Operators

Exercises and solutions

1. What is an operator? What are unary, binary, and ternary operator? Give an example of each type of operator in Java.

An operator is a symbol that performs a specific kind of operation on its operands and produces a result.

Unary operator takes one operand. The following expression uses a unary operator called decrement operator (--), which will decrement the value of an integer variable named num by 1:

--num

Binary operator takes two operands. The following expression uses a binary operator named multiplication operator (\*), which multiplies 10 and 5:

10 \* 5

Ternary operator takes three operands. The following expression uses a ternary operator (?:), which will evaluate to noWork if isHoliday is true and to work if isHoliday is false:

isHoliday ? noWork : work

1. What is the difference between prefix, postfix, and infix operators? Give an example of such operators in Java.

A prefix operator appears before its operand, for example:

++index // ++ is an operator and index is an operand

A postfix operator appears after its operand, for example,

size-- // -- is an operator, size is an operand

An infix operator appears in between the two operands, for example:

a ^ b // ^ is an operator, a and b are operands

1. What is an arithmetic operator, what types of operands do they take, and what type of results do they produce?

An arithmetic operator takes numeric values as its operands and performs an arithmetic operation to compute another numeric value.

Both operands must be one of six types - byte, short, char, int, long, float, double

1. Name two operators in Java, which take only boolean operands and produces a boolean value.

* && (Logical AND operator)
* || (Logical OR operator)

1. What is the difference between the two operators: = and ==?

= is an assignment operator. It is used to assign a value to a variable.

== is an equality operator. It is used to test two operands for equality.

1. Consider the following snippet of code:  
     
   boolean done;  
   /\* Some code goes here \*/  
   your-code-goes-here;  
     
   Using a boolean logical operator, invert the current value stored in the done variable. That is, write a statement that will assign true to the done variable if its current value is false and false if its current value is true.

done = !done; // usea a logical NOT (!)

1. Consider the following snippet of code:  
     
   int x = 23;  
   int y = ++x % 3;  
     
   What will be the value of y after this snippet of code is executed?

Zero. x becomes 24. 24 % 3 is 0.

1. Consider the following snippet of code:  
     
   int x = 23;  
   x = x++ % x;  
     
   What will be the value of x after this snippet of code is executed? Explain your answer with steps performed explaining how the value of x changes during the execution of the second statement.

The answer is 23.

The modulus operator (%) takes two operands. First operand is 23. Then x is incremented by 1. Thus second operand becomes 24. 23 % 24 is 23.

1. Explain the reason why the following snippet of code does not compile.   
     
   int x = 10;  
   boolean yes = (x = 20);

**Answer:**

In Java, = is an assignment operator. (x = 20) assigns 20 to x and evaluates to 20. This results into a statement like

boolean yes = 20;

Java does not allow assigning a non-boolean value to a boolean variable. Therefore, assigning 20, which is an int, to a boolean variable yes causes a compile-time error.

1. What will be the value assigned to the variable named yes when the following snippet of code is executed:   
     
   int x = 10;  
   boolean yes = (x == 20);

**Answer:**

false

1. What will be the value of y when the following snippet of code is executed?  
     
   int x = 19;  
   int y = x > 10 ? 69 : 68;

**Answer:**

69

1. You have a short variable named x, which is declared and initialized as follows:  
     
   short x = -19;  
     
   You want to assign 19 to x using the following statements both of which do not compile:  
     
   x = -x;  
   x = -1 \* x;  
     
   How will you rewrite the above two statements to make them compile? What is wrong with the following statement that attempts to fix the compile-time error in the above statements, but fails to assign 19 to x?  
     
   x -= x;

**Answer:**

In "x = -x", the right hand side operand x is converted to int to apply negation operator. To assign result 19 to left hand side operand x, it has to be explicitly cast to short. The following statement fixes this problem:

x = (short) -x;

Similarly, the following statement fixes the compile-time error in the second statement:

x = (short)(-1 \* x);

1. What will be the output when the following snippet of code is executed?  
     
   boolean b = true;  
   String str = !b +" is not " + b;  
   System.out.println(str);

**Answer:**

false is not true

1. What will be the output when the following snippet of code is executed:  
     
   boolean b = true;  
   String str = (b ^= b) + " is " + b;  
   System.out.println(str);

**Answer:**

false is false

1. What will be the output when you execute the following snippet of code?  
     
   int x = 10;  
   int y = x++;  
   int z = ++x;  
   System.out.println("x = " + x + ", y = " + y + ", z = " + z);

**Answer:**

x = 12, y = 10, z = 12

1. Complete the second statement using the ternary operator (?:) and the bitwise AND operator (&) that will make a message "x is odd". Your code must contain of the following tokens in any order: x, &,==, ?, :, "odd", "even". You may use additional tokens as needed.  
     
   int x = 19;  
   String msg = your-code-goes-here ;  
   System.out.println("x is " + msg);

**Solution:**

String msg = (x & 1) == 1 ? "odd" : "even" ;

1. Which of the following assignments will fail to compile and why?  
     
   int i1 = 100;  
   int i2 = 10.6;  
   byte b1 = 90;  
   byte b2 = 3L;  
   short s1 = -90;  
   float f1 = 12.67;  
   float f2 = 0.00f;  
   double d1 = 12.56;  
   double d2 = 12.78d;  
   boolean bn1 = true;  
   boolean bn2 = 0;  
   char c1 = 'A';  
   char c2 = "A";  
   char c3 = 0;  
   char c4 = '\u0000';

Answer:

int i1 = 100; // ok  
int i2 = 10.6; // fails, cannot assign double to int  
byte b1 = 90; // ok  
byte b2 = 3L; // fails, cannot assign long to byte  
short s1 = -90; // ok  
float f1 = 12.67; // fails, cannot assign double to float  
float f2 = 0.00f; // ok  
double d1 = 12.56; // ok  
double d2 = 12.78d; // ok  
boolean bn1 = true; // ok  
boolean bn2 = 0; // fails, cannot assign int to boolean  
char c1 = 'A'; // ok  
char c2 = "A"; // fails, cannot assign string to char  
char c3 = 0; // ok  
char c4 = '\u0000'; // ok

1. Write down the value assigned to the declared variable in each of the following statement. If a statement generates a compile-time error, explain the reason behind the error and, if possible, provide a solution to fix the error.   
     
   int i1 = 10/4;  
   int i2 = 10.0/4.0;  
   int i3 = 0/0;  
   long l1 = 10/4;  
   long l2 = 10.0/4.0;  
   float f1 = 10/4;  
   float f2 = 10.0/4.0;  
   double d1 = 10/4;  
   double d2 = 10.0/4.0;  
   double d3 = 0/0;  
   double d4 = 0/0.0;  
   double d5 = 2.9/0.0;

**Solution:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statement | Compiles | Reason | Fix | Result |
| int i1 = 10/4; | ok |  |  | 2 |
| int i2 = 10.0/4.0; | fails | cannot assign double to int | int i2 = (int)(10.0/4.0); | 2 |
| int i3 = 0/0 | ok |  |  | ArithmeticException |
| long l1 = 10/4; | ok |  |  | 2L |
| long l2 = 10.0/4.0; | fails | cannot assign double to long | long l2 = (long)(10.0/4.0); | 2L |
| float f1 = 10/4; | ok |  |  | 2.0f |
| float f2 = 10.0/4.0; | fails | cannot assign double to float | float f2 = (float)(10.0/4.0); | 2.5f |
| double d1 = 10/4; | ok |  |  | 2.0 |
| double d2 = 10.0/4.0; | ok |  |  | 2.5 |
| double d3 = 0/0; | ok |  |  | ArithmeticException |
| double d4 = 0/0.0; | ok |  |  | NaN |
| double d5 = 2.9/0.0; | ok |  |  | Infinity |

1. Complete the following snippet of code that will assign a 2's complement of x to y. You must use the bitwise complement operator.  
      
   int x = 19;  
   int y = your-code-goes-here;

**Solution:**

int y = ~x + 1;

1. What will be the output of the following snippet of code:  
     
   int x = 19;  
   int y = (~x + 1) + x;  
   System.out.println(y);

**Solution:**

It will print 0.