Der Prozessor eines Computers kann nur Befehle ausführen, welche in der Maschinensprache vorliegen. Jeder Befehl ist eine Folge von Nullen und Einsen. Es handelt sich hier um Dualzahlen, welche auch als Hexadezimalzahlen dargestellt werden können. Für einen Menschen wäre es extrem schwierig, ein Programm in dieser Form zu schreiben. In der Anfangszeit der Computertechnik wurde allerdings tatsächlich in der Maschinensprache programmiert. Danach wurde die Assemblersprache entwickelt. Hier sind die Befehle durch leicht zu merkende Kürzel (Mnemonics) ersetzt wurden. Für komplexere Programme ist allerdings auch die Assemblersprache schwierig zu handhaben. Sie wird nur noch in Spezialfällen benutzt, beispielsweise für Programme welche bzgl. Geschwindigkeit und Ressourcennutzung maximal optimiert sein müssen. In der heutigen Zeit werden die meisten Programme in einer Hochsprache (z.B. C/C++, Java, PHP) geschrieben, welche für den Menschen relativ leicht verständlich ist. Allerdings kann ein solches Programm nur dann ausgeführt werden, wenn dessen Quellcode vorher in die Maschinensprache übersetzt wurde. Je nachdem, welche Programmiersprache (Hochsprache) verwendet wird, kommen verschiedene Verfahren zum Einsatz, welche jetzt erläutern werden:

Compiler

Zunächst mal ist der Compiler ein Computerprogramm. Mit ihm wird der Quellcode eines Programms in die Maschinensprache (manchmal auch Assemblersprache) übersetzt. Das Ergebnis ist eine direkt ausführbare Datei, z. B. eine EXE-Datei auf Windows-Systemen. Möchte man das Programm benutzen, so startet man die ausführbare Datei. Während der Übersetzung wird der Code auch optimiert und auf Fehler untersucht. Eine typische Compilersprache ist C++.

Interpreter

Auch der Interpreter ist ein Computerprogramm. Im Gegensatz zum Compiler erzeugt er keine ausführbare Datei. Möchte man ein Programm nutzen, so wird es vom Interpreter gestartet und während der Laufzeit des Programms Schritt für Schritt in ausführbare Anweisungen (Maschinensprache) übersetzt. Eine typische Interpretersprache ist PHP.

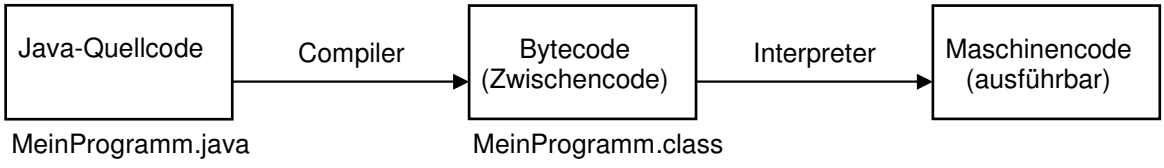
Vergleich von Compiler- und Interpretersprachen

Vorteile der Compilersprachen:
- Liegt ein Programm bereits als ausführbare Datei vor, so läuft es deutlich schneller.
- Während der Laufzeit des Programms wird weniger Arbeitsspeicher benötigt. Der bei Interpretersprachen erforderliche Interpreter belegt während der Laufzeit des Programms einen Teil des Arbeitsspeichers.

Nachteile der Compilersprachen:
- Eine ausführbare Datei ist immer plattformabhängig. So kann z. B. eine unter Windows erzeugte EXE-Datei nicht direkt auf anderen Betriebssystemen (z. B. Linux) ausgeführt werden. (Sollte der Quellcode des Programms zur Verfügung stehen, kann dieser für das andere Betriebssystem neu kompiliert werden.)
- Für die Entwicklung eines Programms wird mehr Zeit benötigt. Für jeden einzelnen Test muss das Programm zunächst neu kompiliert werden, was bei komplexeren Programmen durchaus einige Zeit in Anspruch nehmen kann.

Java-Compiler und Java-Bytecode-Interpreter

Die Entwickler der Programmiersprache Java haben eine Kompromisslösung gefunden, um die jeweiligen Vorteile von Compiler- und Interpretersprachen zu kombinieren. Der Java-Quellcode liegt als Datei mit der Endung "java" vor. Mittels Java-Compiler wird ein Bytecode erzeugt, welcher in einer Datei mit der Endung "class" gespeichert wird. Diese Bytecode-Datei ist nicht ausführbar, deshalb ist der Java-Compiler auch kein "echter" Compiler. Allerdings ist dieser Bytecode bereits optimiert und auf Fehler geprüft. Startet man das Java-Programm, so wird der Bytecode mit Hilfe des Java-Bytecode-Interpreters in Maschinencode übersetzt und ausgeführt.



Der Bytecode (class-Datei) kann auf jedem beliebigen Betriebssystem ausgeführt werden, sofern es dafür einen Java-Bytecode-Interpreter gibt. Damit ist Java wie eine "echte" Interpretersprache plattformunabhängig. Allerdings laufen Java-Programme schneller als diese, weil der Bytecode schneller als der Quellcode übersetzt werden kann. In einer "echten" Compilersprache geschriebene Programme laufen allerdings noch schneller, weil für die hier erzeugten ausführbaren Dateien keine Übersetzung während der Programmlaufzeit erforderlich ist.