Java Magazine

Written by the Java community for Java and JVM developers

Quiz

Quiz yourself: Defining the structure of a Java class

June 6, 2022 | 4 minute read


Mikalai Zaikin


Simon Roberts

Which of the following statements are correct about a Java class? Choose two.

A. A Java class must have a name shown in the source code. The answer is A.

B. A Java class may have several local variables with the same name inside the same method. The answer is B.

C. A Java class may have several `import` statements. The answer is C.

D. An underscore character "_" is a valid Java class name. The answer is D.

Answer. Not all classes have an explicit name shown in the source code. For example, Java provides anonymous classes, such as the following:

```
Runnable r = new Runnable(){ public void run(){
  System.out.print("Do nothing!");
  }
};
r.run();
```

The `run` method of the `Runnable` interface is abstract, and yet you can see that there is a real object because you instantiated it and can invoke the `run` method. This shows that some concrete class, which implements the `Runnable` interface, exists. The variable `r` is a reference to an instance of that class, but the class name is not known in the source code. Therefore, option A is incorrect.

Variables are visible only within the *scope* in which they were defined. However, since a block bounded by curly braces defines a scope, you can create two sibling scopes inside one method. If you do this, two variables with the same name can coexist without a problem, such as in the following:

```
void twoVars() {
  { int i = 0; }
  { int i = 1; } // OK
}
```

In view of this, option B is correct.

Option C discusses multiple `import` statements. Having multiple `import` statements is not merely permitted—in most cases, it's *necessary* to have many `import` statements to provide access to classes in different packages. It's also typical to have multiple `import` statements providing access to each of several classes in the same package, rather than using wildcards.

Even repeating `import` statements for the same class or package is syntactically valid, though it would probably trigger a request during code review to tidy up the code.

The following is completely legal:

```
import java.util.*;
import java.util.*;

public class MyClass { // OK
}
```

By the way, if a class defines more than one `package` statement—whether specifying the same package name or a different package name—compilation would fail. Thus, the following would not compile:

```
package a.b.c;
package a.b.c; // NOT OK

public class MyClass {
}
```

Because option C asks only if multiple `import` statements are permitted, option C is correct.

As for option D, through Java 8 the single underscore character was a valid identifier and could be used as a class name, a method name, or a variable name.

In Java 8, a warning during compilation indicated that this character was reserved for future language changes. However, the warning did not prevent its use. But then, beginning with Java 9, the single underscore character was defined as a keyword and therefore is no longer valid as an identifier.

This change was described in the Java 9 summary of changes, which stated that the underscore character was not a legal name and warned that if you use the underscore character (_) as an identifier, your source code cannot be compiled.

By the way, it is still legal to use a double underscore (__) as an identifier, such as for a class name, a method name, or a variable name. You can also *start* a variable name with a single underscore.

```
public class MyClass {
    int __; // Double underscore is OK
}
```

Because the single underscore is a keyword and is not a legal identifier on its own any longer, option D is incorrect.

Conclusion. The correct answers are options B and C.

Related quizzes

- Quiz yourself: Java records, constructors, and the canonical constructor
- Quiz yourself: The syntax of generated record classes in Java
- Quiz yourself: Nested types and Java records

Mikalai Zaikin
Mikalai Zaikin is a lead Java developer at IBA IT Park in Vilnius, Lithuania. 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.

‹ Previous Post