CSE3040 Java Language
Lecture #04

Dept. of Computer Engineering,
Sogang University

1.4. Arithmetic Operators

- Conditional operators

```
time < 12 ? "am" : "pm"
```

  - If time < 12, the result of this statement is "am".
  - If time >= 12, the result of this statement is "pm".

  - If condition (first operand) is true, the result is the second operand.
  - If condition (first operand) is false, the result is the third operand.

1.4. Arithmetic Operators

```
class op3Hang {
   public static void main(String[] args)
   {
        int   x=30, y=10, z;
        char  op;

        System.out.println(" x:" + x + "  y:" + y);
        //----------------------------------------
        op = '+';
        if (op == '+')  z = x+y;
        else            z = x-y;

        System.out.println(" z:" + z);
        //----------------------------------------
        op = '-';
        z = (op == '+') ?  x+y:  x-y;

        System.out.println(" z:" + z);
   }
}
```

1.4. Arithmetic Operators

- Bit-wise operators and shift operators
  - op1 & op2
    - The AND operator compares two bits and generates a result of 1 if both bits are 1; otherwise, it returns 0.
  - op1 | op2
    - The OR operator compares two bits and generates a result of 1 if either or both bits are 1; otherwise, it returns 0.
  - op1^ op2
    - The EXCLUSIVE-OR operator compares two bits and generates a result of 1 if the bits are complementary; otherwise, it returns 0.
  - ~op1
    - The COMPLEMENT operator is used to invert all of the bits of the operand.
  - op1 >> op2
    - The SHIFT RIGHT operator moves the bits to the right, discards the far right bit, and assigns the leftmost bit a value of 0. Each move to the right effectively divides op1 in half.
  - op1 << op2
    - The SHIFT LEFT operator moves the bits to the left, discards the far left bit, and assigns the rightmost bit a value of 0. Each move to the left effectively multiplies op1 by 2.

1.4. Arithmetic Operators

– Example) when a = 0xA7

a & F0 =	a | F0 =	a ^ F0 =	~a =
1010 0111	1010 0111	1010 0111	
1111 0000	1111 0000	1111 0000	1010 0111
1010 0000 = A0	1111 0111 = F7	0101 0111 = 57	0101 1000 = 58

x << 3 Each bit of x moves 3 bits to the left

y >> 4 Each bit of y moves 4 bits to the right

– Bit assignment operator

&= |= ^= <<= >>=

a &= 0x0F ≡ a = a & 0x0F

1.4. Arithmetic Operators

```
class opBit {
   public static void main(String[] args)
      {
      char  a = 0xA7;  // 0xA7 is hexadecimal number.
                                          // \n : move cursor next line
      System.out.println(" a       : " + Integer.toString(a, 16) + '\n');
      System.out.println(" a  & F0 : " + Integer.toString(a & 0xF0, 16));
      System.out.println(" a  | F0 : " + Integer.toString(a | 0xF0, 16));
      System.out.println(" a  ^ F0 : " + Integer.toString(a ^ 0xF0, 16));
      }
}
```

1.4. Arithmetic Operators

- Operator priority

Level	Operator	Description	Associativity

16	`[]`
	`.`
	`()`
		access array element
		access object member
		parentheses
			left to right

15	`++` `--`	unary post-increment unary post-decrement	not associative

14	`++`
	`--`
	`+`
	`-`
	`!`
	`~`
		unary pre-increment
		unary pre-decrement
		unary plus
		unary minus
		unary logical NOT
		unary bitwise NOT
			right to left

13	`()` `new`	cast object creation	right to left
12	`* / %`	multiplicative	left to right
11	`+ -` `+`	additive string concatenation	left to right
10	`<< >>` `>>>`	shift	left to right

9	`< <=`
	`> >=`
	`instanceof`
		relational	not associative

8	`==` `!=`	equality	left to right
7	`&`	bitwise AND	left to right
6	`^`	bitwise XOR	left to right
5	`|`	bitwise OR	left to right
4	`&&`	logical AND	left to right
3	`||`	logical OR	left to right
2	`?:`	ternary	right to left

1	`= += -=`
	`*= /= %=`
	`&= ^= |=`
	`<<= >>= >>>=`
		assignment	right to left

1.4. Arithmetic Operators

```
class opWooSun {
   public static void main(String[] args)
      {
      int    x=1, y=2, z;

      z = x + y*2 - ++x + (y += 3);

      System.out.println("x:" + x + "    y:" + y + "    z:" + z + '\n');
      System.out.println("x / y * z  : " + (x / y * z));
      System.out.println("x = y += z : " + (x = y += z));
      }
}
```

1.5. Strings

- A string is a sequence of characters. Although Java does not provide a native type for string, the standard Java library provides a class called String.

- Two String variables can be concatenated using '+' operator.

```
String location = "Java";
String greeting = "Hello " + location;
```

- If we concatenate a String with a different type variable, that variable becomes a String.

```
int age = 42;
String output = age + " years"
```

1.5. Strings

```
class dtStrPlus {
   public static void main(String[] args)
   {
        String s = "Ja";
        s = s + "va";
        System.out.println(s);

        s = "square of 2 : " + 2*2;
        System.out.println(s);

        s = "Unicode of A : " + (int)'A';
        System.out.println(s);
   }
}

Results
Java
square of 2 : 4
Unicode of A : 65
```

1.5. Strings

- Static methods
  - join: concatenate strings using a delimiter character.

```
String names = String.join(", ", "Peter", "Paul", "Mary");
// names becomes "Peter, Paul, Mary".
```

- Instance methods
  - substring: return a substring of a string.

```
String greeting = "Hello, World!";
String location = greeting.substring(7,12);
// location becomes “World”.
```

  - split: return an array of strings by dividing a string using delimiter.

```
String names = "Peter, Paul, Mary";
String[] result = names.split(", ");
// result becomes ["Peter", "Paul", "Mary"]
```

1.5. Strings

```
class dtStrMethod {
   public static void main(String[] args)
      {
      String s = "JavaJAVA";

      int n = s.length();                          // length of String s : 8
      System.out.println("length of String s: " + n + "\n");

      for (int i=1 ; i<=n ; i++)
          System.out.println(s.substring(0, i)); // characters from 0 to i-1

      System.out.println('\n' + s.substring(2, 4));// characters from 2 to 3
      System.out.println(       s.substring(2));
                            // characters from 2 to the end of the string
      }
}
```

1.5. Strings

- Instance methods
  - equals: tests if the two strings are equal.

```
location.equals("World")
// if location is actually "World", this statement returns true.
```

- Converting an integer into a String
  - Use a static method Integer.toString.
  - Integer is a wrapper classes provided by the Java library.

```
int n = 42;
String str = Integer.toString(n)
```

- Converting a String into an integer
  - Use a static method Integer.parseInt.

```
String str = "101010";
int n = Integer.parseInt(str);
```

- For converting between String and double
  - Use Double.toString, Double.parseDouble.

Programming Lab #04

04-1. Conditional Operators

- What will be printed on the display when you execute this program?
- Guess first, and then run this program and see the result for yourself.
- Try changing the statement `op = '+';`.

```
public class Ex04_1 {
  public static void main(String[] args) {
    int x = 30, y = 10, z;
    char op;
    System.out.println("x: " + x + " y: " + y);

    op = '+';
    if(op == '+') z = x + y;
    else z = x - y;
    System.out.println("z: " + z);

    op = '-';
    z = (op == '+') ? x + y : x - y;
    System.out.println("z: " + z);
  }
}
```

04-2. Bitwise and Shift Operators

- What will be printed on the display when you execute this program?
- Guess first, and then run this program and see the result for yourself.

```
public class Ex04_2 {
  public static void main(String[] args) {
    char a = 0xA7;
    System.out.println("a      : " + Integer.toString(a, 16));
    System.out.println("a & F0 : " + Integer.toString(a & 0xF0, 16));
    System.out.println("a | F0 : " + Integer.toString(a | 0xF0, 16));
    System.out.println("a ^ F0 : " + Integer.toString(a ^ 0xF0, 16));

    int b = 63;
    System.out.println("b >> 1 : " + (b >> 1));
    System.out.println("b >> 2 : " + (b >> 2));
    System.out.println("b >> 3 : " + (b >> 3));
    System.out.println("b >> 4 : " + (b >> 4));
  }
}
```

04-3. Operator Precedence and Associativity

- What will be printed on the display when you execute this program?
- Guess first, and then run this program and see the result for yourself.

```
public class Ex04_3 {
  public static void main(String[] args) {
    int x = 1, y = 2, z;

    z = x + y * 2 - ++x + (y += 3);

    System.out.println("x: " + x + " y: " + y + " z:" + z);
    System.out.println("x / y * z: " + (x / y * z));
    System.out.println("x = y += z: " + (x = y += z));
  }
}
```

04-4. String Concatenation

- What will be printed on the display when you execute this program?
- Guess first, and then run this program and see the result for yourself.

```
public class Ex04_4 {
  public static void main(String[] args) {
    String s = "Ja";
    s = s + "va";
    System.out.println(s);
    s = "square of 2: " + 2*2;
    System.out.println(s);
    s = "Unicode of A: " + (int)'A';
    System.out.println(s);
  }
}
```

04-5. String Operations

- What will be printed on the display when you execute this program?
- Guess first, and then run this program and see the result for yourself.

```
public class Ex04_5 {
  public static void main(String[] args) {
    String s = "JavaJAVA";
    int n = s.length();
    System.out.println("length of String s: " + n + '\n');

    for(int i=1; i<=n; i++)
      System.out.println(s.substring(0, i));

    System.out.println();
    System.out.println(s.substring(2, 4));
    System.out.println(s.substring(2));
  }
}
```

End of Class

Instructor office: AS818A

Email: jso1@sogang.ac.kr