ORACLE
Menu

Search Java Magazine

Topics ⌄   Archives   Downloads ⌄   Subscribe

Java
magazine

JAVA SE

Quiz yourself: Evaluation order, the assignment operator, and array access in Java

Prepare for a tricky adventure when evaluating expressions that contain arrays.

by Mikalai Zaikin and Simon Roberts

May 3, 2021

This quiz looks at evaluation order, the assignment operator, and array access. Given the following code

```
int[] a = {3,2,1,0,4};
int[] b = {2,4,2,2,1};
System.out.print(a[(a=b)[2]]); // line n1
System.out.print(a[(a=b)[2]]); // line n2
```

What is the result of running that code? Choose one.

A. Compilation fails at line n1.

The answer is A.

B. The `java.lang.ArrayIndexOutOfBoundsException` is thrown at line n2.

The answer is B.

C. `11`

The answer is C.

D. `12`

The answer is D.

E. `22`

The answer is E.

Answer. To decipher what's going on, it might be helpful to decompose the expression into two parts.

```
1.  a[<expr>]
2.  (a=b)[2]
```

The first part of the expression represents the outer part of the overall expression in the `print` statements. It accesses an element of an array identified by the reference variable `a`, at the subscript identified by the value of the expression `<expr>`. For both line n1 and line n2, you need to answer two questions: Which array does a refer to? And what's the value of `<expr>`?

The second part of the expression determines the value of `<expr>`.

In deciding which array is referred to, a rule in *Java Language Specification* section 15.10.4, "Run-time evaluation of array access expressions," applies. This rule states that the array reference expression is evaluated first, prior to evaluating the index

expression. Therefore, at line n1—even though you will see that the value of `a` is changed when the index value is calculated—the outer expression will cause access to the first array, the one with the values `{3,2,1,0,4}`.

In evaluating the second part, you need to determine the value of the expression `(a=b)`. The operation simply makes an assignment, so the variable `a` refers to the second array (but to reiterate, this does *not* alter the array that will be accessed by the outer expression). In doing so, the expression *as a whole* takes on the value that is assigned.

In other words, `(a=b)` is an expression that refers to the *second* array; this is true for both line n1 (in which the value of `a` will be changed) and line n2 (in which the value of `a` is not changed, because it already has the same value as `b` by this point).

Now that the assignment expression `(a=b)` has been evaluated, the subscript operation selects the third element (that is, the element at index `2`) from that second array (the one to which both variables `a` and `b` now refer). The item at index `2` has the value `2` in both line n1 and line n2.

Are you confused yet? Let's keep going.

Now you can evaluate the first of the two elements, the outer expression `a[<expr>]`. You already know that *in line n1* this expression will access an element from the *first* array; therefore, this will retrieve the value `1` from the array, and that's what the code will subsequently print first.

When line n2 executes, both `a` and `b` already refer to the same array—that is, the second array. At this point, there's really no uncertainty: The subexpression `(a=b)[2]` evaluates to `2` (the third item in the second array, as before), and `a[2]` also produces the value `2` because the variable `a` now unambiguously refers to the second array. That value `2` is then printed.

Consequently, the output is `12`, which means option D is correct and options A, B, C, and E are incorrect.

Conclusion. The correct answer is option D.

Mikalai Zaikin

Mikalai Zaikin is a lead Java developer at IBA IT Park in Minsk, Belarus. During his career, he has helped Oracle with development of Java certification exams, and he has been a technical reviewer of several Java certification books, including three editions of the famous *Sun Certified Programmer for Java* study guides by Kathy Sierra and Bert Bates.

Simon Roberts

Simon Roberts joined Sun Microsystems in time to teach Sun's first Java classes in the UK. He created the Sun Certified Java Programmer and Sun Certified Java Developer exams. He wrote several Java certification guides and is currently a freelance educator who publishes recorded and live video training through Pearson InformIT (available direct and through the O'Reilly Safari Books Online service). He remains involved with Oracle's Java certification projects.

Share this Page

Contact
US Sales: +1.800.633.0738
Global Contacts
Support Directory
Subscribe to Emails

About Us
Careers
Communities
Company Information
Social Responsibility Emails

Downloads and Trials
Java for Developers
Java Runtime Download
Software Downloads
Try Oracle Cloud

News and Events
Acquisitions
Blogs
Events
Newsroom