Ein Glossar wichtiger Begriffe zur Java-Technologie

Denis Antonioli, Stefan Berner, Clemens H. Cap, Markus Pilz,
Lutz Richter, Mathias Richter, Takashi Suezawa

Institut für Informatik der Universität Zürich

September 1996

Abstract Window Toolkit:

Es handelt sich dabei um den Teil des → *API*, der alle Klassen zur Implementation von graphischen Benutzeroberflächen (→ *GUI*) beinhaltet. Abstrakt ist das AWT, weil die zur Verfügung gestellten Klassen unabhängig von ihrer jeweiligen Implementation auf einer bestimmten Plattform (z.B. Macintosh, Windows, usw.) sind.

API:

→ *Application Programming Interface*

Applet:

Name für Java-Programme, die in HTML-Seiten eingebunden, über das Netz verschickt und in einem javafähigen → *WWW-Browser* wie z.B. → *Netscape* oder → *HotJava* ausgeführt werden können. Applets werden als Erweiterung der Klasse Applet programmiert, indem Methoden wie init(), start(), stop(), paint() und Methoden zum "event handling" überschrieben werden. Diese Methoden werden bei Bedarf durch den WWW-Browser aufgerufen. Aus Sicherheitsgründen ist die Funktionalität von Applets gegenüber Java Applikationen eingeschränkt. Welche Funktionen im konkreten Fall zur Verfgung stehen, hängt vom Browser und der gewählten Sicherheitsstrategie ab. Typischerweise können Applets aber keine Dateien lesen oder schreiben und nur Netzwerkverbindungen zu dem Host aufbauen, von dem sie heruntergeladen wurden.

AppletViewer:

Programm zur lokalen Ausführung von → *Applets* ohne einen → *Web-Browser*. Wird meistens zu Testzwecken bei der Entwicklung von Applets verwendet.

Application Programming Interface:

Im allgemeinen handelt es sich um die Schnittstelle von Prozeduraufrufen und Datenstrukturen eines Soft- oder Hardware-Elementes, die Programmierern zur Verfügung steht, um Anwendungen auf Basis dieses Elementes zu erstellen. Im Falle von Java handelt es sich beim API um einen Sammelbegriff für alle Java-Klassen, die für die Programmierung von Anwendungen zur Verfügung

stehen. Unterschieden werden → *Core API* und → *Standard Extension API*.

AWT:

→ *Abstract Window Toolkit*

Binding:

Vorgang, durch den ein Dienst-Benutzer (→ *Client*) in Verbindung mit einem Anbieter (→ *Server*) eines benötigten Dienstes gebracht wird. Auf den unteren Protokoll-Schichten beinhaltet dies die Auswahl eines bestimmten Dienst-Anbieters und die Information über die physikalische Komponente, auf welche dieser verteilt wurde.

Bridge Mechanismus:

Bei portablen Programmen soll die Benutzerschnittstelle nicht auf allen Rechnern gleich aussehen, sondern sich in ihren Eigenschaften an dem auf dem jeweiligen Rechnertyp etablierten Standard orientieren. Dazu wird die Realisierung von Buttons, Fenstern und anderen Interaktionselementen einem jeweils plattformabhängigen "Peer" überlassen und auf eine plattformunabhängige Realisierung verzichtet.

Byte Code:

Codierung eines Instruktionssatzes, bei dem die meisten Instruktionen genau ein Byte lang sind. Da acht Bits nicht ausreichen, neben dem Opcode auch noch die Operanden zu codieren, impliziert der Byte Code faktisch eine Stackmaschine, bei der die Operanden implizit durch den Stack gegeben sind. Die nur ein Byte langen Instruktionen führen zu kleinen Programmen, auch wenn Stackmaschinen mehr Instruktionen benötigen als 2- oder 3- Adressmaschinen. Kleine Programme sind für Java essentiell, da kleinere Programme schneller über ein langsames Netz geladen werden können. Der Byte Code der → *Java Virtual Machine* ist allerdings nicht besonders kompakt: er enthält neben der Instruktion auch noch Typinformation, die vom Byte Code Verifier benötigt wird, und viele Instruktionen sind 2 oder 3 Byte lang, weil Operanden oder Konstanten im zweiten und dritten Byte codiert werden.

Cafe:

Integrierte → *Entwicklungsumgebung* von Symantec. Besteht aus einem Projektmanager, → *Klassenbrowser*, Java-Compiler, Debugger und einem → *AppletViewer*.

Class Browser:

Werkzeug zur strukturierten Darstellung des Quellcodes einer Klassendefinition. Hypertext-Elemente erlauben die Navigation zwischen den verschiedenen Elementen einer Klassendefinition bzw. direkt zwischen verschiedenen Klassendefinitionen.

Client:

Logische Komponente einer Anwendung, die einen Dienst anfordert (→ *Client-Server Konzept*).

Client-Server Konzept:

Architektur, die eine Anwendung in Dienstanbieter (→ *Client*) und Dienstbenutzer (→ *Server*) strukturiert.

Code Shipping:

Code Shipping bedeutet, dass Programmcode in einem heterogenen Netzwerk verschickt und auf der Zielmaschine "sofort" ausgeführt wird. Code Shipping impliziert einen neuen Portabilitätsbegriff, da etwelche Portierungsaufgaben zur Laufzeit und für den Benutzer transparent durch das System ausgeführt werden. Diese Portabilität wird erreicht, indem die Programme nicht direkt auf die Hardware und das Betriebssystem zugreifen. Die Programme laufen vielmehr in einer virtuellen und standardisierten Umgebung (Hardware und Betriebssystem) ab. Bei Java wird diese virtuelle Umgebung → *JavaPlattform* genannt. Die Zielmaschine übernimmt die Abbildung der virtuellen Umgebung auf real existierende Hardware und Systemsoftware. Die technische Herausforderung besteht beim Code Shipping darin, die virtuelle Laufzeitumgebung so zu definieren, dass sie effizient und mit tragbarem Leistungsverlust auf alle existierende Hardware und alle gängigen Betriebssysteme abgebildet werden kann. Code Shipping ist zur Zeit noch ein aktives Forschungsgebiet. Konsequenz von Code Shipping sind wirklich offene Systeme, da Software-Entwickler ihre Applikationen unabhängig von der konkreten Hardware und dem konkreten Betriebssystem schreiben können, während die Anwender genauso unabhängig einkaufen können. Daneben hat Code Shipping grosse, noch nicht völlig abschätzbare Auswirkungen auf verteilte Rechnerumgebungen, da nicht nur Daten, sondern auch (die sie bearbeitenden) Programme über das Netz geschickt werden können.

CodeWarrior:

→ *Entwicklungsumgebung* von Metrowerks für C++ und Java.

CORBA:

Corba steht für *Common Object Request Broker Architecture*. Corba ist ein händlerunabhängiger Standard welcher Mechanismen definiert, mit deren Hilfe Objekte in einem verteilten System Nachrichten austauschen können. Die OMG (*Object Management Group*) ist für die Entwicklung von Corba zuständig. Die Firmen DEC, HP, HyperDesk, NCR, Object Design und Sun unterstützen Corba aktiv. Zur Zeit sind es bereits über 600 Firmen, welche Mitglied der OMG sind. Ziel von Corba ist es, auf einfache Art und Weise die Entwicklung von Anwendungen in heterogenen verteilten Systemen zu unterstützen. Hersteller sind frei, eigene Corba-Implementierungen zu entwickeln und zu vermarkten. Corba sieht die Möglichkeit vor, Anwendungen welche auf Corba-Implementierungen unterschiedlicher Hersteller beruhen, zu vernetzen. Die OMG bietet einen Corba-IDL-Compiler (→ *Interface Definition Language*) kostenlos an, um die Verbreitung des Standards zu beschleunigen. Corba-kompatible Produkte werden u.a. von DEC, Iona, SuiteSoftware, Sun, HP, HyperDesk und IBM vertrieben.

Core API:

Das Core → *API* ist integraler Bestandteil der Java-Implementation auf jeder Plattform. Der Umfang des Core API wird laufend vergrössert. Für Java-Applikationen, die ausschliesslich das Core API verwenden, ist die Lauffähigkeit auf allen Java-Plattformen jetzt und in Zukunft gewährleistet (→ *Standard Extension API*).

Datagram :

In sich geschlossene Informationseinheit, welche über verbindungslose Protokolle wie (→ *IP*) oder (→ *UDP*) versandt wird. Im Unterschied zu Netzwerk-Paketen sind Datagramme von variabler Grösse.

Design Patterns:

Design- und Programmier-Erfahrungen sind wichtig, um robuste, (verteilte) Anwendungen innert nützlicher Frist zu realisieren. Es hat sich gezeigt, dass es in der objektorientierten Programmierung Idiome und Klassenstrukturen gibt, welche allgemein eingesetzt werden können und dazu verhelfen, ein Systemdesign flexibler zu gestalten. Ein Beispiel eines wohlbekannten solchen Idiomes ist z.B. das Model-View-Controller- Konzept in Smalltalk. Um Design-Erfahrungen zu fixieren, zu dokumentieren und weiterzugeben haben sich Design Patterns etabliert. Wir gehen davon aus, dass eine Bibliothek von probaten Design Patterns für Design und Implementierung komplexer verteilter Systeme nützlich wäre. Diesbezügliche Bestrebungen sind unseres Wissens aber noch nicht im Gange.

Entwicklungsumgebung:

Eine Entwicklungsumgebung besteht aus einer Reihe von → *Programmierwerkzeugen*, deren Funktionen den Entwicklungszyklus (Editieren, Übersetzen, Testen, Fehlerkorrektur) teilweise oder ganz abdecken. Je nach Kopplung der einzelnen Werkzeuge (z.B. unter einer einheitlichen Benutzeroberfläche) spricht man von einer integrierten Entwicklungsumgebung.

Exception:

Eine Exception signalisiert das während der Programmausführung unerwartete Auftreten eines Ereignisses oder einer Ausnahmebedingung. Programmiersprachen, die Exceptions unterstützen, stellen Konstrukte zur Verfügung, um auf solche Ausnahmebedingungen zu reagieren. In Java sind dies Unterklassen der Klasse "Exception" oder der Klasse "Error", welche wiederum Unterklassen der Klasse Throwable sind. Exceptions können in Java mit dem Schlüsselwort `throw` ausgeloest (engl. "to raise/throw an exception") und mit dem Schlüsselwort `catch` behandelt (engl. "to receive/catch an exception") werden.

Explorer:

Explorer ist ein Internet- bzw. → *WWW-Browser*, der von Microsoft entwickelt und angeboten wird.

Garbage Collection:

Der Prozess, der Speicher von Objekten, die nicht mehr gebraucht werden, wieder freigibt. Ein Objekt wird nicht mehr gebraucht, wenn keine Referenzen mehr darauf existieren, d.h. wenn keine anderen Objekte oder Variablen mehr darauf verweisen.

GC:

→ *Garbage Collection*

Graphical User Interface:

Graphische Benutzeroberfläche (z.B. Macintosh, Windows oder Motif).

GUI:

→ *Graphical User Interface*

Heterogeneity:

Im Bereich verteilter Rechnersysteme bezeichnet man ein Rechnernetz als heterogen, wenn die Rechner im Netz verschieden sind. Die Unterschiede reichen von relativ geringfügigen, wie unterschiedliche Leistung bei Binärkompatibilität, bis hin zu unterschiedlichen Architekturen mit verschiedenen Datenrepräsentationen, die bei der Kommunikation eine Konvertierung erfordern, unterschiedlichen Betriebssytemen, Netzwerk-Interfaces, usw. (→ *Homogeneity*).

Hierarchie-Browser:

→ *Hierarchy Browser*

Hierarchy Browser:

Werkzeug zur Darstellung der Vererbungshierarchie einer Sammlung von Klassen. Interaktion mit dem → *Class Browser* zur Darstellung der entsprechenden Klassendefinitionen.

Homogeneity:

Im Bereich verteilter Rechnersysteme bezeichnet man ein Rechnernetz als homogen, wenn alle Rechner im Netz gleich sind (→ *Heterogeneity*).

HotJava:

Erster javafähiger → *Web-Browser* von SUN, der selbst in Java geschrieben ist (bis heute nur in einer Betaversion erhältlich). Der HotJava Browser sollte den Leuten die Möglichkeiten von Java plastisch vor Augen führen; was im sehr gut gelungen ist - HotJava installieren; sehen, was mit Applets alles möglich ist; selbst solche Applets schreiben - war für viele der Zugang zu Java.

IDE:

→ *Integrated Development Environment*

IDL:

→ *Interface Definition Language*

Integrated Development Environment:

Integrierte → *Entwicklungsumgebung*.

Interface Definition Language:

Eine Interface Definition Language (IDL) dient dazu, die Schnittstelle eines Dienstes eines verteilten Systems, z.B. eines → *Nameservers*, zu deklarieren. Bestandteile dieser Deklaration sind die RPCs (→ *Remote Procedure Calling*) welche der Dienst exportiert, die Parameter zu den einzelnen RPCs, Angaben für die Fehlerbehandlung (→ *Exceptions*), Vererbungs-Beziehungen usw. Mittels einer IDL wird ein deklarativer Programmierstil ermöglicht, bei dem die Details einer verteilten Anwendung vorerst ausser Acht gelassen werden können und lediglich die Komponenten der Anwendungen im Zentrum stehen. Ein Stub-Generator kann aus einer vorgegebenen IDL-Spezifikation den gesamten → *Marshaling*-Programmcode automatisch generieren. Beispiel ist die IDL von Corba (→ *CORBA*).

Interface-Klasse:

Unter einem Interface versteht man die Signaturen aller Methoden einer Klasse. Ein Interface legt die Schnittstelle zu einer Klasse fest, und nicht deren Implementation. In Java werden Interface-Klassen wie "normale" Klassen über das Schlüsselwort `interface` deklariert; allerdings können Interface-Klassen keine Instanzvariablen und nur abstrakte Methoden, also Methoden ohne Implementation, enthalten. Klassen können über das Schlüsselwort `implements` angeben, dass sie ein bestimmtes Interface unterstützen. In diesem Fall müssen sie jede in der Interface-Klasse enthaltene abstrakte Methode implementieren. Eine Java-Klasse kann beliebig viele Interfaces unterstützen. Interface Klassen sind eine eleganter Weg zur Mehrfachvererbung: eine Klasse, die ein Interface implementiert, erbt nicht nur von den direkten Oberklassen, sondern auch von den Oberklassen der Interface-Klasse.

Internet versus Intranet:

Das Internet ist ein weltweiter Zusammenschluss von Rechnern zum Austausch von Daten. Es hat sich ursprünglich aus einer Initiative des amerikanischen Verteidigungsministeriums entwickelt, war bis 1991 von militärischen und akademischen Interessen dominiert und ist seit 1993 ein mehrheitlich kommerziell orientiertes Netzwerk. Ein Intranet nutzt die Technologie des Internets, ist aber meist auf den Bereich einer Firma beschränkt. Ein Intranet kann durch spezielle Zugangsrechner an das Internet angeschlossen werden. Generell kann ein Intranet sicherer und schneller als das Internet realisiert sein, es ist dann aber auf die intern angebotenen Dienste beschränkt.

Internet Protocol:

Dieses Protokoll definiert die Kommunikation auf der Internet-Netzwerkebene zum Austausch von

→ *Datagrammen*, der grundlegenden Informationseinheit, welche über das Internet verschickt wird.

IP:

→ *Internet Protocol*

Java 1.0:

Erster stabiler Release der Programmiersprache Java, der Java → *Virtual Machine* und des Java → *API*. Gegenüber Java 1.0 (Alpha) und Java 1.0 (Beta) sind einige Änderungen am API vorgenommen worden. Aufgrund von Sicherheitslücken sind einige "bug fixes" schon sehr kurz nach dem erscheinen von Java 1.0 nötig geworden. Die derzeit aktuelle Version ist 1.0.2. Aus Sicherheitsgründen sollte nur diese Version eingesetzt werden. Für Oktober 1996 ist eine neue Version 1.1 mit wesentlichen Erweiterungen des API angekündigt.

javac:

Java Compiler des → *JDK*.

javadoc:

javadoc als Bestandteil des → *JDK* extrahiert speziell formatierte Kommentare aus einer Java-Klassendefinition und erstellt daraus eine Hypertext-Dokumentation der entsprechenden Klasse, die mit jedem → *Web-Browser* dargestellt werden kann.

javah:

Werkzeug des → *JDK* zur Erstellung der für die Implementation von nativen Methoden einer Klasse in C notwendigen Dateien (.h und .c).

javap:

Werkzeug des → *JDK* zur Darstellung von Java → *Byte Code* als Mnemonics, d.h. ein Byte Code-Disassembler.

Java Beans API:

→ *API* für die Software-Komponentenarchitektur von Sun. Mit Java Beans entwickelte Komponenten sind kompatibel zu anderen Komponenten-Architekturen wie z.B. → *ActiveX* von Microsoft, OpenDoc oder LiveConnect von → *Netscape*. Wird in Zukunft Teil des → *Core API* sein.

Java Commerce API:

→ *API* für sichere Kauf-, Verkaufs- und Finanztranskationen im → *WWW*. Wird in Zukunft Teil des → *Core API* sein.

Java Database Connectivity API:

Standard SQL-Schnittstelle für den einheitlichen Zugriff auf ein breites Spektrum relationaler Datenbanken. Wird in Zukunft Teil des → *Core API* sein.

Java Development Kit:

Erste Java-Programmierumgebung. Das JDK von SUN ist faktisch die Referenzimplementierung von Java und wurde von vielen Hard- und Software-Herstellern lizenziert. Das JDK ist eine Sammlung von Programmen, mit denen eigene → *Applets* und Applikationen erstellt werden können. Er enthält: → *Appletviewer*, java (→ *Java Interpreter*), → *javac* (Java Compiler), → *javadoc* (documentation generator), → *javah* (native method C file generator), → *javap* (Java Disassembler), jdb (Java Debugger).

Java Embedded API:

Das Java Embedded API ist der Teil des → *API*, den Plattformen zur Verfügung stellen werden, die aufgrund ihrer Leistungsfähigkeit nicht das gesamte API anbieten können. Die Spezifikation des Embedded API ist noch nicht veröffentlicht.

Java IDL API:

API zur Unterstützung der → *Interface Definition Language* (IDL) der Object Management Group (OMG). IDL erlaubt die sprachneutrale Spezifikation von Schnittstellen zwischen Objekten und deren Klienten auf verschiedenen Plattformen. Wird in Zukunft Teil des → *Core API* sein.

Java Interpreter:

Der Java Interpreter implementiert die virtuelle Maschine, die den → *Bytecode* von Java ausführt. Der Java Interpreter ist Bestandteil des → *JDK*. Oft beinhalten Entwicklungsumgebungen wie → *Cafe* oder → *Roaster* eigene Implementationen des Java Interpreters.

Java Management API:

→ *API* für Implementation von Applikationen und → *Applets* zur Verwaltung und Überwachung von Unternehmensnetzwerken über das Internet. Wird in Zukunft Teil des → *Standard Extension API* sein.

Java Media API:

→ *API* für die Verwendung einer breiten Palette von Multimedia-Technologien in → *Applets* und Applikationen. Das Java Media API setzt sich zusammen aus mehreren APIs für Bereiche wie 2D-Graphik, 3D-Graphik, Audio, Video, Animation, usw. Teile des Java Media API werden in das → *Core API*, andere Teile in das → *Standard Extension API* aufgenommen.

JavaOS:

Ein kleines Betriebssystem, welches die JavaPlattform zum ausführen von Java Applets und Java Applikationen bereitstellt und zum grössten Teil selbst in Java geschrieben ist. Das JavaOS enthält eine Java → *Virtual Machine*, das → *API* bzw. → *Java Embedded API* (je nach Hardware-Plattform) und die grundlegende Funktionalität für Fenster-System, Netzwerk und die benötigten Hardware-Treiber. JavaOS soll in Netzwerk-Computern, in Geräten der Heimelektronik, Personal Digital Assistants und anderen vernetzten, "embeded devices" wie Drucker, Fotokopierer etc. eingesetzt werden. JavaOS wird aus dem ROM geladen, da solche Geräte schnell booten und ohne explizite Installation sowie ohne Systemadministration laufen müssen. JavaOS soll auf eine ganze

Reihe von Microprozessoren portiert werden, auch auf die Java Chip-Familie. Wenn JavaOS auf einem Microprozessor der Java Chip-Familie läuft, so wird die in der Hardware implementierte virtuelle Maschine benutzt.

JavaPlattform:

Die JavaPlattform garantiert die Ausführbarkeit und die Portabilität von Java → *Applets* und Applikationen. Java-Programme können auf allen Maschinen ausgeführt werden, welche die Java-Plattform implementieren. Die JavaPlattform besteht aus der Java → *Virtual Machine* und der Java Klassenbibliothek (→ *API*). Über die virtuelle Maschine wird von der konkreten Hardware abstrahiert, über die Klassenbibliothek vom Betriebssystem.

JavaSoft:

JavaSoft ist eine Tochterfirma von Sun, die mit der Entwicklung und Vermarktung von Produkten und Dienstleistungen rund um Java betraut ist.

Java Virtual Machine:

→ *Virtuelle Maschine* für die Ausführung von Java-Programmen.

JavaWorkshop:

Integrierte Java-Entwicklungsumgebung von SunSoft. Besteht aus einem Projektmanager, → *Klassenbrowser*, Java-Compiler, Debugger und einem → *AppletViewer*.

JDBC:

→ *Java Database Connectivity*

JDK:

→ *Java Development Kit*

Jeeves:

→ *Standard Extension API* zur Entwicklung von Java-basierenden Internet- und Intranet-Servern.

Just in Time Compiler (JIT):

Eine Technologie bei der Java Bytecode nicht interpretiert wird sondern kurz vor der Ausführung in Assemblercode des jeweiligen Prozessors compiliert wird. JIT Compiler gestatten meist eine effizientere Programmausführung.

JWS:

→ *JavaWorkshop*

Kaffe:

Ein public domain → *Just in Time* Compiler. Siehe auch http://www.sarc.city.ac.uk/ tim/kaffe.

Klassenbrowser:

→ *Class Browser*

Leichtgewichtiger Prozess:

→ *Thread*

Marshaling:

Unter Marshaling versteht man das Übertragen der Parameter eines RPCs (→ *Remote Procedure Calling*) in einen Buffer, welcher dann anschliessend zum Empfänger des RPCs übermittelt werden kann. Die Übermittlung des Buffers geschieht mittels low-level Kommunikationsprimitiven wie `send()/receive()` und ist für den RPC-Programmierer nicht ersichtlich. Beim Marshaling sind auch Probleme der Datenrepräsentation zu lösen, wenn Sender und Empfänger unterschiedliche Datenformate vorsehen (z.Bsp. ASCII vs. EBCDIC, little endian vs. big endian usw.). Beim Empfänger findet ein komplementärer Vorgang statt, bei dem die Daten im erhaltenen Buffer für den Aufruf der effektiven Prozedur aufbereitet werden (Unmarshaling). In der Regel wird der für einen RPC benötigte Marshaling-Programmcode automatisch aus der Spezifikation (→ *Interface Definition Language*) des RPCs von einem Stub-Generator erzeugt.

Multithreading:

→ *Thread*

Nameserver:

Dienst in einem verteilten System, welcher symbolische Namen von Objekten des Systemes in deren physikalische Adressen umwandelt und umgekehrt. Ziele des Nameserving in verteilten Systemen sind Realisierung eines globalen Namensraumes, Transparenz, Vereinfachung des Zugriffs auf Objekte und gemeinsame Benutzung von Objekten. Ein verteiltes System sieht normalerweise mehrere Nameserver vor. Jeder dieser Server verwaltet seinen bestimmten Namensbereich (administrative domain). Anfragen, die sich auf Objekte ausserhalb seines Bereiches beziehen, leitet der Nameserver dann an den Nameserver des entsprechenden Bereiches weiter. Beispielsweise erhält ein Server als Auftrag, den Namen `swissair.ch:/pub/flugplan7` aufzulösen. Der Auftrag wird an den Nameserver des Bereiches `swissair.ch` weitergeleitet, welcher als Antwort beispielsweise eine Adresse der Form `130.77.48.65,9047,17` zurückgeben wird.

NC:

→ *Network Computer*

Netscape:

Software-Hersteller von Netzwerk-Produkten rund um das Internet und → *WWW*. Bekannt geworden ist Netscape durch seinen → *WWW-Browser*, den → *Netscape Navigator*. Mittlerweile bietet

Netscape aber auch WWW-Server-Software und WWW-Authoring-Software (u.a.) an.

Netscape Navigator:

Sehr weit verbreiteter → *Web-Browser* von → *Netscape*.

Netscape ONE:

Das Netscape Open Network Environment definiert einen plattformunabhaïgigen, offenen Standard für Netzwerkapplikationen.

Network Computer (NC):

Ein Computer mit einem Prozessor, auf welchem unmittelbar die Befehle der → *Java Virtual Machine* ablaufen können. Der NC ist primär eine Art Internet Terminal, kann vom Netz aber auch verschiedene Anwendungsprogramme herunterladen. Es wird spekuliert, in wie weit der NC den PC ablösen wird.

Object Serialization:

Umwandlung von Objekten in formatierte Byteströme, die dann z.B. im Sekundärspeicher abgelegt oder über Netzwerke versendet werden können. Aus dem entsprechenden Bytestrom kann jederzeit wieder ein identisches Objekt rekonstruiert werden, wobei auch Abhängigkeiten zwischen ganzen Konglomeraten von Objekten erhalten bleiben.

Objektorientierte Programmierung:

Objektorientierte Programmierung (OOP) unterstützt Abstraktion, Kapselung, Polymorphismus und Vererbung. *Abstraktion* bedeutet, dass ein System modelliert werden kann ohne dessen Details von Anfang an berücksichtigen zu müssen. OOP erlaubt eine deklarative Programmierweise, in der vor allem das "was", also die Eigenschaften des Systems im Vordergrund stehen. Bei anderen Programmiermethoden, z.Bsp. bei der imperativen oder prozeduralen Programmierung, steht meist das "wie", also die innere Funktionsweise des Systems im Vordergrund. *Kapselung* bedeutet, dass Daten und die zugehörigen Funktionen zu sauber abgegrenzten Objekten zusammengefügt werden können. Dank Kapselung kann die Implementierung eines Objektes von externen Aspekten abgegrenzt werden. Dadurch kann Software übersichtlicher gestaltet werden und die Wartbarkeit des Systems wird verbessert. *Polymorphismus* (dynamic binding) bedeutet, dass wir dieselbe Nachricht an unterschiedliche Objekte senden können und diese typspezifisch darauf reagieren. Beispielsweise kann dieselbe Nachricht `draw` an ein Kreis-Objekt und an ein Viereck-Objekt gesandt werden. Das Objekt "weiss" dabei selber wie es sich auf dem Bildschirm darstellen muss. Polymorphismus erlaubt oft Systeme übersichtlicher zu strukturieren und Kontrollstrukturen zu vereinfachen. *Vererbung* bedeutet, dass neue Objekte die Eigenschaften bestehender Objekte übernehmen können. Dadurch kann vor allem die Wiederverwendung bestehender Software unterstützt und die Erweiterung komplexer Systeme erleichtert werden. Beispiele von bekannten, objektorientierten Programmiersprachen sind C++, Eiffel, Modula-3, Oberon, Object-Pascal, Objective-C, Self und Smalltalk.

Package:

Ein Package enthält eine oder mehrere Klassen und definiert den entsprechenden Namensraum für diese Klassen. Packages innerhalb eines Java-Programms sind hierarchisch organisiert. Mit dem Importieren eines Packages in ein anderes wird der Namensraum des importierenden Packages um denjenigen des importierten Packages erweitert.

Port:

Ein Port ist eine Abstraktion, die eine einfache Form lokaler Orts-Transparenz gewährleistet: Um einen Server zu kontaktieren braucht man nicht seine Prozess-Nummer zu kennen, man wendet sich nur an den Port, unter dem der betreffende Dienst angeboten wird. Im Gegensatz zur Prozess-Nummer, die von Maschine zu Maschine und von Tag zu Tag variieren kann, haben Dienste meist feste Port-Nummern. Im UNIX-Betriebssystem stehen diese in der Datei `/etc/services` verzeichnet. Die Umsetzung von Ports auf Prozesse wird durch den Mechanismus des Portmappers bewerkstelligt.

Programmierwerkzeug:

Zusammenfassende Bezeichnung für Hilfswerkzeuge bei der Entwicklung von Software. Dazu gehören typsicherweise Editoren, Übersetzer (Compiler), Debugger (Fehlerkorrektur) sowie Werkzeuge, die im weitesten Sinne dem Verständnis existierender Programmstrukturen dienen (z.B. → *Hierarchie-Browser* oder → *Klassenbrowser* bei der Verwendung objektorientierter Programmiersprachen). Siehe auch → *Entwicklungsumgebung*.

Proxy:

Ein Proxy ist ein lokales Objekt, welches ein rechnerfremdes Objekt repräsentiert. Zugriffe auf das Proxy werden dabei zum rechnerfremden Objekt, welches mit dem Proxy gekoppelt ist, transparent weitergeleitet. Proxies werden vor allem dazu eingesetzt, um einen logisch lokalen Zugangspunkt für ein Objekt zu schaffen, welches rechnerfremd ist. Im weiteren können Proxies auch dazu eingesetzt werden, einen logisch nicht-replizierten Zugangspunkt für ein repliziertes Objekt (→ *Objektgruppe*) zu schaffen, lokales Caching durchzuführen, transparente Objekt-Migration zu ermöglichen usw.

Remote Method Calling:

Unter RMC wird eine objektorientierte Variante des RPC-Paradigmas (→ *Remote Procedure Calling*). RMC unterscheidet sich dadurch, dass ein objektorientierter Programmierstil erlaubt wird. Über lokale → *Proxy*-Objekte kann der Programmierer die Methoden eines rechnerfremden Objektes aufrufen. Als Parameter können komplexe Objekte dienen. Name-overloading, Vererbung und Polymorphismus gelten nun auch für die objektorientierte Programmierung verteilter Systeme. Mittels desselben RMC-Mechanismus können auch Objektgruppen angesprochen werden, wodurch erreicht wird, dass ein Einzelobjekt auf einfachste Art und Weise durch eine Objektgruppe ersetzt werden kann, um Ausfalltoleranz oder Leistung einer verteilten Anwendung schrittweise zu erhöhen.

Remote Method Invocation:

RMI ist eine von Sun entwickelte Variante von RMC (→ *Remote Method Calling*, → *Remote Procedure Calling*) für die Programmiersprache Java.

RMC:

→ *Remote Method Calling*

RMI:

→ *Remote Method Invocation*

Remote Procedure Calling:

Der Grundmechanismus, welcher der Kommunikation in verteilten Systemen zugrundeliegt, ist input/output. Mittels Operationen wie `send()` und `receive()` können dabei Nachrichten zwischen Prozessen versandt und empfangen werden. Input/output ist aber meist kein geeignetes Programmiermodell, um komplexe, verteilte Anwendungen zu realisieren: Der Abstraktionsgrad ist zu niedrig und das Konzept führt in Programmen, welche mittels Prozeduren, Modulen oder Klassen strukturiert sind, oft zu einem Stilbruch. Diese Problematik führte zur Entwicklung des RPC. RPC ermöglicht das Aufrufen von Prozeduren eines rechnerfremden Programmes. Die Syntax eines Aufrufes ist dabei derjenigen eines herkömmlichen Prozeduraufrufes ähnlich oder gar identisch dazu. Vorteile des RPC sind, dass das Kommunikations-Modell für Programmierer, welche mit einer prozeduralen Sprache vertraut sind, einfacher verständlich ist, dass Programme übersichtlicher und besser lesbar werden, dass das → *Marshaling* von RPC-Parametern meist automatisch erfolgt und dass Typenprüfung der RPC-Parameter zur Compilierzeit vorgenommen werden kann. Probleme des RPC sind die Semantik eines RPCs angesichts von Ausfällen und Pointer (Zeiger) als RPC-Parameter (→ *Remote Method Calling*).

RPC:

→ *Remote Procedure Calling*

Runtime Code Generation:

Der Begriff "Runtime Code Generation" (RTCG) ist nicht eindeutig definiert. Je nachdem können Begiffe wie dynamic compilation, just in time compilation, self- modifying code etc. das gleiche oder etwas ähnliches bedeuten. RTCG ist aber sicher eine Technik, welche den Objektcode eines laufenden Programmes verändert. Oft wirt RTCG verwendet, um schnelle Interpreter zu implementieren. Wird eine Codesequenz genügend oft ausgeführt, so lohnt es sich, für diesen Programmteil zur Laufzeit Maschinencode zu erzeugen und dynamisch in das laufende Programm zu linken. Dieser Programmteil kann dann direkt auf der Hardware ablaufen, was natürlich viel schneller ist als die Ausführung im Interpreter. Die Schwierigkeit besteht darin, nur für die Teile Code zu generieren, die genügend oft ausgeführt werden, um den Laufzeitaufwand der Codegenerierung auch wieder zu amortisieren. RTCG ist zur Zeit noch ein aktives Forschungsgebiet. RTCG ist nur sinnvoll, wenn eine statische Übersetzung nicht möglich ist. (z.B. in Java: Code ist nicht vorhanden, weil er erst zur Laufzeit über das Netz heruntergeladen wird). Zukünftige Implementationen der Java Virtual Maschine werden Spielarten von RTCG und Just In Time Compilation verwenden, um eine schnelle Interpretation des → *Byte Codes* zu erreichen.

Server:

Logische Komponente einer Anwendung, die einen Dienst (Service) anbietet. (→ *Client-Server Konzept*).

Sniff+:

→ *Entwicklungsumgebung* von TakeFive Software. Sniff+ besteht aus einem → *Klassenbrowser*/-editor und einem umfangreichen Projektmanager.

Stackmaschine:

Computerarchitektur, auch Null-Adressmaschine genannt. Die Operanden einer Operation werden von einem Stack geholt und das Resultat auch wieder auf dem Stack abgelegt. Eine Addition hat z.B die Form: `push(add(pop(), pop()))`. Ein bedingter Sprung hat die folgende Form: `jmpConditional (ifLess( pop(), pop()), pop())`, wobei das dritte `pop()` die Zieladresse vom Stack holt. Im Instruktionsformat einer Stackmaschine müssen nur die Operationen und nicht auch die Operanden codiert werden, weshalb das Instruktionsformat nur wenige Bits lang sein muss (z.B. → *Byte Code*). Dies führt zu sehr kleinen Programmen, auch wenn Stackmaschinen mehr Instruktionen benötigen als 2- oder 3-Addressmaschinen. Die meisten realen Stackmaschinen unterstützen aber auch eine Reihe von 1-Addressinstruktionen; weil diese z.B. für das Laden von Konstanten sehr praktisch sind, also z.B.: `push(3.1415926)`.

Stub:

Bei einem → *Remote Procedure Call* erscheint es dem Programmierer so, als ob auf einer entfernten Maschine eine Prozedur aufgerufen wird. Tatsächlich wird die Kontrolle lokal an eine sogenannte Stub-Prozedur übertragen, welche die Kommunikation mit der entfernten Maschine durchführt. Solche Stubs gibt es auf der Server- und der Client-Seite. Bei moderneren RPC-Systemen können sie auch aus einer formalen Beschreibung der Schnittstelle generiert werden (→ *IDL*).

TCP:

→ *Transmission Control Protocol*

Thread:

In den meisten traditionellen Betriebssystemen beherbergt jeder Prozess einen einzigen Ausführungspfad (thread of execution). In vielen Situationen ist es aber vorteilhaft, mehrere Ausführungspfade pro Prozess zu haben. Dabei spricht man von *Multithreading*. Ein solcher Ausführungspfad wird als *Thread* oder als leichtgewichtiger Prozess bezeichnet. Die Threads eines Prozesses laufen in der Regel *quasi*-parallel ab. Normalerweise läuft ein Thread solange bis dieser, z.Bsp. während einer I/O Operation, blockiert wird. Wird ein Thread blockiert, so kommt sofort der nächste Thread innerhalb des Prozesses zum Zuge. Welcher der "nächste" Thread ist, entscheidet ein Scheduling-Algorithmus, der vom Programmierer beeinflusst werden kann. Der Kontext-Wechsel zwischen Threads kommt dabei wesentlich rascher Zustande als der Kontext-Wechsel zwischen Prozessen. Dies weil die mit einem Thread assoziierte Kontext-Information gering ist und in der Regel lediglich den Programm-Zähler, den Stack-Pointer und Register-Inhalte umfasst. Multithreading erlaubt, die einem Prozess zugeordnete Prozessor-Zeit wesentlich besser zu nutzen. Der Prozess kann nun solange aktiv bleiben wie ein Thread in ihm arbeiten kann und die dem ganzen Prozess zugeordnete Zeitscheibe noch nicht abgelaufen ist. Ohne Multithreading wird ein in einer I/O Operation blockierter Prozess gleich vollständig suspendiert. Moderne Betriebssystem-Kerne wie Mach oder Amoeba unterstützen Multi-threading. Bei anderen Betriebssystemen muss Multithreading vollständig im *user space* implementiert werden. Der Nachteil dabei ist, dass ein vom

Prozess empfangenes Signal sämtliche Threads blockieren kann, falls keine besonderen Vorkehrungen getroffen werden (Abfangen und weiterleiten sämtlicher Signale, spezielle non-blocking I/O Bibliotheken etc.). Multithreading erlaubt es, die Effizienz zahlreicher verteilter Anwendungen signifikant zu steigern, Ergebnisse von asynchronen Operationen (z.B. von asynchronem → *RPC* oder asynchronem `send()`) zu empfangen und einen reaktiven Programmierstil zu praktizieren.

Thread-Gruppen:

Wenn sehr viele Prozessgruppen von erheblicher Grösse notwendig sind, können Prozessgruppen zu ineffizienten, schlecht skalierbaren Lösungen führen. Dies vor allem deshalb, weil Prozesse in der Art und Weise wie BSD 4.3 UNIX und Derivate sie vorsehen, "schwergewichtige" Konstrukte sind. Das heisst, mit einem Prozess ist viel Kontext-Information assoziiert und ein Prozesswechsel kommt daher teuer zu stehen. Neuerdings wird der Ansatz verfolgt, Gruppen von "leichtgewichtigen" Prozessen (→ *Thread*), sprich Thread-Gruppen, zu realisieren. Das Hinzutreten zu einer Gruppe, die Verwaltung der Gruppe und die Kontext-Wechsel werden dabei erheblich effizienter. Thread-Gruppen werden beispielsweise im HORUS Toolkit realisiert.

Transmission Control Protocol:

Ein Kommunikationsprotokoll das zuverlässige, bidirektionale, verbindungsorientierte Kommunikation zwischen zwei Prozessen auf verschiedenen Rechnern gestattet. Die TCP-Protokoll-Schicht basiert auf → *IP* zur Kommunikation auf der Internet-Netzwerkebene. Seit seinem Einsatz durch das amerikanische Verteidigungsministerium ist TCP/IP ein wichtiger Standard für die Verbindung verteilter Systeme geworden.

UDP:

→ *User Datagram Protocol*

User Datagram Protocol:

Dieses Kommunikationsprotokoll des → *TCP*/IP-Standards erlaubt die Kommunikation von → *Datagrammen* zwischen Prozessen auf verschiedenen Rechnern mittels des → *IP*-Protokolls. UDP erlaubt im Gegensatz zu TCP selektive Kommunikation, gewährleistet aber nicht den Empfang eines gesendeten Packets.

Uniform Resource Locator:

Schema zur Adressierung und Lokalisierung von Information auf dem → *WWW*. Jeder → *Web-Browser* benutzt URLs, um auf Informationen im WWW zuzugreifen,

URL:

→ *Uniform Resource Locator*

Verteiltes System:

Leslie Lamport soll ein verteiltes System einst wie folgt definiert haben: "A distributed system is one that stops you from getting any work done when a machine you've never heard of crashes". Dies mag darauf hindeuten, dass heute allerhand unter dem Etikett "Verteiltes System" angepriesen

wird. Tanenbaum definiert ein verteiltes System wie folgt: "A distributed system is one that runs on a collection of machines that do not have shared memory, yet looks to its users like a single computer". Damit wird vor allem die Transparenz unterstrichen. Mullender betont bei seiner Definition vor allem die Ausfalltoleranz: "A distributed operating system must not have any single point of failure — no single part failing should bring the whole system down". Wichtige Anforderungen an ein verteiltes System sind sicherlich Transparenz, Flexibilität, Erweiterbarkeit, Zuverlässigkeit (Ausfalltoleranz), Leistung und Skalierbarkeit.

Virtuelle Maschine:

1. Eine virtuelle Maschine ist eine abstrakte Maschine, für die ein Interpreter existiert. Virtuelle Maschinen werden oft gebraucht, um ein portables Ausführungsmodell für eine high-level Sprache zu implementieren. Die high-level Sprache wird in Objektcode für die virtuelle Maschine übersetzt (compiliert) und dieser Objektcode wird dann durch den Interpreter ausgeführt. Um die Sprache auf eine neue Hardware zu portieren genügt es, den Interpreter zu portieren. Der Interpreter selbst ist in Assembler oder einer portablen Sprache wie z.B. C geschrieben.Beispiele sind Java, Core War, OCODE, OS/2, POPLOG, Portable Scheme Interpreter, Portable Standard Lisp, Parallel Virtual Machine, Sequential Parlog Machine, SNOBOL Implementation Language, SODA, UCSD-Pascal.
2. Eine virtuelle Maschine ist ein Software-Emulator für ein Computing Environment. Der Begriff stammt von IBM's VM Betriebssystem, welches gleichzeitig mehrere Computing Environments zur Verfügung stellt, in welchen unterschiedliche Betriebssysteme oder andere Programme gleichzeitig ablaufen können, als wären sie alleine auf der Maschine. Die Hauptanwendung von VM war der gleichzeitige Betrieb von mehreren Betriebssystemversionen, so dass die Software migriert werden konnte, ohne einen zweiten Prozessor (Computer) anschaffen zu müssen.

Virtual Machine:

→ *Virtuelle Maschine*

VM:

→ *Virtuelle Maschine*

Web:

→ *World Wide Web*

Web-Browser:

Programm zur Darstellung von Information, die in HTML formatiert ist (→ *Netscape Navigator*, → *Explorer*).

World Wide Web:

Ein hypertext-basierendes Informations- und Kommunikationssystem, das das Internet als Übertragungsmedium benutzt. Mittels des HyperText Transfer Protocol (HTTP) wird es WWW-Klienten ermöglicht, in Interaktion mit WWW-Servern zu treten und verschiedenste Arten von Daten (v.a. Multimedia-Daten) zu übertragen.

WWW:

→ *World Wide Web*

WWW-Browser:

→ *Web-Browser*