Programmierung

Programme, Compiler,
virtuelle Maschinen, Java

Programme

- Ein Programm ist eine Folge von Anweisungen, die einem Computer sagen, was er tun soll

tuwas.c

```
for(int i=0; i<10; i++)
  System.out.println
        ("Hello World");
Beep();
Play(nextAudiofile);
```

- Programme werden in einer künstlichen Sprache verfasst und in Textdateien gespeichert. Sie sollen
  - für Menschen lesbar sein
  - so präzise sein, dass sie unmissverständliche Handlungsanweisungen für einen Rechner beinhalten.

Programmiersprachen

- Es gibt Tausende von Programmier-sprachen
  - Allgemeine Sprachen, mit denen prinzipiell alle Fähigkeiten eines Rechners zugänglich sind
    - Pascal, C, Java, C#, Prolog, LISP, FORTRAN, …
  - Spezialsprachen für bestimmte Anwendungen
    - SQL, HTML, TeX, TCL/TK, …

Maschinensprache

- Rechner verstehen nur sehr einfache Kommandos

  ADD AX FFh, INC AX, JMP 0A3, MOV AX 5

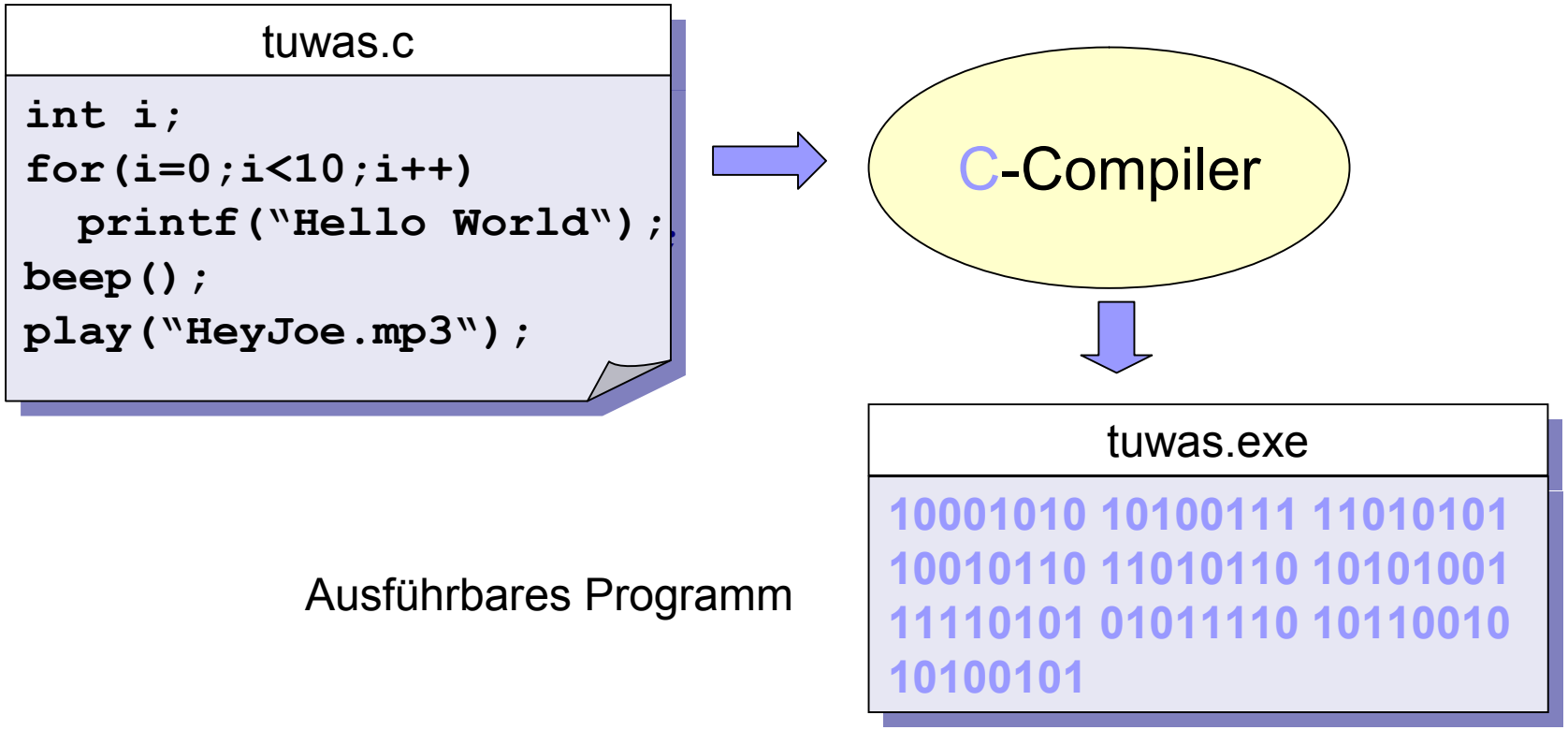
- Diese Kommandos sind als Zahlen im Binärsystem (d.h. mit den Ziffern 0 und 1) kodiert, etwa:

  10001010 10100111 11010101 10010110 11010110
  10101001 11110101 01011110 10110010 10100101

- Für Menschen sind Programme in Maschinen-sprache nicht mehr direkt lesbar.

Compiler

- Ein *Compiler* übersetzt Programme aus einer höheren Sprache in die Maschinensprache des Computers.



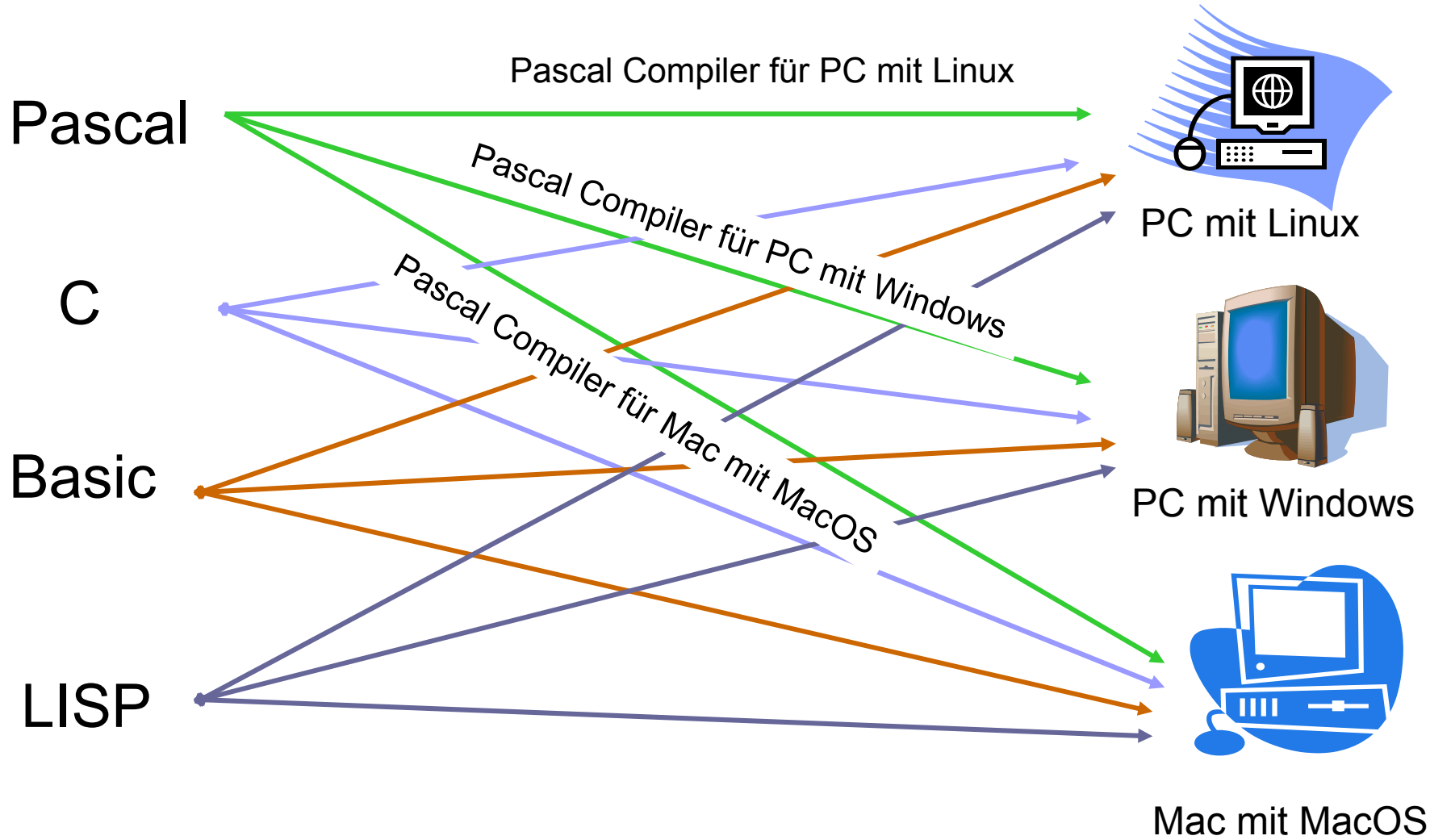
Maschinenabhängigkeit

- Jede Maschine hat andere Befehle
  - Ein Programm für einen PC läuft nicht auf dem Mac und umgekehrt
- Betriebssysteme stellen eine Infrastruktur bereit, die Programme nutzen können:
  - Dateiverwaltung, Speicherverwaltung, Prozessverwaltung, Input/Output, Hilfsprogramme…
- Programme greifen direkt auf die Ressourcen von Betriebssystemen zu
  - writefile, print, readfile, send, receive, out, …

Konsequenz:

- Jedes Programm läuft nur auf einem bestimmten Rechnertyp mit einem bestimmten Betriebssystem
  - z.B. nur auf PC mit Linux, oder nur auf Mac mit MacOS

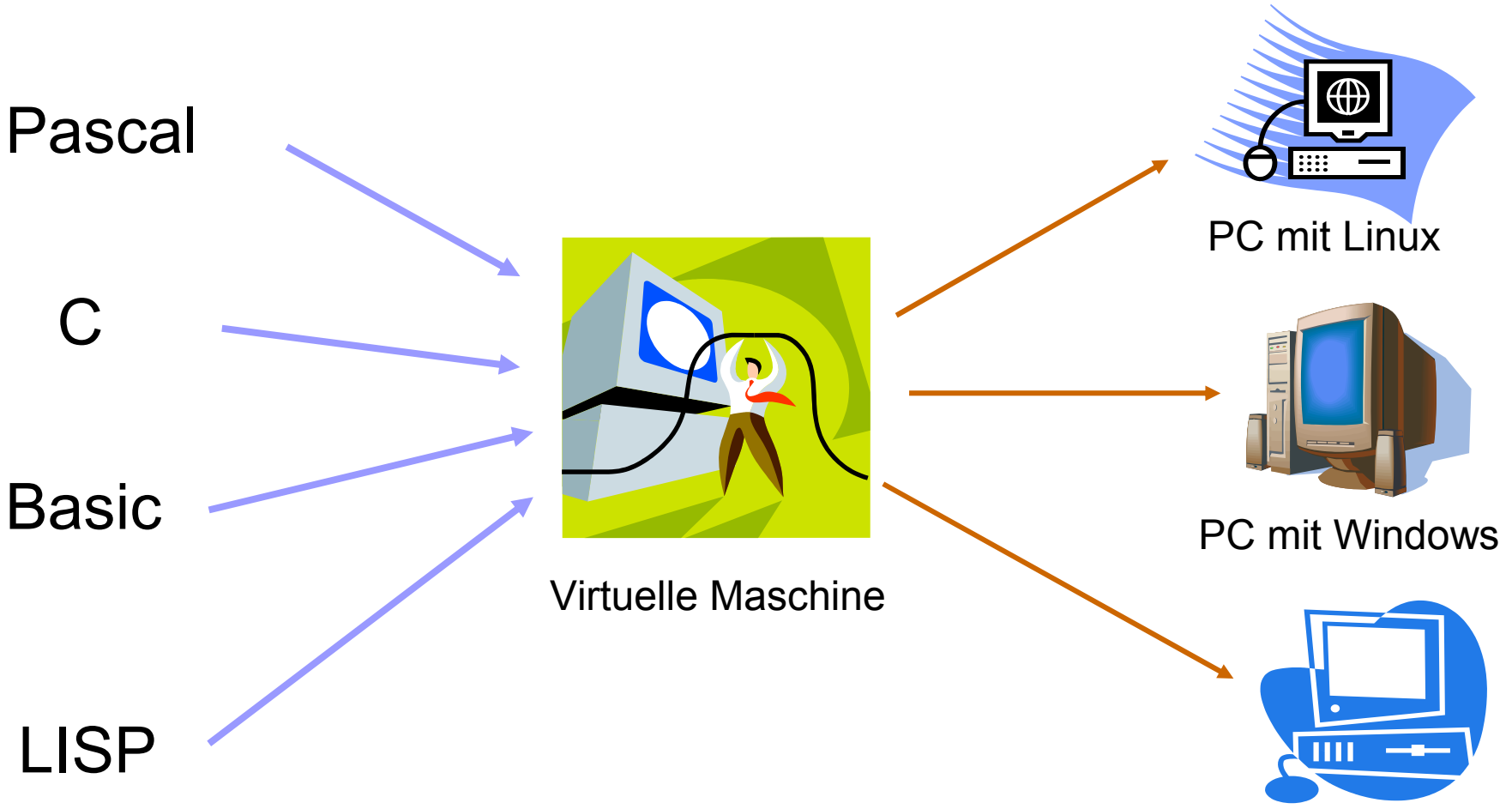
m Sprachen, n Plattformen = m*n Compiler



Virtuelle Maschine

- Eine Virtuelle Maschine ist ein gedachter Computer VM.
- VM wird auf jedem realen Computer simuliert (emuliert).
- Man kann eine VM auch als Zwischensprache auffassen. Anweisungen in dieser Sprache heißen Bytecode.
- Für jede Sprache L benötigt man nur einen Compiler von L nach VM.

Virtuelle Maschine – ein Traum ?



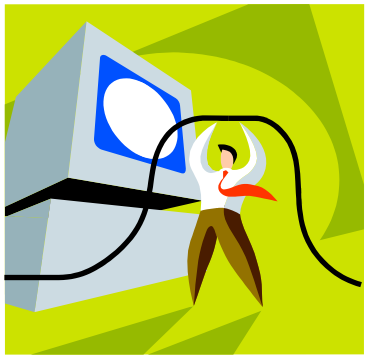
M Sprachen , N Plattformen
= M Compiler + N Implementierungen der VM

Nachteil einer virtuellen Maschine

- Effizienzverlust
  - Spezielle Stärken bestimmter Hardware werden vielleicht nicht genutzt
    - z.B. spezielle Graphikbefehle, Textbefehle, …
  - Spezielle Features bestimmter Sprachen kommen auf bestimmten (virtuellen) Maschinen mehr auf anderen weniger zur Geltung
    - Garbage Collection, Rekursion, …
- Fazit: Eine ideale virtuelle Maschine gab es bisher nur im Traum.

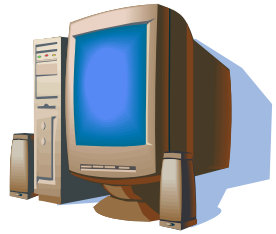
Java Virtuelle Maschine – nur für Java

- Die Java-Runtime-Engine ist eine virtuelle Maschine – speziell für die Sprache Java.
- Sie ist auf fast allen Plattformen implementiert.
- Unter Windows heißt sie `java.exe`.



JVM
(Java Virtual Machine)

PC mit Linux



PC mit Windows

Mac mit MacOS

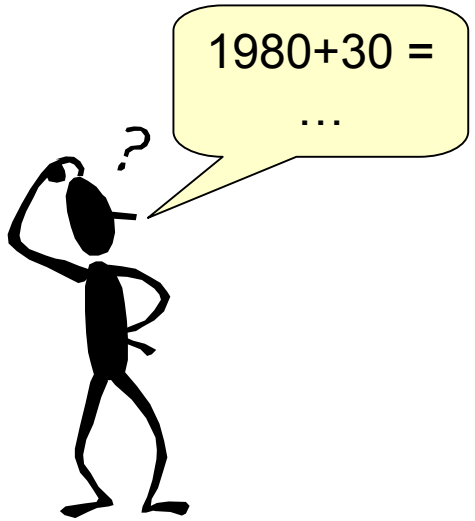
Alles gab es schon mal …

- Virtuelle Maschinen für
  - **eine** Sprache und
  - **multiple** Plattformen

  gab es schon früher, sie haben sich aber nie durchgesetzt.

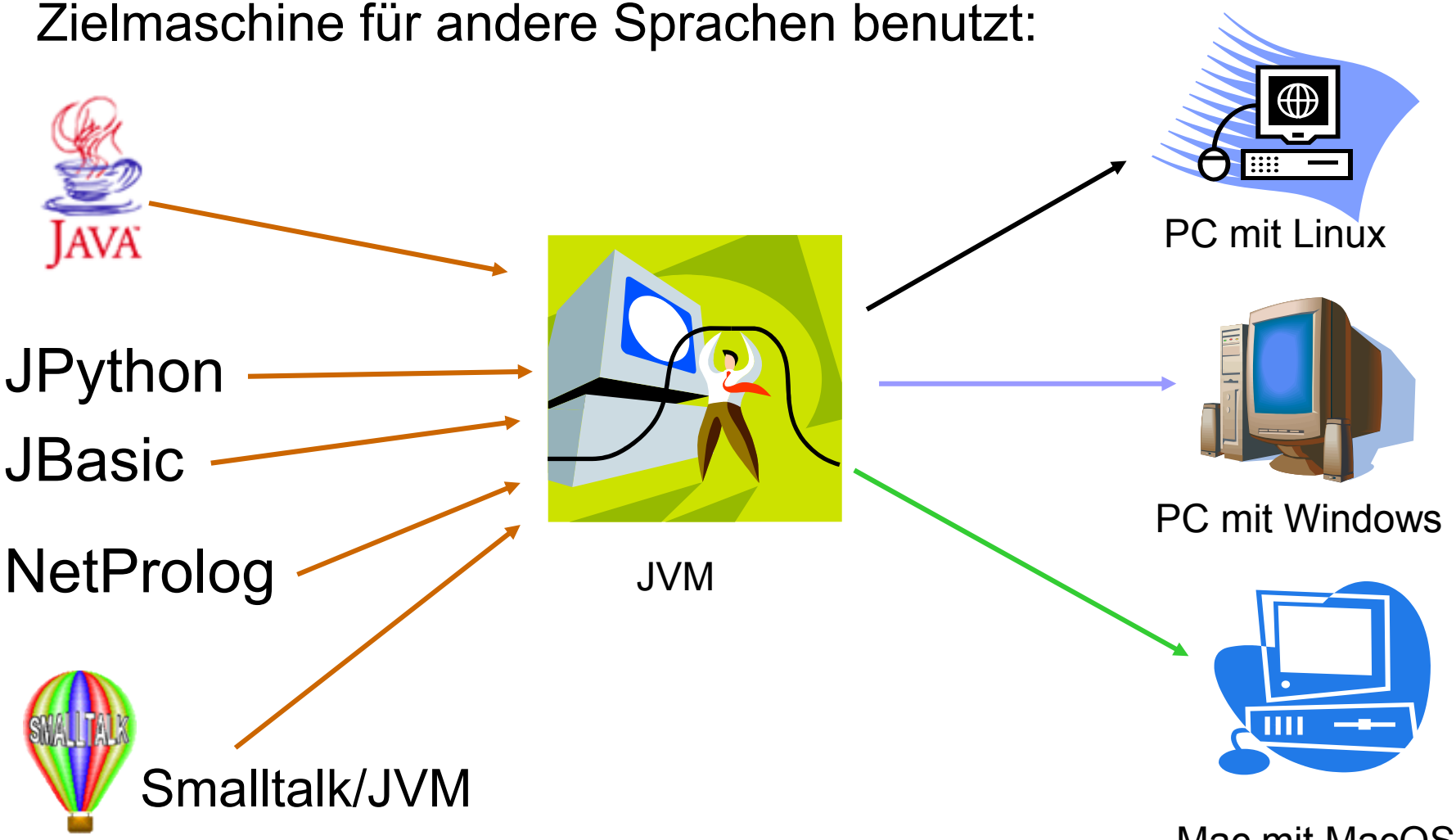
- Früher existierende virtuelle Maschinen für verschiedene Sprachen:
  - Pascal : p-Maschine (Mitte der 80-er Jahre)
  - Smalltalk : Smalltalk Bytecode Interpreter
  - Prolog : WAM

- *Smalltalk 80* kann man als Vorläufer von Java ansehen. *Smalltalk* ist Java in vieler Beziehung weit überlegen. Es war seiner Zeit ca. 30 Jahre voraus …



Erfolg steckt an …

- Seit sich Java durchgesetzt hat, wird die JVM auch als Zielmaschine für andere Sprachen benutzt:



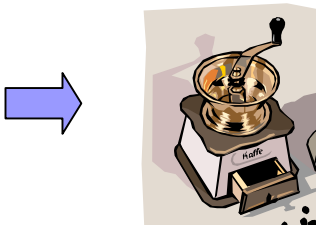
Compilation von Java-Programmen

- Aus einer Textdatei mit der Endung „**.java**“ erzeugt der Compiler **javac** eine Datei mit gleichem Namen, aber Endung „**.class**“
- Diese enthält den *Bytecode* für die JVM

Hallo.java

```
public class Hallo{
 public static void
  main(String[] args)
   { System.out.println
       (“Hallo Leute“); }
}
```

Quellprogramm als Textdatei **Hallo.java**



Compiler **javac.exe**

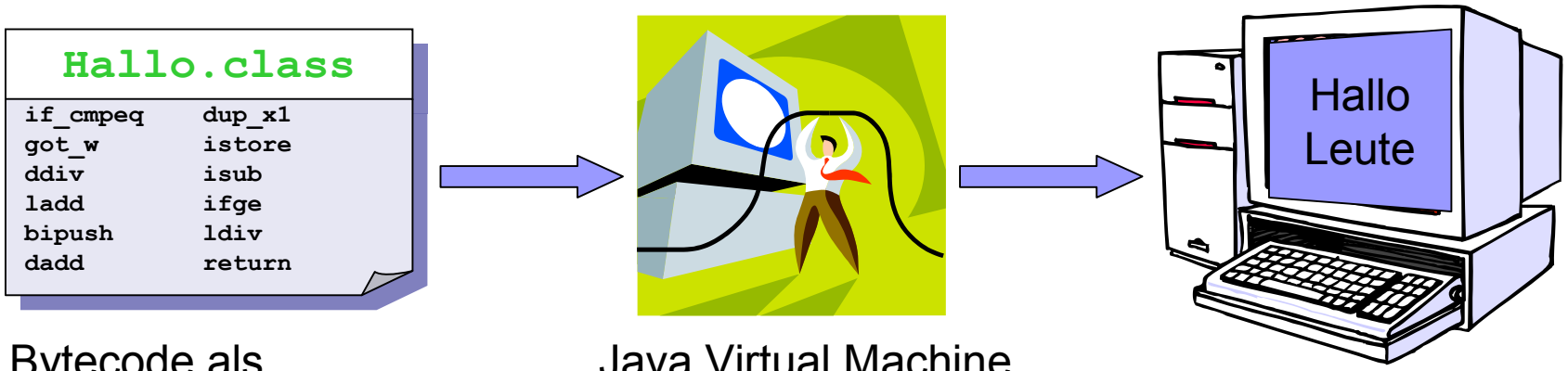
Hallo.class

```
if_cmpeq   dup_x1
got_w      istore
ddiv       isub
ladd       ifge
bipush     ldiv
dadd       return
```

Bytecode als Datei **Hallo.class**

Ausführung

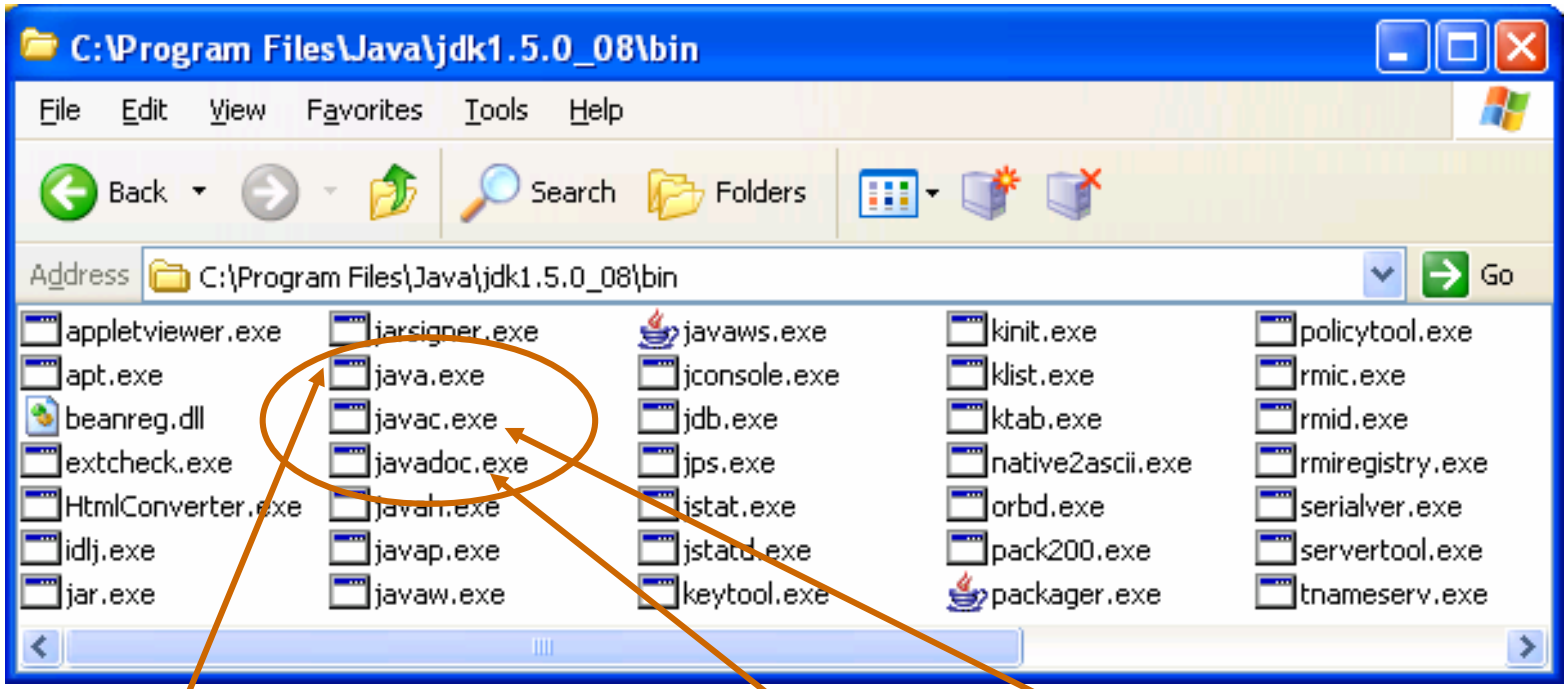
- Die Datei mit dem Bytecode wird der JVM übergeben.
- Der Bytecode könnte auch von einem anderen System, etwa von *JPython*, *JBasic* oder *Smalltalk/JVM* erzeugt stammen.



Bytecode als
Datei `Hallo.class`

Java Virtual Machine
`java.exe`

Die Werkzeuge des SDK



Java Virtual Machine
`java.exe`

Dokumentationstool
`javadoc.exe`

Java-Compiler
`javac.exe`

Wie geht's praktisch unter Windows?

1. Ein Textdatei mit Namen „`Hallo.java`“ erstellen

Hallo.java

```
public class Hallo{
 public static void main(String[] args)
   { System.out.println("Hallo Leute");
   }
}
```

2. Compilieren mit dem Befehl:

   `javac.exe Hallo.java`

   Es entsteht eine Datei

   `Hallo.class`

In der Konsole (DOS-Box/Shell) :

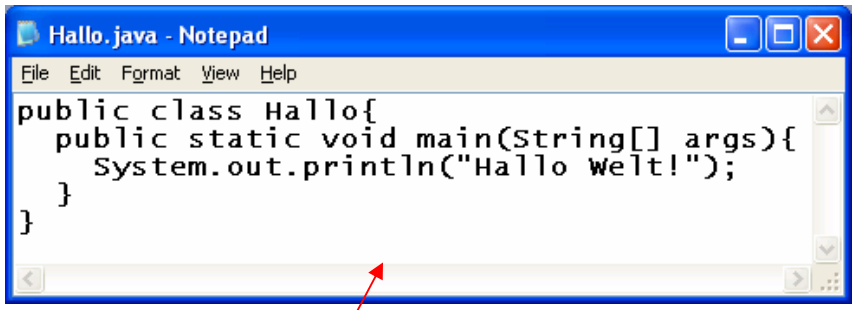
```
C:>   javac Hallo.java
```

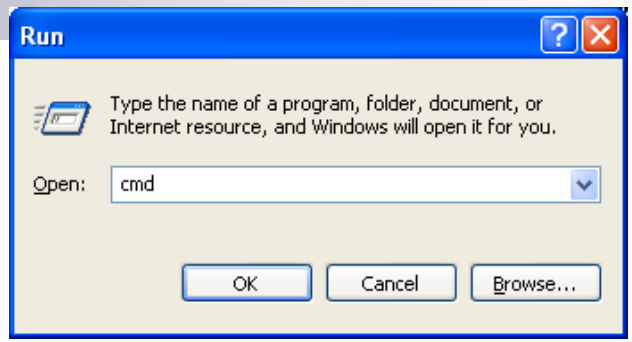
3. Ausführen:

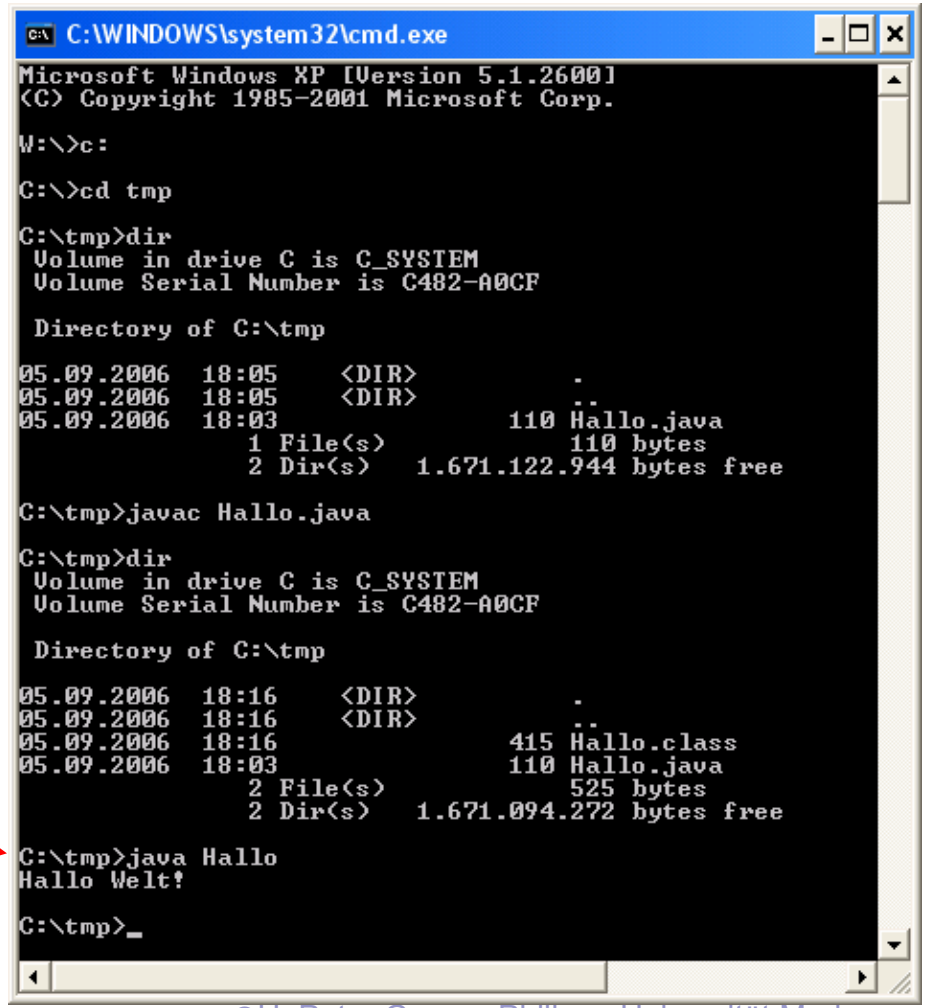
   `java.exe Hallo`

```
C:>   java Hallo
```

In Windows XP:





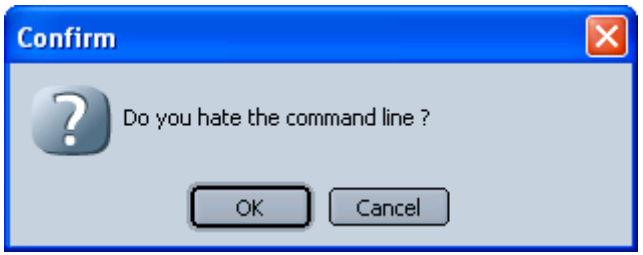


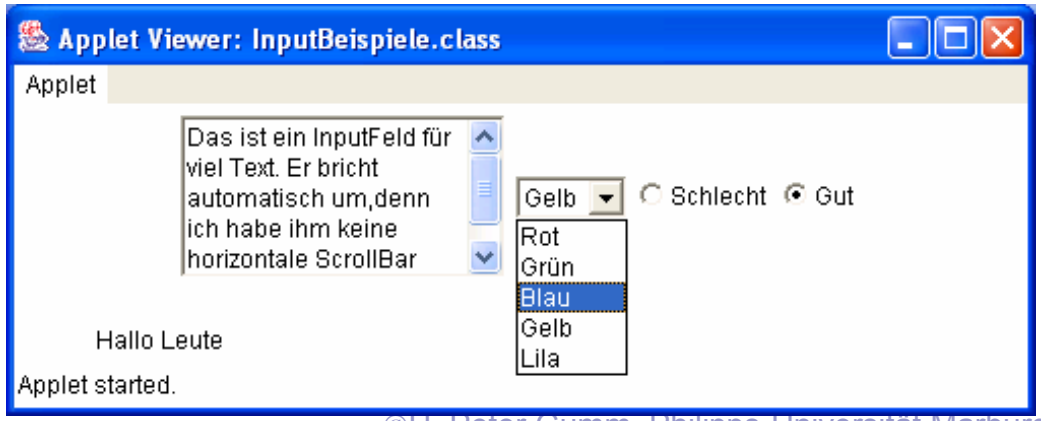
1. Mit beliebigem Texteditor die Datei **`Hallo.java`** erstellen
2. Kommmandointerpreter **`cmd.exe`** aufrufen
3. Compilieren mit **`javac`**
4. Ausführen mit **`java`**
5. Ergebnis bestaunen

Solche Programme sind heute untypisch

- Niemand schreibt mehr auf die Konsole
  - Rückmeldungen kommen in
    - MessageBoxen
    - Statuszeilen
    - Klängen

- Kein Mensch liest mehr von der Konsole.
  - Eingabe kommt aus
    - Bestätigungen
    - Menüs
    - TextFeldern
    - RadioButtons
    - CheckBoxes
    - ListBoxes
    - Slider
    - …





Die Programmerstellung ist vorsintflutlich

- Man muss Kommandos auf der Kommandozeile eintippen
  - Wo ist die ?
  - Sind die Pfade richtig gesetzt ?
- Man wechselt dauernd zwischen Programmaufrufen
  - Editor
  - Compiler
  - Virtuelle Maschine
- Compilermeldungen erscheinen auf der Kommandozeile
  - Error in Line 737
  - ; expected (wo?)
- Testen der Programme (durch Fehlermeldungen) ist umständlich
  - ```
    System.out.println(
    "Das dürfte jetzt nicht sein"
    );
    ```


    ```