Quiz Yourself: String Manipulation (Intermediate)

The subtleties of string concatenation

by Simon Roberts and Mikalai Zaikin

November 21, 2019

If you have worked on our quiz questions in the past, you know none of them is easy. They model the difficult questions from certification examinations. The “intermediate” and “advanced” designations refer to the exams, rather than to the questions, although in almost all cases, “advanced” questions will be harder. We write questions for the certification exams, and we intend that the same rules apply: Take words at their face value, and trust that the questions are not intended to deceive you but to straightforwardly test your knowledge of the ins and outs of the language.

Given the following code:

```java
String s = "abc";
s += 1 + 2;
s += (2 + 2);
s.concat("1").concat("2");
s.concat(" + 2 + 2");
System.out.print(s);
```

What is the output? Choose one.

A. abc12
B. abc34124
C. abc124124
D. abc1241222

**Answer.** This question is fairly straightforward and a favorite topic for distinguishing those who have used Java to create a few examples from those who have merely looked over some code. The essence of this question is that `java.lang.String` objects are immutable (at least when you use “proper” coding techniques), but variables of the reference type, unless marked `final`, are mutable.

Let’s walk through the behavior of the code. First, the initializing assignment sets the value of the reference variable `s` so it refers to a string object that contains the character sequence "abc".

Next, a compound expression is evaluated. Because there are two `+` operators and no parentheses, this evaluates the subexpressions from left to right. The first subexpression takes the string referred to by `s` (which has the text "abc") and concatenates the result of converting the literal `int` value 1 to a `String`. Therefore, this subexpression refers to a new `String` containing "abc1". Next, the second conversion and concatenation produces an expression that refers to a `String` containing the text "abc12". This reference is then assigned to the variable `s`. Notice that the original `String` was never changed, but you built new strings and changed the variable `s` to refer to the newest one.
It is perhaps interesting to compare the preceding evaluation with this variation:

```java
String s = "abc";
s = s + 3 * 2;
```

In this case, the * operator has higher precedence than +, so the subexpression 3 * 2 is evaluated first, resulting in 6. That value is then concatenated with "abc", producing "abc6".

Anyway, back to the code in the question. The next statement is very similar, but uses parentheses to give higher priority for the numeric addition operation: (2 + 2). This results in a subexpression of int type and the value 4. When the second subexpression is evaluated, the result is a reference to another new String containing the text "abc124". Finally, that reference value is stored in the variable s.

The rest of the code is essentially a distraction; although several intermediate new strings are created, the variable s is never updated to refer to any of those strings. Indeed, no durable copy of these references is kept, and the objects are effectively lost immediately after the statement is completed. Therefore, they become eligible for garbage collection, although that’s not relevant to this question. You must always keep in mind that java.lang.String produces immutable objects and old operand strings always remain untouched. So, at the end of the code, in the print statement, the variable s still refers to the string that contains the text "abc124" and so the output is abc124. Therefore, you can conclude that option A is correct and options B, C, and D are incorrect.

**The correct answer is option A.**

**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.

**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.