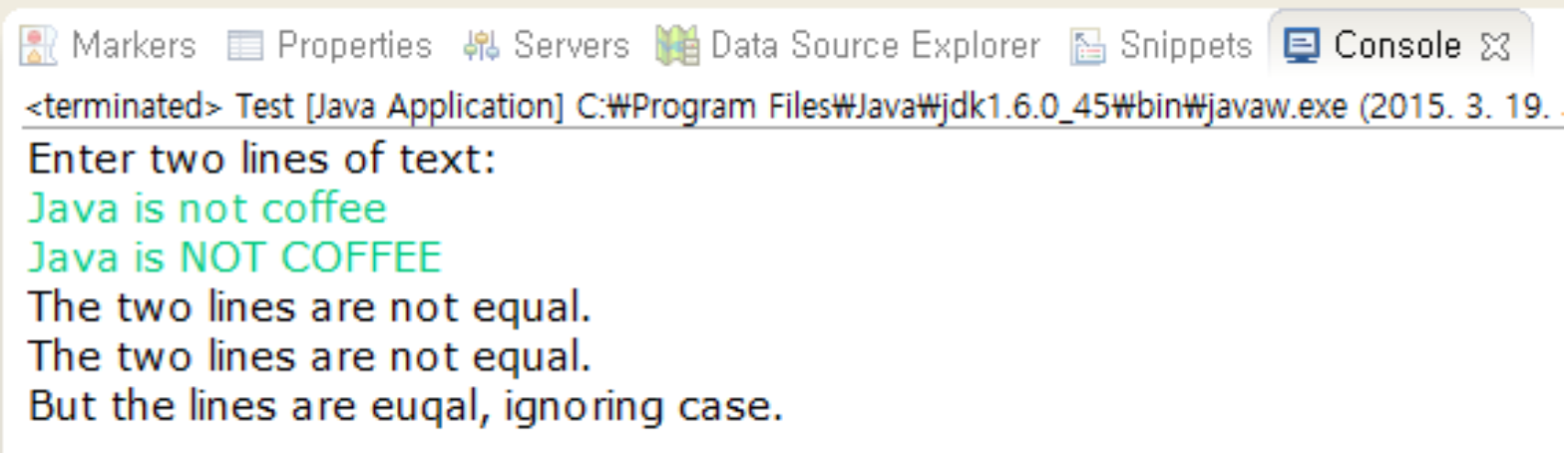
  else

        other command;

• Simple Java Program

```
import java.util.Scanner;

public class StringEqualityDemo
{
        public static void main(String[] args)
        {
                String s1, s2;
                System.out.println("Enter two lines of text:");
                Scanner keyBoard = new scanner(System.in);
                s1 = keyBoard.nextLine();
                s2 = keyBoard.nextLine();

                if(s1.equals(s2))
                        System.out.println("The two lines are equal.");
                else
                        System.out.println("The two lines are not equal.");
                if(s2.equals(s1))
                        System.out.println("The two lines are equal.");
                else
                        System.out.println("The two lines are not equal.");
                if(s1.equalsIgnoreCase(s2))
                        System.out.println("But the lines are euqal, ignoring case.");
                else
                        System.out.println("Lines are not equal, even ignoring case.");
        }
}
```

- # Simple Java Program



• Simple Java Program

http://docs.oracle.com/javase/7/docs/api/java/lang/String.html

	Returns a String that represents the character sequence in the array specified.
boolean	**endsWith**(**String** suffix) Tests if this string ends with the specified suffix.
boolean	**equals**(**Object** anObject) Compares this string to the specified object.
boolean	**equalsIgnoreCase**(**String** anotherString) Compares this String to another String, ignoring case considerations.
static **String**	**format**(**Locale** l, **String** format, **Object**... args) Returns a formatted string using the specified locale, format string, and arguments.
static **String**	**format**(**String** format, **Object**... args) Returns a formatted string using the specified format string and arguments.
byte[]	**getBytes**() Encodes this String into a sequence of bytes using the platform's default charset, storing the
byte[]	**getBytes**(**Charset** charset)

- ***Exercise1-Factorial***

• Exercise1-Factorial

- Recursive factorial
  - $n! = 1 \cdot 2 \cdot 3 \cdot \cdots \cdot (n-1) \cdot n$
  - Recursive definition of $f(n) = n!$

$$f(n) = \begin{cases} 1 & \text{where } n = 0 \\ n \cdot f(n-1) & \text{otherwise} \end{cases}$$

  - Factorial implementation in Java using recursion

```
// recursive factorial function
public static int recursiveFactorial(int n) {
   if (n == 0) return 1;
   else return n * recursiveFactorial(n - 1);
}
```

  - Result requirements :
    - Input : integer value using Scanner class.
    - Output : Execution time, Factorial results.

• Exercise1-Factorial

```
import java.util.Scanner;


public class FactorialTest {

    public static int recursiveFactorial(int n){
        if(n==0) return 1;
        else return n*recursiveFactorial(n-1);
    }

    public static void main(String []args){


                ...



    }
}
```

- ***Exercise2-Fibonacci***

• Exercise2-Fibonacci

- Fibonacci

$$F_0 = 0$$
$$F_1 = 1 \qquad \text{where } n \leq 1$$
$$F_n = F_{n-1} + F_{n-2} \qquad \text{where } n > 1$$

– Specification of Fibonacci numbers as a binary recursive algorithm according to its definition

```
Algorithm BinaryFib(n)
Input: Nonnegative integer n
Output: The nth Fibonacci number F_n
    if n ≤ 1 then return n
    else return BinaryFib(n - 1) + BinaryFib(n - 2)
```

– Result requirements :
  - Input : integer value using Scanner class.
  - Output : Execution time, Factorial results.

• Exercise2-Fibonacci(con't)

- Fibonacci
  - A better Fibonacci Algorithm(1)

```
Algorithm LinearFib(n):
    Input: A nonnegative integer n
    Output: Pair of Fibonacci numbers (F_n, F_{n-1})
      if n ≤ 1 then
          return (n, 0)
      else
          (i, j) = LinearFib(n - 1)
          return (i + j, i)
```

  - Result requirements :
    - Input : integer value using Scanner class.
    - Output : Execution time, Fibonacci results.

• Exercise2-Fibonacci(con't)

- Fibonacci
  - A better Fibonacci Algorithm(2)

```
Algorithm LinearFib2(n):
    Input: A nonnegative integer n
    Output: The nth Fibonacci number F_n
      if n ≤ 1 then return (n)
      prev = 0, curr = 1
      for 2 ≤ i ≤ n
        next = curr + prev
        prev = curr, curr = next
      return next
```

  - Result requirements :
    - Input : integer value using Scanner class.
    - Output : Execution time, Fibonacci results.

• Two return value.

Class 버전

```
public class Example {
  static int i;
  static int j;
  Example r = new Example();
  static Example LinearFibonacci(){
    r.i = 10;
    r.j = 100;
    return r;
  }

  public static void main(String[] args) {
    LinearFibonacci();
    System.out.println(r.i +", "+ r.j);
  }
}
```

Array 버전

```
public class Example {
  static int result[]= new int[2];
  static int[] LinearFibonacci(){
    result[0] = 1;
    result[1] = 100;
    return result;
  }

  public static void main(String[] args) {
    LinearFibonacci();
    System.out.println(result[0]+", "+ result[1]);
  }
}
```

• Submission

e-mail : 2017kudatastructure@gmail.com

Attach methods :

1. Create zip file (java project folder)
2. Modify the file names :
   ID_name_datastructure[2403 or 2404 or 2405].dat
   ex) 201173378_이명재_datastructure[2403].dat
   (2403:A반 / 2404:B반 / 2405:C반)
3. e-mail title :
   datastructure_name_chapter
   ex) datastructure_최수용_chapter2[2404]