Using Operators

Estimated time needed: 20 minutes

In this lab, you will learn about the different operators that are available for you to use in a Java program.

You are currently viewing this lab in a Cloud based Integrated Development Environment (Cloud IDE). It is a fully-online integrated development environment that is pre-installed with JDK 21, allowing you to code, develop, and learn in one location.

Learning Objectives

After completing this lab, you will be able to:

- Understand the different types of operators in Java
- Use operators in a Java program
- Create a program using the operators with different data types
- Explore use cases for using operators

Mathematical operators

Mathematical operators are what most of us are familiar with. These include the `+`(plus), `-`(minus), `*` (multiplication), `/` (division), `%` (modulus).

1. Create a project directory by running the following command.

   ```
   mkdir my_operator_proj
   ```

2. Run the following code to create the directory structure.

   ```
   mkdir -p my_operator_proj/src
   mkdir -p my_operator_proj/classes
   mkdir -p my_operator_proj/test
   cd my_operator_proj
   ```

3. Now create a file named `MathOperators.java` inside the src directory.

```
touch /home/project/my_operator_proj/src/MathOperators.java
```

4. Click the following button to open the file for editing.

Open **MathOperators.java** in IDE

5. Read each statement in the following program and understand how each of the operator is used. Paste the following content in `MathOperators.java`.

```
public class MathOperators {
    public static void main(String[] args) {
        int num1 = 20;
        int num2 = 10;
        System.out.println("Mathematical operators with numbers " + num1 + " and " + num2 );
        // Addition
        System.out.println("Addition: + operator " + (num1 + num2) );
        // Subtraction
        System.out.println("Subtraction: - operator " + (num1 - num2) );
        // Multiplication
        System.out.println("Multiplication: * operator " + (num1 * num2) );
        // Division
        System.out.println("Division: / operator " + (num1 / num2) );
        // Modulus (remainder)
        System.out.println("Modulus: % operator " + (num1 % num2) );
        // Use double for more precise division
        double num3 = 20.0;
        double num4 = 7.0;
        System.out.println("Precise Division: " + (num3 / num4));
    }
}
```

6. Compile the java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/MathOperators.java
```

7. Set the `CLASSPATH` variable.

   ```
   export CLASSPATH=$CLASSPATH:/home/project/my_operator_proj/classes
   ```

8. Now, when you run the java program, it will run seamlessly as expected.

   ```
   java MathOperators
   ```

You will see the output as below:

```
Mathematical operators with numbers 20 and 10
Addition: + operator 30
Subtraction: - operator 10
Multiplication: * operator 200
Division: / operator 2
Modulus: % operator 0
Precise Division: 2.857142857142857
```

Relational operators

Relational operators return a boolean value and compare the relation between the operands. Relational Operators in Java are:

- == (Equal to)
- != (Not equal to)
- > (Greater than)
- < (Less than)
- >= (Greater than or equal to)
- <= (Less than or equal to)

1. Create a file named `RelationalOperators.java` inside the src directory.

   ```
   touch /home/project/my_operator_proj/src/RelationalOperators.java
   ```

2. Click the following button to open the file for editing.

Open **RelationalOperators.java** in IDE

3. Read each statement in the following program and understand how the each of the relational operator is used. Paste the following content in `RelationalOperators.java`.

   ```
   public class RelationalOperators {
       public static void main(String[] args) {
           int a = 10;
           int b = 20;
           // Equal to (==)
           System.out.println("a == b: " + (a == b)); // false
           // Not equal to (!=)
           System.out.println("a != b: " + (a != b)); // true
           // Greater than (>)
           System.out.println("a > b: " + (a > b)); // false
           // Less than (<)
           System.out.println("a < b: " + (a < b)); // true
           // Greater than or equal to (>=)
           System.out.println("a >= b: " + (a >= b)); // false
           // Less than or equal to (<=)
           System.out.println("a <= b: " + (a <= b)); // true
       }
   }
   ```

Please note that `=` is an assignment operator. `==` is an equality check operator.

3. Compile the java program, specifying the destination directory as the `classes` directory that you created.

   ```
   javac -d classes src/RelationalOperators.java
   ```

4. Now run the java program.

   ```
   java RelationalOperators
   ```

You will see the output as below:

```
a == b: false
a != b: true
a > b: false
a < b: true
a >= b: false
a <= b: true
```

As you can see, all the output are boolean.

Ternary operators

Ternary operator is a concise way to evaluate a condition and choose one of two values based on whether the condition is true or false.

1. Create a file named `TernaryOperator.java` inside the src directory.

   ```
   touch /home/project/my_operator_proj/src/TernaryOperator.java
   ```

2. Click the following button to open the file for editing.

Open **TernaryOperator.java** in IDE

3. Read each statement in the following program and understand how the each of the ternary operator is used. Paste the following content in `TernaryOperator.java`.

   ```
   public class TernaryOperator {
       public static void main(String[] args) {
           String hasArgs = args.length == 0 ? "No arguments passed" : args[0];
           System.out.println(hasArgs);
       }
   }
   ```

```
String hasArgs = args.length == 0 ? "No arguments passed" : args[0];
```

If the length of the command line arguments is 0 then assign the value "No arguments passed" to the string hasArgs. Else set the value of the first command line argument to hasArgs.

```
System.out.println(hasArgs);
```

Print hasArgs.

3. Compile the java program, specifying the destination directory as the `classes` directory that you created.

   ```
   javac -d classes src/TernaryOperator.java
   ```

4. Now run the java program.

   ```
   java TernaryOperator
   ```

You will see the output as below:

```
No arguments passed
```

5. Now run the java program with an argument `something`.

   ```
   java TernaryOperator something
   ```

You will see the output as below:

```
something
```

Practice Exercise

1. Create a file in the `src` folder.

2. Create and initialize an int array with 10 elements.

3. Apply the mathematical operator on each number pairs such as:

- addition elements at index 0,1
- subtraction elements at index 2,3
- multiplication elements at index 4,5
- division elements at index 6,7
- modulus elements at index 8,9

4. Print output of each of the operation.

5. Use a for loop to check if the next number in the array is greater than, less than or equal to the current number. User ternary and relational operator.

▶ Click here for solution

You will now be able to appreciate the use of `for` loops in array manipulation.

Conclusion

In this lab, you learned how to use various operators.

Author(s)

Lavanya