Chapter 4: Operators (Java)

1. Arithmetic Operators

These operators are used for basic mathematical calculations.

Operator	Description	Example
`+`	Addition	`a + b`
`-`	Subtraction	`a - b`
`*`	Multiplication	`a * b`
`/`	Division	`a / b`
`%`	Modulus (remainder)	`a % b`

2. The Modulus Operator

- Returns the remainder of a division.

  Example:
  ```
  int result = 10 % 3; // result is 1
  ```

3. Arithmetic Compound Assignment Operators

Combine arithmetic and assignment in one step.

Operator	Equivalent To
`+=`	`a = a + b`
`-=`	`a = a - b`
`*=`	`a = a * b`
`/=`	`a = a / b`
`%=`	`a = a % b`

4. Increment and Decrement

- **Increment (`++`)**: Adds 1 to the operand.
  - Pre-increment (`++a`): Increment before use.
  - Post-increment (`a++`): Increment after use.
- **Decrement (`--`)**: Subtracts 1 from the operand.

- Pre-decrement (`--a`): Decrement before use.
- Post-decrement (`a--`): Decrement after use.

---

5. The Bitwise Operators

Operate directly on bits of integers.

| Operator | Description | Example | |----------|-------------------------|-----------|| `&` | Bitwise AND | `a & b` || `|` | Bitwise OR | `a | b` || `^` | Bitwise XOR | `a ^ b` || `~` | Bitwise complement | `~a` || `<<` | Left shift | `a << 2` || `>>` | Right shift | `a >> 2` || `>>>` | Unsigned right shift | `a >>> 2` |

---

6. Bitwise Logical Operators

- Perform logical operations on bits.
  Example:
```
int a = 5; // 0101
int b = 3; // 0011
System.out.println(a & b); // 0001 (1)
```

---

7. Using Logical Bitwise Operators

- Use `&` , `|` , `^` to combine logical conditions.
  Example:
```
if ((a & b) > 0) {
    // Perform action
}
```

---

8. Shift Operators

- **Left Shift (`<<`)**: Shifts bits to the left, filling with 0s on the right.
  Example:

```
int a = 5; // 0000 0101
int result = a << 2; // 0001 0100 (20)
```

- **Right Shift (`>>`)**: Shifts bits to the right, maintaining the sign bit.
  Example:

```
int a = -5; // 1111 1011
int result = a >> 2; // 1111 1110 (-2)
```

- **Unsigned Right Shift (`>>>`)**: Shifts bits to the right, filling with 0s.
  Example:

```
int a = -5; // 1111 1011
int result = a >>> 2; // 0011 1110 (1073741822)
```

---

9. Bitwise Operator Compound Assignment

- Combine bitwise and assignment operations.

Operator	Equivalent To
`&=`	`a = a & b`
`|=`	`a = a | b`
`^=`	`a = a ^ b`
`<<=`	`a = a << b`
`>>=`	`a = a >> b`
`>>>=`	`a = a >>> b`

---

10. Relational Operators

Compare two values and return a boolean result.

Operator	Description	Example
`==`	Equal to	`a == b`
`!=`	Not equal to	`a != b`
`>`	Greater than	`a > b`
`<`	Less than	`a < b`
`>=`	Greater or equal	`a >= b`
`<=`	Less or equal	`a <= b`

---

11. Boolean Logical Operators

Operator	Description	Example
`&&`	Logical AND	`a && b`
`||`	Logical OR	`a || b`
`!`	Logical NOT	`!a`

---

12. Short Circuit Logical Operators

- `&&` **(AND)** and `||` **(OR)** evaluate the second operand only if necessary.
  Example:
  ```
  if (a > 5 && b < 10) {
      // Executes if both conditions are true
  }
  ```

---

13. The Assignment Operators

Operator	Example
`=`	`a = b`
`+=`	`a += b`
`-=`	`a -= b`

---

14. The Ternary (`? :`) Operator

- A shorthand for `if-else` conditions.
  Syntax:

  ```
  result = (condition) ? value1 : value2;
  ```
  Example:

  ```
  int max = (a > b) ? a : b;
  ```

---

15. Operator Precedence

Determines the order in which operators are evaluated.

| Operator Type | Operators | |---|---| | Postfix | `expr++` `expr--` | | Unary | `++expr` `--expr` `+ - !` | | Multiplicative | `* / %` | | Additive | `+ -` | | Relational | `< > <= >=` | | Equality | `== !=` | | Logical AND | `&&` | | Logical OR | `\|\|` | | Assignment | `= += -= *= /= %=` |

---

16. Using Parentheses

- Use parentheses to explicitly define the order of evaluation.
  Example:
  ```
  int result = (a + b) * c; // Ensures a + b is evaluated first
  ```

These notes summarize all the key points of Chapter 4.