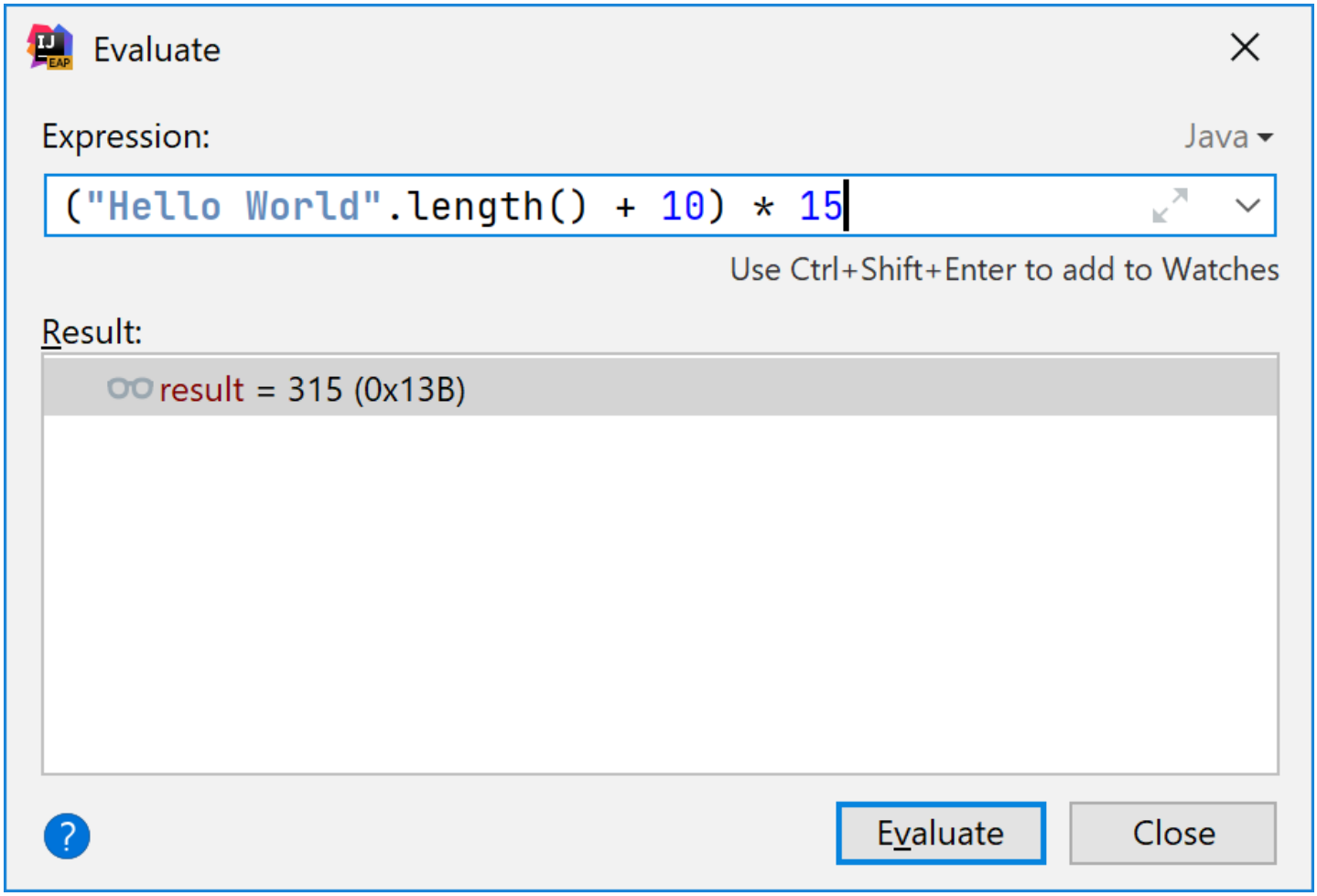
Twitter Thread by Tagir Valeev



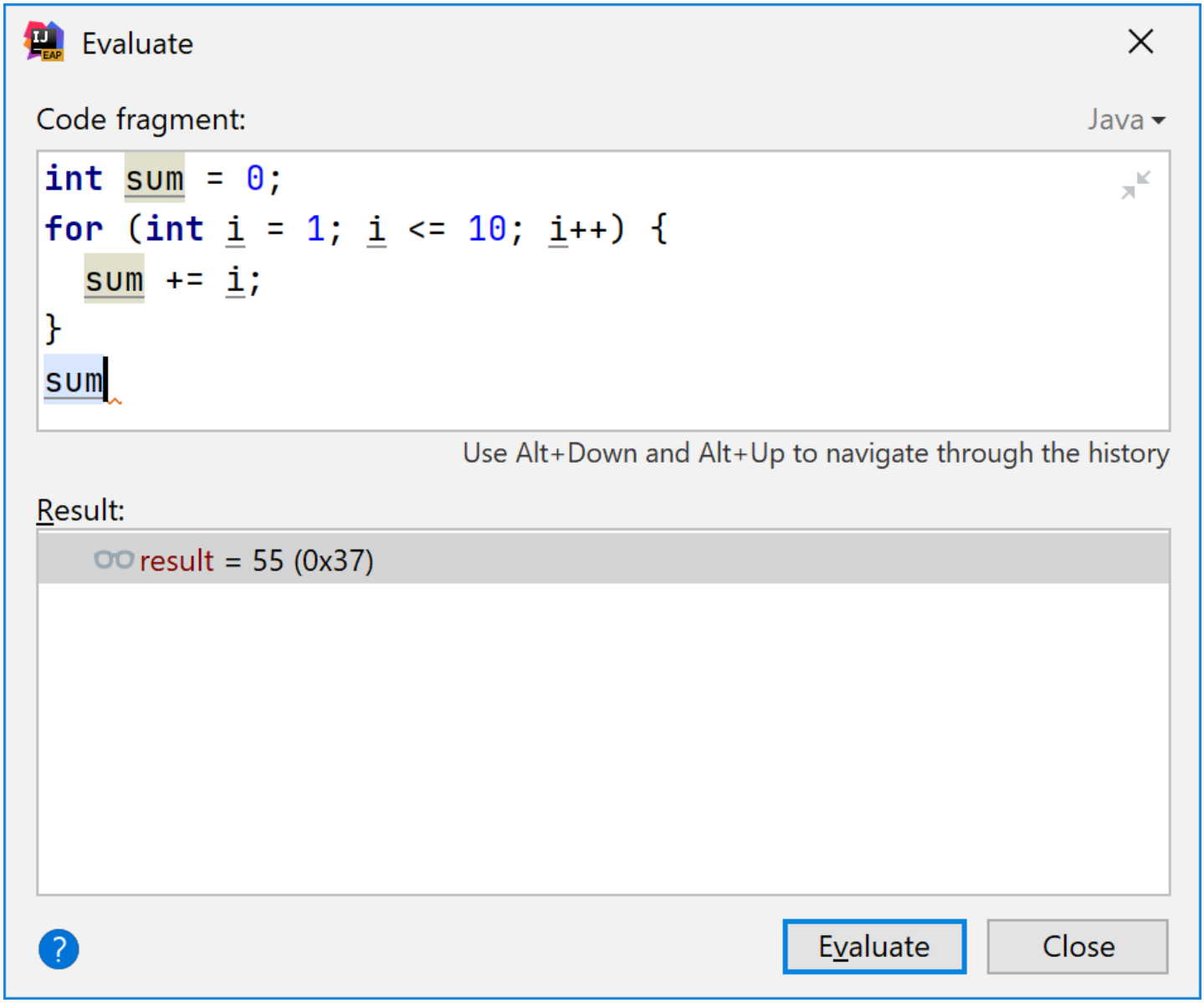
Tagir Valeev
@tagir_valeev

Ever wondered how the JVM debugger 'Evaluate' feature works in IDEs? Long ago I thought that the Java debug interface (JDI) actually allows you to specify the expression and get back its value.



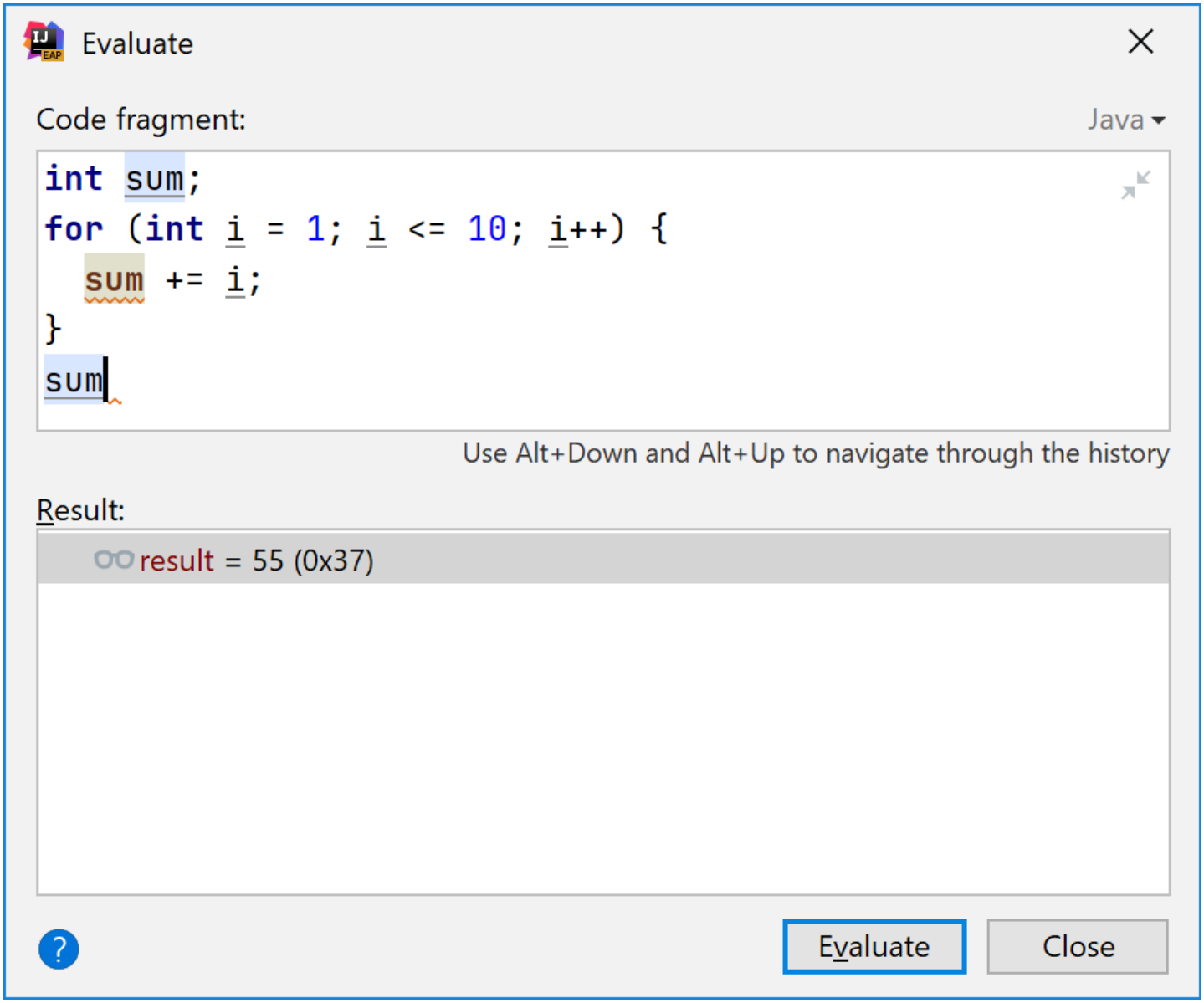
However, this is not quite possible: there's no Java language compiler inside the JVM. When the process is paused JDI allows you to read or write any variables and fields or even execute an existing method with given arguments. But nothing like addition or multiplication.

On the other hand, #IntelliJIDEA debugger allows you to evaluate even statements! You can write there `for`, `if`, `while`, `switch` - whatever. You can even declare local variables. So how this works?

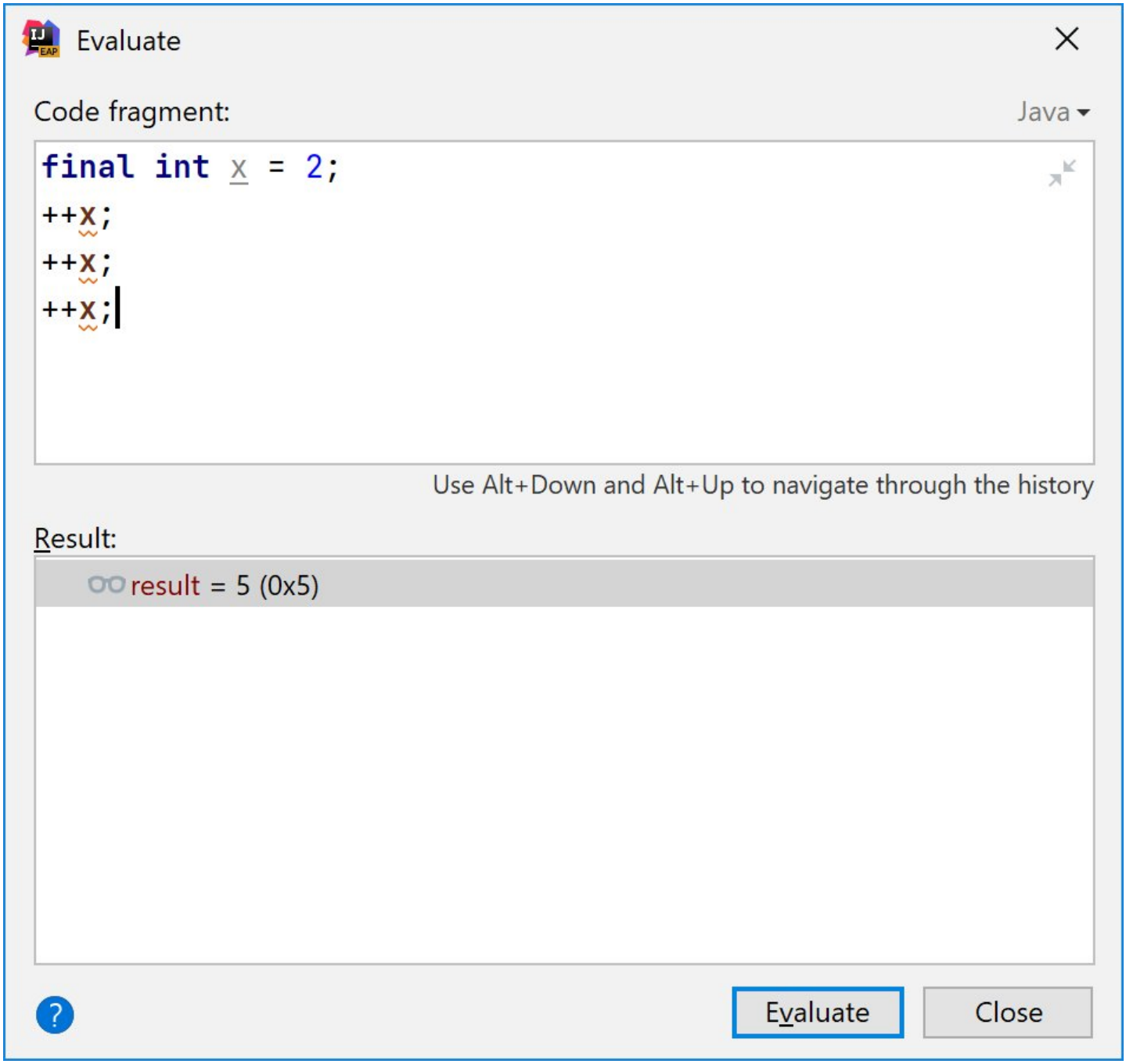


The answer is simple: IDEA debugger has a built-in interpreter for Java! If you access an existing variable or a field, call a method or instantiate an object, it queries JDI. Otherwise, IDE itself does all the calculations.

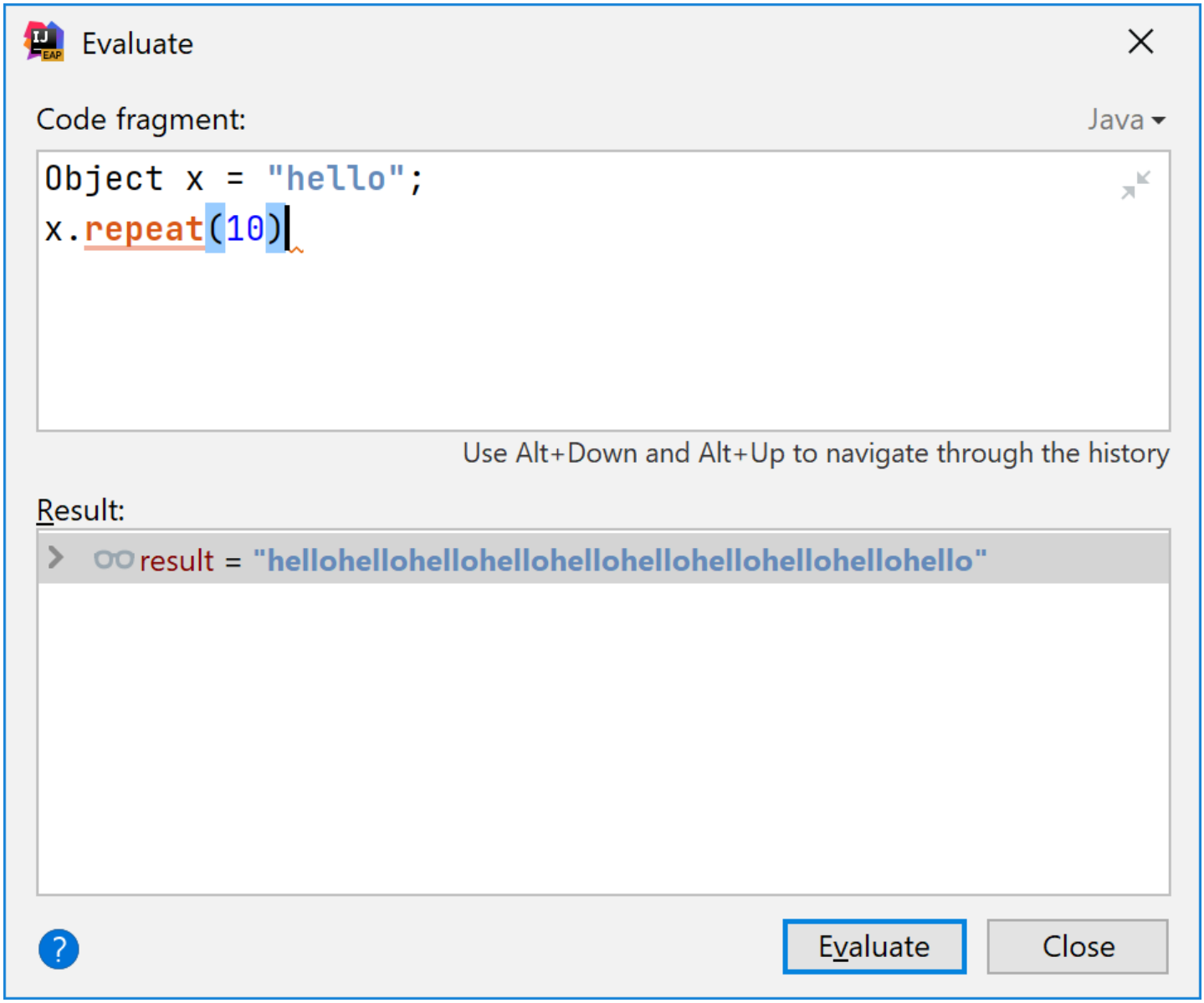
The amusing thing is that debugger authors didn't bother to make it as strict as Java itself, so the language inside the interpreter is a much more permissive version of Java. For example, you don't need to initialize variables there! Default values like 0 are used automatically.



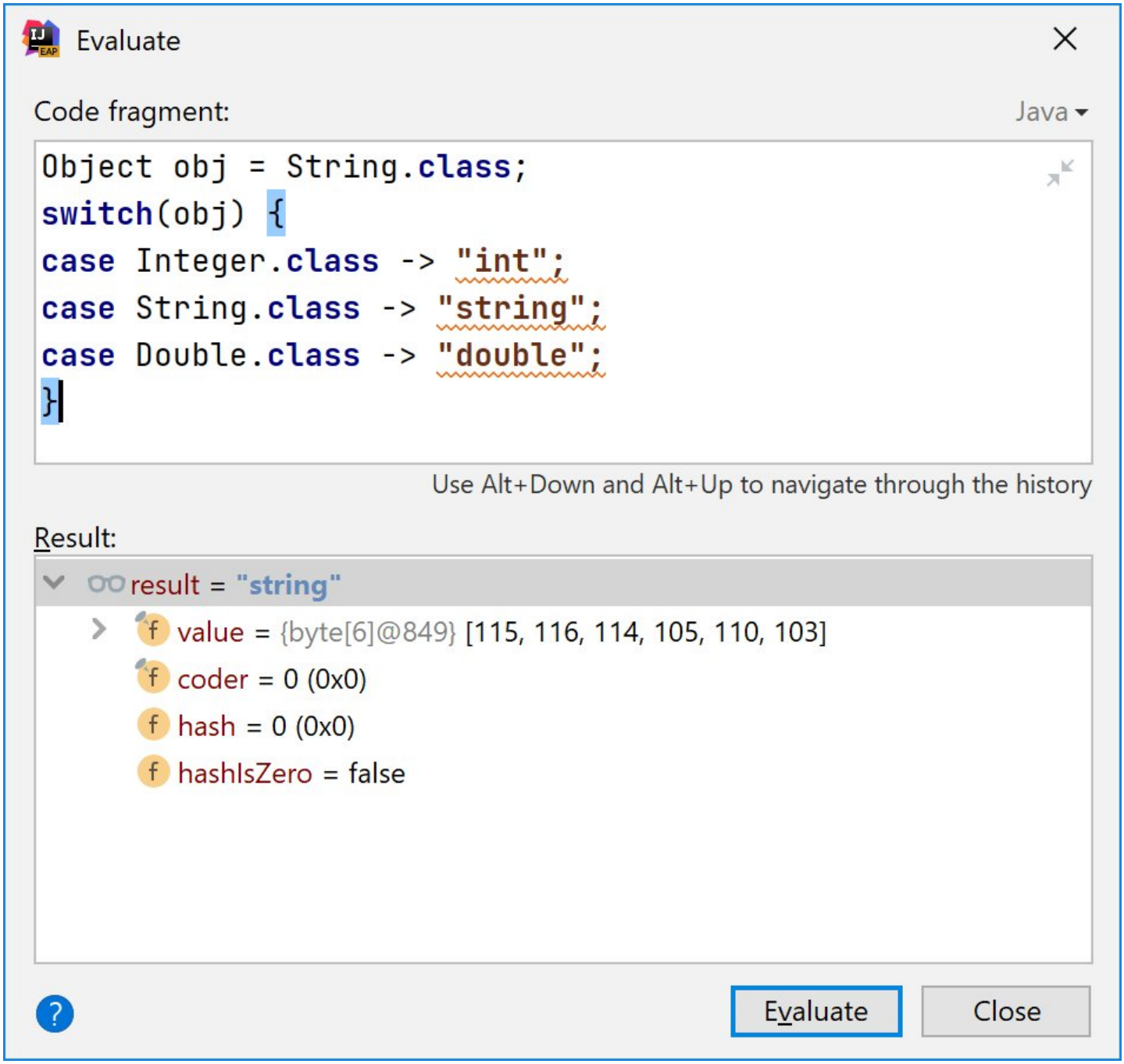
Of course, final variables could be changed. Why artificial limitations?



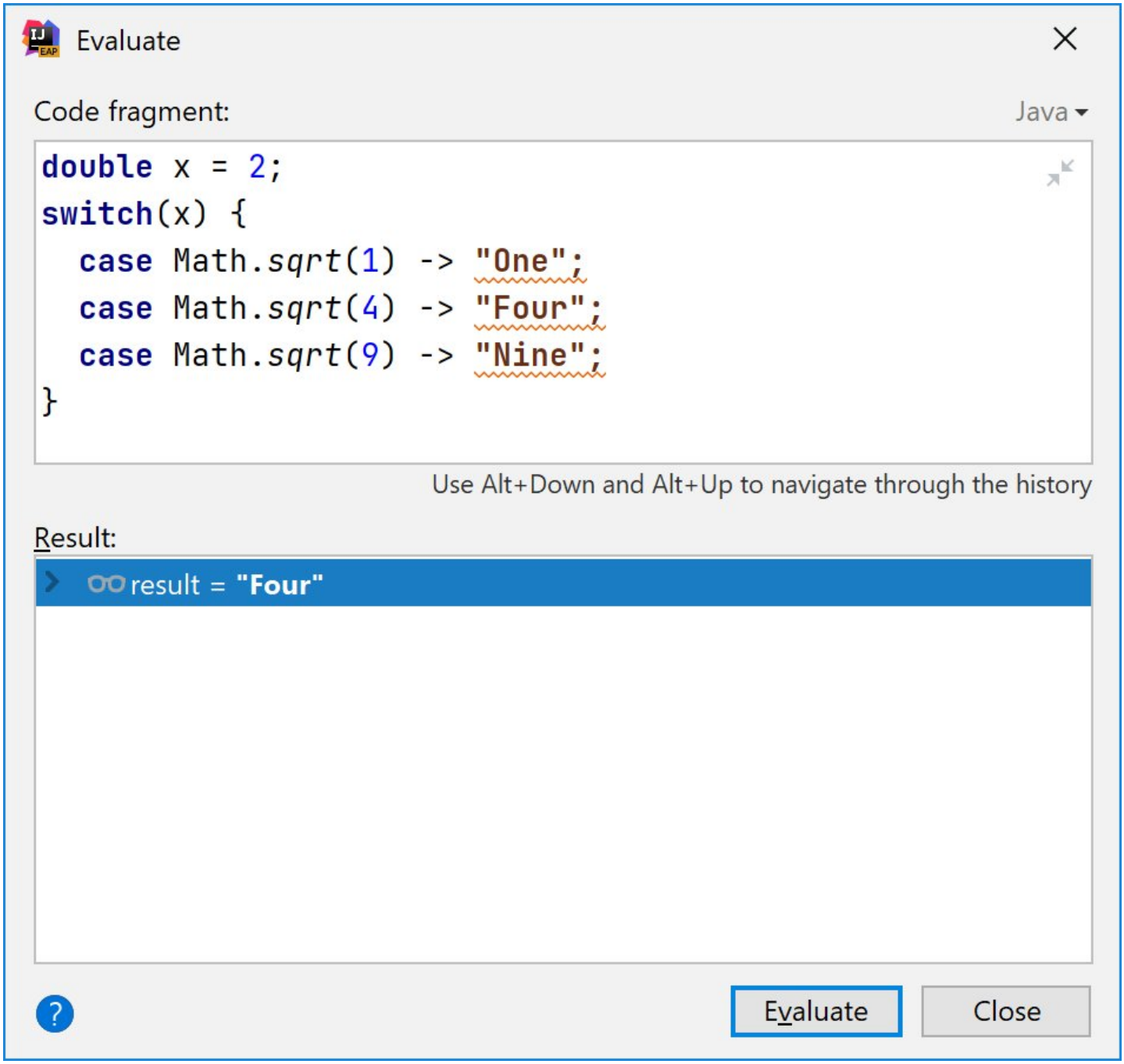
Another thing: most of the casts are not necessary. This interpreted Java more resembles dynamically typed languages like JavaScript. Duck-typing works: if there's a method, we can call it. Even if the highlighter complains.



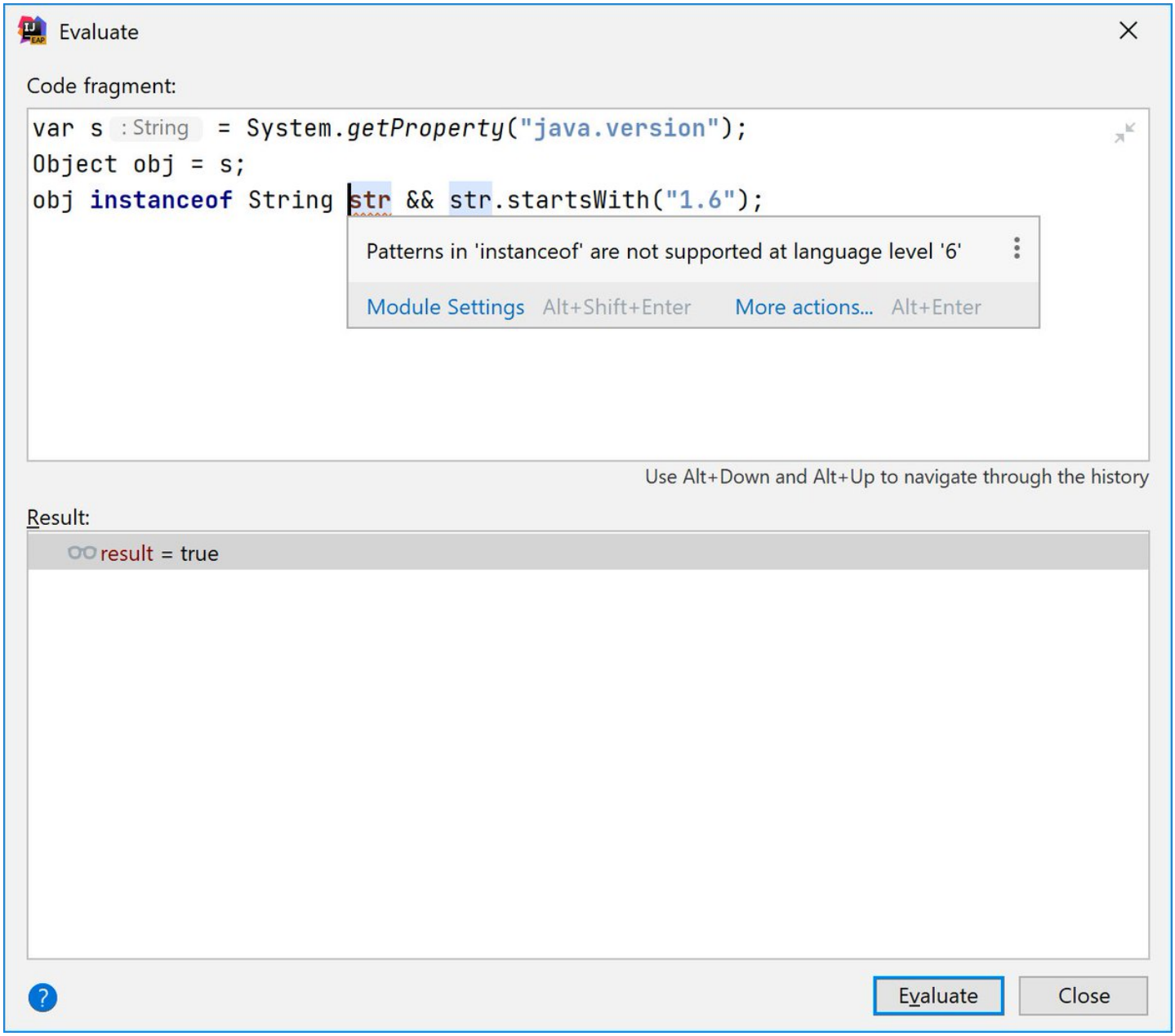
Switch statements and expressions are more permissive there: you can switch over any type. Also, no default is necessary for switch expressions. If it's not visited, the evaluation will be successful.



In addition to use any type, you can use any expressions as case labels, not just constants! The first matching branch will be executed.



Finally, even if your project still uses Java 6 JDK, it doesn't stop you from using new language features. For example, 'var' declarations and patterns in instanceof work perfectly, because they are interpreted by IDE, not by your JDK. Just ignore error messages!


Evaluate
Code fragment:
var s : String = System.getProperty("java.version");
Object obj = s;
obj instanceof String str && str.startsWith("1.6");
Patterns in 'instanceof' are not supported at language level '6'
Module Settings Alt+Shift+Enter
More actions... Alt+Enter
Use Alt+Down and Alt+Up to navigate through the history
Result:
result = true
Evaluate
Close