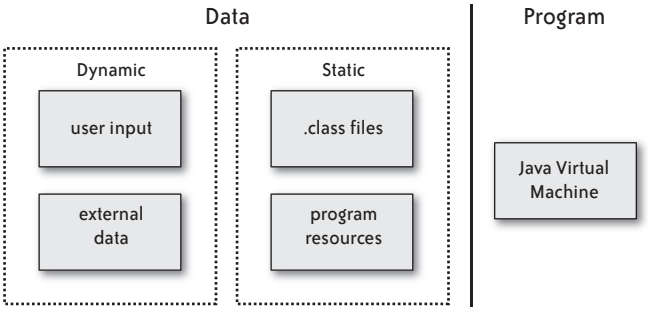

FIGURE 9-6. Program and data regions in a Java process

In a "Just-In-Time" compiled environment such as Sun HotSpot, the commonly used sections of bytecode are translated or dynamically compiled into the native instruction set of the host machine. This moves the class bytes from the data region into the code region. These classes then execute as native code that accelerate the program to native speed. See Figure 9-7.

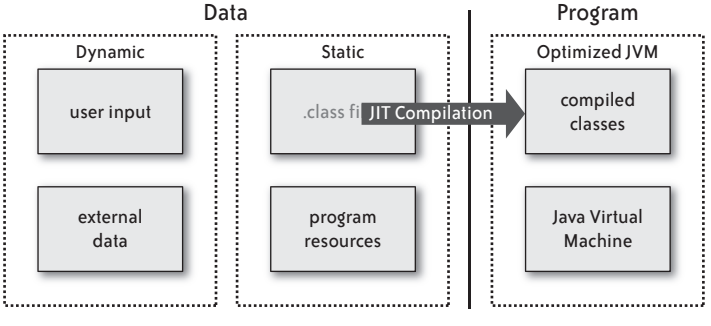

FIGURE 9-7. Just-In-Time compilation in a Java environment

In JPC we take advantage of the fact that not all class data has to be known at JVM startup. In fact, in Java-speak, "static data" would be better referred to as "final data." When a class is loaded, its class bytes are fixed and cannot be changed (let's ignore the JVM TI[#] for the sake of convenience). This allows us to define new classes at runtime, a concept that will be immediately familiar to those who work with plug-in architectures, applets, or J2EE web containers.

[#] For those of us that like mucking around with the naughty bits of the JVM, the Tool Interface (*http://java.sun.com/javase/6/docs/technotes/guides/jvmti/*) can do some very interesting things, including class file redefinition.