Statements

Exercises and solutions

1. What is a statement?

**Answer:**

A statement specifies an action in a Java program, such as assigning the sum of x and y to z, printing a message to the standard output, etc. Statements are written using keywords, operators, and expressions.

1. What is an expression? How do you convert an expression into an expression statement in Java? Can you convert all types of expressions in Java into expression statements?

**Answer:**

An expression consists of literals, variables, operators, and method invocation and they are the building blocks of a Java program. An expression statement is an expression with a semicolon at the end. Not all types of expressions cannot be converted to expression statements.

1. What are control statements and why do you use them?

**Answer:**

A control flow statement controls the order in which other statements are executed. They can be used to execute a statement or set of statements only if a condition is met. Also, they can be used to execute a statement or set of statements repeatedly as long as a condition is met.

1. What is a block statement and how to you create a block statement?

**Answer:**

A block statement is a compound statement (or a sequence of zero or more statements) that is treated as one statement. It is created by enclosing statements in braces.

1. What is an empty statement?

**Answer:**

A semicolon that is treated as a statement is called an empty statement.

1. What is the difference between while-loop and do-while statements?

**Answer:**

The statement associated with a while-loop statement may not be executed even once if the condition-expression evaluates to false for the first time. However, the statement associated with a do-while statement is executed at least once.

1. A switch statement contains a switch-expression. List all the types that a switch-expression must evaluate to.

**Answer:**

The switch-expression must evaluate one of the following types: byte, short, char, int, enum, or String

1. When can you use a switch statement in place of an if-else statement?

**Answer:**

A switch statement is a cleaner way of writing an if-else statement when the condition-expression in an if-else statement compares the value of the same variable or expression for equality.

1. Consider the following snippet of code. The valid value of the count variable must be in the range 11 (inclusive) and 20 (inclusive). Write the condition for the if-else statement, so a correct message is printed.  
     
   int count = 20;  
   if(<your-code-goes-here>)   
    System.out.println("Count is valid.");  
   else   
    System.out.println("Count is invalid");

**Solution:**

int count = 20;  
if (count >= 11 && count <= 20)   
 System.out.println("Count is valid.");  
else   
 System.out.println("Count is invalid");

1. Fix the compile-time errors in the following snippet of code. Make sure the fixed code prints the value of y.  
     
   int x = 10;  
   int y = 20;  
   if (x = 10)  
    y++;  
    System.out.println("y = " + y);  
   else   
    y--;  
    System.out.println("y = " + y);

**Solution:**

There are two errors. Change assignment operator to equality operator in the if staement. Remove the first System.out.println() statement to ensure proper if-else statement.

int x = 10;  
int y = 20;  
if (x == 10)  
 y++;

else   
 y--;

System.out.println("y = " + y);

Alternatively, you can use a block statement with the if and else parts as follows:

int x = 10;  
int y = 20;  
if (x == 10) {  
 y++;  
 System.out.println("y = " + y);

} else {  
 y--;  
 System.out.println("y = " + y);

}

1. Rewrite the following snippet of code using an if-else statement. Make sure that both switch and if-else statements will have the same output when you initialize the variable x to another value. (Hint: This is a tricky question because there are no break statements in any case labels.)  
     
   int x = 50;  
   switch (x) {  
    case 10:  
    System.out.println("Ten");  
    default:  
    System.out.println("No-match");   
    case 20:  
    System.out.println("Twenty");  
   }

**Solution:**

The following if-else statement is a direct translation of this switch statement:

int x = 50;

if (x == 10) {

System.out.println("Ten");

System.out.println("No-match");

System.out.println("Twenty");

} else if (x == 20) {

System.out.println("Twenty");

} else {

System.out.println("No-match");

System.out.println("Twenty");

}

1. The following snippet of code is a modified version of the previous one. Rewrite it using an if-else statement. Make sure that both switch and if-else statements will have the same output when you initialize the variable x to another value.  
      
   int x = 50;  
   switch (x) {  
    case 10:  
    System.out.println("Ten");  
    break;  
    default:  
    System.out.println("No-match");   
    break;  
    case 20:  
    System.out.println("Twenty");  
    break;  
   }

**Solution:**

int x = 20;

if (x == 10) {

System.out.println("Ten");

} else if (x == 20) {

System.out.println("Twenty");

} else {

System.out.println("No-match");

}

1. A programmer was learning about the switch statement and he tried to use it everywhere he could. The following snippet of code is an example of such a forced use where it is not needed. Rewrite the following snippet of code using no control flow statements. That is, you need to get rid of the switch statement leaving the program logic intact.  
     
   int x = 10;  
     
   // Some logic goes here...  
     
   switch(x) {  
    default:   
    x++;  
   }

**Solution:**

int x = 10;

// Some logic goes here...

x++;

1. How do you write an infinite loop using a for, while, and do-while statements? Give an example of each.

**Solution:**

// Using a for loop

for( ; ; ) {

// Code goes here

}

// Using a while loop

while (true) {

// Code goes here

}

// Using a do-while loop

do {

// Code goes here

} while (true);

1. The intent of the following for statement is to print integers from 1 to 10 in reverse order. The code does not print the numbers as intended. Point the logical error and fix the code, so it prints 10, 9, 8,…1.  
     
   for(byte b = 10; b >= 1; b++)  
    System.out.println(b);

**Solution:**

Once intial value of b, which is 10, is printed, before the second iteration the value of b is incremented. Thus all values 10, 11, 12, etc are printed. To fix the error, variable b should be decremented (b--) after each iteration, not increment (b++).

for (byte b = 10; b >= 1; **b--**)  
 System.out.println(b);

1. Write a for statement that prints all odd numbers from 13 to 1 in reverse order. The body of the for statement must be an empty statement. That is, you can use only the initialization, condition-expression, and expression-list of the for statement to write all your logic. The template of your for statement is as follows:  
     
   for(<your-code>; <your-code>; <your-code>);

**Solution:**

for (short s = 13; s >= 1; System.out.println(s), s -= 2);

1. Write a snippet of code using a for statement that calculates the sum of all integers from 1 to 10 and prints it on the standard output. The template for your code is as follows:  
     
   int sum = 0;  
   for(<your-code>; <your-code>; <your-code>);  
   System.out.println("Sum = " + sum);

**Solution:**

int sum = 0;

for (int i = 1; i <= 10; sum += i, i++);

System.out.println("Sum = " + sum);

1. Use a nested for statement to print the following pyramid.  
     
    \*  
    \*\*\*  
    \*\*\*\*\*  
    \*\*\*\*\*\*\*

**Solution:**

String s = "\*";

for (int i = 1; i <= 4; i++) {

for (int j = i; j < 4; j++) {

System.out.print (" ");

}

System.out.println(s);

s += "\*\*";

}

1. Write a nested for statement that will print the following  
     
    1  
    22  
    333  
    4444  
    55555  
   666666

**Solution:**

// for right-aligned output

for (int i = 1; i <= 6; i++) {

// comment below for loop for left-aligned output

for (int j = 1; j <= 6 - i; j++) {

System.out.print (" ");

}

for (int j = 0; j < i; j++) {

System.out.print (i);

}

System.out.println();

}

1. Complete the following snippet of code. It is supposed to print a comma-separated list of all integers from lower to upper. For example, if lower is 1 and upper is 4, it should print: 1, 2, 3, 4. (Hint: Use System.out.print() to print a message without a new line.)  
     
   int lower = 1;  
   int upper = 4;  
     
   for(<your-code-goes-here>) {  
    <your-code-goes-here>  
   }

**Solution:**

for (int i = lower; i <= upper; i++) {

System.out.print(i);

if (i != upper)

System.out.print(", ");

}