Quiz Yourself: Understanding enums (Advanced)

The subtleties of using enums in a switch statement

by Simon Roberts and Mikalai Zaikin

January 10, 2020

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 enumeration:

```
enum Size { SMALL, MEDIUM, LARGE; }
```

and the following code fragment:

```
public static void main(String[] args) {
    final var size = Size.SMALL; // line n1
    switch (size) { // line n2
        case SMALL: { System.out.print(size);} // line n3
    }
}
```

What is the result? Choose one.

A. Compilation fails at line n1 due to invalid assignment syntax.
B. Compilation fails at line n2 because both $MEDIUM$ and $LARGE$ case expressions are missing.
C. Compilation fails at line n3 because you must use a qualified constant name (that is, $Size.SMALL$).
D. $SMALL$ is output.
E. $Size@3cb5cdba$ is output.

Answer. First, a clarification. The objectives for the exam include mention of "enumerations." This is perhaps rather ambiguous in that it might refer to types that are declared as enum or classes that implement the java.util.Enumeration<E> interface. This question is about types that are declared as enum, and such classes are definitely a topic that is addressed by questions in the exams.

Option A suggests there is a syntax problem in line n1. However, the syntax used here is correct; a local variable can be declared as final and in Java 10, the type of a local variable may generally be declared as final (there are some restrictions, but none that apply here). In such a case, the type of the variable is inferred from the type of the expression
Option A suggests that an ExpressionStatement could not have a value in only a limited number of situations, and in that form, it actually is a compilation error if the compiler cannot prove that all possible values of the control expression have a case (or default) that matches. However, the current version of the exam is written against Java 11, and this new feature has a different syntax from that of the standard switch statement, so this observation does not alter the fact that option B is incorrect.

Option C is also incorrect; it suggests that the case keyword should be followed by the fully qualified name Size.SMALL. In fact, just the opposite is true: When an enum constant is used in a case expression, it must always be used in the unqualified—that is, short—form.

Option D asks about the default toString behavior of an enum type. When you declare an enum type, you implicitly define a class that extends java.lang.Enum<E>. This base class implements a toString method that prints the simple name of the constant as it's declared in the source. In this case, the output is therefore SMALL, and option D is correct. As a side note, it's possible to define and override methods in an enum type and, therefore, the behavior of the class's toString can be overridden to provide some other output if desired.

Option E is incorrect: As explained in the preceding discussion about option D, the default toString behavior of an enum instance returns the constant's simple name. Therefore, this option, which suggests an output reminiscent of the behavior of the default java.lang.Object toString method, is incorrect.

The correct answer is option D.