**Operators in Java**

**Operators:**

- Operators are special symbols used to perform operations on variables and values.

- Ex: int a = 10;

int b = 5;

int sum = a + b;

System.out.println(“Sum: ” + sum);

**Types of Operators:**

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. **Arithmetic Operators:**

**-** Arithmetic operators used to perform mathematical operations on numeric values.

1. **Addition (+):** This operator is used to adds two numbers together.

Ex: int a = 2, b = 4;

int sum = a + b;

1. **Subtraction (-):** This operator is used to subtracts one number from another.

Ex: int a = 5, b = 2;

int sum = a - b;

1. **Multiplication (\*):** This operator is used to multiplies two numbers together.

Ex: int a = 2, b = 4;

int sum = a \* b;

1. **Division (/):** This operator is used for divides one number by another.

Ex: int a = 5, b = 3;

int sum = a / b;

1. **Modulus (%):** This operator returns the remainder of a division operation.

Ex: int a = 10, b = 5;

int sum = a % b;

1. **Relational Operators:**

**-** Relational operators is used to compare two values and return a Boolean result.

1. **Equal to (==):** Returns true if two values are equal else it returns false.

Ex1:int x = 2; y = 2;

System.out.println(x == y); //true

Ex2: int x = 4; y = 2;

System.out.println(x == y); //false

1. **Not equal to (!=):** Returns true if two values are not equal else it returns false.

Ex1:int x = 2; y = 3;

System.out.println(x != y); //true

Ex2: int x = 2; y = 2;

System.out.println(x != y); //false

1. **Greater than (>):** Returns true if one value is greater than another else it returns false.

Ex1:int x = 5; y = 2;

System.out.println(x > y); //true

Ex2: int x = 2; y = 8;

System.out.println(x > y); //false

1. **Less than (<):** Returns true if one value is less than another else it returns false.

Ex1:int x = 5; y = 8;

System.out.println(x < y); //true

Ex2: int x = 10; y = 4;

System.out.println(x < y); //false

1. **Greater than or equal to (>=):** Returns true if one value is greater than or equal to another else it returns false.

Ex1: int x = 10, y = 5;

System.out.println(x>=y); //true

Ex2: int x = 5; y = 10;

System.out.println(x>=y); //false

1. **Less than or equal to (<=):** Returns true if one value is less than or equal to another else it returns false.

Ex1: int x = 4, y = 8;

System.out.println(x<=y); //true

Ex2: int x = 10; y = 4;

System.out.println(x<=y); //false

1. **Logical Operators:**

**-** Logical operators is used for combines conditional statements and return a Boolean result.

1. **Logical AND (&&):** Returns true if both statements are true else it returns false.

Ex1: int x = 5;

System.out.println(x > 3 && x < 10); //true

Ex2: int x = 5;

System.out.println(x < 3 && x > 10); //false

1. **Logical OR (||):** Returns true if any one of the statements is true and if both statements are false it returns false.

Ex1: int x = 5;

System.out.println(x > 3 || x < 4); //true

Ex2: int x = 5;

System.out.println(x < 3 || x > 10); //false

1. **Logical NOT (!):** Reverse the result, returns false if the result is true and returns true if the result is false.

Ex1: int x = 5;

System.out.println(! (x > 3 && x < 10)); //false

Ex2: int x = 5;

System.out.println(x < 3 && x < 10); //true

* **Operator Precedence:**
* Java evaluates operators in a specific order
* Highest precedence → Lowest precedence: ◦
* () Parentheses
* \* / % Multiplication/Division/Modulus
* + - Addition/Subtraction
* < > <= >= Relational
* == != Equality
* && AND
* || OR
* Example: int result = 10 + 5 \* 2; // 20, not 30

**Introduction to Classes and Objects**

**Classes:**

- In java, classes are basic concepts of Object-Oriented Programming (OOPs) that are used to represent real-world concepts and entities.

- A blueprint for creating objects.

- Defines attributes and behaviors (methods).

**Objects:**

- An object is an instance of a class.

- For example, the Animal type Dog is a class, while a particular dog named Tommy is an object of the Dog class.

- Declaration: Dog tommy = new Dog(arguments);

- Multiple objects can be created from the same class.

* **Memory Representation:**
* Class: Only definition (no memory until object created).
* Object: Occupies memory when created.
* Example: Two car objects each with their own color.

**Constructors and Methods**

**Constructors:**

- A constructor in Java is a Specialized method used for initializing objects of a class.

- It is automatically invoked when an instance of a class is created using new keyword.

- Same Name as the class.

- No return type.

**Types of Constructors:**

1. **Default constructor:** Takes no parameters and is used to create object with default initial values.
2. **Parameterized constructor:** Takes one or more parameters, allowing you to initialize an object’s fields with specific values provided during object creation.

**Methods:**

- A method in Java is a block of code designed to perform a specific task or action.

- All methods in java must be defined within a class.

- A method contains a set of instructions that are executed when the method is called.

- Methods allow you to write a block of code once and call it multiple times.

- Any name and it has return type.

- Syntax: returnType mthodName(parameters) {

//body

}

**Types of Methods:**

1. **Instance Method:** Access the instance data using the object name. Declared inside a class.

Example: void method\_name(){

// instance method body

}

1. **Static Method:** Access the static data using class name. Declared inside class with static keyword.

Example: static void method\_name(){

//static method body

}

* **Difference between Constructor VS Method.**

|  |  |
| --- | --- |
| **Constructor** | **Method** |
| Initialize the state of object | Defines behaviour of object |
| No return type | Has return type |
| Same name as class | Any name other than keywords |
| If the developer does not write a constructor, the default constructor will be called | There are no default methods |
| Automatically called when an object is instantiated. | Method are called during the execution of a program |