Java Basics

Lab 2

TA : Hyuna Seo, Kichang Yang, Minkyung Jeong, Jaeyong Kim

SEOUL NATIONAL UNIVERSITY

Announcement

- You should finish the lab practice and submit your job to eTL before the next lab class starts(Wednesday, 7:00 PM).
- The answer of the practice will be uploaded after the due.

Overview

- Recap: Java basic review
  - Arrays
  - if-else / ternary / switch
  - while / for / foreach
- Problem 1 - Reverse Print
- Problem 2 - Student ID Checker

Recap: Arrays

Main Function

```java
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars[0]);
cars[0] = "Opel";
System.out.println(cars[0]);
System.out.println(cars.length);
```

Output

```
Volvo
Opel
4
```

Recap: `if`/`else` Statement

Main Function

```
int time = 22;
if (time < 10) {
  System.out.println("Good morning.");
} else if (time < 20) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
```

Output

```
Good evening.
```

Recap: for/for-each

Main Function

```
String[] cars = { "Volvo", "BMW", "Ford", "Mazda" };

for (int i = 0; i < 4; i++) {
  System.out.print(cars[i] + " ");
}
System.out.println();

for (String car : cars) { System.out.print(car + " "); }
System.out.println();
```

Output

```
Volvo BMW Ford Mazda
Volvo BMW Ford Mazda
```

Recap: while/do-while

Main Function

```
int i = 0;
while (i < 5) { System.out.print(i++ + ","); }
System.out.println();

i = 0;
do { System.out.print(i++ + ","); }
while (i < 5);
System.out.println();
```

Output

```
0,1,2,3,4,
0,1,2,3,4,
```

Objectives

- Get used to Java basics
- Problem 1 - Reverse Print
    - arrays
    - for / foreach
- Problem 2 - Student ID Checker
    - if-else
    - functions
    - while

Problem Overview

- Problem 1 - Reverse Print
  - 1-1 Get String Input (0:05)
  - 1-2 Save Strings in an Array (0:05)
  - 1-3 Reverse Print (0:05)
- Problem 2 - Student ID Checker
  - 2-1 Student ID Validator (0:05)
  - 2-2 Refactoring Validator (0:05)
  - 2-3 Get repeated Input (0:05)

Problem 1 : Array Printer

- Write a program which inputs strings and outputs in the opposite order.
    - Get the number of input strings
    - Declare a string array
    - Get input strings and put them into the array
    - Print the strings of the array
    - Print the strings of the array in the opposite order

Problem 1-1 : Get String Input (5 min)

- Import `java.util.Scanner`
- Create a scanner which get inputs from console
  - Use `Scanner scanner = new Scanner(System.in);`
- Get the number of input strings
  - Use `.nextInt` to get the input as "int" type
  - Print the input (this line will be removed after problem1-1)

Problem 1-1 : Get String Input (5 min)

Console

Input
Output

```
3
3
```

Console

Input
Output

```
wronginput

Exception in thread "main" java.util.InputMismatchException
	at java.base/java.util.Scanner.throwFor(Scanner.java:939)
	at java.base/java.util.Scanner.next(Scanner.java:1594)
	at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
	at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
	at ArrayPrinter.main(ArrayPrinter.java:13)
```

Problem 1-2 : Save Strings in an Array (5 min)

- Declare a string array
- Get input strings and put them into the array
    - Use the input from Problem 1-1 as the number of iteration
- Print the strings of the array
    - Iterate on each element in the array
    - Print the element at each iteration

Problem 1-2 : Save Strings in an Array (5 min)

Console

Input

```
3
apple
pear
orange
```

Output

```
apple pear orange
```

Problem 1-3 : Reverse Print (5 min)

- Print the strings of the array in the reverse order

Problem 1-3 : Reverse Print (5 min)

Console

Input

```
3
apple
pear
orange
```

Output

```
apple pear orange
orange pear apple
```

Problem 2 : Student ID Checker

- Write a program which checks whether an input string is a valid student ID (XXXX-XXXXX).
- Input a string from the console and save the string into a variable.

Problem 2 : Student ID Checker

- Check whether the input string is a valid student ID or not in following order, and print a corresponding message.
  1. The length of the input should be 10.
     -> `The input length should be 10.`
  2. The 5th character of the input should be ‘-’.
     -> ``Fifth character should be `-`.``
  3. All characters of the input but 5th should be digits.
     -> `Contains an invalid digit.`

Problem 2-1 : Student ID Validator (5 min)

- Use `.charAt` to get a nth character of a string
- Pass an `int` variable as a index of the character you want to get.
- Return type of `.charAt` is `char`.
- `IndexOutOfBoundsException` is thrown if the index argument is negative or not less than the length of this string.

Problem 2-1 : Student ID Validator (5 min)

- Each Java character matches to a number called ASCII code (https://en.wikipedia.org/wiki/ASCII)
- You can check whether a character is a digit or alphabet with ASCII code comparison.
- This boolean expression is `true` if `char` type variable `ch` is a
  - digit: `ch >= '0' && ch <= '9'`
  - non-digit: `ch < '0' || ch > '9'`
  - lower alphabet: `ch >= 'a' && ch <= 'z'`
  - upper alphabet: `ch >= 'A' && ch <= 'Z'`

Problem 2-1 : Student ID Validator (5 min)

- Invalid student IDs

Console

Input
Output

```
2018-1234
The input length should be 10.
```

Console

Input
Output

```
2018_12345
Fifth character should be `-`.
```

Console

Input
Output

```
e018-12345
Contains an invalid digit.
```

Problem 2-1 : Student ID Validator (5 min)

- Valid student ID

Console

Input
Output

```
2018-12345
2018-12345 is a valid student ID
```

Problem 2-2 : Refactoring Validator (5 min)

- Refactor (Make the code clean) student ID checker by
  - moving each validation checking logic into new functions, `isProperLength`, `hasProperDivision`, and `hasProperDigits`.
    - `isProperLength` : Checks whether the length of the input is 10.
    - `hasProperDivision` : Checks whether the 5th character of the input is '-'.
    - `hasProperDigits` : Checks whether all characters of the input but 5th is digits.

Problem 2-2 : Refactoring Validator (5 min)

- Refactor (Make the code clean) student ID checker by
  - moving top-level `if`/`else` statements into a new function `validateStudentID`.
    - All validation functions (`isProperLength`, `hasProperDivision`, and `hasProperDigits`) would be called in `validateStudentID` function to conduct validation

Problem 2-3 : Get Repeated Input (5 min)

- Upgrade your student ID checker to get input repeatedly until the input is “exit”.

Problem 2-3 : Get Repeated Input (5 min)

Console

```
Input    2018-1234
Output   The input length should be 10.
Input    2018_12345
Output   Fifth character should be `-`.
Input    ee18-12345
Output   Contains an invalid digit.
Input    2018-12345
Output   2018-12345 is a valid student ID.
Input    exit
```

Submission

- Compress your final StudentIDValidator.java file into a zip file.
- Rename your zip file as 20XX-XXXXX_{name}.zip - for example, 2021-12345_JeongMinkyung.zip
- Upload it to eTL - Lab 2 assignment.
- Your program should contain main function that can be properly executed and print desired outputs.