operators-and-conditional-statements

Pranjal Aggarwal

Masters from NIT Allahabad
Upcoming Data Analyst at Citi

October 26, 2024

Agenda

- Data Types Recap
- Operators in Java
- If-Else Statements
- Loops

Examples of Using Data Types

Integer (int):
Stores whole numbers without decimal points, commonly used in cases where precision is not required.
`int age = 25;`
Scenario: Integer values are used for age, year, or item count, like `int itemCount = 100;` for stock in a warehouse.

Double (double):
Stores decimal numbers, allowing us to work with more precise values, like currency or measurements.
`double price = 19.99;`
Scenario: Product prices, distances, and weights often require decimal values, like `double distance = 3.75;` for kilometers.

Examples of Using Data Types

Boolean (boolean):
Holds only true or false, useful for making decisions and setting conditions in a program.
`boolean isStudent = true;`
Scenario: Booleans work well for yes/no questions like `boolean isMember = false;` or checks like `boolean isLoggedIn = true;` for a user's status.

Character (char):
Stores single letters or symbols, commonly used in grading systems or initials.
`char grade = 'A';`
Scenario: Grades in exams, initials, or simple codes like `char code = 'X';`.

Examples of Using Data Types

String (String):
Stores a sequence of characters, useful for names, addresses, and messages.
`String name = "Alice";`
Scenario: Strings can hold names like `String fullName = "Alice Johnson";`, addresses, or descriptions such as `String productDescription = "High-quality headphones";`.

Data types help in organizing and storing different forms of information, ensuring that each type is handled effectively in the program.

Printing to the Screen

In Java, we use the `System.out.println()` method to print text or values to the console.

This method is essential for debugging and displaying results.

Example:

- `System.out.println("Hello, World!");`

This will output: `Hello, World!`

Taking User Input

To get input from the user, we can use the `Scanner` class.

This allows us to read various types of data, like strings and numbers.

Example:

- `Scanner scanner = new Scanner(System.in);`
- ```
  System.out.print("Enter your name:  ");
  String name = scanner.nextLine();
  ```

Here, the program prompts the user to enter their name, which is then stored in the `name` variable.
  ```

Categories of Operators in Java

- **Arithmetic Operators:** Used for basic math operations.
  - Example: `int sum = 5 + 3;` (Addition)
- **Relational Operators:** Compare values and return true or false.
  - Example: `boolean isEqual = (5 == 3);` (Equality check)
- **Logical Operators:** Combine multiple conditions.
  - Example: `boolean result = (5 > 3) && (3 < 8);` (AND operator)
- **Assignment Operators:** Assign values to variables.
  - Example: `int number = 10;` (Assignment)

Categories of Operators in Java (Continued)

- **Increment and Decrement Operators:** Increase or decrease values by 1.
  - Example: `int count = 5; count++;` (Increment)
- **Conditional (Ternary) Operator:** Short-hand for simple if-else.
  - Example: `int min = (a < b) ? a : b;` (Ternary conditional)

Each operator type provides different functionality, helping in calculations, decisions, and efficient code structure.

If-Else: Decisions Ahead!

"If it's raining, take an umbrella.

Else, wear sunglasses!"

If-Else Statements

In Java, **if-else statements** are used to execute different blocks of code based on certain conditions. This allows the program to make decisions and take actions accordingly.

The `if` statement checks a condition:

- If the condition is `true`, the code inside the `if` block runs.
- If the condition is `false`, the code inside the `else` block runs (if provided).

This structure is essential for implementing logic in programs and handling different scenarios.

Example:

```
if (age > 18) {
   System.out.println("Adult");
} else {
   System.out.println("Not Adult");
}
```

LOOOOOOPS Ahead!

"Hey, could you order me 1 pizza?"

"Wait, how about 100 pizzas?"

"Oh no, not 1000 pizzas!"

"Chill! Loops are here to handle all those orders without breaking a sweat!"

For Loop

For Loops are useful when we know the exact number of times we want to repeat a set of actions.

A `for` loop consists of:

- **Initialization:** Set a starting point.
- **Condition:** Define when the loop should stop.
- **Increment/Decrement:** Modify the loop variable each time.

Example:
```
for (int i = 0; i < 5; i++) {
    System.out.println(i);
}
```

This example prints numbers from 0 to 4.

While Loop

While Loops are used when we want to repeat a set of actions as long as a specific condition is true.

A `while` loop will continue until the condition becomes false.

Example:
```
int i = 0;
while (i < 5) {
    System.out.println(i);
    i++;
}
```

This example prints numbers from 0 to 4, as the condition `i < 5` is checked before each loop.

Do-While Loop

Do-While Loops are similar to while loops but always execute at least once, as the condition is checked after the loop's body.

This is useful when we want the action to run at least once, regardless of the condition.

Example:
```
int i = 0;
do {
    System.out.println(i);
    i++;
} while (i < 5);
```

This example also prints numbers from 0 to 4 but ensures the loop body executes once before checking `i < 5`.