02_2 Operators and Control Structures

Object-Oriented Programming

Arithmetic Operators

perform mathematical operations

- **Addition (+)**: Adds two operands.
- **Subtraction (-)**: Subtracts the second operand from the first.
- **Multiplication (*)**: Multiplies two operands.
- **Division (/)**: Divides the numerator by the denominator.
- **Modulus (%)**: Returns the remainder of a division.
- **Increment (++)**: Increases the value of an operand by 1.
- **Decrement (--)**: Decreases the value of an operand by 1.

Example of Arithmetic Operation

```
public class ArithmeticOperators {
    public static void main (String[] args) {
        int a = 11, b = 5;
        int sum = a + b;        // 16
        int difference = a − b; // 6
        int product = a ∗ b;    // 55
        int quotient = a / b;   // 2
        int remainder = a % b;  // 1
        int c = a++; // c = 11
        int d = ++a; // d = 13
        int e = b−−; // e = 5
        int f = −−b; // f = 3
        System.out.println("sum = " + sum + “\n"+ … );
    }
}
```

Comparison Operators

Compare two values and return a boolean result

- **Equal to (==)**: Checks if two operands are equal.
- **Not equal to (!=)**: Checks if two operands are not equal.
- **Greater than (>)**: Checks if the first operand is greater than the second.
- **Less than (<)**: Checks if the first operand is less than the second.
- **Greater than or equal to (>=)**: Checks if the first operand is greater than or equal to the second.
- **Less than or equal to (<=)**: Checks if the first operand is less than or equal to the second.

Example of Comparison Operators

```java
public class ComparisonOperators {
    public static void main(String[] args) {
        int x = 10, y = 20;
        boolean isEqual = (x == y);          // false
        boolean isNotEqual = (x != y);       // true
        boolean isGreater = (x > y);         // false
        boolean isLess = (x < y);            // true
        boolean isGreaterOrEqual = (x >= y); // false
        boolean isLessOrEqual = (x <= y);    // true
    }
}
```

Logical Operators

- Combining multiple boolean expressions
- **AND (&&)**: Returns true if both operands are true.
- **OR (||)**: Returns true if at least one operand is true.
- **NOT (!)**: Reverses the logical state of its operand.

```java
public class LogicalOperators {
    public static void main(String[] args) {
        boolean a = true, b = false;
        boolean andResult = a && b; // false
        boolean orResult = a || b;  // true
        boolean notResult = !a;     // false
    }
}
```

Assignment Operators

- assign values to variables.
- **Simple assignment (=)**:
  - Assigns the right-hand side value to the left-hand side variable.
- **Compound assignment (+=, -=, *=, /=, %=)**:
  - Combines an arithmetic operation with assignment.

```java
public class AssignmentOperators {
    public static void main(String[] args) {
        int a = 10;
        a += 5; // a = a + 5;  // 15
        a -= 3; // a = a - 3;  // 12
        a *= 2; // a = a * 2;  // 24
        a /= 4; // a = a / 4;  // 6
        a %= 3; // a = a % 3;  // 0
    }
}
```

Negative Integer (1/2)

- Negative number representation = 2’s complement of positive number
  - 1’s complement of positive a = ~a
  - 2’s complement of positive a = ~a + 1

```
public class NegativeInteger {
    public static void main(String[] args) {
        int a = 5;   // a =  5     (00000101)
        a = ~a;      // a = ~a     (11111010) 1's complement of a
        a += 1;      // a = ~a + 1 (11111011) 1's complement + 1 = 2's complement
        a = -5;      // a = -5     (11111011) 2's complement 5 = -5
    }
}
```

Negative Integer (2/2)

- From 2’s complement to negative integer
  - (-a) - 1 = 1’s complement of a
  - ~((-a) - 1) = a
  - a is positive integer, so the original binary number represents the negative integer -a
- Ex) A = 11111011
  - A - 1 = 11111010
  - ~(A - 1) = 00000101 = +5
  - So A represents -5

Bitwise Operators

Bitwise operators perform operations on bits.

- **AND (&)**: Performs a bitwise AND.
- **OR (|)**: Performs a bitwise OR.
- **XOR (^)**: Performs a bitwise XOR.
- **NOT (~)**: Performs a bitwise NOT.
- **Left shift (<<)**: Shifts bits to the left.
- **Right shift (>>)**: Shifts bits to the right.

Example of Bitwise Operators

```
public class BitwiseOperators {
    public static void main(String[] args) {
        int a = 5, b = -3;          // a =  5       (00000101)
                                    // b = -3       (11111101)
        int andResult = a & b;      // a & b =  5   (00000101)
        int orResult = a | b;       // a | b = -3   (11111101)
        int xorResult = a ^ b;      // a ^ b = -8   (11111000)
        int notResult = ~a;         // ~a = -6      (11111010)
        int leftShift = a << 1;     // a << 1 = 10  (00001010)
        int leftShiftB = b << 1;    // b << 1 = -6  (11111010)
        int rightShift = a >> 1;    // a >> 1 =  2  (00000010)
        int rightShiftB = b >> 1;   // b >> 1 = -2  (11111110)
        int right2Shift = a >> 2;   // a >> 2 =  1  (00000001)
        int right2ShiftB = b >> 2; // b >> 2 = -1  (11111111)
    }
}
```

Ternary Operator

- The ternary operator is a shorthand for the if-else statement.
- **Syntax**: condition ? value if true : value if false

```java
public class TernaryOperator {
    public static void main(String[] args) {
        int a = 10, b = 20;
        int max = (a > b) ? a : b; // 20
    }
}
```

if Statement (1/2)

```java
int num = 5;
if (num > 5) {
    System.out.println("Number is greater than 5");
}

num = 2;
if (num > 5) {
    System.out.println("Number is greater than 5");
}
else {
    System.out.println("Number is not greater than 5");
}
```

if Statement (2/2)

```
if (A) { U }
else if (B) { V }
else if (C) { X }
else { Y }
Z

if condition A is true, U and goto Z
if condition B is true, V and goto Z
if condition C is true, X and goto Z
else Y and goto Z
```

switch Statement (1/3)

```java
import java.util.Scanner;

public class SwitchStatement {
    public static void main(String[] args) {
        int score;
        char grade;
        Scanner keyboard = new Scanner(System.in);

        System.out.print("What is your score? ");
        score = keyboard.nextInt();
        int scoreOverTen = score / 10;
```

```
Prompt: What is your score? 73
```

switch Statement (2/3)

```
switch (scoreOverTen) {
    case 10:
    case 9:
        grade = 'A';
        break;
    case 8:
        grade = 'B';
        break;
    default:
        grade = 'C';
}

System.out.println("Score: " + score + " Grade: " + grade + "\n");
```

switch Statement (3/3)

```
        System.out.print("Choose your menu (Americano, CafeLatte): ");
        String menu = keyboard.next();
        int sales = 0;

        switch (menu) {
            case "Americano":
                sales += 3500;
                break;
            case "CafeLatte":
                sales += 4500;
                break;
            default:
                System.out.println("Wrong coffee menu.. system exit...");
                System.exit(0);
        }

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

for Loop (1/2)

```java
public class ForLoop {
    public static void main(String[] args) {
        // basic for loop
        for (int i = 0; i < 5; i++) {
            System.out.println(i);
        }
        // nested for loop
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.println("i: " + i + ", j: " + j);
            }
        }
```

```
0
1
2
3
4

i: 1, j: 1
i: 1, j: 2
i: 1, j: 3
i: 2, j: 1
i: 2, j: 2
i: 2, j: 3
i: 3, j: 1
i: 3, j: 2
i: 3, j: 3
```

for Loop (2/2)

```
        // for-each
        int[] numbers = {1, 2, 3, 4, 5};
        for (int n : numbers) {
            System.out.println(n);
        }
    }
}
```

while Loop

```
public class WhileLoop {
    public static void main(String[] args) {
        int i = 0;
        while (i < 5) {
            System.out.println(i);  // 0 1 2 3 4
            i++;
        }

        i = 0;
        while (true) {  // infinite loop
            if (i >= 5) {  // using break statment to exit from the loop
                break;
            }
            System.out.println(i);  // 0 1 2 3 4
            i++;
        }
    }
}
```

do-while Loop

```
public class DoWhileLoop {
    public static void main(String[] args) {
        int i = 0;
        do {
            System.out.println(i); // 0 1 2 3 4
            i++;
        } while (i < 5);

        i = 0;
        do {
            System.out.println("printed at least once.");
            i++;
        } while (i < 0);

    }
}
```

break and continue

```
public class LoopControl {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            if (i == 5) {
                break; // exit out the for loop
            }
            System.out.println(i);
        }
        for (int i = 0; i < 10; i++) {
            if (i % 2 == 0) {
                continue; // go up to for
            }
            System.out.println(i);
        }
    }
}
```

```
0
1
2
3
4
1
3
5
7
9
```