PROGRAMMING IN JAVA

Lecture 3

Expressions and Operators

Arithmetic, Relational, Logical, Bitwise, Assignment Operators, Precedence, and Expressions

Lecturer: Arshyn Altybay

Department of Computer Engineering,
Almaty Technological University, Almaty, Kazakhstan

1. Learning Objectives

By the end of this lecture, students should be able to:

- define expressions and operators;
- use arithmetic, unary, relational, equality, logical, and assignment operators;
- explain integer division and the remainder operator;
- use prefix and postfix increment/decrement correctly;
- use compound assignment operators;
- explain short-circuit evaluation;
- use bitwise and shift operators at an introductory level;
- apply operator precedence and associativity;
- use explicit parentheses to clarify expressions;
- use the conditional operator;
- explain string concatenation with `+`;
- identify common expression-related errors.

2. Expressions and Operators

An *expression* is a combination of literals, variables, operators, and method calls that produces a value.

Examples:

```
a + b
price * quantity
score >= 50
x > 0 && y > 0
```

An operator specifies the operation to perform on one or more operands.

3. Arithmetic Operators

Java provides the standard arithmetic operators.

Operator	Name	Example
+	addition	a + b
-	subtraction	a - b
*	multiplication	a * b
/	division	a / b
%	remainder	a % b

Example 1. Basic arithmetic

```
1  public class ArithmeticExample {
2      public static void main(String[] args) {
3          int a = 17;
4          int b = 5;
5
6          System.out.println("a + b = " + (a + b));
7          System.out.println("a - b = " + (a - b));
8          System.out.println("a * b = " + (a * b));
9          System.out.println("a / b = " + (a / b));
10         System.out.println("a % b = " + (a % b));
11     }
12 }
```

Expected output

```
a + b = 22
a - b = 12
a * b = 85
a / b = 3
a % b = 2
```

4. Integer and Floating-Point Division

When both operands are integer types, Java performs integer division.

Example 2. Division depends on operand types

```java
public class DivisionExample {
    public static void main(String[] args) {
        int a = 5;
        int b = 2;

        System.out.println(a / b);
        System.out.println((double) a / b);
        System.out.println(5.0 / 2);
    }
}
```

Expected output

```
2
2.5
2.5
```

The remainder operator `%` is useful for parity tests, cyclic calculations, and divisibility checks.

5. Unary Operators

Unary operators act on one operand.

```
+x
-x
!flag
++count
--count
```

The logical negation operator `!` changes `true` to `false` and vice versa.

6. Increment and Decrement

The operators ++ and – add or subtract one.

Prefix form changes the value before the surrounding expression uses it; postfix form yields the previous value first.

Example 3. Prefix and postfix increment

```
public class IncrementExample {
    public static void main(String[] args) {
        int x = 5;

        System.out.println(x++);
        System.out.println(x);
        System.out.println(++x);
        System.out.println(x);
    }
}
```

Expected output

```
5
6
7
7
```

> Avoid placing many increment/decrement operations inside one complicated expression. Clear separate statements are easier to read and debug.

7. Assignment Operators

The simple assignment operator is =.

```
int score = 80;
score = 90;
```

Compound assignment operators combine an operation and assignment.

Operator	Example	Approximate meaning
`+=`	`x += 5`	`x = x + 5`
`-=`	`x -= 5`	`x = x - 5`
`*=`	`x *= 5`	`x = x * 5`
`/=`	`x /= 5`	`x = x / 5`
`%=`	`x %= 5`	`x = x % 5`

8. Relational and Equality Operators

Relational and equality operators produce Boolean results.

Operator	Meaning
`<`	less than
`<=`	less than or equal to
`>`	greater than
`>=`	greater than or equal to
`==`	equal to
`!=`	not equal to

Example 4. Comparing numeric values

```
public class ComparisonExample {
    public static void main(String[] args) {
        int score = 78;

        System.out.println(score >= 50);
        System.out.println(score == 100);
        System.out.println(score != 0);
    }
}
```

Expected output

```
true
false
true
```

Common mistake: Do not confuse assignment `=` with equality comparison `==`.

9. Logical Operators

Logical operators combine or negate Boolean expressions.

Operator	Meaning
`&&`	logical AND
`||`	logical OR
`!`	logical NOT

Example 5. Combining conditions

```
public class LogicalExample {
    public static void main(String[] args) {
        int age = 20;
        boolean documentsComplete = true;

        boolean eligible =
                age >= 18 && documentsComplete;

        System.out.println(eligible);
        System.out.println(age < 18 || !documentsComplete);
    }
}
```

Expected output

```
true
false
```

10. Short-Circuit Evaluation

The operators `&&` and `||` use short-circuit evaluation.

For `&&`, if the left operand is false, the right operand does not need to be evaluated.

For `||`, if the left operand is true, the right operand does not need to be evaluated.

Example 6. Using short-circuit evaluation safely

```
public class ShortCircuit {
    public static void main(String[] args) {
        int denominator = 0;

        boolean safe =
                denominator != 0
                && 10 / denominator > 1;

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

Expected output

```
false
```

The division is never evaluated because the first condition is false.

11. Bitwise Operators

Bitwise operators manipulate the binary representation of integer values.

Operator	Operation
`&`	bitwise AND
`|`	bitwise OR
`^`	bitwise XOR
`~`	bitwise complement

Example 7. Bitwise AND and OR

```
public class BitwiseExample {
    public static void main(String[] args) {
        int a = 6; // binary 110
        int b = 3; // binary 011

        System.out.println(a & b);
        System.out.println(a | b);
        System.out.println(a ^ b);
    }
}
```

Expected output

```
2
7
5
```

12. Shift Operators

Java provides left and right shift operators.

```
x << n
x >> n
x >>> n
```

For positive values, shifting left by one position often corresponds to multiplication by two when overflow does not occur.

Example 8. Simple shifts

```
1 public class ShiftExample {
2     public static void main(String[] args) {
3         int value = 8;
4 
5         System.out.println(value << 1);
6         System.out.println(value >> 1);
7     }
8 }
```

Expected output

```
16
4
```

13. Operator Precedence and Associativity

Precedence determines which operators bind more strongly.

For example:

```
int result = 2 + 3 * 4;
```

Multiplication is performed first, so the result is 14.

A simplified precedence order from higher to lower includes:

Level	Examples
postfix	`x++`, `x--`
unary	`++x`, `--x`, `!`, unary `+` and `-`
multiplicative	`*`, `/`, `%`
additive	`+`, `-`
shift	`<<`, `>>`, `>>>`
relational	`<`, `<=`, `>`, `>=`
equality	`==`, `!=`
logical AND	`&&`
logical OR	`||`
conditional	`?:`
assignment	`=`, `+=`, `-=`, etc.

> Use parentheses when they make the intended calculation clearer, even when operator precedence already gives the correct result.

14. The Conditional Operator

The conditional operator is Java's three-operand operator.

```
condition ? valueIfTrue : valueIfFalse
```

Example 9. Choosing a value with ?:

```
1  public class ConditionalOperator {
2      public static void main(String[] args) {
3          int score = 72;
4
5          String result =
6                  score >= 50 ? "Pass" : "Fail";
7
8          System.out.println(result);
9      }
10 }
```

Expected output

```
Pass
```

Use it for concise value selection, not for complex multi-step logic.

15. String Concatenation with +

When one operand of `+` is a `String`, Java performs string concatenation.

Example 10. Concatenation and arithmetic

```
public class ConcatenationExample {
    public static void main(String[] args) {
        int a = 5;
        int b = 3;

        System.out.println("Sum = " + a + b);
        System.out.println("Sum = " + (a + b));
    }
}
```

Expected output

```
Sum = 53
Sum = 8
```

The first expression is evaluated from left to right after string concatenation begins.

16. Comprehensive Example: Student Result Calculation

Example 11. Combining several operator categories

```
public class StudentResult {
    public static void main(String[] args) {
        double assignment = 82.0;
        double midterm = 76.0;
        double finalExam = 90.0;

        double weighted =
                assignment * 0.30
                + midterm * 0.30
                + finalExam * 0.40;

        boolean passed =
                weighted >= 50.0;

```

Example 11 (continued)

```
        String status =
                passed ? "PASS" : "FAIL";

        int rounded =
                (int) Math.round(weighted);

        System.out.println(
                "Weighted score = " + weighted
        );
        System.out.println(
                "Rounded score = " + rounded
        );
        System.out.println(
                "Status = " + status
```

Example 11 (continued)

```
        );
    }
}
```

Expected output

```
Weighted score = 83.4
Rounded score = 83
Status = PASS
```

17. Common Errors and Good Practices

- Do not confuse `=` and `==`.
- Remember that integer division truncates the fractional part.
- Use short-circuit logical operators for Boolean conditions unless bitwise behavior is specifically required.
- Avoid excessively complicated increment/decrement expressions.
- Use parentheses to make important grouping explicit.
- Be careful when concatenating numbers with strings.
- Do not assume arithmetic can never overflow.
- Use meaningful intermediate variables when an expression becomes hard to read.

18. Lecture Summary

- Expressions produce values by combining operands and operators.
- Arithmetic operators perform numeric calculations.
- Relational and equality operators produce Boolean results.
- Logical operators combine Boolean conditions.
- `&&` and `||` use short-circuit evaluation.
- Compound assignment combines calculation and assignment.
- Bitwise and shift operators operate on integer bit patterns.
- Precedence and associativity determine expression grouping.
- The conditional operator selects one of two values.
- String concatenation changes the meaning of `+` when a `String` is involved.

19. Review Questions

1. What is an expression?
2. What is an operator?
3. What does the remainder operator do?
4. Why is `5 / 2` equal to 2 when both operands are integers?
5. What is the difference between prefix and postfix increment?
6. What is compound assignment?
7. What is the difference between `=` and `==`?
8. Which operators compare numeric values?
9. What do `&&`, `||`, and `!` mean?
10. What is short-circuit evaluation?
11. What is a bitwise operator?
12. What is the purpose of shift operators?
13. What is operator precedence?
14. Why are parentheses useful?
15. What is the conditional operator?
16. How does `+` behave when a string is involved?

20. Programming Exercises

1. Calculate the area and perimeter of a rectangle.
2. Convert Celsius to Fahrenheit.
3. Given total seconds, calculate hours, minutes, and remaining seconds.
4. Determine whether a number is even using the remainder operator.
5. Demonstrate prefix and postfix increment.
6. Use compound assignment operators with one integer variable.
7. Compare two numbers and print all relational results.
8. Combine age and document-status conditions using logical operators.
9. Demonstrate short-circuit evaluation safely.
10. Apply bitwise AND, OR, and XOR to two small integers.
11. Demonstrate left and right shifts.
12. Rewrite a simple `if-else` value choice using `?:`.
13. Predict and verify the output of several mixed string/numeric expressions.
14. Create a weighted-score calculator using several arithmetic operators.

21. Homework

Create a program called `SalaryCalculator`. Store:

- basic salary;
- bonus percentage;
- tax percentage;
- number of overtime hours;
- overtime hourly rate.

Calculate gross income, tax, and final income. Use arithmetic operators, compound assignment where meaningful, relational/logical expressions to check whether the final salary is positive, and the conditional operator to display a simple status message.

22. Key Terms

Term	Meaning
expression	combination of values, variables, calls, and operators that produces a value
operand	value on which an operator acts
arithmetic operator	operator used for numeric calculation
relational operator	operator that compares ordered values
logical operator	operator that combines or negates Boolean values
short-circuit evaluation	evaluation strategy that may skip the right operand of `&&` or `||`
compound assignment	combined calculation and assignment such as `+=`
precedence	rule describing which operators bind more strongly
associativity	rule describing grouping when operators have equal precedence
conditional operator	three-operand `?:` operator used to select a value