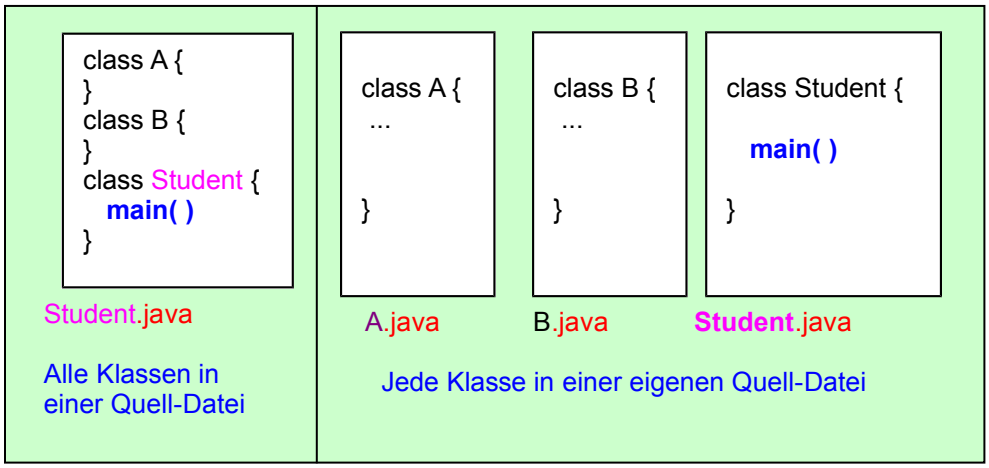
Erstellung, Kompilierung und Ausführung eines Java-Programms

1. Erstellen und Abspeichern des Java-Quelltextes

Java-Quellcode muss die Endung `.java` haben. Die kleinste Einheit, aus der eine Java-Datei bestehen kann, ist eine Klasse. Bei kleineren Übungsbeispielen werden meist alle benötigten Klassen in einer einzigen Datei gespeichert. Der Dateiname kann nicht beliebig gewählt werden, sondern muss mit dem Namen derjenigen Klasse übereinstimmen, die die Methode `main()` enthält. Groß- / Kleinschreibung ist zu beachten.

Wenn ein Programm aus mehreren Klassen besteht, dann können diese in getrennten Dateien abgespeichert werden. Dies ist bei größeren Anwendungen üblich. Auch dann muss der Dateiname dem Namen der Klasse, die `main` enthält, exakt entsprechen.

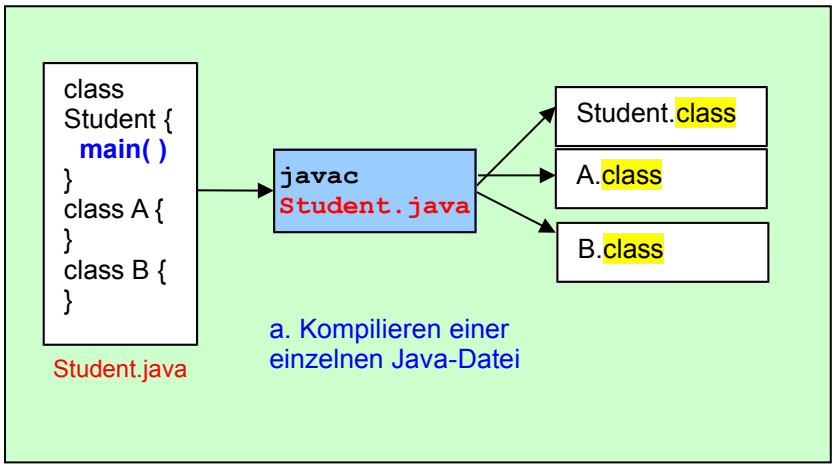


Alternativen bei der Erstellung eines Java-Quelltexts

2. Kompilieren von Java-Dateien

a. Wenn alle benötigten Klassen in einer *einzigen* Datei vorliegen, dann wird der Name dieser Datei (`Student.java`) dem Java-Compiler übergeben. Beim Java SDK heißt der Compiler `javac`. Der Java-Compiler arbeitet in einer kommandozeilenorientierten Umgebung (Linux-Shell, DOS-Box) und kann weitere Parameter/Optionen übernehmen.

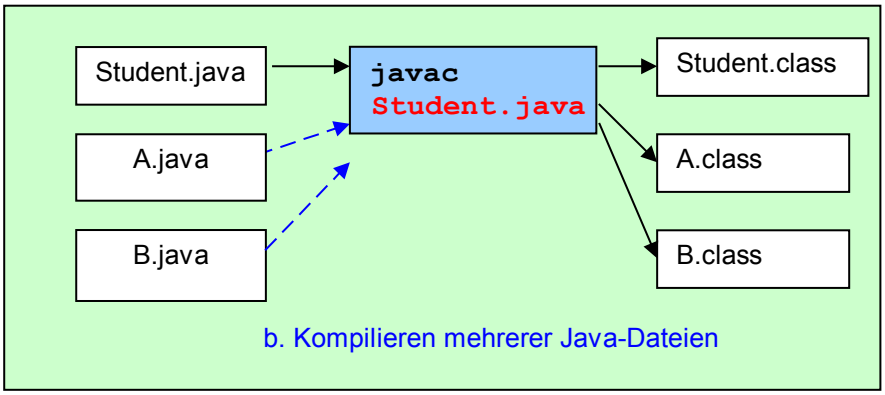
Der Compiler erstellt für jede Klasse, die im Quelltext enthalten ist, eine eigene class-Datei. Diese Datei wird automatisch unter dem Namen der Klasse (**`.class`**) abgespeichert. Nach dem Kompiliervorgang liegen im aktuellen Verzeichnis also unter Umständen eine größere Anzahl von class-Dateien.. Diese können in ein Archiv (`JAR`) komprimiert werden, so dass nur noch eine einzige Datei vorliegt.



b. Wenn zum Programm *mehrere* getrennte Java-Dateien gehören, dann muss dem Compiler **nur** der Dateiname derjenigen Klasse übergeben werden, die `main` enthält. Es ist *nicht* nötig, alle Dateien anzugeben, die eventuell benötigte Klassen enthalten. Der Compiler erkennt an den in main() erzeugten Objekten, welche Klassen gebraucht werden. Er sucht die entsprechenden Java-Dateien im aktuellen Verzeichnis, kompiliert sie und speichert sie unter ihrem Klassennamen mit der Endung **`.class`** ab.

Der Java-Compiler findet die benötigten Klassen aber nur, wenn diese unter ihrem Klassennamen (`.java`) abgespeichert werden. Er prüft nicht andere Dateien, ob die Klasse dort vorkommt. Gesucht wird prinzipiell nur innerhalb des gerade aktiven Verzeichnisses. Wenn es nötig ist, kann ein *untergeordnetes* (kein übergeordnetes) Verzeichnis oder ein Paket (`package`) importiert werden.

Wenn die class-Dateien bereits kompiliert vorliegen, dann wird vom Compiler geprüft, ob sie auf neuestem Stand (gegenüber der entsprechenden Java-Datei) sind. Wenn dies nicht der Fall ist, werden sie neu kompiliert (ähnlich `make` bei C-Programmen).

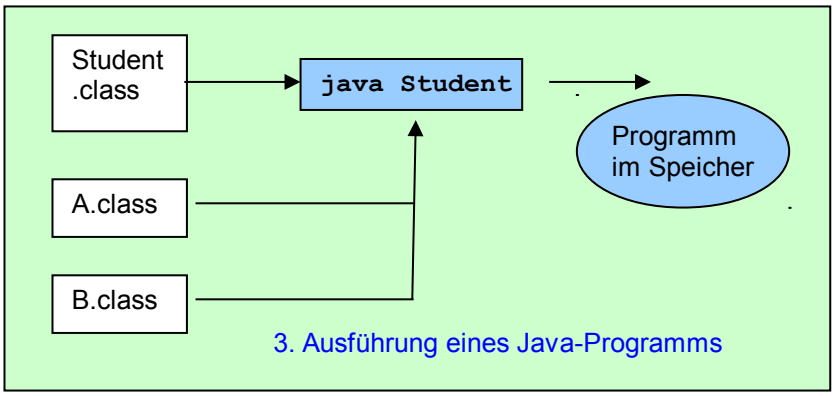


Da durch den Compiler-Durchlauf jede Klasse in genau eine class-Datei verwandelt wird, ist es letztendlich gleichgültig, ob der Java-Quellcode in einer einzigen Datei oder in mehreren Klassen-Dateien vorlag. Die Entscheidung richtet sich nach praktischen Gesichtspunkten (Länge des Quelltextes, Anzahl der Klassen ...).
Eine class-Datei enthält alle Informationen über die jeweilige Klasse (Attribute, Methoden) in sogenanntem Byte-Code. Dieser stellt eine prozessorunabhängige Codierung einer Java-Klasse dar, die zur Übertragung im Internet geeignet ist. Die strikte Trennung erleichtert es der Java-Laufzeitumgebung (Interpreter), Klassen zu finden, indem einfach an den Name der gesuchten Klasse die Endung .class angehängt wird. Class-Dateien sind nicht direkt menschenlesbar. Sie können aber mit einem Decompiler in lesbaren Java-Code zurück verwandelt werden. Hierfür gibt es zahlreiche Programme (JDC, Decafé ...).

3. Java-Interpreter - Ausführung

Da eine class-Datei keinen direkt ausführbaren Prozessorcode enthält, kann sie nur unter ganz bestimmten Rahmenbedingungen gestartet werden. Zur Ausführung einer Java-Applikation muss auf dem jeweiligen Rechner ein plattformspezifischer Java-**Interpreter** installiert sein. Dieser arbeitet den Byte-Code zeilenweise ab und wandelt ihn dabei in ausführbare Maschinenanweisungen um. Der Befehl zum Aufruf des Java-Interpreters lautet **`java`**. Dem Interpreter wird beim Aufruf der Name der class-Datei übergeben, die main enthält. Die Endung `.class` muss weggelassen werden (**`java Student`**)

Die zur Ausführung eines Java-Programms nötigen Laufzeit-Module müssen auf dem Rechner vorhanden sein, auf dem die Anwendung ausgeführt werden soll. Jede Java-Anwendung setzt also die Installation einer prozessorspezifischen Java-Runtime-Library (JRL) bzw. Java-Runtime-Environment (JRE) oder Java-Virtual-Machine (JVM) voraus. Ein Java-Applet wird immer innerhalb eines HTML-Kontexts dargestellt und benötigt deshalb einen geeigneten Internet-Browser.



3. Ausführung eines Java-Programms

Der Java-**Interpreter** sucht in der angegebenen class-Datei nach der Methode **`main`**. Wenn er sie findet, werden die darin stehenden Java-Anweisungen ausgeführt. Wenn zur Ausführung des Programms weitere Klassen benötigt werden, dann sucht sie der Interpreter als `class`-Dateien in demselben Verzeichnis, in importierten (untergeordneten) Verzeichnissen bzw. in angegebenen Packages. Dies setzt immer voraus, dass alle zum Java-Projekt gehörigen Klassen kompiliert wurden (Schritt 2), denn der Java-Interpreter kann nicht kompilieren, sondern nur Byte-Code ausführen.

Wenn die Methode **`main()`** NICHT gefunden wird, erscheint die Fehlermeldung:

```
java.lang.NoSuchMethodError: main
Exception in thread "main"
```

Diese Fehlermeldung deutet darauf hin, dass die angegebene class-Datei keine Methode names **`main()`** enthält. In diesem Fall muss der Name der class-Datei überprüft und die Java-Datei auf evtl. Schreibfehler getestet werden.