OBJECT-ORIENTED SYSTEMS DESIGN
[Exercise]: Console Input and Output

2022/03/23

Department of Computer Science,
Hanyang University

Lecturer: Taeuk Kim
TA: Taejun Yoon, Seong Hoon Lim, Young Hyun Yoo

Today's Plan

1. **Screen Output: 10 min.**
2. **Console Input Using the `Scanner` Class: 10 min.**
3. **Practice: 40 min.**

Screen Output

Chapter 2.1

`System.out.println` for Console Output

- **`System.out` is an object that is part of the Java language.**

- **`println` is a method invoked by the `System.out` object that can be used for *console output.***
  - The data to be output is **given as an argument in parentheses**.
  - A plus sign is used to connect more than one item.
  - Every invocation of `println` ends a line of output.

```
System.out.println("The answer is " + 42);
```

- **There are other printing methods supported by `System.out` object**
  - `println`
  - `print`
  - `printf`

println vs. print

- **The `println` method moves the cursor to the next line after printing out all of its content.**

```
public static void main(String[] args){
    String str1 = "hello java";
    String str2 = "Tomorrow is Tuesday";
    System.out.println(str1);
    System.out.println(str2);
    System.out.print(str1);
    System.out.print(str2);
}
```

```
hello java
Tomorrow is Tuesday
hello javaTomorrow is Tuesday
```

Formatting Output with `printf`

- **`System.out.printf` can be used to produce output in a specific format.**
  - It can have any number of arguments.
  - The first argument is always a *format string* that contains one or more *format specifiers* for the remaining arguments.
  - All the arguments except the first are values to be output to the screen.

```
public static void main(String[] args){
    String str = "abc";
    System.out.printf("START%sEND \n",str);
    System.out.printf("START%4sEND %n",str);
    System.out.printf("START%2sEND %n",str);
    System.out.println();

    char oneC = 'Z';
    System.out.printf("START%cEND \n",oneC);
    System.out.printf("START%4cEND \n",oneC);
    System.out.println();

    double d = 12345.123456789;
    System.out.printf("START%f \n",d);
    System.out.printf("START%.4f \n",d);
    System.out.printf("START%.2f \n",d);
    System.out.printf("START%12.4f \n",d);
    System.out.printf("START%e \n",d);
}
```

```
STARTabcEND
START  abcEND
STARTabcEND

STARTZEND
START   ZEND

START12345.123457
START12345.1235
START12345.12
START   12345.1235
START1.234512e+04
```

The value is always output. If the specified field width is too small, extra space is taken.

Note that the output is rounded, not truncated, when digits are discarded.

String.format

- **We have another way of printing out formatted results.**
  - The **`format`** method of the **`String`** class.
  - **`System.out.printf = System.out.print + String.format`**

```
public class PrintWithFormat {
    public static void main(String[] args) {
        String str1 = " The value of pi is ";
        double pi = Math.PI;

        System.out.println("[original]\t" + str1  + pi);
        System.out.printf("[printf]\t" + str1 + "%4.2f\n", pi);
        String formatted = String.format("[format]\t" + str1 + "%4.2f", pi);
        System.out.print(formatted);
    }
}
```

```
[original]   The value of pi is 3.141592653589793
[printf]     The value of pi is 3.14
[format]     The value of pi is 3.14

Process finished with exit code 0
```

Format Specifiers

Display 2.1 Format Specifiers for `System.out.printf`

CONVERSION CHARACTER	TYPE OF OUTPUT	EXAMPLES
d	Decimal (ordinary) integer	`%5d` `%d`
f	Fixed-point (everyday notation) floating point	`%6.2f` `%f`
e	E-notation floating point	`%8.3e` `%e`
g	General floating point (Java decides whether to use E-notation or not)	`%8.3g` `%g`
s	String	`%12s` `%s`
c	Character	`%2c` `%c`

Console Input Using the `Scanner` Class

Chapter 2.2

Console Input Using the `Scanner` Class (1)

- **Starting with version 5.0, Java includes a class for doing simple keyboard input named the `Scanner` class.**
- **In order to use the `Scanner` class, a program must include the following line near the start of the file:**

```
import java.util.Scanner;
```

- **The following line creates an object of the class `Scanner` and names the object `keyboard`:**

```
Scanner keyboard = new Scanner(System.in);
```

Console Input Using the `Scanner` Class (2)

- **The method `nextInt` reads one `int` value typed in at the keyboard and assigns it to a variable:**

```
int numberOfPods = keyboard.nextInt();
```

- **The method `nextDouble` reads one `double` value typed in at the keyboard and assigns it to a variable:**

```
double d1 = keyboard.nextDouble();
```

- **Multiple inputs must be separated by *whitespace* and read by multiple invocations of the appropriate method.**
  - Whitespace is any string of characters, such as blank spaces, tabs, and line breaks that print out as white space.

Console Input Using the `Scanner` Class (3)

- **The method `next` reads one string of non-whitespace characters delimited by whitespace characters such as blanks or the beginning or end of a line.**

- **Given the code**

```
String word1 = keyboard.next();
String word2 = keyboard.next();
```

and the input line

```
jelly beans
```

  - The value of `word1` would be `jelly`, and the value of `word2` would be `beans`.

Console Input Using the `Scanner` Class (4)

- **The method `nextLine` reads an entire line of keyboard input.**

- **The code,**

```
String line = keyboard.nextLine();
```

  - reads in an entire line and places the string that is read into the variable `line`.

- **The end of an input line is indicated by the escape sequence `'\n'`.**
  - This is the character input when the **Enter** key is pressed.
  - On the screen it is indicated by the ending of one line and the beginning of the next line.

- **When `nextLine` reads a line of text, it reads the `'\n'` character, so the next reading of input begins on the next line.**
  - However, the `'\n'` does not become part of the string value returned (e.g., the string named by the variable `line` above does not end with the `'\n'` character).

Examples

- `int intVal = keyboard.nextInt();`
- `double dVal = keyboard.nextDouble();`
- **Multiple inputs must be separated by whitespace.**

```
public static void main(String[] args){
    Scanner keyboard = new Scanner(System.in);
    int  intVal = keyboard.nextInt();
    double dVal = keyboard.nextDouble();
    System.out.println(intVal);
    System.out.println(dVal);
```

```
123
3.14
123
3.14
```

nextLine

- **nextLine**
  - This method reads the remaining strings up to the end of the line and discards the EOL (end of line) character.

```
public static void main(String[] args){
    Scanner keyboard = new Scanner(System.in);
    String s = keyboard.nextLine();
    int  intVal = keyboard.nextInt();
    System.out.println(s);
    System.out.println(intVal);
```

```
hello java!
1234
hello java!
1234
```

next vs nextLine

- **Note that `next` cannot erase '\n' from buffer.**

```
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    System.out.println("next enter two word");
    String word1 = keyboard.next();
    String word2 = keyboard.next();
    System.out.println("You entered \"" + word1 + "\" and \"" + word2 + "\"");
    System.out.println();

    keyboard.nextLine(); //To get rid of '\n'

    System.out.println("next enter two word");
    String str = keyboard.nextLine();
    System.out.println("You entered \""+ str+"\"");
}
```

```
next enter two word
jelly beans
You entered "jelly" and "beans"

next enter two word
jelly beans
You entered "jelly beans"
```

Changing Delimiters

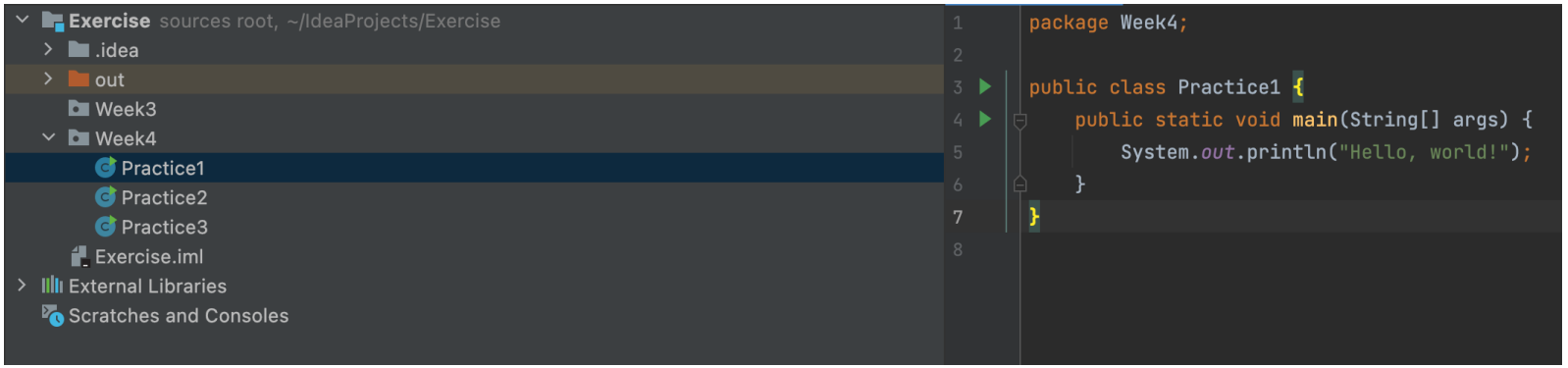
- **It is also possible to change the delimiter of the `Scanner` class.**

```
Scanner keyboard = new Scanner(System.in);
keyboard.useDelimiter("#");
```

```
public static void main(String[] args){
    Scanner keyboard = new Scanner(System.in);
    keyboard.useDelimiter("#");
    int  intVal1 = keyboard.nextInt();
    int  intVal2 = keyboard.nextInt();

    System.out.println(intVal1);
    System.out.println(intVal2);
```

```
123#456#
123
456
```



Practice

Construct a separate class for each problem!

Exercise/Week4/Practice1.java,
Practice2.java,
Practice3.java

Practice 1 (`Practice1.java`)

- **Write a program that reads in a string containing three words separated by commas and then outputs that string with each word in a different line.**

Practice 2 (`Practice2.java`)

- **Write a program that reads in two numbers typed on the keyboard and divides the first number by the second number.**
  - The program should output the **dividend**, **divisor**, **quotient**, and **remainder** on the screen.

Practice 3 (`Practice3.java`)

- **Grade point average (GPA) is a 4-point scale is calculated by using the following formula:**

$$GPA = \left(\frac{Percentage}{100}\right) \times 4$$

- **Write a program that takes as input the percentage (an integer) from a user.**

- **The program should then output the user's GPA on the screen.**
  - The format of the output should be as follows, assuming the user's percentage is 85:

$$\mathbf{(85/100) * 4 = 3}$$

  - Note that the original score is 3.4 (i.e., use some functions to match the format).

Time for Practice

Get it started, and ask TAs if you are in a trouble.