02_3 Screen I/O

Object-Oriented Programming

이번 강의에서는 Screen I/O 에 관해 강의하겠습니다.

Keyboard Input (1/4)

```java
import java.util.Scanner;

public class ScannerClass {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter an integer: ");         // Enter an integer:
        int number = scanner.nextInt();                 // Enter an integer: 243
        System.out.println("You entered: " + number);   // You entered: 243

        System.out.print("Enter a double number: "); // Enter a double number: 32.534
        double dnumber = scanner.nextDouble();       // Enter a double number: 32.534
        System.out.println("You entered: " + dnumber); // You entered: 32.534
```


페이지 2

Keyboard input을 받기 위해 필요한 Scanner class는
java.util package에 들어 있습니다.
java.util package는 Java에서 default로 import를 해 주지 않기 때문에
명백하게 import를 해야 합니다.
그래서 첫줄에 보면 'import java.util.Scanner' 로 Scanner class를 포함시켰습니다.
import 할 때 java.util package의 모든 class를 포함하고 싶으면
'import java.util.*' 이라고 하면 되는데,
여기서는 Scanner class만을 필요로 하기 때문에
그것만을 import 하였습니다.
main method 내에서 scanner object를 하나 생성하기 위해
new Scanner(System.in); 을 실행합니다.
이를 통해 scanner는 하나의 Scanner object를 가리키는 reference type variable이 됩니다.
Program이 keyboard input을 받아들이기 위해
프로그램을 사용하는 user에게 안내를 해 줄 필요가 있습니다.
그래서 어떤 입력을 어떤 형식으로 하라는 안내를 해 주는 말을 화면에 보여주는데
이를 prompt라 부릅니다.
이 예제에서는 먼저 정수 하나를 입력받기 위해
"Enter an integer: " 라는 prompt를 프린트 하였습니다.
keyboard 입력을 받기 위한 cursor가 prompt 뒤에 바로 위치하게 하기 위해
System.out.println 이 아닌 Sytem.out.print 를 사용하여 prompt를 보여줍니다.
정수를 하나 입력받는 Scanner의 method는 nextInt() 입니다.
Keyboard로 정수를 하나 입력하고 enter를 치면
그 정수 값이 읽혀져 return 되고 program의 variable 'number' 에 저장됩니다.
입력이 잘 되었는지를 알아보기 위해 입력된 값을 "You entered: " 라는 String과 함께
screen에 출력하여 확인하도록 하였습니다.
그 아래에는 double number를 하나 keyboard를 통해 입력 받는 코드가 있습니다.
double number를 하나 입력받는 Scanner의 method는 nextDouble() 입니다.
이 예제에서는 예를 들어 32.534라는 double number를 keyboard로 입력하였습니다.

Keyboard Input (2/4)

```
System.out.print("Enter a word: ");           // Enter a word:
String word = scanner.next();                 // Enter a word: Korea
System.out.println("You entered: " + word);   // You entered: Korea

System.out.print("Enter a line of text: ");   // Enter a line of text:
String line = scanner.nextLine();
System.out.println("You entered: (" +
                   line + ")");       // Enter a line of text: You entered: ()
```


페이지 3

이 부분에서는 String 하나를 keyboard로 입력하는 것을 보여줍니다.
이 때 사용하는 Scanner의 method는 next() 입니다.

그 아래 부분에 나오는 것은
blank 를 무시하고 현재 cursor에서 new line 전까지의 모든 character를
하나의 String으로 읽어 오는 부분으로
nextLine() method를 사용합니다.

결국 이 프로그램의 의도는 blank가 없는 단어를 word로 하나 읽어온 후
그 다음에 또 다른 line을 입력하여 한 줄 전체를 nextLine() 으로 입력하는 것이었을 겁니다.
그러나 comment 부분의 output을 보면
"Korea" 라는 하나의 단어를 String으로 읽어오고 난 후
nextLine() 이 자동으로 읽혀지면서
Enter a line of text: prompt 뒤에 You entered: () 가 나옵니다.
즉, 사용자가 입력을 하지도 않았는데 nextLine() 이 empty String을 읽었다는 뜻입니다.
이것은 next() 로 하나의 단어를 읽어간 후, 남아 있던 "\n" 을
nextLine() 이 하나의 valid한 input으로 취급하여서
\n을 제외한 empty String을 nextLine() 으로 읽어 버리기 때문입니다.

Keyboard Input (3/4)

```
System.out.print("Enter a word: ");         // Enter a word:
String word2 = scanner.next();              // Enter a word: Korea
System.out.println("You entered: " + word2);// You entered: Korea

String dummy = scanner.nextLine();          // read out dummy = "\n";
System.out.print("Enter a line of text: "); // Enter a line of text:
String line2 = scanner.nextLine();          // Enter a line of text: Seoul Busan
System.out.println("You entered: (" + line2 + ")"); // You entered: (Seoul Busan)
```


페이지 4

따라서 이러한 현상에 대한 대비가 필요합니다.
이 slide에서는 간단한 solution을 제시하고 있습니다.
앞의 경우와 마찬가지로 String 하나인 word를 scanner.next() 로 입력 받았습니다.
그런데 그 이후에 온전한 한 line의 text input을 입력 받기 전에
현재 남아 있는 "\n" 을 제거해야 할 필요가 있는 것이지요.
그래서 이 예제에서는 scanner.nextLine() 을 한번 call하여
"\n" 전까지의 남아 있는 입력 부분을 dummy String으로 받아 무시하는 과정을 추가하였습니다.
이렇게 하게 되면 기존에 남아 있는 입력 부분은 모두 사라지게 되며
그 후에 한 라인을 온전히 새로운 입력으로 받아 들이는 부분은
정상 작동하게 됩니다.

Keyboard Input (4/4)

```
System.out.print("Enter an integer: ");    // Enter an integer:
int num = scanner.nextInt();               // Enter an integer: 243
String dummy2 = scanner.nextLine();        // read out dummy2 = "\n"
System.out.print("Enter a line of text: ");// Enter a line of text:
String line3 = scanner.nextLine();         // Enter a line of text: Seoul Busan
System.out.println("You entered: (" + num +
                ") (" + line3 + ")");      // You entered: (243) (Seoul Busan)
```


페이지 5

이 부분에서도 비슷하게 dummy String으로 input out 하는 또 다른 예를 보여주고 있습니다.
먼저 int input 하나를 nextInt() 로 받습니다.
뒤에 한 줄을 온전히 읽기 전에 남아 있는 "\n" 를 읽어서 버리기 위하여
dummy2 String을 하나 nextLine()으로 읽었습니다.
그 후에 nextLine() 으로 새로운 한줄 텍스트를 읽습니다.

Input by String

```java
import java.util.Scanner;

public class InputByString {
    public static void main(String[] args) {
        String input = "Korea 123.456 5678";
        Scanner scanner = new Scanner(input);
        String str = scanner.next();
        float fnum = scanner.nextFloat();
        int inum = scanner.nextInt();
        System.out.println(str + " " + fnum + " " + inum);
        // OUTPUT: Korea 123.456 5678
    }
}
```


페이지 6

입력을 keyboard에서 받는 것이 아니라 String으로 input을 대신하게 할 수도 있습니다.
java.util.Scanner를 역시 import 하였습니다.
input으로 사용할 String을 정의합니다.
이 예제에서는 input 이라는 String안에 String, double, int data 하나씩을 넣어 정의하였습니다.
그리고 Scanner object를 생성할 때 Scanner의 constructor parameter로
System.in 대신 String input을 지정하였습니다.
이렇게 하면 keyboard에서 읽어 오는 것이아니라
input을 String으로 부터 읽어올 수 있습니다.
그 다음에는 정상적으로 String, float, int data를 하나씩 읽어 옵니다.

Changing Delimiters

```java
import java.util.Scanner;

public class InputDelimiter {
    public static void main(String[] args) {
        String input = "10 20 30";
        Scanner scanner = new Scanner(input);
        while (scanner.hasNextInt()) {
            System.out.println(scanner.nextInt());  // 10 20 30
        }
        String input2 = "10,20,30";
        Scanner scanner2 = new Scanner(input2).useDelimiter(",");
        while (scanner2.hasNextInt()) {
            System.out.println(scanner2.nextInt()); // 10 20 30
        }
    }
}
```


페이지 7

blank space 이외에도 다른 delimiter를 사용하여 input data를 구분할 수 있습니다.
프로그램의 전반부는 정상적으로 String으로 부터 input을 받아
10 20 30 의 세 개의 정수를 입력 받아 출력하였습니다.
두번째 Scanner object는 useDelimiter method를 사용하여
쉼표를 delimiter로 지정하였습니다.
따라서 "10, 20, 30" 이라는 input String을
10 20 30 이라는 세개의 정수로 나누어 입력 받을 수 있습니다.

System.out.print(), System.out.println()

```java
public class SystemOutPrint {
    public static void main(String[] args) {
        System.out.print("Hello, "); // Hello,
        System.out.print("World!");  // Hello, World!
        System.out.println("Hello, World!"); // Hello, World!Hello, World!
        System.out.println("Welcome to Java programming.");
        // Welcome to Java programming.
        System.out.println("Number: " + 123); // Number: 123
        System.out.println("Boolean: " + true); // Boolean: true
        System.out.println("Character: " + 'A'); // Character: A
        System.out.println("" + false); // false
    }
}
```


페이지 8

지금까지 많이 사용했던 System.out.print와 System.out.println을 사용하는
예제 프로그램 입니다.
System.out.print는 print가 끝난 후 줄을 바꾸지 않아서
같은 줄에 출력이 계속 이어지게 합니다.
System.out.println은 출력이 끝난 이후 줄을 바꿉니다.
String과 int, boolean, char 등 다른 type과 concatenation 하여 출력이 가능합니다.
맨 마지막 줄에 보면 “” + false 와 같이 하여 print를 했는데
이 경우는 System.out.println(false) 와 같이 하면 compile error가 나기 때문에
empty String과 concatenation하면서 false를 String type으로 바꾸어 print가 가능합니다.

Formatted Output: System.out.printf (1/2)

- Format Specifiers: %d: Integer %f: Floating-point %s: String %c: Character %b: Boolean

```java
public class SystemOutPrintf {
    public static void main(String[] args) {
        System.out.printf("Formatted number: %d\n", 123);
        // Formatted number: 123
        System.out.printf("Width 10: %10d\n", 123);
        // Width 10:        123
        System.out.printf("Two decimal places: %.2f\n", 123.456);
        // Two decimal places: 123.46
        System.out.printf("Left justified: %-10d|\n", 123);
        // Left justified: 123       |
```


페이지 9

output에서 줄을 맞추거나 여러가지 유형으로 프린트를 할 수 있게 해 주는 method가
System.out.printf 입니다.
사실 printf는 원래 C와 C++에서 사용되던 것인데
Java의 단순한 출력 기능을 보완해 주기 위하여
Java에 추가된 것은 오래되지 않았습니다.
printf 에는 format specifier라는 기능이 있습니다.
print 할 data의 type에 따라 적절한 specifier를 선택할 수 있습니다.
integer, floating-point, String, character, boolean 등의
형식으로 print할 수 있습니다.
%d 를 사용하여 integer 123을 출력하였습니다.
output이 차지하는 width를 지정할 경우 %10d 와 같이 할 수 있습니다.
이 예에서는 width를 10으로 하여 integer를 print하며
default로 오른쪽에 붙어서 print가 되며
width 중에 남는 부분은 blank space로 보여지게 됩니다.
실수 출력의 경우, 정수부와 소수부의 자리수를 지정할 수 있는데
%5.3f %10.2f 와 같이 하면 됩니다.
이 예에서는 정수부는 제약이 없고 소수부는 두 자리로 한정한 예를 보여주고 있습니다.
왼쪽으로 당겨 프린트하고 싶다면 퍼센트 다음에 음의 정수를 붙여서
칸수를 잡아주면 됩니다.

Formatted Output: System.out.printf (2/2)

```
        System.out.printf("%-10s %10s %10s\n", "Name", "Age", "Score");
        System.out.printf("%-10s %10d %10.2f\n", "Alice", 30, 88.5);
        System.out.printf("%-10s %10d %10.2f\n", "Bob", 25, 91.75);
        System.out.printf("Price: $%.2f\n", 19.99);
    }
}
```

```
Name              Age       Score
Alice              30       88.50
Bob                25       91.75
Price: $19.99
```


페이지 10

이 예에서는 output의 칸을 잘 맞춰서
표와 같은 형태로 print하는 것을 보여주고 있습니다.
표의 field 제목들인 Name, Age, Score와
표의 data entry들을
열칸씩 잡고 좌우로 justify하면서
줄을 잘 맞추었습니다.
마지막 statement에서는 퍼센트 앞에 달러 싸인을 놓으면서
output 앞에 달러 싸인이 프린트 되게 하였습니다.