Programming with Python and Java
(CS 29008)

School of Electronics Engineering, KIIT DU

Chapter 2

Lab 2: Operators, Selectors and Iterators

2.1 Operators in Java

An *operator* is a symbol which instructs computer to perform the specified manipulation over some data. Depending on the type of operations, operators are classified into the following types:

- Assignment operator $(=)$
- Arithmetic operators $(+, -, *, /, \%,)$
- Relational operators $(<, <=, >, >=, ==, !=)$
- Increment and decrement operators $(++, --)$
- Logical operators $(\&\&, ||, !)$
- Conditional operator $(? :)$
- Bitwise operators $(\&, |, \wedge, <<, >>, >>>)$

2.2 Relational operators

The *relational operators* are used to construct relational expressions which are used to compare two quantities. Figure 2.1 shows a Java program to illustrate the relational operators.

Figure 2.2 shows the console input-output of the program **Relational.java**. In this program, all the assignment statements (i.e. line numbers 10, 11, 12,

```
class Relational {
        public static void main(String args[]) {
                int a = 50, b = 60, x = 30, y = 30;
                boolean L, G, LE, GE, E, NE;
                System.out.println("a = " + a);
                System.out.println("b = " + b);
                System.out.println("x = " + x);
                System.out.println("y = " + y);
                System.out.println();
                L = a < b;
                G = a > b;
                LE = a <= b;
                GE = x >= y;
                E = x == y;
                NE = a != b;
                System.out.println("a < b is " + L);
                System.out.println("a > b is " + G);
                System.out.println("a <= b is " + LE);
                System.out.println("x >= y is " + GE);
                System.out.println("x == y is " + E);
                System.out.println("a != b is " + NE);
        }
}
```

Figure 2.1: Java program to illustrate relational operators

13, 14, and 15) are of the form **variable_name = operand1 relational-operator operand2**. In each of the statements, the expression on the right-hand side of the assignment gets evaluated and the result of the expression, which is either **true** or **false** is assigned to the variable on the left-hand side of the assignment operator.

```
D:\KIIT\Programming_With_Python_Java\My_Codes\Java_Program>javac Relational.java

D:\KIIT\Programming_With_Python_Java\My_Codes\Java_Program>java Relational
a = 50
b = 60
x = 30
y = 30

a < b is true
a > b is false
a <= b is true
x >= y is true
x == y is true
a != b is true
```

Figure 2.2: Input-output of program Relational

2.3 Logical operators

The *logical operators* are used to construct compound conditional expressions. The operators logical AND (&&) and logical OR (——) are used to combine two conditional expressions. The operator logical NOT (!) is used to negate a conditional expression. Figure 2.3 shows a Java program to illustrate the logical operators.

```
class Logical {
        public static void main(String args[]) {
                int a = 50, b = 60, c = 70;
                boolean x, y, z;
                System.out.println();
                System.out.println("a = " + a + " b = " + b + " c = " + c);
                System.out.println();
                System.out.println("Working of relational operators...");
                System.out.println();
                x = (a < b) || (a > c);
                System.out.println("(a < b) || (a > c) is " + x);
                y = (a < b) && (a > c);
                System.out.println("(a < b) && (a > c) is " + y);
                z = !(b > c);
                System.out.println("!(b > c) is " + z);
        }
}
```

Figure 2.3: Java program to illustrate logical operators

Figure 2.4 shows the console input-output of the program **Logical.java**.

```
D:\KIIT\Programming_With_Python_Java\My_Codes\Java_Program>javac Logical.java

D:\KIIT\Programming_With_Python_Java\My_Codes\Java_Program>java Logical

a = 50 b = 60 c = 70

Working of relational operators...

(a < b) || (a > c) is true
(a < b) && (a > c) is false
!(b > c) is true
```

Figure 2.4: Input-output of program Logical

2.4 Conditional operator

The *conditional operator* helps in decision-making. The general syntax of a conditional operator is

(condition) ? statement1 : statement2;

Here, **condition** is evaluated. If it is *true*, then **statement1** is executed, otherwise, **statement2** is executed. Since three expressions or operands are involved in this operator, the operator is also called *ternary operator*. Figure 2.5 shows a Java program to illustrate the conditional operator.

```
class Conditional {
        public static void main(String args[]) {
                int a = 120, b = 60;
                int largest;
                System.out.println();
                System.out.println("a = " + a + " b = " + b);
                System.out.println();
                System.out.println("Largest number using conditional operator...");
                System.out.println();
                largest = (a > b) ? a : b;
                System.out.println("Largest = " + largest);
        }
}
```

Figure 2.5: Java program to illustrate conditional operator

Figure 2.6 shows the console input-output of the program **Conditional.java**.

```
D:\KIIT\Programming_With_Python_Java\My_Codes\Java_Program>javac Conditional.java

D:\KIIT\Programming_With_Python_Java\My_Codes\Java_Program>java Conditional

a = 120 b = 60

Largest number using conditional operator...

Largest = 120
```

Figure 2.6: Input-output of program Conditional

The above program finds out the largest of two integers stored in the variables, a and b. The expression **largest = (a > b) ? a : b;** is executed in the following manner:

- If the expression **a > b** evaluates to *true*, **a** is assigned to the variable **largest**, i.e. **largest = a**.

- If the expression **a** > **b** evaluates to *false*, **b** is assigned to the variable **largest**, i.e. **largest** = **b**.

2.5 Bitwise operators

The *bitwise operators* allow us to perform operations on data at bit-level. Table 2.1 lists the symbols and names of the bitwise operators.

Table 2.1: Symbols and names of the bitwise operators

Operator symbol	Operator name
&	bitwise AND
|	bitwise OR
∧	bitwise exclusive OR
−	bitwise complement
<<	bitwise left shift
>>	bitwise right shift
>>>	bitwise right shift with zero fill

2.6 Conditional Execution Statements

Java provides built-in decision-making structures to implement conditional execution in the form of

- **if** statement
- **if-else** statement
- **if-elseif-else** statement
- **switch** statement

The syntax of all the decision-making statements are shown as follows.

Listing 2.1: *if* syntax

```
if(condition) {
  // statements;
}
```

Listing 2.2: *if-else* syntax

```
if(condition) {
  // statements;
}
else {
  // statements;
}
```

Listing 2.3: *if-elseif-else* syntax

```
if(condition1) {
  // statements;
}
else if(condition2) {
  // statements;
}
else {
  // statements;
}
```

Listing 2.4: *switch* syntax

```
switch(condition) {
    case v1:
      // statements;
      break;
    case v2:
      // statements;
      break;
    case v3:
      // statements;
      break;
      .
      .
     case vn:
      // statements;
      break;
     default:
      // statements;
      break;
}
```

Important points related to switch statements:

- The **case** labels should not be float values or Boolean expressions.
- The **case** labels **v1, v2**,$\cdots$,**vn** should be distinct.
- After each **case**, there should be a **break** statement.
- **default** case is always the last case statement.

2.7 Iterative statements

The repeated execution of a block of statements as long as some condition is *true* is called **looping** or **iteration**. Java provides three looping structures. They are

- **while** loop
- **do-while** loop
- **for** loop

The syntax of the looping structures are given below.

Listing 2.5: *while* syntax

```
while(test-expression) {
  // statements;
}
```

Listing 2.6: *do-while* syntax

```
do {
  // statements;
} while(test-expression);
```

Listing 2.7: *for* syntax

```
for(initialization; test-expression; updation) {
  // statements;
}
```

2.8 Lab 2 Exercises

Objectives:

- To learn writing, executing and debugging programs related to Java operators.
- To learn writing, executing and debugging programs related to Java decision control and loop control statements.

Outcomes:

- After completing this, the students would be able to develop Java programs using operators, if-else, for loop, etc.

Lab Assignments

1. Write a Java program to print a table of values of the function $y = e^{-x}$ for x varying from 0 to 1 in steps of 0.1. The table appears as follows.

x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y											

2. Write a Java program to find the largest of three numbers using a conditional operator.

3. Write a Java program to accept a point (x, y) and find whether it lies on the circle or inside the circle or outside the circle. The center of the circle is $(0, 0)$ and the radius of the circle is 5. Equation of a circle with $(0, 0)$ as the center and r as the radius is given by $x^2 + y^2 = r^2$.

1. If $x^2 + y^2 < r^2$, then the point (x, y) lies within the circle.
2. If $x^2 + y^2 > r^2$, then the point (x, y) lies outside the circle.
3. If $x^2 + y^2 = r^2$, then the point (x, y) lies on the circle.

4. Write a Java program to find whether a number is an Armstrong number or not. (Hint: A number is an Armstrong number if the sum of the cubes of the digits of the number is equal to the number itself. For example, $153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27$).

5. Write a Java program to generate a Fibonacci series.