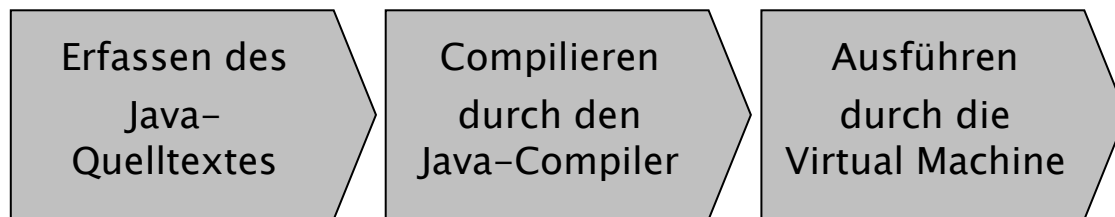


Informatik: Einführung in Java

Ziele

- Verstehen wie ein Java-Programm erstellt, compiliert und ausgeführt werden kann
- Einfache Programm erstellen können
- Programmierregeln zum Erstellen des Programms anwenden können
- Begriffe wie Compiler, Bytecode, Virtual Machine, JDK, JRE einordnen können
- Der unterschiedlichen Java-Versionen auseinander halten können
- Weiters soll ein kurzer Einblick in Programmiersprachen gegeben werden
- Die Begriffe Syntax, Semantik und Pragmatik einer Programmiersprache verstehen

Prinzipielle Vorgangsweise



Problemstellung

n Zahlen von 1 beginnend sollen addiert werden

z. B. n = 10 Ergebnis 55

Erfassen des Java-Quelltextes

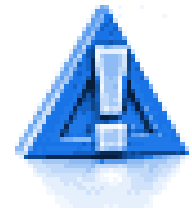
```
Summe.java x
/**
 * n Zahlen von 1 beginnend werden addiert
 * @author Sepp
 */
public class Summe
{
    public static void main(String[] args) {
        /**
         * Eine Anpassung von n bestimmt die Länge der zu
         * addierenden Zahlenfolge
         */
        int n = 10;
        int summe = 0;
        // Zahlen werden ab 1 addiert
        int i = 1;
        while (i <= n) {
            summe = summe + i;
            i = i + 1;
        }
        System.out.println(summe);
    }
}
```

Grundgerüst eines Java-Programms

```
public class Programmname
{
    public static void main(String[] args) {
        ...
    }
}
```

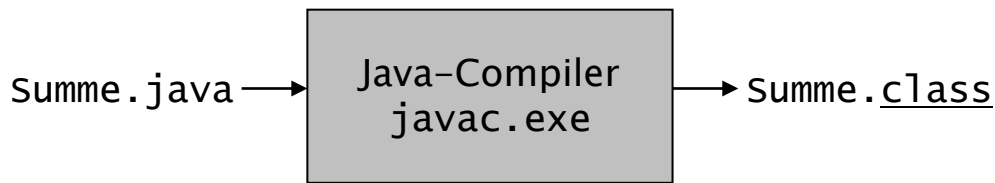
Programmierregeln

- Positionierung der Klammern { } und Einrückungen
- Programname beginnt mit Großbuchstaben
- Quelltext speichern in Textdatei mit Dateiname
Programname.java
- Variablennamen (summe, n, i) beginnen mit Kleinbuchstaben
- **Sprechende** Variablennamen verwenden
- **Reservierte Wörter** (**public**, **class**, **while**, **static**, **void**) werden klein geschrieben
- **Kommentare** machen das Programm leserlicher
Einzeilen-, Block- und Dokumentationskommentar
- **Kommentieren** Sie Ihr Programm so dass Sie und Außenstehende es auch nach längerer Zeit sofort verstehen
- **Schreiben** Sie Ihre Kommentare während der Programmierung und nicht nachher
- Der **Inhalt einer Programmzeile** sollte nie über die Bildschirmbreite hinausragen



Compilieren durch den Java-Compiler

Java-Compiler übersetzt den Java-Quelltext in **Bytecode**



Problem: Bytecode kann **NICHT** direkt vom Prozessor ausgeführt werden

Ausführen durch die Virtual Machine

Bytecode ist nur für hypothetischen Prozessor die **Virtual Machine** bestimmt

Damit Bytecode trotzdem gestartet werden kann, wird ein Programm benutzt, welches den hypothetischen Prozessor auf realem Prozessor simuliert

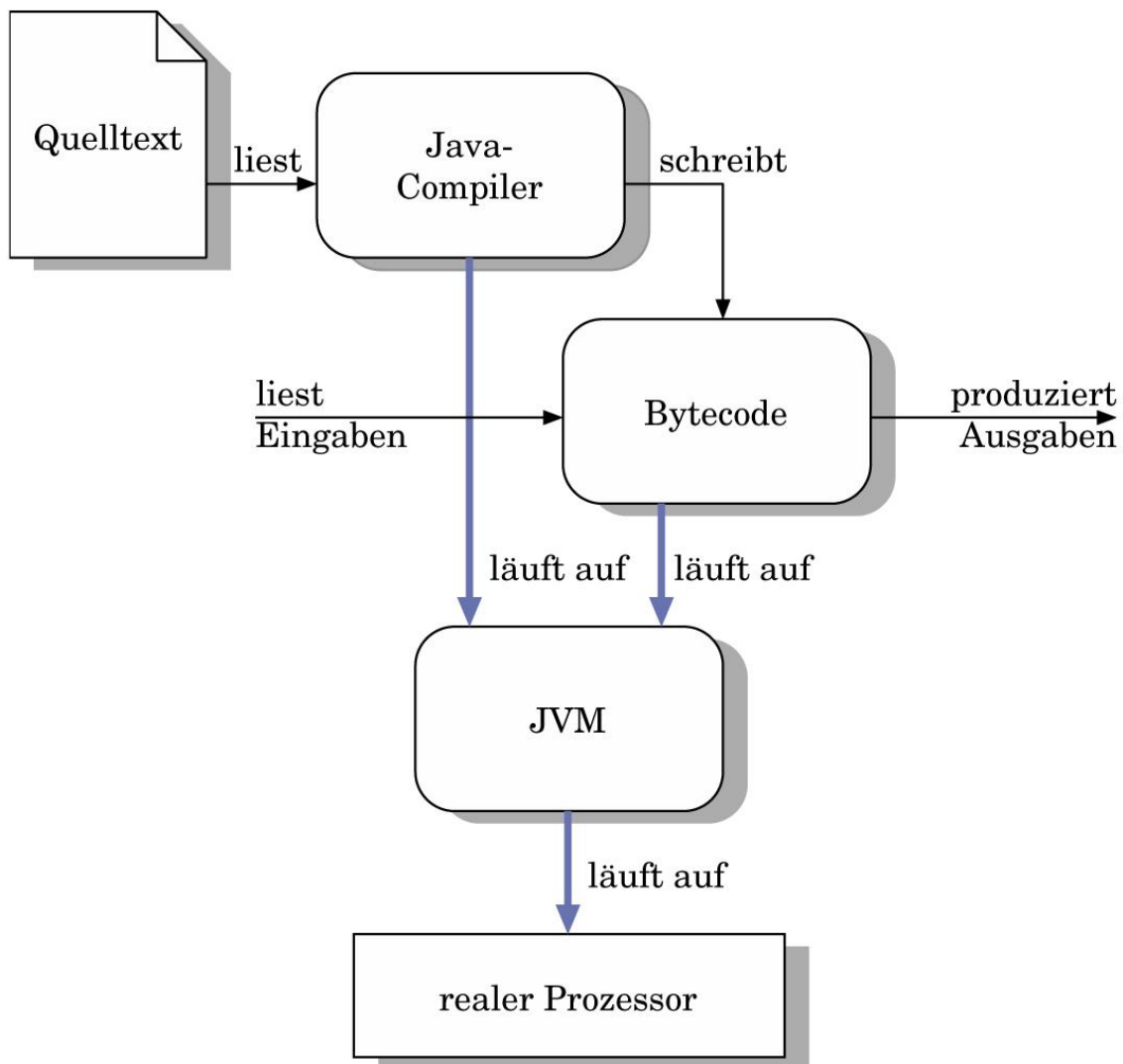
Programm heißt **Java Virtual Machine (JVM)**



Vor- und Nachteile

- ☺ Bytecode kann unverändert auf jedem System verwendet werden für das JVM existiert
- ☹ Hoher Ressourcenverbrauch (Speicherplatz, Rechenleistung)

Zusammenfassung



Java-Compiler javac ist selbst ein Javaprogramm, besteht aus Bytecode und läuft auf der JVM

Java-Versionen

Entwickelt von Sun Microsystems, Inc. und seit 1996 verfügbar

Für Entwickler – **Java Development Kit (JDK)**

umfasst `javac.exe`, `java.exe`, `appletviewer.exe`, `javadoc.exe`, `jar.exe` usw. und Laufzeitbibliothek

Standard Edition (SE)

Dient zum Programmieren auf Einzelplatzcomputern

Enterprise Edition (EE)

Dient zum Einsatz von Java auf Servern

Micro Edition (ME)

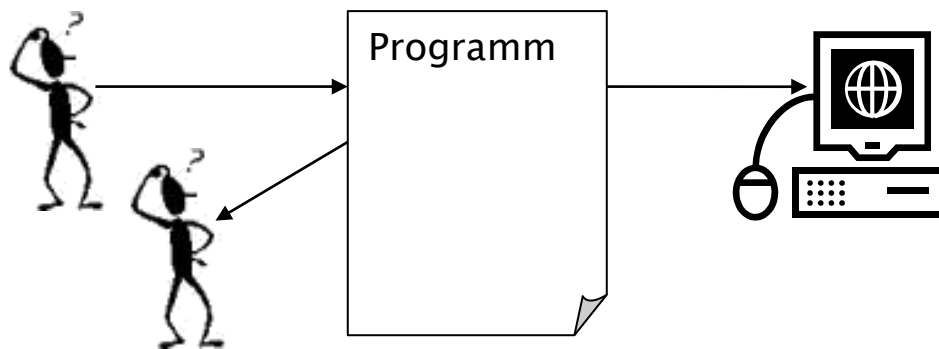
Java zum Einsatz auf Geräten mit begrenzten Ressourcen (Palm, Handys, Waschmaschine)

Für Anwender – **Java Runtime Environment (JRE)**

umfasst `java.exe` und Laufzeitbibliothek damit auch Java-Applets im Webbrowser gestartet werden können



Programmiersprachen



- Dienen zur Formulierung von Programmen
- Dienen zur Kommunikation zw. Mensch und Maschine aber auch zw. Mensch und Mensch

Generationen

Maschinenorientierte Sprachen (ab 1950er Jahre) – assembler

Nutzen Möglichkeiten des Rechners ideal aus

Prozedurale Sprachen (ab 1960er Jahre) – Fortran, Basic, Pascal, C

Orientieren sich am menschlichen Denken

Objektorientierte Sprachen (ab 1990er Jahre) – C++, Java

Zusammenhang zw. Daten und Programm wird in Objekten

abgebildet. Lassen große Programmsysteme leichter erstellen und warten

Betrachtungsweisen einer Programmiersprache:

Syntax

Regelt Form und Struktur von Sprachelementen

natürliche Sprache

Der Apfel ist grün

Apfel ist grün der

Programmiersprache

while (i <= n)

(i n <=) **while**

Semantik

Regelt das sinnvolle Zusammenwirken syntaktisch richtiger Befehle

natürliche Sprache

Der Apfel ist grün

Das Schiff singt grün

Programmiersprache

while (i <= n)

```
int sum = 0;
int i = 1;
while (i <= n) {
    sum = sum + i;
}
```

Pragmatik

Beschreibt Schablonen und Muster, die sich allgemein bewährt haben und erfahrungsgemäß funktionieren um Programmieraufgaben zu lösen

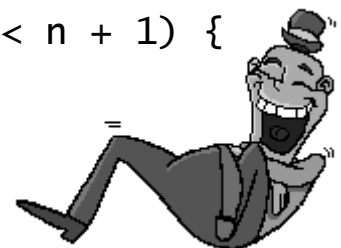
Fundierte Kenntnisse der Pragmatik unterscheiden einen professionellen Entwickler von einem Anfänger

Gebräuchliche Schablone

```
int n = 2;
int i = 0;
while (i <= n) {
    ...
    i = i + 1;
}
```

Lächerlich

```
int n = 2;
int i = 1;
while (i - 1 < n + 1) {
    i = i - 1;
    ...
    i = i + 2;
}
```



Beide Programmfragmente produzieren dieselbe Ausgabe...