

# Debugging

The generated classes can be debugged interactively, even though they were created on-the-fly.

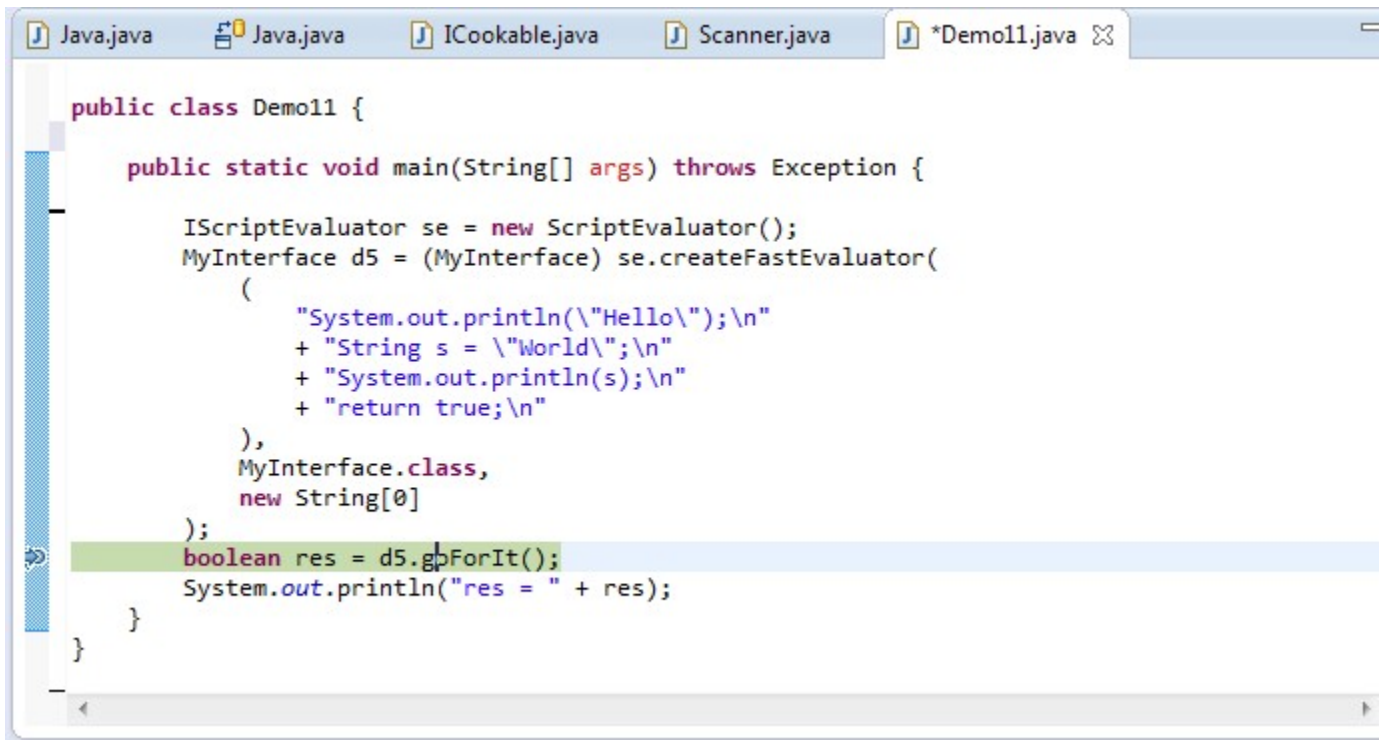
All that needs to be done is set two system properties, e.g. when starting the JVM:

```
java -Dorg.codehaus.janino.source_debugging.enable=true  
-Dorg.codehaus.janino.source_debugging.dir=C:\tmp
```

(The second property is optional; if not set, then the temporary files will be created in the default temporary-file directory.)

When JANINO scans an expression, script, class body or compilation unit, it stores a copy of the source code in a temporary file which the debugger accesses through its source path. (The temporary file will be deleted when the JVM terminates.)

Then when you debug your program



```
public class Demo11 {  
    public static void main(String[] args) throws Exception {  
        IScriptEvaluator se = new ScriptEvaluator();  
        MyInterface d5 = (MyInterface) se.createFastEvaluator(  
            (  
                "System.out.println(\"Hello\");\n"  
                + "String s = \"World\");\n"  
                + "System.out.println(s);\n"  
                + "return true;\n"  
            ),  
            MyInterface.class,  
            new String[0]  
        );  
        boolean res = d5.gpForIt();  
        System.out.println("res = " + res);  
    }  
}
```

, you can step right into the generated code

```
Java.java | ICookable.java | Scanner.java | *Demo11.java | janino28563... | »1  
System.out.println("Hello");  
String s = "World";  
System.out.println(s);  
return true;
```

, and debug it:

```
*Demo11.java | janino28563... | »4  
System.out.println("Hello");  
String s = "World";  
System.out.println(s);  
return true;
```

| Name   | Value           |
|--------|-----------------|
| ● this | SC (id=78)      |
| ▶ ○ s  | "World" (id=81) |

World

As you can see, you can even inspect and modify fields and variables - everything your debugger supports.