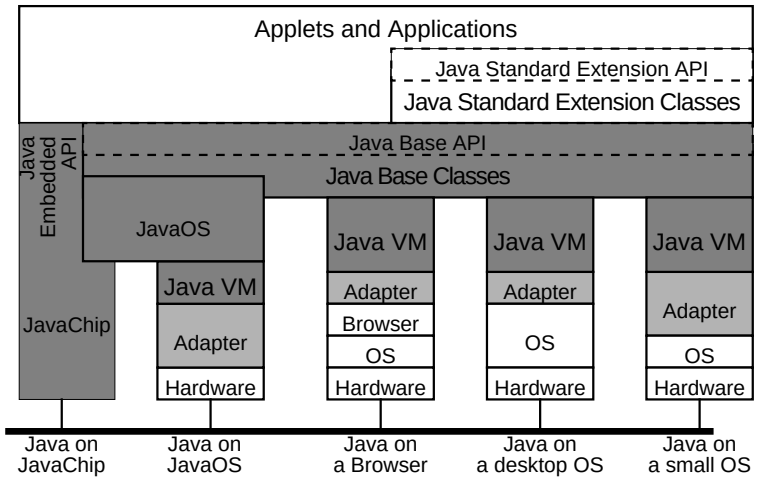
Grundlagen der Informatik für Ingenieure I

2.1 Java - Entwicklungsgeschichte

- Java wurde zu Beginn der 90er Jahre von Sun Microsystems entwickelt, mit dem Ziel, eine einfache, objektorientierte, plattformunabhängige Sprache für sogenannte *“Embedded Systems”* bereitzustellen - in diesem Fall insbesondere für den *Consumer*-Marktbereich.
- Der Einstieg zum Erfolg war die Entwicklung eines Java-Browsers für das Internet (WWW) - ***HotJava***.
  Der endgültige Durchbruch gelang dadurch, daß sich die Fa. Netscape bereit erklärte, in Ihren WWW-Browser die ***Java-Virtuall Maschine (Java-VM)*** zu integrieren. Hierdurch stand die Java-VM auf allen gängigen Rechnerplattformen zur Verfügung; ein erstes Ziel - die Plattformunabhängigkeit - war erreicht.
- Wie bereits erwähnt haben die Eigenschaften “objektorientiert”, “plattformunabhängig” und “sicher” (letztere beiden Eigenschaften realisiert u. a. durch **interpretative Abarbeitung**) auch ihren Performancepreis. Nicht zuletzt deswegen wurde an einem Prozessor gearbeitet, der als Hardwareschnittstelle die Java-VM bereitstellen sollte. Es stellte sich jedoch heraus, daß die ***sun***-eigene "SPARC-Technologie" eher ungeeignet war. Ob es wirklich noch zu einer Hardwarelösung kommen wird ist z. Zt. nicht abzusehen.

2.2 Java Architektur Überblick

- API: Application Programming Interface



2.3 Netzwerk-Computer: Ein neues Betriebsparadigma

- Mit der Einführung von Netzwerk-Computern (Java-Stations) wurde auch ein neues “Betriebsparadigma” für Rechnernetze eingeführt.
- Bisher kennen wir in vernetzten Systemen drei (idealisierte) Modelle:
  - Klassische PC-Netze:
    Jeder PC ist nach den persönlichen Bedürfnissen des Besitzers ausgestattet; zentrale Server stellen zentrale Dienstleistungen, wie zum Beispiel Druckerservice; e-mail oder ähnliches zur Verfügung; manchmal sind auch diese Services lokal organisiert.
    Der Administrationsaufwand ist überproportional zu N (= Anzahl der PC’s), da lokal über Systemsoftwarekonstrukte und ihre Versionen entschieden wird und daher Konfigurationen verschiedener PC’s z. B. hinsichtlich der Netzdienste häufig nicht konform sind.

2.3 Netzwerk-Computer: Ein neues Betriebsparadigma

- Workstationnetze:
  Workstationnetze waren von jeher darauf ausgelegt zentral administriert zu werden; lokal werden in der Regel nur “unveränderliche” Daten, auf den Servern die veränderlichen gehalten. Durch geeignete Mechanismen läßt sich die Konformität der Installationen erzwingen.
  Administrationsaufwand ist proportional zu den Servern oder Clustern bzw. zusätzlich proportional zu verschiedenen Architekturen von Workstations im Netz.
- X-Windows-Terminal-Netze:
  Diese Netze entsprechen den klassischen Zentralrechnerkonfigurationen mit dem Unterschied, daß auf den Terminals eine graphische Oberfläche zur Verfügung steht.
  Der Nachteil gegenüber den oben genannten Lösungen besteht darin, daß der Server auch die Rechenleistung erbringen muß. Alle Benutzer teilen sich also die Rechenleistung eines Rechners und natürlich auch andere Ressourcen, wie z. B. Arbeitsspeicher und Plattenperipherie.
  Administrationsaufwand proportional zur Anzahl der Server.

2.3 Netzwerk-Computer: Ein neues Betriebsparadigma

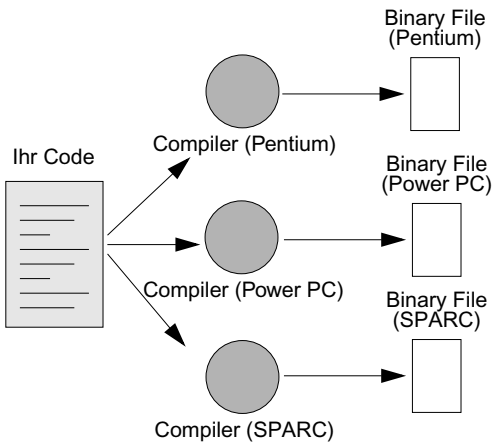
- Neu: Netzwerk-Computer (NC) - Netze:
  Zu dieser Klasse gehören die Java-Stations. Von der Netz-Architektur her entsprechen diese Netze den X-Window-Terminal-Netzen, jedoch werden die auszuführenden Applikationen jeweils zu den Stations “heruntergeladen” und kommen auf diesen zur Ausführung.
  Man kann Applikationen auch so gestalten, daß je nach Leistungsfähigkeit und Last ein Teil der Applikation auf dem NC abläuft und ein Teil auf dem Server. Dieses Konzept vereinigt die Vorteile der verteilten Rechenlast bei den Workstations mit dem geringen Administrationsaufwand bei X-Window-Terminal-Netzen.
  Administrationsaufwand proportional zur Anzahl der Server.
- Neu: Ultra-Thin-Clients - Netze(Sun-Ray):
  Angesichts der immer schnelleren Netzwerke ist man auf den Gedanken gekommen, gegenüber einem X-Terminal auch noch die grafische Aufbereitung der Oberfläche auf den Server zu verlagern. Auf dem - vom Restnetz disjunkten - Verbindungsnetzwerk werden also nur Grafikinformationen ausgetauscht.

2.3 Netzwerk-Computer: Ein neues Betriebsparadigma

- Zusatzbemerkung:
  Heutzutage ist es potentiell möglich, auch PC-Netze aus administrativer Sicht wie Workstationnetze zu betreiben. Leider ist man aber auch in der Lage Workstationnetze wie PCs zu betreiben.

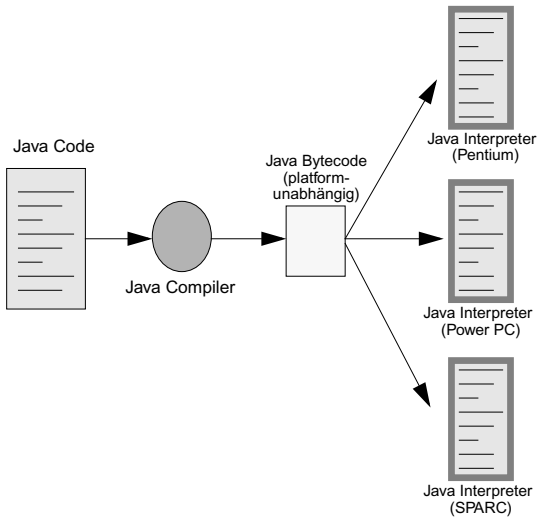
2.4 Java - Eigenschaften

- Java ist plattformunabhängig, realisiert durch die Bereitstellung einer Virtuellen Maschine (Java-VM), die den übersetzten Javacode (sog. ***Bytecode***) interpretativ abarbeitet. Die Java-VM steht auf allen gängigen Rechensystemen zur Verfügung.
  - die traditionelle Methode:



2.4 Java - Eigenschaften

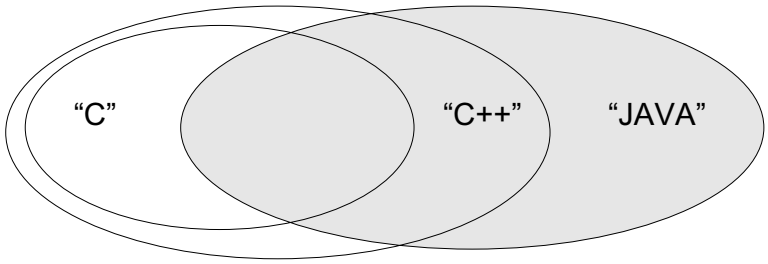
- die Java-VM-Methode



2.4 Java - Eigenschaften

- Java ist objektorientiert

Viele Eigenschaften von Java stammen aus der Programmiersprache C++, eine objektorientierte Erweiterung der Sprache C, welche die Implementierungssprache von UNIX ist. C entstand Anfang der 70er Jahre als hardwarenahe Systemprogrammiersprache.
Darüberhinaus wurden weitere Konzepte von existierenden objektorientierten Sprachen in Java übernommen; hervorzuheben ist, daß die Objektorientierung erzwungen wird. Sie ist nicht quasi optional wie bei C++.



2.4 Java - Eigenschaften

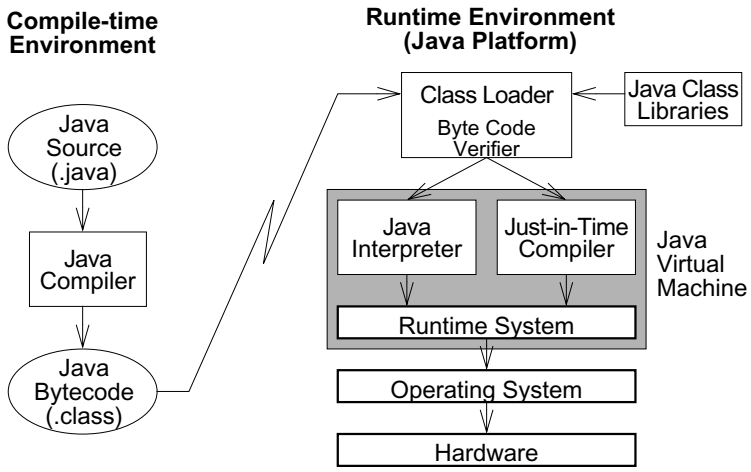
- Java ist sicher, einfach und robust
  Auf die Sicherheitsaspekte werden wir in einem der späteren Kapitel eingehen; ob die beiden anderen Aussagen zutreffen, werden Sie im Laufe des Semesters selber beurteilen können.

2.5 Java - Entwicklungsumgebung

- Als Entwicklungsumgebung verwenden wir Sun's JDK (***Java Development Kit***), welches u. a. alle notwendigen Klassen- (Programm-) Bibliotheken enthält, den Compiler, die Java-VM zum Ablauf von Java-Applikationen[1] und einen ***Java-Viewer*** zum Test von Java-Applets.
- Wir verwenden den ***Browser "netscape"*** für Java-Applets, die in ***HTML***-Seiten eingebettet sind.
- Last but not Least verwenden wir Rechnersysteme unter dem Unix-Betriebssystem. Soweit notwendig werden wir im Anschluß an dieses Kapitel und in der 1.Übung alle für uns wichtige Eigenschaften des Systems kennenlernen, insbesondere den Umgang mit dem Dateisystem und den Texteditoren.

1. alle Ihnen unbekannte Termini werden in den nächsten Abschnitten erläutert.

2.5 Java - Entwicklungsumgebung



2.6 Application vs. Applet

- Eine Java-Applikation ist ein Programm, daß keinerlei Unterstützung durch Webbrowser benötigt. Es läuft in einer eigenen Laufzeitumgebung ab, wie Programme anderer Sprachen auch, z. B. in C oder Fortran.
  In der Regel wird der Bytecode von der Java-VM interpretativ abgearbeitet. Um den Perfomanceverlust in Grenzen zu halten hat man jedoch ***Just-in-Time*** - Compiler ***(JIT)*** entwickelt, die zur Laufzeit den Bytecode in Maschinencode übersetzen, sodass beim erneuten Durchlauf dieses Codes der Code in Machinensprache vorliegt. Das bringt Leistungssteigerungen insbesondere beim Abarbeiten von Programmschleifen.
- Ein Java-Applet ist ein Programm, dessen Aufruf in einer ***HTML*** - Seite eingebettet ist. Dieses Programm benötigt zu seinem Ablauf einen ***WWW-Browser*** mit integrierter Java-VM, wie z. B. *netscape*.

2.7 Ein erstes Programm

- Eine erste Java-Applikation
  wird mit Hilfe eines Editors eingegeben. Der Quellcode ***(source)*** wird in eine Datei mit Namen *classname*.java gespeichert, in diesem Fall also: “xemacs HelloWorld.java”.
  Der Quellcode:

```
/* Hello World, the first Java application */
class HelloWorld {

    public static void main (String args[]) {
      System.out.println("Hello World!");
    }
}
```

- Compilieren des Quellcodes; Starten der Application
  (Die Dateien liegen im *current directory*!)

```
javac HelloWorld.java
java HelloWorld
```


2.7 Ein erstes Programm

- Der Compiler hinterlegt den übersetzten ***Source-Code***, also den ***Bytecode***, in eine Datei mit Namen *classname*.class. Diese Datei wird vom Java-Interpreter interpretiert. Bei der Angabe des Dateinamens als Parameter wird die Extension *“class”* jedoch nicht mit angegeben!
  - Sollten Sie Fehlermeldungen bekommen, kontrollieren Sie Namen der Klasse und Namen der Datei. Beide müssen exakt übereinstimmen, auch bezüglich der Groß- und Kleinschreibung!
  - Sollte das einwandfrei sein, sind Ihnen möglicherweise Tippfehler im Quellcode unterlaufen; schauen Sie sich den Quellcode noch einmal genau an, insbesondere hinsichtlich der Sonderzeichen “{“, “}”, “;”.

2.7 Ein erstes Programm

- Ein erstes Java-Applet
  wird mit Hilfe eines Editors eingegeben. Der Quellcode (source) wird in eine Datei mit Namen *classname*.java gespeichert, in diesem Fall also: “xemacs HelloWorldApplet.java”.
  Der Quellcode:

```
/* First Hello World Applet */
import java.awt.Graphics;

public class HelloWorldApplet extends java.applet.Applet{

    public void paint(Graphics g) {
      g.drawString("Hello world!", 5, 25);
    }
}
```

- Compilieren des Quellcodes
  (Die Dateien liegen im *current directory*!)

```
javac HelloWorldApplet.java
```


2.7 Ein erstes Programm

- Starten des Java-Applets
  Wie bereits erwähnt wird ein Java-Applet in eine Webseite integriert und aus dieser heraus durch den Browser, in unserem Fall *“netscape”*, gestartet.

- Ein erstes HTML-Script
  wird mit Hilfe eines Editors eingegeben. Der Quellcode wird (sinnvoller Weise) in eine Datei mit Namen *classname*.html gespeichert, in diesem Fall also: “xemacs HelloWorldApplet.html”.
  Der Quellcode:

```
<HTML>
<HEAD>
<TITLE>Hello to Everyone!</TITLE>
</HEAD>
<BODY>
<P>My Java Applet says:
<APPLET CODE="HelloWorldApplet.class" WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

2.7 Ein erstes Programm

- Starten des Browsers

  `netscape`

- Starten des Java-Applets
  - Wähle aus Menü *File "Open Page"*
  - In dem Pop-Up-Menü klicke *"choose file"* an. Durch Anklicken der *directory*-Namen werden Sie in das *directory* gelangen, in dem Ihre *html*-Datei liegt. Dann diese Datei anklicken und *"OK"* klicken.
  - Als letztes *"Open in Navigator"* anklicken; das Applet wird ausgeführt!
- Will man nur das Applet selbst testen, genügt auch der im *jdk* vorhandene *appletviewer.*

  `appletviewer HelloWorldApplet.html`

  (der ***HTML***-Text wird vom ***appletviewer,*** bis auf die für das Applet relevanten Zeilen, ignoriert!)

2.7 Ein erstes Programm

- Ein erstes Java-Applet (cont)
  - Sollten Sie Fehlermeldungen bekommen, kontrollieren Sie Namen der Klasse und Namen der Datei. Beide müssen exakt übereinstimmen, auch bezüglich der Groß- und Kleinschreibung!
  - Sollte das einwandfrei sein, sind Ihnen möglicherweise Tippfehler im Quellcode unterlaufen; schauen Sie sich den Quellcode noch einmal genau an, insbesondere hinsichtlich der Sonderzeichen "{","}",";".