Java Operators :

Certainly! Java operators are symbols that perform operations on variables and values. They are used to manipulate data and perform various calculations. Java operators can be categorized into different groups based on their functionality.

Here, I'll explain each category of Java operators with examples:

1. **Arithmetic Operators**:
   These operators perform basic arithmetic calculations.

   - **+ (Addition)**: Adds two values.
   ```java
   int sum = 5 + 3; // sum will be 8
   ```

   - **- (Subtraction)**: Subtracts one value from another.
   ```java
   int difference = 10 - 5; // difference will be 5
   ```

   - *** (Multiplication)**: Multiplies two values.
   ```java
   int product = 4 * 6; // product will be 24
   ```

   - **/ (Division)**: Divides one value by another (integer division).
   ```java
   int quotient = 20 / 4; // quotient will be 5
   ```

   - **% (Modulus)**: Returns the remainder of division.
   ```java
   int remainder = 17 % 5; // remainder will be 2
   ```

2. **Assignment Operators**:
   These operators are used to assign values to variables.

   - **= (Assignment)**: Assigns a value to a variable.
   ```java
   int x = 10; // x is assigned the value 10
   ```

- **+= (Add and Assign)**: Adds a value to the variable and assigns the result back to the variable.
  ```java
  int y = 5;
  y += 3; // y will be 8
  ```

  (Similar compound assignment operators exist for other arithmetic operators like -=, *=, /=, %=, etc.)

3. **Comparison Operators**:
  These operators compare two values and return a boolean result.

  - **== (Equal to)**: Checks if two values are equal.
  ```java
  boolean isEqual = 5 == 5; // isEqual will be true
  ```

  - **!= (Not equal to)**: Checks if two values are not equal.
  ```java
  boolean isNotEqual = 6 != 5; // isNotEqual will be true
  ```

  - **< (Less than)**, **> (Greater than)**, **<= (Less than or equal to)**, **>= (Greater than or equal to)**: Compare values for ordering.

4. **Logical Operators**:
  These operators are used to perform logical operations.

  - **&& (Logical AND)**: Returns true if both operands are true.
  ```java
  boolean bothTrue = true && false; // bothTrue will be false
  ```

  - **|| (Logical OR)**: Returns true if at least one operand is true.
  ```java
  boolean eitherTrue = true || false; // eitherTrue will be true
  ```

  - **! (Logical NOT)**: Negates the value of a boolean expression.
  ```java
  boolean notValue = !true; // notValue will be false
  ```

5. **Increment and Decrement Operators**:
   These operators are used to increase or decrease the value of a variable.

   - **++ (Increment)**: Increases the value of a variable by 1.
   ```java
   int count = 5;
   count++; // count will be 6
   ```

   - **-- (Decrement)**: Decreases the value of a variable by 1.
   ```java
   int score = 10;
   score--; // score will be 9
   ```

6. **Bitwise Operators**:
   These operators perform bitwise operations on individual bits of values.

   - **& (Bitwise AND)**, **| (Bitwise OR)**, **^ (Bitwise XOR)**: Perform bitwise operations.
   ```java
   int a = 5; // 0101 in binary
   int b = 3; // 0011 in binary
   int resultAND = a & b; // 0001, which is 1 in decimal
   int resultOR = a | b; // 0111, which is 7 in decimal
   int resultXOR = a ^ b; // 0110, which is 6 in decimal
   ```

7. **Shift Operators**:
   These operators shift the bits of a value left or right.

   - **<< (Left Shift)**: Shifts the bits of a number to the left.
   ```java
   int value = 5; // 0101 in binary
   int shifted = value << 2; // 010100, which is 20 in decimal
   ```

   - **>> (Right Shift)**: Shifts the bits of a number to the right (with sign extension).
   ```java
   int value = -8; // 11111111111111111111111111111000 in binary (32-bit two's complement)
   int shifted = value >> 2; // 11111111111111111111111111111110, which is -2 in decimal
   ```

- **>>> (Unsigned Right Shift)**: Shifts the bits of a number to the right (without sign extension).
  ```java
  int value = -8; // 11111111111111111111111111111000 in binary (32-bit two's complement)
  int shifted = value >>> 2; // 00111111111111111111111111111110, which is 1073741822 in decimal
  ```

These are the main categories of Java operators, and they allow you to perform a wide range of operations on variables and values. Remember that the specific behavior of operators can vary depending on the data types involved and the rules of the programming language.