OBJECT-ORIENTED SYSTEMS DESIGN
[Exercise]: Flow of Control

2022/03/30

Department of Computer Science,
Hanyang University

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

Today's Plan

1. **Chapter Review: 20 min.**

2. **Practice : 40 min.**

Branching Mechanism

Chapter 3.1

Branching with an `if-else` Statement

- **An `if-else` statement chooses between two alternative statements based on the value of a Boolean expression.**
  - `Yes(or no)_statements` can be composed of compound statements.

```
package com.company;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int myInt = sc.nextInt();

        if(myInt == 5) {
            System.out.println("Input integer is 5");
            System.out.println("So, add 1 to your input");
            myInt++;
        }
        else {
            System.out.println("Input integer is not 5");
            System.out.println("So, subtract 1 to your input");
            myInt--;
        }
        System.out.printf("Final myInt = %d\n", myInt);
    }
}
```

```
Main
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Prog
5
Input integer is 5
So, add 1 to your input
Final myInt = 6
```

If we pick an integer other than 5, this `no_statement` will be executed

As the if-statement's boolean_expression is true, `yes_statement` has been executed.

Variants of `if-else` statements

- **Nested Statements**

```
package com.company;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int myInt = sc.nextInt();

        if(myInt > 5) {
            System.out.println("Input integer is bigger than 5");
            if(myInt == 10) {
                System.out.println("Input integer is 10");
            } else {
                System.out.println("Input integer is not 10");
            }
        } else {
            System.out.println("Input integer is less than or equal to 5");
            if(myInt == 3) {
                System.out.println("Input integer is 3");
            } else {
                System.out.println("Input integer is not 3");
            }
        }
```

```
Main
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\Je
4
Input integer is less than or equal to 5
Input integer is not 3
```

- **Multiway `if-else` statements**

```
package com.company;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int myInt = sc.nextInt();

        if(myInt == 5) {
            System.out.println("Input integer is 5");
        } else if(myInt == 6) {
            System.out.println("Input integer is 6");
        } else if(myInt == 7) {
            System.out.println("Input integer is 7");
        } else if(myInt == 8) {
            System.out.println("Input integer is 8");
        }
    }
}
```

```
Main
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagen
7
Input integer is 7
```

The `switch` Statement

- **Similar to multiway `if-else` statements**
  - Its action is determined by *controlling expression.*

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int myInt = sc.nextInt();

    // myInt = Controlling expression
    // determines which branch to execute
    switch(myInt) {
        // if myInt == 5, then case 5 is executed
        case 5 :
            System.out.println("Input integer is 5");
            break;

        // if myInt == 6, then case 6 is executed
        case 6 :
            System.out.println("Input integer is 6");
            break;

        // if myInt == 7, then case 7 is executed
        case 7 :
            System.out.println("Input integer is 7");
            break;

        // if myInt matches none of these cases, then default is executed
        default :
            System.out.println("I don't know what your input is!");
            break;
    }
```

```
Main
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBr
6
Input integer is 6
```

```
Main
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBr
1
I don't know what your input is!
```

The `switch` Statement

- **Don't forget to add *`break`*!**

```java
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int myInt = sc.nextInt();

        // myInt = Controlling expression
        // determines which branch to execute
        switch(myInt) {
            // if myInt == 5, then case 5 is executed
            case 5 :
                System.out.println("Input integer is 5");

            // if myInt == 6, then case 6 is executed
            case 6 :
                System.out.println("Input integer is 6");

            // if myInt == 7, then case 7 is executed
            case 7 :
                System.out.println("Input integer is 7");

            // if myInt matches none of these cases, then default is executed
            default :
                System.out.println("I don't know what your input is!");
        }
    }
```

```
Main
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetB
6
Input integer is 6
Input integer is 7
I don't know what your input is!
```

When **`myInt == 6`**, all of this part will be executed.

Boolean Expressions

Chapter 3.2

Boolean Expressions

- **A Boolean expression is an expression that is either `true` or `false`.**

```java
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int myInt = sc.nextInt();
        int myInt2 = 2;

        // Kind of boolean expression
        // True/False itself is also boolean expression
        boolean myBool1 = myInt != 1;
        boolean myBool2 = myInt >= myInt2;
        boolean myBool3 = true;
        boolean myBool4 = false;

    }
}
```

Java Comparison Operators

Display 3.3 Java Comparison Operators

MATH NOTATION	NAME	JAVA NOTATION	JAVA EXAMPLES
=	Equal to	==	`x + 7 == 2*y` `answer == 'y'`
≠	Not equal to	!=	`score != 0` `answer != 'y'`
>	Greater than	>	`time > limit`
≥	Greater than or equal to	>=	`age >= 21`
<	Less than	<	`pressure < max`
≤	Less than or equal to	<=	`time <=limit`

Pitfall: Using == with Strings

- **When applied to two *objects* such as objects of the `String` class, `==` tests to see if they are stored in the same memory location, not whether or not they have the same value.**
  - Use the method **`equals`**, or **`equalsIgnoreCase`** to compare two strings.

```
public class Main {
    public static void main(String[] args) {
        String s1 = new String("hello world");
        String s2 = new String("hello world");

        System.out.println(s1 == "hello world");
        System.out.println(s2 == "hello world");
        System.out.println(s1 == s2);
        System.out.println(s1.equals(s2));
    }
}
```

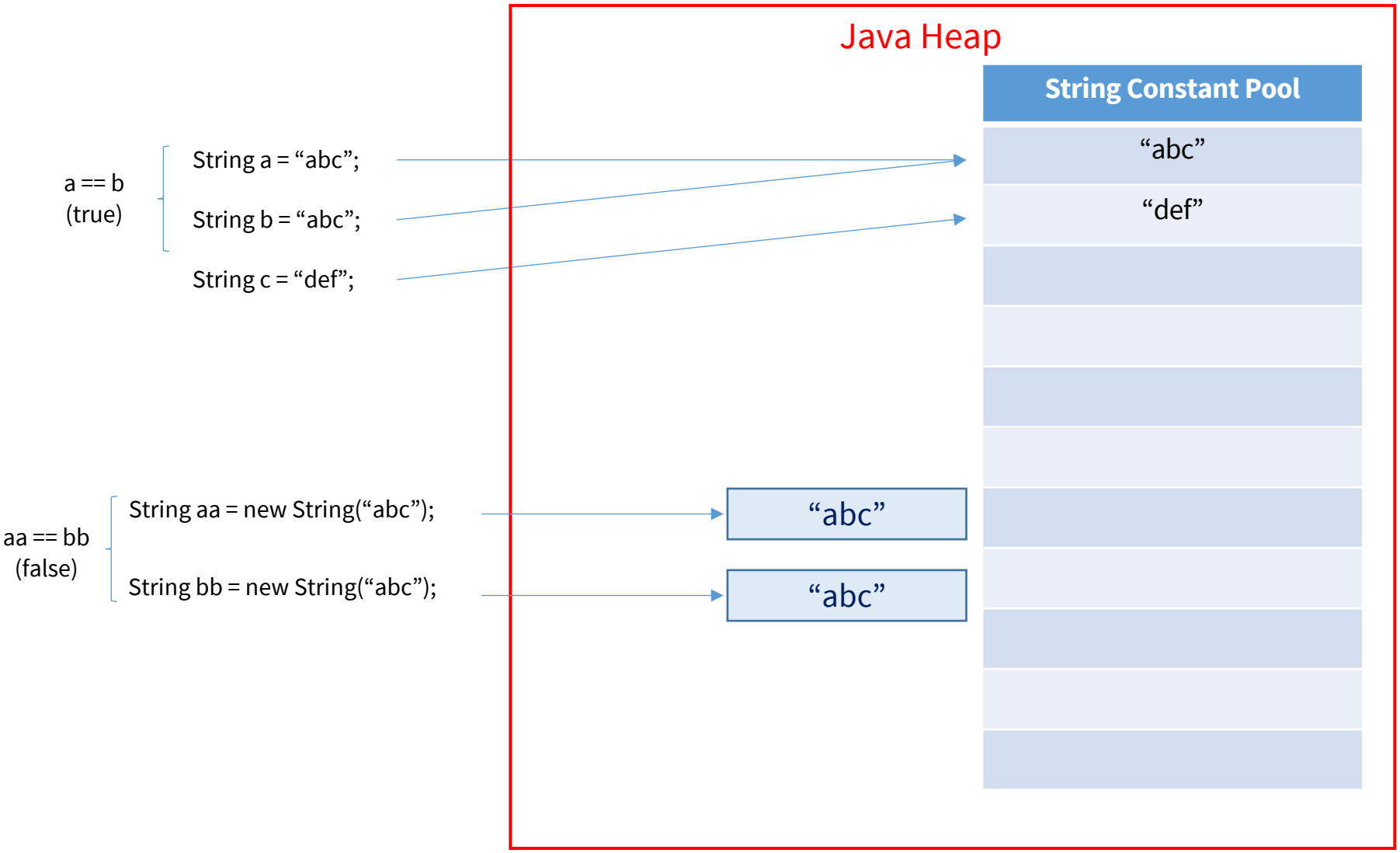
```
Main ×
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-java
false
false
false
true
```

Because s1 and s2 are not in the same memory location, 1~3 print results are “false”.

Pitfall: Java String Pool

- **String class has a special data structure, "String Constant Pool"**
  - When a String class is defined with " "(e.g., `String a = "abc";`),
    - The value of `String a` (= `"abc"`) is stored in **String Constant Pool**.
    - And the variable `a` points to the location of the value.
  - If another String class is defined to the same value of String Constant Pool's component (e.g., `String b = "abc";`),
    - It doesn't allocate a new memory location: it points to the same location as `a.`
    - So, the `String` objects `a` and `b` points to the same location of String Constant Pool.
  - But when a String class is defined to the same value of String Constant Pool's component, with "`new String( )`" (e.g., `String c = new String("abc");`),
    - It allocates its new independent memory location, not in the String Constant Pool
    - So, `c` points to another location in memory, not identical to that of `a` and `b`.

Pitfall: Java String Pool



Combining multiple Boolean expressions

- **Operator "or"**
  - Denoted as "||".
  - Check at least one of expressions is true.
  - Check expressions sequentially.
    - Short-circuit evaluation : If true expression is found, it skips remaining expressions.
      - Even though one of remaining expressions causes Runtime Error.

```
public class Main {
    public static void main(String[] args) {
        if(3 + 7 == 10 ||
                4 + 6 == 11 ||
                9 + 3 == 6  ||
                123 / 0 == 10)  // It causes DivideByZeroException but...
        {
            System.out.println("Yes_Statement");
        } else {
            System.out.println("No_Statement");
        }
```

```
Main ×
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\
Yes_Statement
```

Combining multiple Boolean expressions

- **Operator “and”**
  - Denoted as “&&”.
  - Check all of expressions are true.
  - Check expressions sequentially.
    - Short-circuit evaluation : If false expression is found, it skips remaining expressions.
      - Even though one of remaining expressions causes Runtime Error.

```
public class Main {
    public static void main(String[] args) {
        if(3 + 7 == 10 &&
                4 + 6 == 11 &&
                9 + 3 == 6  &&
                123 / 0 == 10)  // It causes DivideByZeroException but...
        {
            System.out.println("Yes_Statement");
        } else {
            System.out.println("No_Statement");
        }
```

```
Main
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\J
No_Statement
```

Combining multiple Boolean expressions

- **Operator “not”**
  - Denoted as “!”.
  - Returns opposite of expression result,

```
public class Main {
    public static void main(String[] args) {
        if(!(3 + 7 == 10))
        {
            System.out.println("Yes_Statement");
        } else {
            System.out.println("No_Statement");
        }
    }
}
```

```
Main ×
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaa
No_Statement
```

Truth Tables

Display 3.5 Truth Tables

AND		
Exp_1	*Exp_2*	*Exp_1 && Exp_2*
true	true	true
true	false	false
false	true	false
false	false	false

OR		
Exp_1	*Exp_2*	*Exp_1 || Exp_2*
true	true	true
true	false	true
false	true	true
false	false	false

NOT	
Exp	*!(Exp)*
true	false
false	true

Complete evaluation

- **Check all of expressions**
  - Denoted as "|" and "&" for "or" and "and" operation.
  - It does not skip remaining expressions.

```
public class Main {
    public static void main(String[] args) {
        if(3 + 7 == 10 |
                4 + 6 == 11 |
                9 + 3 == 6  |
                123 / 0 == 10)  // It causes DivideByZeroException
        {
            System.out.println("Yes_Statement");
        } else {
            System.out.println("No_Statement");
        }
```

```
Main ×
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program F
Exception in thread "main" java.lang.ArithmeticException Create breakpoint :
    at com.company.Main.main(Main.java:6)
```

Precedence and Associativity Rules

Display 3.6 Precedence and Associativity Rules

Highest Precedence

Lowest Precedence

PRECEDENCE	ASSOCIATIVITY
From highest at top to lowest at bottom. Operators in the same group have equal precedence.	
Dot operator, array indexing, and method invocation., `[ ]`, `( )`	Left to right
`++` (postfix, as in x++), `--` (postfix)	Right to left
The unary operators: `+`, `-`, `++` (prefix, as in ++x), `--` (prefix), and `!`	Right to left
Type casts (Type)	Right to left
The binary operators `*`, `/`, `%`	Left to right
The binary operators `+`, `-`	Left to right
The binary operators `<`, `>`, `<=`, `>=`	Left to right
The binary operators `==`, `! =`	Left to right
The binary operator `&`	Left to right
The binary operator `|`	Left to right
The binary operator `&&`	Left to right
The binary operator `||`	Left to right
The ternary operator (conditional operator) `?:`	Right to left
The assignment operators `=`, `*=`, `/=`, `%=`, `+=`, `-=`, `& =`, `|=`	Right to left

Loops

Chapter 3.3

`while` statement

- **Iterate until `while` statement's condition becomes false**
  - First check condition, then iterate.
  - If condition is false before iteration, it does nothing.

```java
public class Main {
    public static void main(String[] args) {
        int myInt = 0;

        // while(boolean expression) {
        //      body
        // }
        while(myInt < 10) {
            myInt++;        // iterated 10 times
        }

        System.out.println(myInt);
    }
}
```

```
Main ×
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaag
10
```

```java
public class Main {
    public static void main(String[] args) {
        int myInt = 10;

        // while(boolean expression) {
        //      body
        // }
        while(myInt < 10) {
            myInt++;
        }

        System.out.println(myInt);
    }
}
```

```
Main ×
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-ja
10
```

`for` Statement

- **Iterate until the given boolean expression becomes true**
  - **`for (Initializing; Boolean_Expression; Update)`**

```
public class Main {
    public static void main(String[] args) {
        int myInt = 10;

        // for(initialize, boolean_expression, update) {
        //      body
        // }
        for(int i = 0; i < 10; i++) {
            myInt++;
            System.out.println("myInt value = " + myInt);
        }
    }
}
```

```
Main
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\
myInt value = 11
myInt value = 12
myInt value = 13
myInt value = 14
myInt value = 15
myInt value = 16
myInt value = 17
myInt value = 18
myInt value = 19
myInt value = 20
```

The integer variable **`i`** starts at 0, and updates its value at each iteration (**`i++`**).

When it iterated 10 times, the value of **`i`** becomes 10, making the Boolean expression false, then the iteration stops.

`break` statement

- **`Break` terminates the innermost iteration immediately**
  - Regardless of whether Boolean expression is true or false.

```java
public class Main {
    public static void main(String[] args) {
        int myInt = 10;

        for(int i = 0; i < 10; i++) {
            myInt++;
            System.out.println("myInt value = " + myInt);
            if(myInt == 15) break;
        }
    }
}
```

```
Main
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:
myInt value = 11
myInt value = 12
myInt value = 13
myInt value = 14
myInt value = 15
```

Although the value of **`i`** is under 10, **`break`** terminates the **`for`** loop immediately.

`continue` statement

- **`Continue` skips innermost iteration's remaining body operation**

```java
public class Main {
    public static void main(String[] args) {
        int myInt = 10;

        for(int i = 0; i < 10; i++) {
            myInt++;
            if(myInt == 15 || myInt == 16) continue;
            System.out.println("myInt value = " + myInt);
        }
    }
}
```

```
Main
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:
myInt value = 11
myInt value = 12
myInt value = 13
myInt value = 14
myInt value = 17
myInt value = 18
myInt value = 19
myInt value = 20
```

15 and 16 were skipped due to the continue command.

Random Number Generation

Chapter 3.5

Generating Random Numbers

```java
import java.util.Random;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Random rnd = new Random();
        int myInt = sc.nextInt();

        for(int i = 0; i < myInt; i++) {
            int randomNum = rnd.nextInt(10);
            System.out.println("My random number = " + randomNum);
        }
    }
}
```

Main × Main2 ×

```
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Prograr
5
My random number = 5
My random number = 8
My random number = 3
My random number = 4
My random number = 7
```

Practice

Construct a separate class for each problem!

Exercise/WeekN/Practice1.java,
Practice2.java

Practice 1 (`Practice1.java`)

- **Write a program that**
  - Take an ***integer*** as an input.
  - Your program should print **the multiples of the input integer from 1 to 100.**
  - Tip : Use the **`for`** statement.

```
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.3.2\lib
3
3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99,
종료 코드 0(으)로 완료된 프로세스
```

```
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe "-javaag
7
7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98,
종료 코드 0(으)로 완료된 프로세스
```

Practice 2 (`Practice2.java`)

- **Write a *"Rock-paper-scissors"* program that**
  - Take ***"rock"***, ***"paper"*** or ***"scissors"*** as an input (no other input is allowed).
  - Your program should print **a computer's random choice** and **the corresponding result of the game.**

```
C:\Users\LSH\.jdks\openjdk-17.0.2\bin\java.exe
scissors
Computer's choice : paper
You win

종료 코드 0(으)로 완료된 프로세스
```

```
C:\Users\LSH\.jdks\openjdk-17.0.2\b
rock
Computer's choice : scissors
You win

종료 코드 0(으)로 완료된 프로세스
```

Time for Practice

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