Unit 2

Boolean Expressions

Relational Operators

Relational operators are used to compare values and determine their relationship.

Relational Operators

Operator	Meaning	Can be used with
<	less than	primitives only
<=	less than or equal	primitives only
>	greater than	primitives only
>=	greater than or equal	primitives only
==	equality (equal)	primitives and references
!=	equality (not equal)	primitives and references

Logical Operators

logical operators are used to combine or manipulate boolean expressions

Logical Operators

Operator	Meaning	Can be used with
`&&`	AND	`boolean` variables or `boolean` expressions
`||`	OR	`boolean` variables or `boolean` expressions
!	NOT	`boolean` variables or `boolean` expressions

Operator Precedence

Refer to the following chart for operator precedence listed from highest to lowest.

Operator	Type of Operator	Associativity
`(), .`	parentheses, object access	left to right
`++, --`	post-increment, post-decrement	left to right
`+, -, !`	unary plus, unary minus, logical NOT (positive) (negative)	right to left
`(), new`	cast, object creation	left to right
`* / %`	multiplication and division	left to right
`+ -`	addition and subtraction	left to right
`<, <=, >, >=`	relational	left to right
`== , !=`	equality	left to right
`&&`	logical AND	left to right
`||`	logical OR	left to right
`=, +=, -=, *=, /=, %=`	assignment, compound assignment	left to right

Operator's Precedence in Java

Operators	Precedence
!, +, - (unary Operators)	First (Highest)
* , / , %	Second
+ , -	Third
< , <= , >=, >	Fourth
== , !=	Fifth
&&	Sixth
||	Seventh
= (assignment Operator)	Lowest

use parentheses for clarity.

Examples

Most left-to-right associate

`1 + 2 * 3` is treated as `1 + (2 * 3)`

`1 * 2 + 3` is treated as `(1 * 2) + 3`

`72 / 2 / 3` is treated as `(72 / 2) / 3`

```
System.out.println("1 + 2 = " + 1 + 2);
System.out.println("1 + 2 = " + (1 + 2));
```

If either (or both) of the operands of the + operator is a string, the other is automatically cast to a string. String concatenation and addition have the *same precedence* and are left-to-right associative. The parentheses in the second statement ensures that the second + operator performs addition instead of string concatenation.

Some right-to-left associative

`x = y = z = 17` is treated as `x = (y = (z = 17))` Assign

What is the value of the expression `+-17` Unary plus, unary minus, logical NOT

Example Relational operator

What is the value of the expression `1 <= 2 <= 3`

`1 <= 2 <= 3` as `(1 <= 2) <= 3`

This leads to a compile-time error because you can't use the `<=` operator to compare a `boolean` to an `int`.

Example Method and Construction

Object construction and method invocation have the same precedence and are left-to-right associative.

```
int n = new String("ABCDEFGHIJ").substring(0, 5).length();
System.out.println(n);
```

```
5
```

- The bitwise NOT operator (~) inverts all the bits of a number. In a typical two's complement system, this is equivalent to the formula -(n+1).
- ~17 = -(17+1) = -18

Practice

```
System.out.println(1 + 2 + "abc");
System.out.println("abc" + 1 + 2);
```

`3abc` and `abc12`

```
year % 4 == 0 && year % 100 != 0 || year % 400 == 0
```

```
((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)
```

Short-circuit evaluation

- Java evaluates every operand of an operator before the operation is performed.
- For the logical AND (&&) and logical OR (||) operators, Java evaluate the second operand only if it is necessary to resolve the result.

```
if ((s != null) && (s.length() < 10))
```

invoke the `length()` method only if `s` is not `null`