26 | ENGEL.ch06 Page 161 Friday, May 19, 2000 2:34 PM

This way of checking external references gives JVM programmers tremendous flexibility to change classes without having to recompile all the classes that use them. You only have to check for the fields and methods that are actually used. You can add or delete fields and methods, without altering the correctness of classes that don't use them.

In addition, because they're checked by name and type, you can move methods around in the source file without affecting other classes that use this class. This helps solve the "fragile base class" problem familiar to C++ programmers, who know that even a seemingly negligible change such as changing the order of methods can require recompilation of every file in the project. It is a cause of many subtle and mysterious bugs in C++ projects.

Many JVM implementations, including Sun's Java Development Kit, don't actually test external references until they are needed. This can save a lot of time. You can start using the class immediately, without having to wait for all the related classes (and all the classes related to those classes, and so on) to load. It's quite possible that many of the references will never need to be checked because the field or method is never used.

6.7 Java Language and Verification Algorithm

The most common programming language for writing Java virtual machine programs is Java. One thing the language guarantees is that anything you can say in Java can be translated into virtual machine code that will pass the verification algorithm.

The verification rules state that the arithmetic instructions (`iadd`, `ddiv`, etc.) always take two values of the same type (two `int`s, two `double`s, etc.). Say you have the following Java program fragment:

```
int i = 70;              // Call this variable 1
float j = 111.1;         // Call this variable 2
double k = i+j;          // Call this variable 3 (and 4)
```

A naïve (and invalid) translation into bytecodes would be

```
bipush 70        ; Initialize i (variable 1) to 70
istore_1
ldc 111.1        ; Initialize j (variable 2) to 111.1
fstore_2
iload_1          ; Push the integer 70
fload_2          ; Push the float 111.1
```