```
 6  import java.util.*
 7
 8  class MyClass {
 9    short s;
10    int i;
11    float f;
12    double d;
13    Object obj;
14    Date aDate;
15    String str;
16    SetOfObject set;
17    BagOfObject bag;
18    VArrayOfObject varray;
19    ListOfObject list;
20    MyClass myObj;
21    boolean aBool;
22    long l;
23
24    // Objekt e aus java.util.Enumeration
25    // wird nicht gespeichert
26
27    transient Enumeration e;
28
29    public MyClass() {
30      set = new SetOfObject();
31      bag = new BagOfObject();
32      varray = new VArrayOfObject();
33      list = new ListOfObject();
34      i = 11;
35      s = 12;
36      d = 3.1415926;
37      str = "Mein erstes POET Objekt";
38      }
39
40    public String toString() {
41      return str + " (" +
42        Integer.toString(i) + ", " +
43        Integer.toString(s) + ", " +
44        Double.toString(d) + ", " +
45        aDate.toString() +")";
46      }
47  }
48  // End of file cl3:/u/bonin/myjava/POET/MyClass.java
```

Beispiel `Bind.java`

```
1  /**
2    Persistence aware class Bind
3  */
4  import COM.POET.odmg.*;
5  import java.util.*;
6
```

```
 7  public class Bind {
 8    Bind(Database db, String name) throws ODMGException {
 9    Transaction txn = new Transaction();
10    txn.begin();
11      try {
12      MyClass myObject = new MyClass();
13      db.bind(myObject, name);
14      }
15      // Fuer den Fehlerfall
16      catch (ObjectNameNotUniqueException exc){
17        txn.abort();
18        throw exc;
19      }
20      catch (ODMGRuntimeException exc) {
21        txn.abort();
22        throw exc;
23      }
24
25    txn.commit();
26    }
27
28    public static void main(String[] argv) throws ODMGException {
29      if (argv.length<2) {
30        System.out.println("Bitte Datenbank und Objektname nennen!");
31        System.exit(1);
32      }
33      Database db = Database.open(argv[0], Database.openReadWrite);
34      try {
35      new Bind(db, argv[1]);
36      }
37      finally {db.close();}
38    }
39  }
40  // End of file cl3:/u/bonin/myjava/POET/Bind.java
```

Beispiel `Lookup.java`

```
 1  /**
 2    Selektieren und Rekonstruieren eines POET-Objektes
 3   */
 4  import COM.POET.odmg.*;
 5
 6  public class Lookup {
 7    Lookup(Database db, String name) throws ODMGException {
 8    Transaction txn = new Transaction();
 9    txn.begin();
10    try {
11      MyClass myObject = (MyClass)db.lookup(name);
12      System.out.println(myObject);
13    }
14    // Fuer den Fehlerfall
15    catch (ObjectNameNotFoundException exc) {
```

```
16      txn.abort();
17      throw exc;
18    }
19    catch (ODMGRuntimeException exc) {
20      txn.abort();
21      throw exc;
22    }
23    txn.commit();
24    }
25
26    public static void main(String[] argv) throws ODMGException {
27      if (argv.length<2) {
28        System.out.println("Bitte Datenbank und Objektname nennen!");
29        System.exit(1);
30      }
31      Database db = Database.open(argv[0], Database.openReadWrite);
32      try {
33        new Lookup(db, argv[1]);
34      }
35      finally {db.close();}
36    }
37  }
38  // End of file cl3:/u/bonin/myjava/POET/Lookup.java
39
```

Beispiel `Delete.java`

```
 1  /**
 2    Selektieren und Löschen eines POET-Objektes
 3   */
 4  import COM.POET.odmg.*;
 5
 6  public class Delete {
 7    Delete(Database db, String name) throws ODMGException {
 8    Transaction txn = new Transaction();
 9    txn.begin();
10    try {
11      ObjectServices.delete(db.lookup(name));
12      db.unbind(name);
13    }
14    // Fuer den Fehlerfall
15    catch (ObjectNameNotFoundException exc) {
16      txn.abort();
17      throw exc;
18    }
19    catch (ODMGRuntimeException exc) {
20      txn.abort();
21      throw exc;
22    }
23    txn.commit();
24    }
25
```

```
26    public static void main(String[] args) throws ODMGException {
27      if (args.length<2) {
28        System.out.println("Bitte Datenbank und Objektname nennen!");
29        System.exit(1);
30      }
31      Database db = Database.open(args[0], Database.openReadWrite);
32      try {
33        new Delete(db, args[1]);
34      }
35      finally {db.close();}
36      }
37  }
38  // End of file cl3:/u/bonin/myjava/POET/Delete.java
39
```

6.9 Verteilte Objekte mit RMI

RPC

CORBA

Wenn Objekte auf mehreren Rechnern zusammen ein Anwendungssystem bilden, dann muß irgend eine Form von Informationsaustausch zwischen ihnen möglich sein. Ein solcher Informationsaustausch wird häufig auf der Basis eines *Remote Procedure Call* (RPC) Mechanismus abgebildet. Das *Java Remote Method Invocation Protocol* (RMI) ist ein solches RPC-basiertes Protokoll. RMI ermöglicht einem Objekt eines Client-Systems vorgegebene Methoden auf einem Server-System genauso aufzurufen als wären es lokale Methoden. RMI löst diese Kommunikation in einer vereinfachten Form des Standards *Common Object Request Broker Architecture* (CORBA). Im Gegensatz zu CORBA setzt RMI eine homogene Java-Umgebung, also Java-Clients und Java-Server voraus.[22] Man kann sich daher RMI annähernd als ein „pure-Java-CORBA“[23] vorstellen. Das einfachere RMI ist CORBA vorzuziehen, wenn gewährleistet ist, daß es nur Java-Objekte gibt. Bei Mehrsprachigkeit im System ist CORBA erforderlich.

RMI ist ein Modell der verteilten Objekte, das bekannte Lösungen in eine durchgängige Java Syntax und Semantik einbaut. Dabei kombiniert RMI Lösungen von *Modula-3 Network Objects System* und von *Spring's Subcontract* (→[SunRMI98]). In diesem Modell ist ein *Remote*-Objekt ein Objekt, dessen Methoden aus einer anderen *Java Virtual Maschine* (JVM) aufgerufen werden können. Diese andere JVM läuft üblicherweise auf einem anderen Rechner im Netz. Ein *Remote*-Objekt wird durch ein oder mehrere *Remote Interfaces* beschrieben. Ein solches Interface deklariert die Methoden des *Remote*-Objektes. Der Client referenziert das *Remote*-Objekt anhand einer RMI-URL[24]-Angabe. Diese hat folgende Form:

RMI-URL

[22] Präziser formuliert: RMI verbindet Systeme, die das *Standard Java Native Method Interface* (JNI) benutzen. Dies könnten prinzipiell auch Systeme in einer anderen Sprache sein.

[23] →[Vanderburg97] p. 525

[24] *Uniform Resource Locator*

```
rmi://hostname[:port]/object
```

Dabei hat der *Default*-Port die Nummer 1099.

Wenn der Server ein Objekt für einen RMI-URL-Zugriff (*lookup*) verfügbar macht, dann muß er eine Objektinstanz an einen Objektnamen binden. Der Objektname ist ein vorgegebener `String`. Die Klassenmethode `lookup(String url)` der Klasse `java.rmi.Naming` verbindet letztlich den Client mit dem entsprechenden Serverobjekt. Dazu sind die beiden Klassen `server_Stub.class` und `server_Skel.class` auf der Serverseite erforderlich. Diese zusätzlichen Kommunikationsklassen werden mit Hilfe des Programms `rmic` erzeugt.[25]

```
rmic ServerClassName
```

rmic

Das Programm `rmic` erzeugt und compiliert die sogenannten *Stub*- und *Skeleton*-Klassen. In der Stub-Klasse sind die Remote-Methoden implementiert. In dem Beispiel „Bank“ sind es die Methoden `abheben()`, `einzahlen()` usw. (→Abschnitt 6.9 auf Seite 161). In der Skeleton-Klasse sind es die Methoden `getOperations()` und `dispatch()`. Die Stub-Klasse dient als eine Art Dummy-Referenz für das Client-System, während die Skeleton-Klasse das eigentliche Server-System verwaltet. Tabelle 6.6 auf Seite 160 skizziert die Zusammenarbeit zwischen Client, Stub, Skeleton und Server.

Bei diesem *Remote Object Model* hält der Server Objekte vor, die der Client „aus der Ferne“ benutzen kann. Der Client wendet eine Methode auf ein entferntes Objekt genauso an, als ob das Remote-Objekt sein lokales Objekt wäre, das in seiner *Java Virtual Maschine* existiert.

```
... MyClientFooClass
⋮
localInstance.lokalMethod(myArgument);
⋮
remoteInstance.remoteMethod(myArgument);
⋮
```

Der RMI-Mechanismus verbirgt die tiefer liegenden Transportmechanismen für das Übermitteln des Methodennamens, der Methodenargumente und des Rückgabewertes. Argumente und Rückgabewert können komplexe Objekte sein, und nicht nur einfache Zeichenketten. Für die Übermittlung müssen sie allerdings serialisiert werden. Daher kommen für RMI alle `Serializable`-Objekte in Betracht (→Abschnitt 6.3 auf Seite 110).

Für die Entwicklung einer RMI-Anwendung sind folgende Schritte erforderlich:

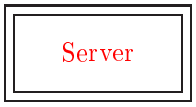
Client

1. Festlegen der Methoden, die auf das Remote-Objekt angewendet werden sollen.
   ↪ Definieren eines Subinterfaces von `java.rmi.Remote`. Dieses Interface definiert die exportierbaren Methoden, die das Remote-Objekt implementiert, das heißt, die Methoden, die der Server implementiert

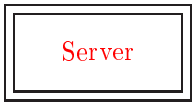
[25] Dieses Programm ist Bestandteil des JDK-Paketes.

und der Client aufrufen kann.

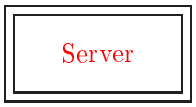
2. Definieren einer Subklasse von `java.rmi.server.UnicastRemoteObject`
   ↪Sie implementiert das `Remote`-Interface.



3. Schreiben der Server-Applikation — Erzeugen einer Instanz des Remote-Objekts und „Exportieren“ dieser Instanz, das heißt, Verfügbarmachen für die Nutzung durch den Client.
   ↪Registrieren des Objektes anhand seines Namens mit einem Registrierungsservice. Üblicherweise erfolgt diese Registrierung mittels der Klasse `java.rmi.Naming` und dem Programm `rmiregistry` (→übernächsten Punkt).



4. Erzeugen von Stub und Skeleton mit dem Programm `rmic` aus der compilierten Server-Klasse.



5. Registrierung

   Windosw-NT-Plattform: `start rmiregistry [port]`
   UNIX-Plattform: `rmiregistry [port] &`

6. Schreiben der Client-Applikation

7. Compilieren und Anwenden der Client-Applikation

Für das Beispiel „Bank“ sehen die Schritte wie folgt aus:

1. Methoden auf dem Bank-Server:

```
public interface RemoteBank extends Remote {
  public void einzahlen
    (String name, String passwort, Euro geld)
    throws RemoteException, BankingException;
  public Euro abheben
    (String name, String passwort, int betrag)
    throws RemoteException, BankingException;
  public int getStand
    (String name, String passwort)
    throws RemoteException, BankingException;
  public Vector getKontoBewegungen
    (String name, String passwort)
    throws RemoteException, BankingException;
  public void eroeffnenKonto
    (String name, String passwort)
    throws RemoteException, BankingException;
  public Euro aufloesenKonto
    (String name, String passwort)
    throws RemoteException, BankingException;
}
```

2. Definieren der Klasse `RemoteBankServer` als Unterklasse von `java.rmi.server.UnicastRemoteObject`. Sie implementiert das Interface `RemoteBank`

```
public class RemoteBankServer extends UnicastRemoteObject
  implements RemoteBank {
  class Konto {...}

  public RemoteBankServer() throws RemoteException {
    super();
  }

  public void einzahlen(String name, String passwort, Euro geld)
    throws RemoteException, BankingException {...}

  public Euro abheben(String name, String passwort, int betrag)
    throws RemoteException, BankingException {...}

  public int getStand(String name, String passwort)
    throws RemoteException, BankingException {...}

  public Vector getKontoBewegungen(String name, String passwort)
    throws RemoteException, BankingException {...}

  public synchronized void eroeffnenKonto(String name, String passwort)
   throws RemoteException, BankingException {...}

  public Konto pruefen(String name, String passwort)
    throws BankingException {...}

  public synchronized Euro aufloesenKonto(String name, String passwort)
    throws RemoteException, BankingException {...}
...
}
```

3. Schreiben von `RemoteBankServer.java` — Erzeugen einer Instanz `bank` des Remote-Objekts `RemoteBankServer` und „Exportieren“ dieser Instanz mittels `Naming.rebind(name, bank)`

```
public static void main(String argv[]) {
  try {
    RemoteBankServer bank = new RemoteBankServer();
    String name = System.getProperty("bankname", "BoninRemote");
    Naming.rebind(name, bank);
    System.out.println(name +
      " ist eroeffnet und bereit fuer Buchungen.");
  }
```

Die Klassenmethode `getProperty(String key, String default)` der Klasse `java.lang.System` sucht in der Systemeigenschaftsliste nach dem Wert von `key`. Wird keiner gefunden, dann ist `default` der Rückgabewert. Beim Aufruf einer Java-Applikation kann ein Eintrag in diese Systemeigenschaftsliste mit Hilfe der Option „`-D`“ erfolgen.

```
java -Dkey1=wert1 -Dkey2=wert2 ... javaClass
```

Zum Beispiel:

```
java -Dbank="rmi://myServer:1111/myRemoteObject" Bank$Client
...
```

4. Erzeugen von `RemoteBankServer_Stub` und `RemoteBankServer_Skel` mit dem Programm `rmic` aus `RemoteBankServer.java`.

```
javac RemoteBankServer.java
rmic RemoteBankServer
```

```
public final synchronized class RemoteBankServer_Stub
  extends java.rmi.server.RemoteStub
  implements Bank$RemoteBank, java.rmi.Remote {
  // Feld(er)
  private static java.rmi.server.Operation[] operations;
  private static final long interfaceHash;
  // Konstruktor(en)
  public RemoteBankServer_Stub();
  public RemoteBankServer_Stub(java.rmi.server.RemoteRef);
  // Methode(n)
  public Bank$Euro abheben(java.lang.String, java.lang.String, int)
    throws Bank$BankingException, java.rmi.RemoteException;
  public Bank$Euro aufloesenKonto(java.lang.String, java.lang.String)
    throws Bank$BankingException, java.rmi.RemoteException;
  public void einzahlen(java.lang.String, java.lang.String, Bank$Euro)
    throws Bank$BankingException, java.rmi.RemoteException;
  public void eroeffnenKonto(java.lang.String, java.lang.String)
    throws Bank$BankingException, java.rmi.RemoteException;
  public java.util.Vector getKontoBewegungen
    (java.lang.String, java.lang.String)
     throws Bank$BankingException, java.rmi.RemoteException;
  public int getStand(java.lang.String, java.lang.String)
    throws Bank$BankingException, java.rmi.RemoteException;
}
```

```
public final synchronized class RemoteBankServer_Skel
  extends java.lang.Object
  implements java.rmi.server.Skeleton {
  // Feld(er)
  private static java.rmi.server.Operation[] operations;
  private static final long interfaceHash;
  // Konstruktor(en)
  public RemoteBankServer_Skel();
  // Methode(n)
  public java.rmi.server.Operation[] getOperations();
  public void dispatch
   (java.rmi.Remote, java.rmi.server.RemoteCall, int, long)
   throws java.rmi.RemoteException, java.lang.Exception;
}
```

5. Registrierung mit Hilfe des Programms `rmiregistry` auf einer UNIX-Plattform bei Nutzung des *Default-Ports* 1099 und Starten des Servers.

```
rmiregistry&
java RemoteBankServer
BoninRemote ist eroeffnet und bereit fuer Buchungen.
```

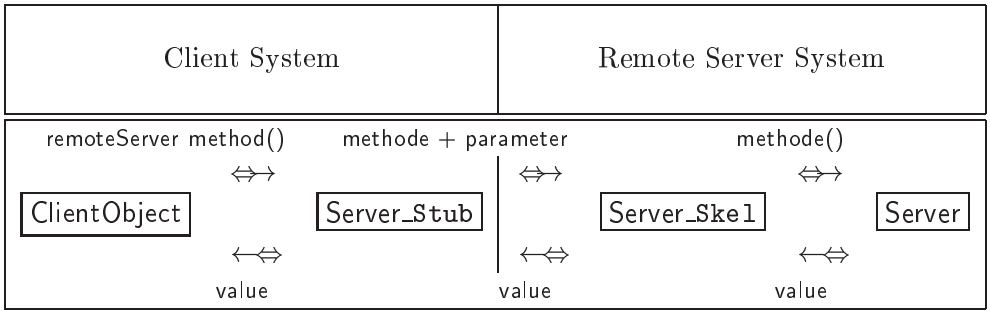
6. Schreiben der Client-Applikation `Bank$Client`

```
public static class Client {
  public static void main(String argv[]) {
  try {
    System.setSecurityManager(new RMISecurityManager());
    String url = System.getProperty("bank", "rmi:///BoninRemote");
    RemoteBank bank = (RemoteBank) Naming.lookup(url);

    if (aktion.equals("einzahlen")) {
      Euro geld = new Euro(Integer.parseInt(argv[3]));
      bank.einzahlen(argv[1], argv[2], geld);
      System.out.println("Eingezahlt: " +
        geld.betrag + " Euro");
    }
    else if (aktion.equals("abheben")) {
      Euro geld = bank.abheben(argv[1], argv[2],
        Integer.parseInt(argv[3]));
      System.out.println("Abgehoben: " +
        geld.betrag + " Euro");
    }
    ...

  }
  catch (RemoteException e)  {...}
  catch (BankingException e) {...}
  catch (Exception e)        {...}
  ...
```

7. Compilieren der Datei `Bank.java` und Anwenden der Java-Applikation, das heißt, Aufruf von `main()` in der Klasse `Bank$Client`. Die Klasse `Bank.class` dient nur als ein Sammelbehälter für das Interface `RemoteBank` und die Klassen `Euro`, `BankingException` und `Client`.[26]

```
javac Bank.java
java Bank$Client eroeffnen otto kh234g
Konto eroeffnet!
...
```

[26] Diese sind mit dem Modifikator `static` versehen, um als Toplevel-Interface bzw. Toplevel-Klassen nutzbar zu sein.



Legende:
`Stub` ≡ lokaler Stellvertreter
`Skel` ≡ Skeleton, Rahmen
Das RMI-System gliedert sich in drei Schichten (*Layer*):

1. *Stub/Skeleton Layer* — eine Proxy-Funktion auf der Client-Steite (Stub)
2. *Remote Reference Layer* — Verhalten bei einem einzelnen Objekt oder bei replizierten Objekten
3. *Transport Layer* — Verbindungsmanagement und Remote Objekt Verfolgung

Näheres →[SunRMI98]

Tabelle 6.6: RMI: *Stub/Skeleton-*, *Remote-Reference-* und *Transport*-Schicht

Client `Bank.java`

```
 1  /**
 2   RMI-Client-Beispiel
 3   Idee von David Flanagan; Java Examples in a Nutshell, 1997, p. 294
 4   Quellcode stark modifiziert.
 5   @author Bonin 13-Jun-1998
 6   @version 1.0
 7  */
 8  import java.rmi.* ;
 9  import java.util.Vector ;
10
11  /**
12   Bank enthält alle Interfaces und Klassen (top-level)
13   */
14  public class Bank {
15  /** Methoden auf dem Bankserver
16     einzahlen
17     abbheben
18     getStand
19     getKontoBewegungen
20     eroeffnenKonto
21     aufloesenKonto
22   */
23    public interface RemoteBank extends Remote {
24      public void einzahlen
25        (String name, String passwort, Euro geld)
26        throws RemoteException, BankingException;
27      public Euro abheben
28        (String name, String passwort, int betrag)
29        throws RemoteException, BankingException;
30      public int getStand
31        (String name, String passwort)
32        throws RemoteException, BankingException;
33      public Vector getKontoBewegungen
34        (String name, String passwort)
35        throws RemoteException, BankingException;
36      public void eroeffnenKonto
37        (String name, String passwort)
38        throws RemoteException, BankingException;
39      public Euro aufloesenKonto
40        (String name, String passwort)
41        throws RemoteException, BankingException;
42    }
43  /**
44    Einfache Klasse die Geld repraesentiert
45   */
46    public static class Euro implements java.io.Serializable {
47      public int betrag;
48      public Euro(int betrag) {
49        this.betrag = betrag;
50      }
51    }
52  /**
```

```
 53    Bankspezifische Ausnahmen
 54   */
 55    public static class BankingException extends Exception {
 56      public BankingException(String nachricht) {
 57        super(nachricht);
 58      }
 59    }
 60
 61  /**
 62    Bank$Client kommuniziert mit dem RMI-Server
 63   */
 64    public static class Client {
 65      public static void main(String argv[]) {
 66      try {
 67        // Sicherheit gegen untrusted stub code über das Netz
 68        System.setSecurityManager(new RMISecurityManager());
 69        // Default-Wert BoninRemote
 70        String url = System.getProperty("bank", "rmi:///BoninRemote");
 71        // Naming Objekt kontaktet rmiregistry
 72        RemoteBank bank = (RemoteBank) Naming.lookup(url);
 73
 74        String aktion = argv[0].toLowerCase();
 75
 76        if (aktion.equals("einzahlen")) {
 77          Euro geld = new Euro(Integer.parseInt(argv[3]));
 78          bank.einzahlen(argv[1], argv[2], geld);
 79          System.out.println("Eingezahlt: " +
 80            geld.betrag + " Euro");
 81        }
 82        else if (aktion.equals("abheben")) {
 83          Euro geld = bank.abheben(argv[1], argv[2],
 84            Integer.parseInt(argv[3]));
 85          System.out.println("Abgehoben: " +
 86            geld.betrag + " Euro");
 87        }
 88        else if (aktion.equals("stand")) {
 89          System.out.println("Kontostand : " +
 90            bank.getStand(argv[1], argv[2]) + " Euro");
 91        }
 92        else if (aktion.equals("bewegungen")) {
 93          Vector bewegungen = bank.getKontoBewegungen(
 94            argv[1], argv[2]);
 95          for (int i = 0; i < bewegungen.size(); i++)
 96            System.out.println(bewegungen.elementAt(i));
 97        }
 98        else if (aktion.equals("eroeffnen")) {
 99          bank.eroeffnenKonto(argv[1], argv[2]);
100          System.out.println("Konto eroeffnet!");
101        }
102        else if (aktion.equals("aufloesen")) {
103          Euro geld = bank.aufloesenKonto(argv[1], argv[2]);
104          System.out.println(geld.betrag +
105            " Euro erhalten Sie noch ausgezahlt!");
106        }
```

```
107         else System.out.println("Unbekannte Aktion!");
108       }
109       catch (RemoteException e) {System.err.println(e);}
110       catch (BankingException e) {
111         System.err.println(e.getMessage());
112       }
113       catch (Exception e) {
114         System.err.println(e);
115         System.err.println(
116           "Usage: java [-Dbank=<url>] Bank$Client " +
117           "<aktion> <name> <passwort> [<betrag>]");
118         System.err.println(
119           "wobei <aktion> einer der folgenden Wert ist: " +
120           "\nEinzahlen, Abheben, Stand, Bewegungen, Eroeffnen, Aufloesen");
121       }
122       }
123     }
124   }
125   //End of file cl3:/u/bonin/myjava/Bank.java
126
```

Server `RemoteBankServer.java`

```
 1  /**
 2   RMI-Server-Beispiel
 3   Quellidee von David Flanagan; Java Examples in a Nutshell, 1997, p. 297
 4   Quellcode modifiziert.
 5   @author Bonin 13-Jun-1998
 6   @version 1.0
 7  */
 8  import java.rmi.* ;
 9  import java.rmi.server.* ;
10  import java.util.* ;
11  import Bank.* ;
12
13  public class RemoteBankServer extends UnicastRemoteObject
14    implements RemoteBank {
15    class Konto {
16      int stand;
17      String passwort;
18      Vector bewegungen = new Vector();
19
20      Konto(String passwort) {
21        this.passwort = passwort;
22        bewegungen.addElement("Kontoeroeffnung am: " + new Date());
23      }
24    }
25
26    Hashtable kontos = new Hashtable();
27    public RemoteBankServer() throws RemoteException {
28      super();
29    }
```

```
30
31    public void einzahlen(String name, String passwort, Euro geld)
32      throws RemoteException, BankingException {
33      Konto myK = pruefen(name, passwort);
34      synchronized(myK) {
35        myK.stand += geld.betrag;
36        myK.bewegungen.addElement("Eingezahlt: " + geld.betrag +
37          " am " + new Date());
38      }
39    }
40    public Euro abheben(String name, String passwort, int betrag)
41      throws RemoteException, BankingException {
42      Konto myK = pruefen(name, passwort);
43      synchronized(myK) {
44        if (myK.stand < betrag)
45          throw new BankingException("Keine Deckung!");
46        myK.stand -= betrag;
47        myK.bewegungen.addElement("Abgehoben: " +
48          betrag + " am " + new Date());
49        return new Euro(betrag);
50      }
51    }
52    public int getStand(String name, String passwort)
53      throws RemoteException, BankingException {
54      Konto myK = pruefen(name, passwort);
55      synchronized(myK) {
56        return myK.stand;
57      }
58    }
59    public Vector getKontoBewegungen(String name, String passwort)
60      throws RemoteException, BankingException {
61      Konto myK = pruefen(name, passwort);
62      synchronized(myK) {
63        return myK.bewegungen;
64      }
65    }
66    public synchronized void eroeffnenKonto(String name, String passwort)
67     throws RemoteException, BankingException {
68     if (kontos.get(name) != null)
69       throw new BankingException("Konto gibt es schon!");
70     Konto myK = new Konto(passwort);
71     kontos.put(name, myK);
72    }
73
74    public Konto pruefen(String name, String passwort)
75      throws BankingException {
76      synchronized(kontos) {
77        Konto myK = (Konto) kontos.get(name);
78        if (myK == null)
79          throw new BankingException("Kein solches Konto!");
80        if (!passwort.equals(myK.passwort))
81         throw new BankingException("Falches Passwort!");
82        return myK;
83      }
```

```
84    }
85    public synchronized Euro aufloesenKonto(String name, String passwort)
86      throws RemoteException, BankingException {
87      Konto myK;
88      myK = pruefen(name, passwort);
89      kontos.remove(name);
90      synchronized (myK) {
91        int wert = myK.stand;
92        myK.stand = 0;
93        return new Euro(wert);
94      }
95    }
96
97
98    public static void main(String argv[]) {
99      try {
100       RemoteBankServer bank = new RemoteBankServer();
101       String name = System.getProperty("bankname", "BoninRemote");
102       Naming.rebind(name, bank);
103       System.out.println(name +
104         " ist eroeffnet und bereit fuer Buchungen.");
105     }
106     catch (Exception e) {
107       System.err.println(e);
108       System.err.println(
109         "Usage: java [-Dbankname=<name>] RemoteBankServer");
110       System.exit(1);
111     }
112   }
113 }
114 //End of file cl3:/u/bonin/myjava/RemoteBankServer.java
115
```

Sessionprotokoll

```
1  --- Windows-NT-Rechner 193.174.33.100
2  >java -fullversion
3  java full version "JDK1.1.5k"
4  >javac RemoteBankServer.java
5  >rmic RemoteBankServer
6  >start rmiregistry
7  >java RemoteBankServer
8  BoninRemote ist eroeffnet und bereit fuer Buchungen.
9
10 --- UNIX-Rechner 193.174.33.106
11
12 cl3:/home/bonin/myjava:>java -fullversion
13 java full version "JDK 1.1.6 IBM build a116-19980529" (JIT: jitc)
14 >javac Bank.java
15 >java -Dbank="rmi://193.174.33.100:1099/BoninRemote" \
16 Bank\$Client doof otto geheim
```

```
Unbekannte Aktion!
>java -Dbank="rmi://193.174.33.100:1099/BoninRemote" \
Bank\$Client eroeffnen otto geheim
Konto eroeffnet!
>java -Dbank="rmi://193.174.33.100:1099/BoninRemote" \
Bank\$Client einzahlen otto geheim 100
Eingezahlt: 100 Euro
>java -Dbank="rmi://193.174.33.100:1099/BoninRemote" \
Bank\$Client einzahlen otto geheim 200
Eingezahlt: 200 Euro
>java -Dbank="rmi://193.174.33.100:1099/BoninRemote" \
Bank\$Client stand otto geheim
Kontostand : 300 Euro
>java -Dbank="rmi://193.174.33.100:1099/BoninRemote" \
Bank\$Client abheben otto geheim 50
Abgehoben: 50 Euro
>java -Dbank="rmi://193.174.33.100:1099/BoninRemote" \
Bank\$Client stand otto geheim
Kontostand : 250 Euro
>java -Dbank="rmi://193.174.33.100:1099/BoninRemote" \
Bank\$Client bewegungen otto geheim
Kontoeroeffnung am: Sat Jun 13 14:14:13 CEST 1998
Eingezahlt: 100 am Sat Jun 13 14:15:03 CEST 1998
Eingezahlt: 200 am Sat Jun 13 14:15:14 CEST 1998
Abgehoben: 50 am Sat Jun 13 14:16:00 CEST 1998
>java -Dbank="rmi://193.174.33.100:1099/BoninRemote" \
Bank\$Client aufloesen otto geheim
250 Euro erhalten Sie noch ausgezahlt!
cl3:/home/bonin/myjava:>
```

6.10 Übungen

6.10.1 Applikation `Rekursion.java`

```
/**
 Kleine Kostprobe fuer eine Rekursion
   Beispiel Fakultaet:
   n! = n * (n - 1)!
 @author Bonin 10-Apr-1998
        update 20-Apr-1998, 28-Apr-1998
 @version 1.0
*/

import java.math.*;

class Fakultaet {
// long-Wertebereich: 64 Bit
// -9223372036854775808 ... 9223372036854775807
// BigInteger von beliebiger Groesse

 BigInteger wert;
```

```
18   BigInteger basisWert = new BigInteger("1");
19
20   static long anzahlAufrufeFac = 0;
21
22   BigInteger fac(long n){
23      anzahlAufrufeFac += 1;
24
25      if (n == 0)
26        return  basisWert;
27      else
28       {
29        System.out.println("Aufruf  n = " + n);
30
31        wert =  this.fac(n  - 1).multiply(new BigInteger(Long.toString(n)));
32
33        System.out.println("Rueckgabe wert = " + wert.toString());
34        return wert;
35       }
36    }
37  }
38
39  public class Rekursion {
40    public static void main(String argv[]) {
41
42      Fakultaet foo = new Fakultaet();
43
44      String in;
45      if (argv.length == 0) in = "0";
46      else
47        in = argv[0].replace('+','0');
48
49      long      k = Long.parseLong(in);
50      long grenze = Long.MAX_VALUE;
51
52      if (k <= grenze && k >= 0)
53        System.out.println("Fakultaetsfunktion: fac(" + k +
54          ") = " + foo.fac(k));
55      else
56         System.out.println("Wert = " + k +
57           " kann nicht berechnet werden!");
58
59      System.out.println("Anzahl der Aufrufe von fac(): " +
60        Fakultaet.anzahlAufrufeFac);
61      }
62  }
63  // cl3:/u/bonin/myjava/Rekursion.java
```

Geben Sie bei dem folgenden Aufruf das Ergebnisse an:

```
>java Rekursion 4
```

6.10.2 Applikation `QueueProg.java`

Hinweis: Aufgabenidee aus [Freeman/Ince96] entnommen. Quellcode stark modifiziert und ergänzt.

```
 1  /**
 2   Kleine Kostprobe fuer eine ,,Zirkulaere Liste''
 3   mit dem Prinzip ,,first-in-first-out''
 4   @author Bonin 10-Apr-1998
 5   @version 1.0
 6  */
 7  final class Queue {
 8    private final int queueCapacity    = 3;
 9    private String circleList[]        = new String[queueCapacity];
10    private int noOfItemsInQueue       = 0;
11    private int frontOfTheQueue        = 0;
12    static int noOfQueues              = 0;
13
14    Queue() {
15     noOfQueues++;}
16
17    public int getQueueCapacity() {
18      return queueCapacity ;}
19
20    public int getNoOfItemsInQueue() {
21      return noOfItemsInQueue ;}
22
23    public boolean isQueueEmpty() {
24      return (noOfItemsInQueue == 0);}
25
26    public boolean isQueueFull() {
27      return (noOfItemsInQueue == queueCapacity);}
28
29    private void addNthItem(int n, String itemToBeAdded) {
30      int index;
31      index = frontOfTheQueue + (n -1);
32      if (index >= queueCapacity)
33        index = index % queueCapacity;
34      circleList[index] = itemToBeAdded;
35    }
36
37    public boolean addItem(String itemToBeAdded) {
38      if (this.isQueueFull()) {
39        System.out.println("Item = " + itemToBeAdded +
40          " nicht aufgenommen!");
41        return false;}
42      else {
43        noOfItemsInQueue++;
44        this.addNthItem(noOfItemsInQueue, itemToBeAdded);
45        return true; }
46    }
47
48    public String getFirstItem() {
49     if (this.isQueueEmpty()) return "";
50     else return circleList[frontOfTheQueue];}
51
```

```
52    public boolean removeFirstItem() {
53      if (this.isQueueEmpty()) {
54        System.out.println("Kein Item entfernbar");
55        return false; }
56      else {
57        noOfItemsInQueue--;
58        if (frontOfTheQueue == (queueCapacity - 1))
59          frontOfTheQueue = 0;
60        else
61          frontOfTheQueue++;
62        return true; };
63    }
64
65    public boolean isItemInQueue(String item) {
66      int count = 0;
67      while (count < noOfItemsInQueue) {
68        count++;
69        if (this.getNthItem(count) == item )
70          return true;
71      }
72      return false;
73    }
74
75    private String getNthItem(int n) {
76      int index;
77      index = frontOfTheQueue + (n -1);
78      if (index >= queueCapacity)
79        index = index % queueCapacity;
80      return (circleList[index]);
81    }
82  }
83
84  public class QueueProg {
85    public static void main(String argv[]) {
86
87    Queue myQ = new Queue();
88    Queue myL = new Queue();
89    System.out.println("Step 0: Queue.noOfQueues = " +
90      Queue.noOfQueues);
91    System.out.println("Step 1: myQ.getQueueCapacity() = " +
92      myQ.getQueueCapacity());
93
94    myQ.addItem("Otto AG");
95    myQ.addItem("Emma AG");
96    myQ.addItem("Klara AG");
97    myQ.removeFirstItem();
98
99    System.out.println("Step 2: myQ.getFirstItem() = " +
100     myQ.getFirstItem());
101
102   myQ.addItem("Willi AG");
103   myQ.addItem("Ernst AG");
104   myQ.removeFirstItem();
105   myQ.removeFirstItem();
```

```
106    myQ.addItem("Ernst AG");
107
108    System.out.println("Step 3: myQ.getFirstItem() = " +
109      myQ.getFirstItem());
110
111    System.out.println("Step 4: myQ.getNoOfItemsInQueue = " +
112      myQ.getNoOfItemsInQueue());
113
114    boolean inCircleList = myQ.isItemInQueue("Ernst AG");
115    System.out.println("Step 5: myQ.isItemInQueue(Ernst AG) = "
116      + inCircleList);
117   }
118  }
119  // cl3:/u/bonin/myjava/QueueProg.java
```

Geben Sie bei dem folgenden Aufruf das Ergebnisse an:

```
>java QueueProg
```

6.10.3 Applet SimpleThread.java

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0//EN//">
<HTML>
<!-- Testbett fuer Applet SimpleThread.class     -->
<!-- Bonin 07-May-1998                           -->
<!--                                             -->
<HEAD>
<TITLE>Mond am Himmel</TITLE>
</HEAD>
<BODY>
<H1>Mond am Himmel</H1>
<APPLET NAME="Mond" CODE="SimpleThread.class"
  WIDTH="450" HEIGHT="120" ALIGN="RIGHT"
  ALT="Mond am Himmel">
</APPLET>
<FONT SIZE=+2 COLOR="#F00000">Der Mond ist aufgegangen ...</FONT>
<P> Copyright Prof. Bonin 07-May-1998 all rights reserved</P>
<ADDRESS>
<A HREF="mailto:hinich-bonin@fbw.fh-lueneburg.de"
  >hinich-bonin@fbw.fh-lueneburg.de</A>
</ADDRES>
</BODY>
<!-- File C:\myjava\SimpleThread.html               -->
</HTML>
```

```
/**
 Kleines Beispiel fuer eine
 ,,Animation mittels Thread'',  Idee aus:
 Hubert Partl; Java-Einfuehrung, Version April 1998, S. 82
 http://www.boku.ac.at/javaeinf/
 Quellcode leicht modifiziert

 @author Bonin 08-Mai-1998
 @version 1.0
*/
import java.applet.*;
import java.awt.* ;

public class SimpleThread extends Applet
  implements Runnable {

  int x, y, width, height;
  Graphics grafik;
  Image bild;
  Color nachtFarbe = new Color(0,0,102);
  Color mondFarbe  = new Color(204,204,255);

  Thread myT = null;

  public void init() {
    Dimension d = getSize();
    width  = d.width;
    height = d.height;
```

```
29      bild   = createImage(width, height);
30      grafik = bild.getGraphics();
31      x = width / 2;
32      y = height / 2;
33      System.out.println("x = " + x + " y = " + y);
34    }
35
36    public void start() {
37      if (myT == null) {
38        myT = new Thread(this);
39        myT.start();
40      }
41      System.out.println("start() appliziert");
42    }
43
44    public void stop() {
45      if (myT != null) {
46        myT.stop();
47        myT = null;
48      }
49      System.out.println("stop() appliziert");
50    }
51
52    public void run() {
53      while (true) {
54        grafik.setColor(nachtFarbe);
55        grafik.fillRect(0,0,width,height);
56        grafik.setColor(mondFarbe);
57        grafik.fillArc(x,y-25,50,50,270,180);
58        x += 2;
59        if (x > width+50) x = -50;
60        repaint();
61        try {
62          System.out.println("In run() vor Thread.sleep(1000)");
63          Thread.sleep(1000);
64          System.out.println("In run() nach Thread.sleep(1000)");
65        }
66        catch (InterruptedException e) {
67          System.out.println("In run() Fehler");
68        }
69      }
70    }
71
72    public void paint (Graphics g) {
73      update(g);
74    }
75
76    public void update (Graphics g) {
77      if (bild != null) g.drawImage(bild,0,0,this);
78    }
79  }
80  //End of file cl3:/u/bonin/mywww/SimpleTread/SimpleThread.java
```

Skizzieren Sie bei dem folgenden Aufruf das Ergebnisse:

```
>appletviewer http://as.fh-lueneburg.de/mywww/SimpleThread/SimpleThread.html
```

Oder nutzen Sie einen Browser mit einem JDK der Version $\geq$1.1.4.

6.10.4 Applet DemoAWT.java

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0//EN//">
<HTML>
<!-- Testbett fuer ein Applet                    -->
<!-- Bonin 19-April 1998                         -->
<!--                                             -->
<HEAD>
<TITLE>Willi liebt Sport</TITLE>
</HEAD>
<BODY>
<H1>Willi liebt Sport</H1>
<APPLET NAME="Willi" CODE="DemoAWT.class"
  WIDTH="350" HEIGHT="120" ALIGN="RIGHT"
  ALT="Willi will Ausdauersport">
Willi will Ausdauersport
</APPLET>
<P> Copyright Prof. Bonin 19-Apr-98 all rights reserved</P>
<ADDRESS>
<A HREF="mailto:hinich-bonin@fbw.fh-lueneburg.de"
  >hinich-bonin@fbw.fh-lueneburg.de</A>
</ADDRES>
</BODY>
<!-- File C:\myjava\ExampAWT.htm                 -->
</HTML>
```

```
/**
 Kleines Beispiel fuer die
 ,,Abstract Window Toolkit'' Bibliothek
 @author Bonin 14-Apr-1998
         update 21-Apr-1998
 @version 1.0
*/
import java.applet.Applet;
import java.awt.* ;
import java.awt.event.ActionListener;

class MaskeAufbau extends Applet{
  Panel topPanel, leftPanel, centerPanel, rightPanel, bottomPanel;

  public void doUserInterface(Frame frame) {
    frame.setLayout(new BorderLayout());
    topPanel    = new Panel();
    leftPanel   = new Panel();
    centerPanel = new Panel();
    rightPanel  = new Panel();
    bottomPanel = new Panel();
    frame.add("North",  topPanel);
    frame.add("West",   leftPanel);
    frame.add("Center", centerPanel);
    frame.add("East",   rightPanel);
    frame.add("South",  bottomPanel);

    MenuBar myMbar = new MenuBar();
```

```
29
30      Menu myMTria = new Menu("Triathlon");
31      myMTria.add(new MenuItem("Schwimmen"));
32      myMTria.add(new MenuItem("Radfahren"));
33      myMTria.add(new MenuItem("Laufen"));
34      myMbar.add(myMTria);
35
36      Menu myMDua = new Menu("Duathlon");
37      myMDua.add(new MenuItem("1. Laufen"));
38      myMDua.add(new MenuItem("Radfahren"));
39      myMDua.add(new MenuItem("2. Laufen"));
40      myMbar.add(myMDua);
41
42      frame.setMenuBar(myMbar);
43    }
44  }
45
46  class MyCanvas extends Canvas {
47    public final int width = 80;
48    public final int height = 120;
49
50    public void paint(Graphics g) {
51      //  x-Achse:  waagerecht von links nach rechts
52      //  y-Achse:  senkrecht von oben nach unten
53      //  rgbWert:  jeweils 0...255
54      int x, y, rgbWert;
55      for (x = 0, y = 0, rgbWert = 0;
56        (x < (width / 2)) && (y < (height / 2) && (rgbWert < 256));
57        x += 2, y += 2, rgbWert += 6) {
58        g.setColor(new Color(255 - rgbWert, rgbWert, 0));
59        g.fillRect(x,y, width - (2 * x), height - (2 * y));
60      }
61      g.setColor(Color.blue);
62      g.drawString("D T U",
63      (width - g.getFontMetrics().stringWidth("D T U"))/2,
64       height / 2);
65    }
66
67    // minimumSize() wird vom Layout-Manger aufgerufen,
68    // um zu erfahren, wie gro der minimale Platz ist,
69    // der bentigt wird.
70    public Dimension minimumSize() {
71      return new Dimension(width + 20, height + 20);
72    }
73
74    // preferredSize() wird vom Layout-Manager aufgerufen,
75    // um zu erfahren, wie gro man es gern htte.
76    public Dimension preferredSize() {
77      return this.minimumSize();
78    }
79  }
80
81  public class DemoAWT extends MaskeAufbau {
82
```

```
 83    Frame myFrame = new Frame("Willi will Ausdauersport!");
 84
 85    public void init() {
 86      DemoAWT myDemo = new DemoAWT();
 87      myDemo.doUserInterface(myFrame);
 88      myFrame.pack();
 89      myFrame.show();
 90    }
 91
 92    public void stop() {
 93      System.out.println("stop() appliziert!");
 94    }
 95
 96    public void doUserInterface(Frame frame) {
 97      super.doUserInterface(frame);
 98
 99      topPanel.setLayout(new GridLayout(1,2));
100      topPanel.add(new Checkbox("DTU-Lizenz"));
101      Choice myC = new Choice();
102      myC.addItem("Kurzdistanz");
103      myC.addItem("Mitteldistanz");
104      myC.addItem("Langdistanz");
105      topPanel.add(myC);
106
107      Button anmelden = new Button("Anmelden!");
108      anmelden.addActionListener(new SimpleListener(frame));
109      leftPanel.add(anmelden);
110
111      centerPanel.add(new MyCanvas());
112
113      Button absagen = new Button("Absagen!");
114      absagen.addActionListener(new SimpleListener(frame));
115      rightPanel.add(absagen);
116
117      int widthDTUinPixel = new MyCanvas().width;
118      bottomPanel.add(new TextArea(
119        "Beschreiben Sie genau Ihren Leistungsstand!",
120        3,widthDTUinPixel / 2));
121    }
122  }
123
124  class SimpleListener implements ActionListener {
125    private Frame fr;
126
127    public SimpleListener(Frame f) {
128      fr = f;
129    }
130
131    public void actionPerformed(java.awt.event.ActionEvent e) {
132      String name = e.getActionCommand();
133      System.out.println("actionPerformed() appliziert: " + name);
134      if (name.equals("Anmelden!")) fr.setTitle("Danke Willi!");
135      if (name.equals("Absagen!")) fr.setTitle("Schade Willi!");
136    }
```

```
137  }
138  //End of file cl3:/u/bonin/myjava/DemoAWT.java
139
```

Skizzieren Sie bei dem folgenden Aufruf das Ergebnis:

```
>appletviewer file://localhost/u/bonin/myjava/ExampAWT.html
```

6.10.5 POET-Beispiel `Buch.java`

In diesem POET-Beispiel läuft der POET-Server auf dem Rechner (IP: 193.174.33.17) mit dem Namen `oodbserver`. Das Erzeugen des Schema `BuchDict` und der Datenbank `BuchDB` erfolgt auf einem anderen Rechner (IP: 193.174.33.100). Damit ein Buchobjekt mit Namen `PKS01` eingespeichert werden kann, müssen entsprechende Zugriffsrechte bei den persistenten Klassen `Buch`, `Person` und `Autor` vorliegen. Da POET in seiner sogenannten Tool-Klasse `PtName` den Namen `PKS01` speichert, muß man auch für die Klasse `PtName` Schreibzugriffsrechte haben.

```
/*
  ptjavac.opt
  Konfiguration fuer Buch-Beispiel
 */

[schemata\dict]
name=BuchDict
server="oodbserver"
onefile=false

[databases\base]
name=BuchDB
schema=dict
location=same
onefile=false

[classes\Buch]
persistent=true
schema=dict

[classes\Autor]
persistent=true
schema=dict

[classes\Person]
persistent=true
schema=dict
```

```
/** Einfaches POET-Uebungsbeispiel
  @author Bonin 10-Jun-1998
  @version 1.0
*/
import COM.POET.odmg.*;
import java.util.*;

public class Buch implements Constraints {
  private String titel;
  private String isbn;
  private int erscheinungsJahr;
  private Autor hauptAutor;
  private String schlagWoerter;
  private transient int alter;
```

```
15    private String verlagsKurzName;
16
17  /* Getter
18   */
19    public String getTitel() {
20      return titel;
21    }
22    public String getIsbn() {
23      return isbn;
24    }
25    public int getErscheinungsJahr() {
26      return erscheinungsJahr;
27    }
28    public Autor getHauptAutor() {
29      return hauptAutor;
30    }
31    public String getSchlagWoerter() {
32      return schlagWoerter;
33    }
34    public int getAlter() {
35      return alter;
36    }
37    public String verlagsKurzName() {
38      return verlagsKurzName;
39    }
40
41  /* Setter
42   */
43    public void setTitel(String titel) {
44      this.titel = titel;
45    }
46    public void setIsbn(String isbn) {
47      this.isbn = isbn;
48    }
49    public void setErscheinungsJahr(int erscheinungsJahr) {
50      this.erscheinungsJahr = erscheinungsJahr;
51    }
52    public void setHauptAutor(Autor hauptAutor) {
53      this.hauptAutor = hauptAutor;
54    }
55    public void setSchlagWoerter(String schlagWoerter) {
56      this.schlagWoerter = schlagWoerter;
57    }
58    public void setAlter(int alter) {
59      this.alter = alter;
60    }
61    public void setVerlagsKurzName(String verlagsKurzName) {
62      this.verlagsKurzName = verlagsKurzName;
63    }
64
65  /* Methode zur Rekonstruktion
66   */
67    public void postRead() {
68      int heute = new Date().getYear();
```

```
69      this.setAlter(heute - this.getErscheinungsJahr());
70    }
71
72  /* Methoden zur Interfaceerfuellung
73   */
74    public void preWrite(){
75      System.out.println("preWrite-Methode appliziert!");
76    }
77    public void preDelete(){
78      System.out.println("preDelete-Methode appliziert!");
79    }
80  }
81
82  // End of File 193.174.34.100 C:\myjava\Buch.java
83
```

```
 1  /** Einfaches POET-Uebungsbeispiel
 2    @author Bonin 10-Jun-1998
 3    @version 1.0
 4  */
 5  import COM.POET.odmg.*;
 6  import java.util.*;
 7
 8  public class Autor extends Person {
 9    private String themen;
10    private String orgKurzName;
11
12    // Konstruktoren
13    public Autor() {};
14    public Autor(String name){
15     this();
16     this.setZuName(name);
17    }
18
19    public String getThemen() {
20      return themen;
21    }
22    public String getOrgKurzName() {
23      return orgKurzName;
24    }
25    public void setThemen(String themen) {
26      this.themen = themen;
27    }
28    public void setOrgKurzName(String orgKurzName) {
29      this.orgKurzName = orgKurzName;
30    }
31  }
32
33  // End of File 193.174.34.100 C:\myjava\Autor.java
```

```
 1  /** Einfaches POET-Uebungsbeispiel
 2    @author Bonin 10-Jun-1998
```

```
 3    @version 1.0
 4  */
 5  import COM.POET.odmg.*;
 6  import java.util.*;
 7
 8  public class Person {
 9    private String zuName;
10    private String vorNamen;
11
12    public String getZuName() {
13      return zuName;
14    }
15    public String getVorNamen() {
16      return vorNamen;
17    }
18    public void setZuName(String zuName) {
19      this.zuName = zuName;
20    }
21    public void setVorNamen(String vorNamen) {
22      this.vorNamen = vorNamen;
23    }
24  }
25  // End of File 193.174.34.100 C:\myjava\Person.java
26
```

```
 1  /** Einfaches Beispiel fuer POET
 2   @author Bonin 10-Jun-1998
 3   @version 1.0
 4   */
 5  import COM.POET.odmg.*;
 6
 7  public class BuchBind {
 8
 9    public static void main(String[] argv) throws ODMGException {
10      Database myDB = Database.open(
11        "poet://oodbserver/BuchDB", Database.openReadWrite);
12      Transaction myT = new Transaction(myDB);
13      myT.begin();
14      try {
15        Buch myBuch = new Buch();
16        myBuch.setTitel(" Softwarekonstruktion mit LISP");
17        myBuch.setIsbn("3-11-011786-X");
18        myBuch.setErscheinungsJahr(91);
19
20        myBuch.setHauptAutor(new Autor("Bonin"));
21        myBuch.setSchlagWoerter("Arbeitstechniken, Qualitaet");
22        myBuch.postRead();
23        myBuch.setVerlagsKurzName("WalterDeGruyter");
24
25        System.out.println("Zuname des Autors: " +
26          myBuch.getHauptAutor().getZuName() +
27          "\nAlter des Buches : " +
28          myBuch.getAlter());
```

```
29        myDB.bind(myBuch, "PKS01");
30      }
31      catch (ObjectNameNotUniqueException exc) {
32        System.out.println("PKS01 gibt es schon!");
33      }
34      catch (ODMGRuntimeException exc) {
35        myT.abort();
36        throw exc;
37      }
38      myT.commit();
39      myDB.close();
40    }
41  }
42  // End of File 193.174.34.100 C:\myjava\BuchBind.java
43
```

```
 1  /** Einfaches Beispiel fuer POET
 2   @author Bonin 10-Jun-1998
 3   @version 1.0
 4   */
 5  import COM.POET.odmg.*;
 6
 7  public class BuchLookup {
 8
 9    public static void main(String[] argv) throws ODMGException {
10      Database myDB = Database.open(
11        "poet://oodbserver/BuchDB", Database.openReadWrite);
12      Transaction myT = new Transaction(myDB);
13      myT.begin();
14      try {
15        Buch myBuch = (Buch) myDB.lookup("PKS01");
16
17        System.out.println("Zuname des Autors: " +
18          myBuch.getHauptAutor().getZuName() +
19          "\nAlter des Buches : " +
20          myBuch.getAlter());
21      }
22      catch (ODMGRuntimeException exc) {
23        myT.abort();
24        throw exc;
25      }
26      myT.commit();
27      myDB.close();
28    }
29  }
30  // End of File 193.174.34.100 C:\myjava\BuchLookup.java
31
```

Skizzieren Sie bei dem folgenden Aufruf das Ergebnis:

```
>java BuchLookup
```

6.10.6 Vererbung in Java

Die folgende Java-Quellcodedatei `Foo.java` wurde fehlerfrei compiliert.

```
>java -fullversion
java full version "JDK 1.1.6 IBM build a116-19980529" (JIT: jitc)
>javac Foo.java
>
```

Foo.java

```
 1  /** Vererbungsbeispiel
 2    @author Bonin 30-Jun-1998
 3    @version 1.0
 4  */
 5  public class Foo {
 6    public static class KlasseA {
 7      public static int updateAnzahlSlot = 0;
 8      private  String slot = "KlasseA";
 9      public String getSlot() {
10        return slot;
11      }
12      public void setSlot(String slot) {
13        updateAnzahlSlot = updateAnzahlSlot + 1;
14        this.slot = slot;
15      }
16    }
17    public static class KlasseB extends KlasseA {
18      public static int updateAnzahlSlot = 0;
19      private  String slot = "KlasseB";
20      public String getSlot() {
21        return slot;
22      }
23      public void setSlot(String slot) {
24        updateAnzahlSlot = updateAnzahlSlot + 1;
25        this.slot = slot;
26      }
27    }
28    public static class KlasseC extends KlasseB {
29      public static int updateAnzahlSlot = 0;
30      private  String slot = "KlasseC";
31      public String getSlot() {
32        return slot;
33      }
34      public void setSlot(String slot) {
35        updateAnzahlSlot = updateAnzahlSlot + 1;
36        this.slot = slot;
37      }
38    }
39    public static class Bar {
40      public static void main(String argv[]) {
41        KlasseA a = new KlasseA();
42        KlasseB b = new KlasseB();
```

```
43        KlasseC c = new KlasseC();
44        b.setSlot(a.getSlot());
45        c.setSlot(b.getSlot());
46        a.setSlot(c.getSlot());
47        System.out.println("Slot-Wert in Instanz c: " +
48         c.getSlot() +
49         "\nAnzahl der Änderungen in KlasseA: " +
50          KlasseA.updateAnzahlSlot);
51     }
52   }
53  }
54  // End of File:cl3:/u/bonin/mywww/anwd/SS98/Foo.java
```

Erzeugte Dateien angeben

Geben Sie an, welche Dateien nach dem Compilieren von `Foo.java` entstanden sind.

Java-Aufruf angeben

Ersetzen Sie im folgenden Aufruf die drei Punkte.
`>java Foo...`
`>`
Geben Sie an, für welche Plattform (AIX oder NT) Ihr Ersatz gilt.

Ergebnis des `java`-Aufrufes angeben

Geben Sie das Ergebnis Ihres Aufrufes (`> java Foo...`) an.

6.10.7 Java-Programm schreiben

Die Datei `TelefonBuchProg.java` enthält den Java-Quellcode für ein sehr einfaches Telefonbuch. Notiert ist dabei primär nur der Teil, der für die Persistenz der Eintragen in das Telefonbuch sorgt.

TelefonBuchProg.java

```
 1  /**
 2   Einfaches permanentes Telefonbuch mit Schreibtest
 3   @author Bonin 29-Jun-1998
 4   @version 1.0
 5  */
 6  import java.io.* ;
 7  import java.util.* ;
 8  /**
 9   TelefonBuch
10   */
11  class TelefonBuch implements Serializable {
12    Hashtable tabelle;
13    public TelefonBuch() {
14      tabelle = new Hashtable();
15    }
16    public TelefonEintrag getEintrag(String key) {
```

```
17      return (TelefonEintrag) tabelle.get(key);
18    }
19    public TelefonEintrag addEintrag(String key, TelefonEintrag te) {
20      return (TelefonEintrag) tabelle.put(key, te);
21    }
22    public int size() {
23      return tabelle.size();
24    }
25    public boolean gleichheit(TelefonBuch t) {
26      if ((t == null) || (size() != t.size()))
27        return false;
28      Enumeration keys = tabelle.keys();
29      while (keys.hasMoreElements()) {
30        String key = (String) keys.nextElement();
31        TelefonEintrag myTe = getEintrag(key);
32        TelefonEintrag otherTe = t.getEintrag(key);
33        if (!myTe.gleichheit(otherTe))
34          return false;
35      }
36      return true;
37    }
38  }
39  /**
40   TelefonEintrag
41   */
42  class TelefonEintrag implements Serializable {
43    String kurzname;
44    String telefon;
45    public TelefonEintrag(String kurzname, String telefon) {
46      if ((kurzname == null) || (telefon == null))
47        throw new IllegalArgumentException();
48      this.kurzname = kurzname;
49      this.telefon = telefon;
50      System.out.println("TelefonEintrag: " +
51        kurzname + " " + telefon);
52    }
53    public boolean gleichheit(TelefonEintrag te) {
54      return (kurzname.equalsIgnoreCase(te.kurzname)) &&
55             (telefon.equalsIgnoreCase(te.telefon));
56    }
57  }
58  /**
59   TelefonBuchProg
60   */
61  public class TelefonBuchProg {
62    public static void main(String argv[]) {
63      TelefonBuch t = new TelefonBuch();
64      t.addEintrag("Key1",
65        new TelefonEintrag("Otto", "+49/4131/677175"));
66      t.addEintrag("Key2",
67        new TelefonEintrag("Emma", "+49/4131/677144"));
68      try {
69        FileOutputStream fout = new FileOutputStream("tbuch.ser");
70        ObjectOutputStream out = new ObjectOutputStream(fout);
```

```
71        out.writeObject(t);
72        out.close();
73        /* Zur Kontrolle: Wiedereinlesen und Vergleichen */
74        FileInputStream fin = new FileInputStream("tbuch.ser");
75        ObjectInputStream in = new ObjectInputStream(fin);
76        TelefonBuch copy = (TelefonBuch) in.readObject();
77        in.close();
78        if (t.gleichheit(copy))
79          System.out.println("OK --- Objekte sind gleich!");
80        else
81           System.out.println("Fehler --- Objekte sind ungleich!");
82        }
83        catch (Exception e) {
84          e.printStackTrace(System.out);
85        }
86    }
87  }
88  //End of file cl3:/u/bonin/myjava/TelefonBuchProg.java
```

Ergebnis von `java TelefonBuchProg` angeben

Geben Sie das Ergebnis von „`java TelefonBuchProg`“ an.

Programmieren von `TelefonLookupProg.java`

Die Java-Applikation `TelefonLookupProg` erfüllt folgende Anforderungen:

1. `TelefonLookupProg` nutzt das permanente Telefonbuch von `TelefonBuchProg`
2. `TelefonLookupProg` nutzt die Klassen `TelefonEintrag` und `TelefonBuch`
3. `TelefonLookupProg` sucht für einen vorgegebenen Kurznamen die Telefonbucheintragung und gibt den Wert von `kurzname` und von `telefon` aus.
4. Der vorgegebene Kurzname (Wert von `kurzname`) wird beim Aufruf als Argument genannt, zum Beispiel:
   `>java TelefonLookupProg Emma`
5. Findet `TelefonLookupProg` keine Eintragung, dann gibt es keine Ausgabe und die Applikation wird beendet.

Notieren Sie einen Quellcode für diese Java-Applikation `TelefonLookupProg`.

6.10.8 Objektbeziehungen notieren

In einer Diskussionsrunde zwischen den Verantwortlichen für die Softwareentwicklung werden die folgenden Aussagen festgestellt:

1. `K1` ist eine Java-Applikation
2. Klasse `K1` implementiert das Interface `I0`
3. `I0` umfaßt die Methoden `m1()` und `m2()`

4. `K1` hat die Instanzvariablen `v1`, `v2`, `v3` vom Typ `K4`
5. Klasse `K2` enthält eine Instanz `s` der Klasse `K1`
6. Klasse `K3` ist Subklasse von `K2`
7. `K3` hat die Klassenvariable `c1` vom Typ `K4`
8. Klasse `K4` hat die Methode `m3()`

Objektbeziehungen in Java abbilden

Bilden Sie die obigen Aussagen in Java-Quellcode ab.

Getter- und *Setter*-Methoden ergänzen

Herr Franz Otto ist Anhänger des Java-Beans-Modell. Er möchte unbedingt die Zugriffsmethoden mit dargestellt sehen. Ergänzen Sie daher Ihren Java-Quellcode um die sogenannten *Getter*- und *Setter*-Methoden.

Kapitel 7

Java-Konstruktionsempfehlungen

Trainingsplan

Das Kapitel „Java-Konstruktionsempfehlungen“ erläutert:

- die Idee von Geschäftsobjekten (*Common Business Objects*) und Geschäftsvorgängen (*Core Business Processes*) und
  ↪ Seite 189 . . .
- Alternativen für die Quellcodegestaltung aus Sicht der Performance.
  ↪ Seite 190 . . .

7.1 Java-Rahmen für Geschäftsobjekte und -prozesse

Text derzeit noch in Bearbeitung! (Stand: 10. November 1998)

Francisco

Das *San Francisco Project*[1] der IBM Corporation stellt bewährte „Musterobjekte“ (*Common Business Objects*) und „Musterprozesse“ (*Core Business Processes*) für verschiedene Anwendungsfelder in der kommerziellen Datenverarbeitung bereit. Dazu zählen zum Beispiel:

Objekte

- Geschäftsobjekte (*Common Business Objects*):
  - Adresse
  - Geschäftspartner (Kunde, Lieferant)
  - Kalender (zum Beispiel perioden-basiert)

[1] Aktuelle Informationen zum San Francisco Projekt der IMB Corporation: `http://www.ibm.com/java/sanfrancisco` (Zugriff: 08-Jul-1998)

  - Indentifierungs-Serien für Dokumente, Konten usw.
  - Währungen

Prozesse

- Geschäftsvorgänge (*Core Business Processes*):
  - Zahlungsverkehr, Finanzwesen (*Business Financials*))
  - Verkaufs- und Angebotsverwaltung (*Order Management*
  - Empfangen und Versenden von Waren *Warehouse Management*

Solche gut getesteten *Common Business Objects* und *Core Business Processes* können direkt in den Quellcode einer Java-Anwendung importiert werden. Sie bilden die eigentliche Basis für den Entwickler. Die Java-Konstrukte des JDKs dienen nur noch als „Mörtel“ für das Zusammenpassen der vorgefertigten Bausteine.

7.2 Java Coding Tips

Text derzeit noch in Bearbeitung! (Stand: 10. November 1998)

7.2.1 Konstante statt Pseudo-Variable ↪ Performance

Immer wenn unstrittig feststeht, daß ein Wert für alle Objekte gleich bleibt, sich also garantiert nicht ändert, ist eine Konstante statt einer Variablen zu wählen. Die Angabe der Modifieren `static` und `final` ermöglicht dem Compiler einen wesentlich effizienteren Code zu erzeugen.

Langsame Lösung

```
String myString = "Bleibt immer so!";
```

Effizientere Lösung

```
static final String myString = "Bleibt immer so!";
```

7.2.2 Anordnen von Ausnahmen (*Exceptions*)

Sind mehrere Ausnahmen zu programmieren, dann stellt sich die Frage ihrer Anordnung. Die Praxis, jeden Methodenaufruf, der eine Ausnahme bewirken kann, in ein eigenes `try-catch`-Konstrukt einzuschließen, macht den Quellcode schwer durchschaubar. Zusätzlich erschwert es eine Ablaufoptimierung des Compilers. Es ist daher besser, in einem größeren `try`-Block die Methodenaufrufe zusammen zu fassen und die `catch`-Blöcke danach zu notieren.

Schlecht durchschaubare `try-catch`-Anordnung

```
private void irgendEtwas() {
  try {
    foo.methodA();
  }
  catch (methodAException eA) {
    // Code zur Behandlung der Ausnahme eA
  }
  try {
    foo.methodB();
  }
  catch (methodBException eB) {
    // Code zur Behandlung der Ausnahme eB
  }
  try {
    foo.methodC();
  }
  catch (methodCException eC) {
    // Code zur Behandlung der Ausnahme eC
  }
}
```

Bessere `try-catch`-Anordnung

```
private void irgendEtwas() {
  try {
    foo.methodA();
    foo.methodB();
    foo.methodC();
  }
  catch (methodAException eA) {
    // Code zur Behandlung der Ausnahme eA
  }
  catch (methodBException eB) {
    // Code zur Behandlung der Ausnahme eB
  }
  catch (methodCException eC) {
    // Code zur Behandlung der Ausnahme eC
  }
}
```

Ersatz durch `throws`-Konstrukt In manchen Fällen kann das `try-catch`-Konstrukt durch `throws` ersetzt werden. Die Aufgabe wird dann dem *Caller* übertragen.

```
private void irgendEtwas()
  throws methodAException, methodBException, methodCException {
```

```
  foo.methodA();
  foo.methodB();
  foo.methodC();
}
```

7.2.3 Häufige String-Modifikationen mit `StringBuffer`

Weil ein Objekt vom Typ `String` per Definition nicht änderbar (*immutable*) ist, wird bei jeder Modifikation ein neues String-Objekt erzeugt. Die Zwischenresulate bei mehreren Manipulationen sind dann alles Objekte, deren Speicherplatz wieder freizugeben ist, also Arbeit für den *Garbage Collector*. Der `StringBuffer` ist ein modifizierbares Objekt. Er sollte daher stets benutzt werden, wenn viele Manipulationen an einer Zeichenkette erforderlich sind.

Aus dem gleichen Grund sollte auch eine Konstruktion mit `StringBuffer` und `append` einer Konstruktion mit dem Konstrukt „`+`“ vorgezogen werden.

Schlechte Lösung

```
String myString = "Alles";
String klar     = "klar?";
myString += " ";
myString += klar;
```

Bessere Lösung

```
StringBuffer myBuffer = new StringBuffer(11);
myBuffer.append("Alles ");
myBuffer.append("klar?");
String myString = myBuffer.toString();
```

Noch bessere Lösung

```
String myString = "Alles" + " " + "klar?";
```

da vom Compiler in denselben Bytecode verwandelt wie (→[IBM-Francisco98]):

```
String myString = "Alles klar?";
```

Kapitel 8

Dokumentieren mit HTML

HyperTextMarkup Language (HTML), Version 4.0, ist eine umfassende Sprache zum Publizieren im *World Wide Web* (WWW). Sie ist hervorragend geeignet für das Dokumentieren eines Softwaresystems, da auf einfache Art und Weise Informationen in verschiednen Dokumenttypen (zum Beispiel: Modell-, Test- und Quellcode-Dokumenten) verknüpft werden können. Mit dem CSS-Konzept (*Cascading Style Sheets*) können die vielen einzelnen Dokumenten aus den verschiednenen Quellen (Entwicklungswerkzeugen) einheitlichen und damit leichter überschaubar präsentiert werden.

Trainingsplan

Das Kapitel „Dokumentation mit HTML“ erläutert:

- Möglichkeiten von HTML 4.0 und
  ↪ Seite 193 ...
- die Layout-Gestaltung mit Hilfe von *Cascading Style Sheets* (CSS).
  ↪ Seite 196 ...

8.1 HTML 4.0

HyperTextMarkup Language (HTML), Version 4.0, ergänzt die ursprünglichen, einfachen HTML-Konstrukten (→Versionen 1 & 2) um leistungsfähige Mechanismen für die Softwaredokumentation. Zu nennen sind hier zum Beispiel:

- einheitliche Layoutgestaltung über mehrere Dokumente (*Cascading Style Sheet* (CSS) (→Abschnitt 8.2 auf Seite 196)
- Einbindung von Script-Sprachen (*Scripting* — zum Beispiel Javascript)

- Bildflächenaufteilung (*Frames*)
- Integrierte Objekte (*Embedding Objects*)
- Maskengestaltung (*Forms*)
- geschachtelte Tabellen (*Tables*)
- Textausrichtung nach rechts, links, mittig.

Solche Gestaltungsmöglichkeiten gab es schon ansatzweise in der Version 3.2 und/oder durch die browser-spezifischen Konstrukte von Netscape und Microsoft. Mit HTML 4.0 sind die Mechanismen als SGML[1]-Konstrukte strikter definiert und in ein logisches Gesamtkonzept integriert.

Diese vielfältigen Möglichkeiten können hier nicht dargestellt werden. Präzise Beschreibungen sind der HTML 4.0 Spezifikation (→[HTML4.0]) entnehmbar. Im folgenden wird nur das CSS-Konzept behandelt, da es ermöglicht, Daten aus verschiedenen Quellen im Softwareerstellungsprozeß einheitlich zu präsentieren. Für die Teamarbeit besteht damit eine leicht umsetzbare Konvention für das Definieren und Einhalten eines gemeinsamen Projekt-Layouts.

Das folgende Beispiel `Overview.html` soll daher nur einige HTML-4.0-Optionen skizzieren.

Beispiel `Overview.html`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
  "http://www.w3c.org/TR/REC-html40/strict.dtd">
<HEAD>
<BASE href="http://cl3.fbw.fh-lueneburg.de:6667/bonin/">
<TITLE>Bonin's Opening</TITLE>
<META http-equiv="Last-Modified"
      content="20-Oct-98 13:00:00 GMT">
<META http-equiv="Expires"
      content="01-Jan-99 00:00:00 GMT">
<META http-equiv="Content-Script-Type"
      content="text/javascript">
<META name="KEYWORDS"
      content="HTML-Beispiele, Arbeiten von H. Bonin">
<META name="DESCRIPTION"
      content="Bonin, Java, Triathlon">
<LINK rev=owns
      title="Hinrich E.G. Bonin"
      href="mailto:hinrich-bonin@fbw.fh-lueneburg.de">
<LINK HREF="http://as.fh-lueneburg.de:6667/bonin/myStyle.css" REL="stylesheet" TYPE="t
<SCRIPT type="text/javascript" language="JavaScript">
<!--
  var browserName    = navigator.appName;
  var browserVersion = parseInt(navigator.appVersion);
  var checkBrowser    = "NichtOK";
  if ((browserName == "Netscape" && browserVersion >= 3) ||
    (browserName.indexOf('Explorer') != -1 && browserVersion >=4))
```

[1] SGML ≡ *Standard Generalized Markup Language*, ISO-Standard 8879:1986

```
27      {
28      checkBrowser = "OK";
29      }
30    function goMyWeb(){
31      if (checkBrowser == 'OK')
32        setTimeout('location="http://cl3.fbw.fh-lueneburg.de:6667/bonin/index.html"',200
33    }
34  // -->
35  </SCRIPT>
36  <STYLE type="text/css">
37    BODY {
38      background:
39      url(http://cl3.fbw.fh-lueneburg.de:6667/bonin/gelberBall.gif);
40      text-align: center
41    }
42    H1 {
43      border-width: 1; border: solid;
44      background-color: teal;
45      color: red;
46    }
47    #myTD {
48      font-size: 24pt;
49      color: black;
50      background-color: white;
51    }
52    #myA {
53      font-size: 24pt;
54      color: black;
55      background-color: white;
56    }
57  </STYLE>
58  </HEAD>
59  <BODY onload="goMyWeb()">
60  <TABLE>
61    <TR>
62      <TD><A href="index.html"><IMG src="krankerhinrich_kl.gif"
63             ALT="Wohnort Reppenstedt"></A></TD>
64    </TR>
65  </TABLE>
66  <TABLE>
67    <TR>
68      <TD id="myTD">Prof. Dr. Hinrich E. G.
69          <A href="index.html">Bonin</A></TD>
70    </TR>
71    <TR>
72      <TD><A href="mailto:hinrich-bonin@fbw.fh-lueneburg.de">
73        hinrich-bonin@fbw.fh-lueneburg.de</A></TD>
74    </TR>
75  </TABLE>
76
77  <H1> <A href="http://www.fh-lueneburg.de">
78    Fachhochschule Nordostniedersachsen</A>
79  </H1>
80  <H1><A href="http://www.fh-lueneburg.de/stadt/k_luene.html">
```

```
81    L&uuml;neburg, Germany</A></H1>
82  <P><A href="http://validator.w3.org/"><IMG
83    src="vh40.gif"
84    ALT="Valid HTML 4.0!" HEIGHT=31 WIDTH=88></A>
85  </BODY>
86  </HTML>
```

8.2 Cascading Style Sheets (*CSS*)

DTP

Cascading Style Sheet (CSS1 ≡ Level 1) ist ein Mechanismus, der es sowohl dem Autor wie dem Leser ermöglicht Farben, Schriftart, Schriftgröße und Zwischenräume (≈ das gewünschte Layout) mit dem Dokument zu verknüpfen. CSS1 ist eine leicht lesbare und schreibbare Spezifikationssprache, die sich an der üblichen Desktop Publishing Terminology orientiert.

8.2.1 CSS-Konzept

Eine einfache CSS-Regel hat folgende Form:

selektor { eigenschaft: wert }

Zum Beispiel wird mit folgender Regel die Hauptüberschrift als roter Text dargestellt:

```
H1 { color: red }
```

Eine Regel besteht aus zwei Hauptteilen:

1. Selektor
   Im Beispiel: `H1`

2. Deklaration
   Im Beispiel: `color: red`
   Die Deklaration hat zwei Teile:

   (a) Eigenschaft (*property*)
   Im Beispiel: `color`

   (b) Wert (*value*)
   Im Beispiel: `red`

Der Selektor bildet die Verknüpfung zwischen dem HTML-Konstrukt und der Spezifikation des *Style Sheet*. Alle HTML-Elementtypen sind mögliche Selektoren. Die Eigenschaft `color` ist beispielsweise eine von ≈ 50 Eigenschaften, die die Präsentation festlegen. Der Autor eines HTML-Dokuments braucht nur seine speziellen Vorstellungen über die spätere Präsentation zu spezifizieren, weil der Web-Browser (*User Agent*) ein *Default Style Sheet* besitzt. Der Autor überschreibt mit seiner Spezifikation die Default-Werte.

8.2.2 HTML-Dokument ⇔ CSS

Das HTML-Dokument kann von dem *Style Sheet* auf verschiedene Weise Kenntnis erhalten. Das folgende Beispiel zeigt vier Möglichkeiten:

1. im `<HEAD>`-Konstrukt
   (a) mit dem `<LINK>`-Konstrukt wird ein Verweis auf eine externe CSS-Datei angeben und diese wird über den Web-Server geladen laden
   (b) mit dem `<STYLE>`-Konstrukt und der `@import`-Angabe wird ein Verweis auf eine externe CSS-Datei angegeben und über den Web-Server geladen
   (c) direkt codiert im `<STYLE>`-Konstrukt
2. im `<BODY>`-Bereich
   - mit dem `STYLE`-Attribut eines HTML-Elementes


```
<HTML>
  <HEAD>
    <LINK  HREF="myStyle.css" REL="stylesheet" TYPE="text/css">
    <TITLE>Mein Dokument</TITLE>
    <STYLE TYPE="text/css">
      @import url(http://as.fh-lueneburg.de/main.css);
      H1 { color: red }
    </STYLE>
  </HEAD>
  <BODY>
    <H1>Mein Dokument in Rot</H1>
    <P STYLE="color: blue">Mein blauer Text</P>
  </BODY>
</HTML>
```

8.2.3 Gruppierung & Vererbung

Zur Verkürzung der CSS-Länge können Selektoren in Form einer Liste gruppiert werden. Zum Beispiel:

```
H1, H2, H3 { font-family: Times }
```

Oder auch in Kurzschreibweise notiert werden. Zum Beispiel:

```
H1 { font: bold 12pt/14pt Arial }
```

statt ausführlich:

```
H1 {
  font-weight:  bold;
  font-size:    12pt;
```

```
  line-height:  14pt;
  font-family:  Arial;
  font-variant: normal;
  font-style:   normal;
}
```

Bei geschachtelten Konstrukten erben die inneren Konstrukte Eigenschaften von den äußeren Konstrukten. Gilt zum Beispiel für den Selektor `<H1>`:

```
H1, H2 {
  font-size:   24pt;
  font-weight: bold;
  font-family: Arial, Helvetica;
  color:       yellow;
  background-color: blue;
  margin: 5px;
}
```

und den Selektor `<EM>`:

```
EM {
  font-style:italic
}
```

dann ist die folgende Überschrift ganz in Gelb auf blauem Hintergrund geschrieben.

```
<H1><EM>CTP</EM>-Dokumentation</H1>
```

Das `EM`-Konstrukt erhält seine Farbe vom „Parent element“, hier: `<H1>`. Nicht jede Eigenschaft wird vererbt. So wird beispielsweise die Eigenschaft `background` nicht vererbet. Es empfiehlt sich daher bei einer Angabe von `color` stets auch eine Angabe für `background` zu machen.

```
BODY {
  background: url(http://as.fh-lueneburg.de/gelberBall.gif) black;
  color: white;
}
```

Im obigen Beispiel ist die Textfarbe weiß und der Hintergrund wird aus dem Bild „gelber Ball“ gebildet. Ist das Bild kleiner als die Hintergrundfläche wird das Bild wiederholt dargestellt. Die „Zwischenräume“ werden mit der zweiten Angabe von `background` aufgefüllt; hier also schwarz dargestellt. Die zweite Angabe wird auch benutzt, wenn das Bild nicht zugreifbar ist.

Bei der Spezifikation von einer Eigenschaft kann man sich auf andere Eigenschaften beziehen. Ein Beispiel ist die Prozentangabe bei der Eigenschaft `line-height`.

```
P {
  font-size:   14pt;
  line-height: 150%;
}
```

8.2.4 Selektor: CLASS & ID

CLASS

Man kann Eigenschaften zu einer Klasse zusammenfassen. Die Klasse wird benannt und mit einem Punkt unmittelbar hinter dem Selektor notiert. Eine Klasse die für mehrere Selektoren genutzt werden soll wird ohne Selektorangabe mit einem Punkt beginnend spezifiziert. Es kann nur eine Klasse pro Selektor spezifiziert werden, wie das folgende Beispiel skizziert.

```
H1.myKopf {
  color:            yellow;
  background-color: black;
}
H1 {
  color:            red;
  background-color: black;
}
.myClass {
  color:            blue;
  background-color: maroon;
}
...
<H1 CLASS="myKopf">Das Gelbe vom Ei</H1>
<H1>Der rote Kopf</H1>
<H1 CLASS="myClass">Das Blaue vom Himmel</H1>
<P CLASS="myClass">Blau, blau ... </P>
...
Geht nicht!
<H1 CLASS="myKopf" CLASS="myClass">Fehler</H1>
```

ID

Mit dem ID-Attribut wird üblicherweise einem einzelnen Element eine CSS-Spezifikation zugeordnet. Der ID-Wert muß im Dokument eindeutig sein. Er wird beginnend mit einem Hashzeichen „#“ notiert, wie das folgende Beispiel zeigt.

```
#ZZ981 { letter-spacing: 0.3em }
#ZZ982 { letter-spacing: 0.5em }
...
<P ID="ZZ982">Buchstaben mit viel Zwischenraum</P>
```

8.2.5 Kontextabhängige Selektoren

Im folgenden CSS-Beispiel sind alle <EM>-Konstrukte im Dokument von der Spezifikation (grüne Textfarbe) betroffen:

```
H1 {
  color: red;
}
EM {
  color: green;
}
```

Soll sich die Spezifikation nur auf <EM>-Konstrukte innerhalb eines <H1>-Konstruktes beziehen, dann kann ein kontextabhängiger Selektor wie folgt notiert werden

```
H1 {
  color: red;
}
H1 EM {
  color: green;
}
```

Auch solche kontextabhängigen Selektoren sind grupperbar. Zum Beispiel entspricht

```
H1 EM, H2 B {
  color: blue;
}
```

der Spezifikation

```
H1 EM {
  color: blue;
}
H2 B {
  color: blue;
}
```

8.2.6 Kommentare im CSS

/*...*/

Ein Kommentar wird innerhalb eines CSS mit der Slash-Sternchen-Kombination gekennzeichnet, ähnlich wie in Java oder C.

```
/* Bild wird häufig nicht angezeigt */
UL {
  list-style-image: url(http://as.fh-lueneburg.de/gelberBall.gif) white;
  list-style-position: inside;
}
```

8.2.7 Pseudo-Konstrukte (A:link, P:first-letter, usw.)

Üblicherweise zeigt ein Web-Browser neue Links (Anker: A-Konstrukte) in anderem Layout an als die schon „besuchten“. Ihr Layout läßt sich mit Hilfe der sogenannten *Anchor Pseudo-Classes* spezifizieren. Pseudo-Konstrukte werden ähnlich wie Klassen angeben, allerdings mit Doppelpunkt und nicht mit Punkt getrennt.

```
A:link {          /* unbesuchte Link */
  color: red;
}
A:visited  {      /* aufgesuchter Link */
```

```
  color: blue;
}
A:active {       /* aktiver Link */
  color: green;
}
```

Die Pseudo-Konstrukte `first-line` und `first-letter` werden benutzt um einen Absatz zu gestalten, zum Beispiel mit einem großen Buchstaben am Anfang.

```
P:first-letter {
  font-size: 24pt;
  float: left;
  color: yellow;
  background-color: black;
}
```

Dabei können Pseudo-Konstrukte mit Klassen in den Selektoren verknüpft werden, wie das folgende Beispiel zeigt:

```
P.anfang:first-letter {
  color: yellow;
  background-color: black;
}
...
<P CLASS="anfang">Erster Ansatz im Text</P>
...
```

8.2.8 Die Kascade & Konflikte

Ein HTML-Dokument kann von mehr als einer CSS-Spezifikation beeinflußt werden. Verantwortlich sind dafür zwei Aspekte:

- Modularität: mehr als eine CSS-Angabe in einem HTML-Dokument
  Zum Beispiel:

  ```
  @import url(http://as.fh-lueneburg.de/mainStlye.css);
  @import url(http://as.fh-lueneburg.de/myStlye.css);
  H1 {
    color: blue /* ueberschreibt importierte Sheets */
  }
  ```

- Autor⇔Leser-Balance
  Der Leser kann mit seinem Style Sheet die Autoren-Vorgaben beeinflussen (→`important`-Deklaration).

Die CSS-Angaben für ein HTML-Dokument können Konflikte aufweisen. Diese werden mit Hilfe von Gewichtungsfaktoren gelöst. So ist normalerweise das Gewicht der Leser-Spezifikation geringer als das Gewicht der Autoren-Spezifikation. Es sei denn, in der Leser-Spezifikation wird eine Eigenschaft-Wert-Angabe mit ! `important` gekennzeichnet.

```
H1 {
  color: black ! important;
}
P {
  font-size: 12pt ! important;
  font-style: italic
}
```

Die Farbangabe eines Lesers für den obigen `H1`-Selektor überschreibt eine Farbangabe des Autors, weil diese mit `! important` markiert ist.

Um CSS-Konflikte zu lösen, werden CSS-Eigenschaft-Wert-Angaben nach folgender Vorgehensweise abgearbeitet (→[LieBos96] Chapter 3):

1. Für einen Selektor werden alle Eigenschaft-Wert-Angaben festgestellt.
2. Gibt es keine entsprechende Angabe wird die geerbte Angabe eingesetzt. Gibt es keine geerbte, dann wird der Initialwert verwendet.
3. Die Angaben werden nach Gewicht sortiert. Als wichtig gekennzeichnet Angaben (`! important`) haben dabei ein höheres Gewicht.
4. Die Angaben werden nach der Quelle sortiert. Dabei gilt: Autoren-Angeben haben mehr Gewicht als Leser-Angaben. Diese haben mehr Gewicht als Einstellungen des Web-Browsers (*User Agent*).
5. Die Angaben werden nach dem „Grad der Spezifizierung“ sortiert. Spezielle Angaben überschreiben generelle Angaben. Dieser Grad wird wie folgt ermittelt:

   α `ID`-Feststellung

   β `CLASS`-Feststellung

   γ Feststellung der Anzahl der HTML-Tags in der Deklaration

   Das folgende Beispiel skizziert die Gewichtsermittlung:

   ```
   LI {...}              /* α = 0 β = 0 γ = 1 ⇔→ Gewicht= 1 /*
   UL LI {...}           /* α = 0 β = 0 γ = 2 ⇔→ Gewicht= 2 /*
   UL OL LI {...}        /* α = 0 β = 0 γ = 3 ⇔→ Gewicht= 3 /*
   LI.red {...}          /* α = 0 β = 1 γ = 1 ⇔→ Gewicht= 11 /*
   UL OL LI.red {...}    /* α = 0 β = 1 γ = 3 ⇔→ Gewicht= 13 /*
   #X123 {...}           /* α = 1 β = 0 γ = 0 ⇔→ Gewicht= 100 /*
   ```

   Dabei zählen die *Pseudo*-Selektoren wie zum Beispiel `A:link` als normale Elemente.
6. Bei Angaben mit gleichem Gewicht wird die zuletzt spezifizierte gewählt.

8.2.9 CSS-Beispiel

Das HTML-Dokument `exampleCSS.html` enthält in seinem `<LINK>`-Konstrukt einen Verweis auf die CSS-Datei `myStyle.css`. Die Abbildung 8.1 auf Sei-

te 204 zeigt die Darstellung eines Auszugs des Dokumentes mit dem Browser Netscape Communicator Version 4.04 auf einer AIX-Plattform.[2]

CSS-Datei: `myStyle.css`

```
/* Basis-Layout fuer das Projekt: FOO  */
/*   Bonin 23-06-98                   */
/*   Update                           */
/* Achtung: Kaum ein Browser zeigt    */
/*          dieses Layout korrekt!    */


/* Bild wird häufig nicht angezeigt */
UL {
  color: white;
  background-color: black;
  list-style-image: url(http://193.174.33.106:6667/bonin/gelberBall.gif);
  list-style-position: inside;
}
OL {
  color: white;
  background-color: black;
  list-style-type: lower-roman;
}

P:first-letter {
  font-size: 24pt;
  float: left;
  color: yellow;
  background-color: black;
}
P {
  font-family:    "Times New Roman", Times, serif;
  font-weight:    bold;
  font-size:      14pt;
  line-height:    150%;
  letter-spacing: 0.5em;
  color: green;
  background-color: yellow;
}
H3 {
  font-family: Arial, Helvetica, sans-serif;
  font-size:   20pt;
  font-weight: bold;
  text-decoration: underline;
  color: green;
  background-color: white;
  margin: 3em;
}
H1, H2 {
  font-size:   24pt;
  font-weight: bold;
```

[2] mit reduzierter Auflösung $120dpi * 120dpi$

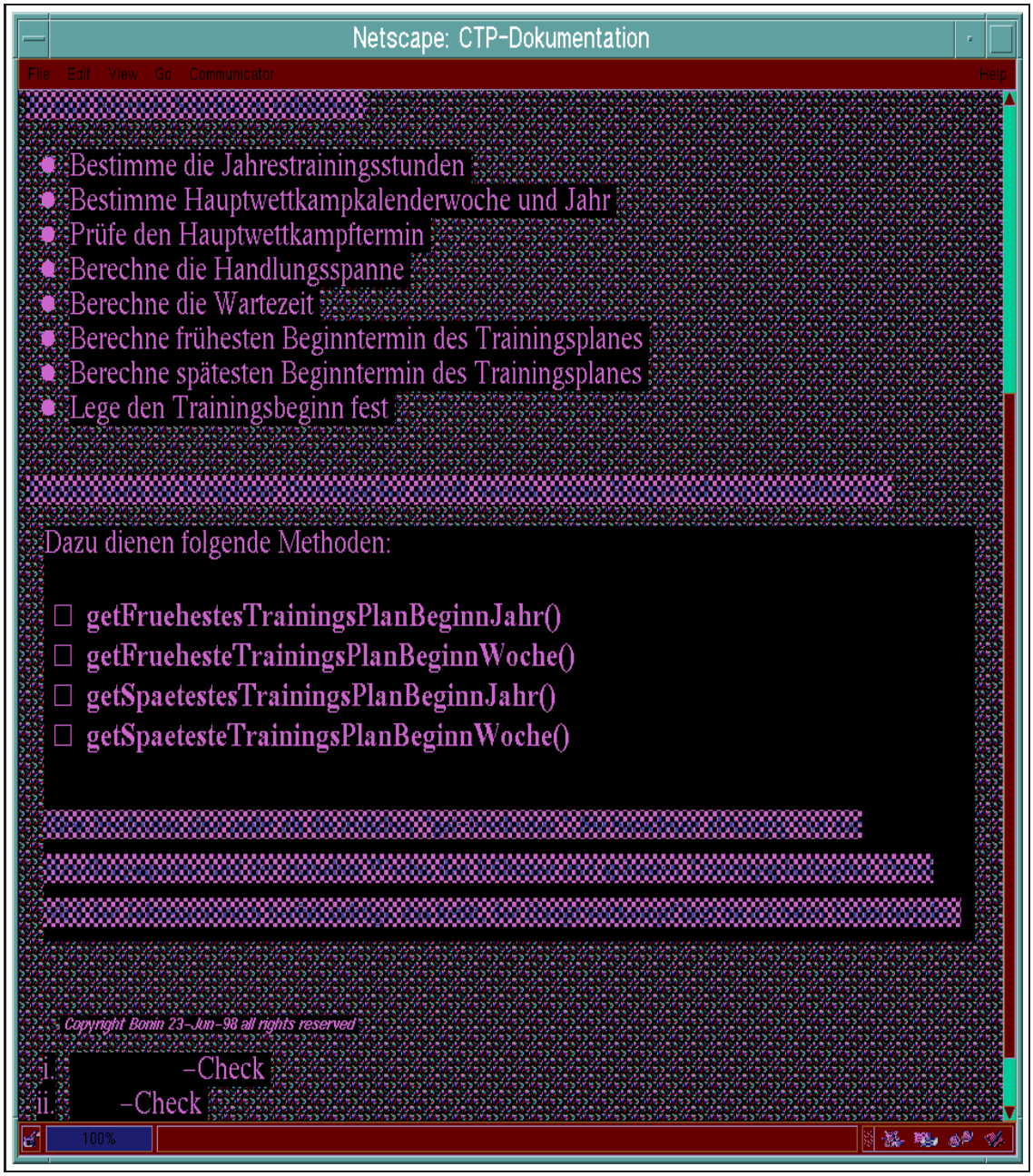


Abbildung 8.1: Ergebnis von `exampleCSS.html`

```
48    font-family: Arial, Helvetica;
49    color:       yellow;
50    background-color: blue ! important;
51    margin: 5px;
52  }
53  BODY {
54    background: url(http://193.174.33.106:6667/bonin/gelberBall.gif) black;
55    color:            white;
56  }
57  A:link {
58    font-weight:     bold;
59    text-decoration: none;
60    color:           red;
61    background-color: black;
62  }
63  A:visited {
64    font-weight:     bold;
65    text-decoration: none;
66    color:           blue;
67    background-color: black;
68  }
69  A:active {
70    font-weight:     bold;
71    text-decoration: none;
72    color:           yellow;
73    background-color: black;
74  }
75  EM {
76    font-style:italic
77  }
78  .hinweis {
79    font-style:  italic;
80    font-weight: bold;
81    color:       yellow;
82    background-color: black;
83  }
84  .hervorhebung {
85    color:            white;
86    background-color: maroon;
87    margin:           10px;
88    border:           none;
89  }
90  .dickeListe {
91    color:           white;
92    background-color: black;
93    list-style-type: square;
94    font-family:     "Times New Roman", Times, serif;
95    font-size:       18pt;
96    font-weight:     bold;
97  }
98  .anhang {
99    color:       white;
100   background-color: black;
101   font-style:  italic;
```

```
102    font-family: Helvetica, sans-serif;
103    font-size:   12pt;
104    text-indent: 25px;
105  }
106  /* End of File C13:/u/bonin/mywww/anwd/HTML40/myStyle.css  */
```

HTML-Dokument: `exampleCSS.html`

```
 1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
 2    "http://www.w3c.org/TR/REC-html40/strict.dtd">
 3  <!-- Bonin: 25-Jun-1998                        -->
 4  <HTML>
 5  <HEAD>
 6  <LINK  HREF="myStyle.css" REL="stylesheet" TYPE="text/css">
 7  <TITLE>CTP-Dokumentation</TITLE>
 8  </HEAD>
 9  <BODY>
10  <H1><EM>CTP</EM>-Dokumentation</H1>
11  <P CLASS="hinweis">
12  Graphische Darstellung der
13  <A HREF="Fachklassen.html">Fachklassen</A>
14  </P>
15  <H3>Vorgehensweise</H3>
16  <P>Es sind folgende Schritte zu vollziehen:</P>
17  <UL>
18   <LI>Bestimme die Jahrestrainingsstunden
19   <LI>Bestimme Hauptwettkampkalenderwoche und Jahr
20   <LI>Prüfe den Hauptwettkampftermin
21   <LI>Berechne die Handlungsspanne
22   <LI>Berechne die Wartezeit
23   <LI>Berechne frühesten Beginntermin des Trainingsplanes
24   <LI>Berechne spätesten Beginntermin des Trainingsplanes
25   <LI>Lege den Trainingsbeginn fest
26  </UL>
27  <P>Danach kann ein kompletter Trainingsplan
28  erstellt werden, dessen Hauptwettkampf erreichbar ist.
29  <DIV CLASS="hervorhebung">
30  Dazu dienen folgende Methoden:
31  <UL CLASS="dickeListe">
32    <LI>getFruehestesTrainingsPlanBeginnJahr()
33    <LI>getFruehesteTrainingsPlanBeginnWoche()
34    <LI>getSpaetestesTrainingsPlanBeginnJahr()
35    <LI>getSpaetesteTrainingsPlanBeginnWoche()
36  </UL>
37  <P>Diese Methoden sind mehr als die üblichen
38  "get-Methoden". Sie berechnen abhängig von der
39  Handlungsspanne und der aktuellen Woche,
40  den frühesten und spätesten Trainingsplanbeginn.
41  Dabei wird der Jahresumbruch berücksichtigt, das hei&szlig;t,
42  der Hauptwettkampf liegt im nächsten Kalenderjahr.
43  </DIV>
44  <P CLASS="anhang">
45  <IMG SRC="../../gelberBall.gif" ALT="*">
46  Copyright Bonin 23-Jun-98 all rights reserved
```

```
47  <OL>
48  <LI><A HREF="http://validator.w3.org/check?uri=http:
49  //193.174.33.106:6667/bonin/anwd/HTML40/exampleCSS.html">
50  HTML4.0</A>-Check
51  <LI><A HREF="http://jigsaw.w3.org/css-validator/validator?uri=http:
52  //193.174.33.106:6667/bonin/anwd/HTML40/myStyle.css">
53  CSS</A>-Check
54  </OL>
55  </BODY>
56  </HTML>
57  <!-- End of File Cl3:/u/bonin/mywww/anwd/HTML40/exampleCSS.html -->
```

8.3 Übungen

8.3.1 Aufgabe: HTML-Dokument mit CSS

Das folgende HTML-Dokument `myPage.html` nutzt die CSS-Datei `myStyle.css`. Außerdem weist es eine Layout-Spezifikation im `<STYLE>`-Konstrukt auf.

myPage.html

```
 1  <HTML>
 2  <HEAD>
 3  <TITLE>Cascading Style Sheet</TITLE>
 4  <LINK HREF="myStyle.css" REL="stylesheet" TYPE="text/css">
 5  <STYLE>
 6  H1 {
 7    color:      white;
 8    background: black;
 9  }
10  </STYLE>
11  </HEAD>
12  <BODY>
13  <H1><EM>CSS</EM> (Cascading Style Sheet)</H1>
14  <UL>
15  <LI>Frage:
16    <P>Wer mag denn nur <EM>CSS?</EM></P>
17  <LI>Antwort:
18    <P>Jeder der HTML-Dokumente schreibt!</P>
19  </UL>
20  </BODY>
21  </HTML>
```

myStyle.css

```
 1  /* Cascading Style Sheet: myStyle.css  */
 2  /* Bonin 30-Jun-98                     */
 3  P {
 4    font-size:  12pt;
 5    color:      red;
 6    background: white;
 7  }
 8  H1 EM {
 9    font-size:  28pt;
10  }
11  H1 {
```

```
12    font-size:  14pt;
13    color:      white;
14    background: blue;
15  }
16  EM {
17    color:      green;
18    background: white;
19    font-style: italic;
20  }
```

<H1>-Konstrukt interpretieren

Beschreiben Sie die Hauptüberschrift (`<H1>`-Konstrukt), wenn diese von einem Web-Browser angezeigt wird, der die W3C-Empfehlungen in Bezug auf HTML 4.0 und CSS erfüllt.

<EM>-Spezifikation

Erläutern Sie, warum in der CSS-Datei einerseits `H1 EM {...}` und andererseits `EM {...}` angegeben sind.

Anhang A

Lösungen zu den Übungen

Lösung Aufgabe 3.7 auf Seite 46:
3.7.1:
Die Abbildung A.1 auf Seite 210 zeigt das Klassendiagramm für die RVE.
3.7.2:
Die Abbildung A.2 auf Seite 210 zeigt die Diagrammerweiterung um den Montageplatz.

Lösung Aufgabe 5.4.1 auf Seite 87:
5.4.1:

```
1  // Klassenname mit kleinem Anfangsbuchstaben, also
2  //        echo statt Echo,
3  // da in Shells echo üblicherweise in
4  // kleinen Buchstaben geschrieben wird.
5
6  public class echo {
7    public static void main(String argv[]) {
8      for (int i = 0; i < argv.length; i++)
9        System.out.print(argv[i] + " ");
10     System.out.print("\n");
11     }
12   }
13 // End of file cl3:/u/bonin/myjava/echo.java
```

5.4.1:

- Leerzeichen zwischen den einzelnen Argumenten werden auf ein Leerzeichen reduziert.
  ```
  >java echo Alles   klar?
  Alles klar?
  ```

- Kein Rückgabewert — Abhilfe durch Ergänzung von `System.exit(0)`

Lösung Aufgabe 5.4.2 auf Seite 87:

```
1  cl3:/home/bonin/myjava:>java -fullversion
2  java full version "JDK1.1.4 IBM build a114-19971209" (JIT on)
```

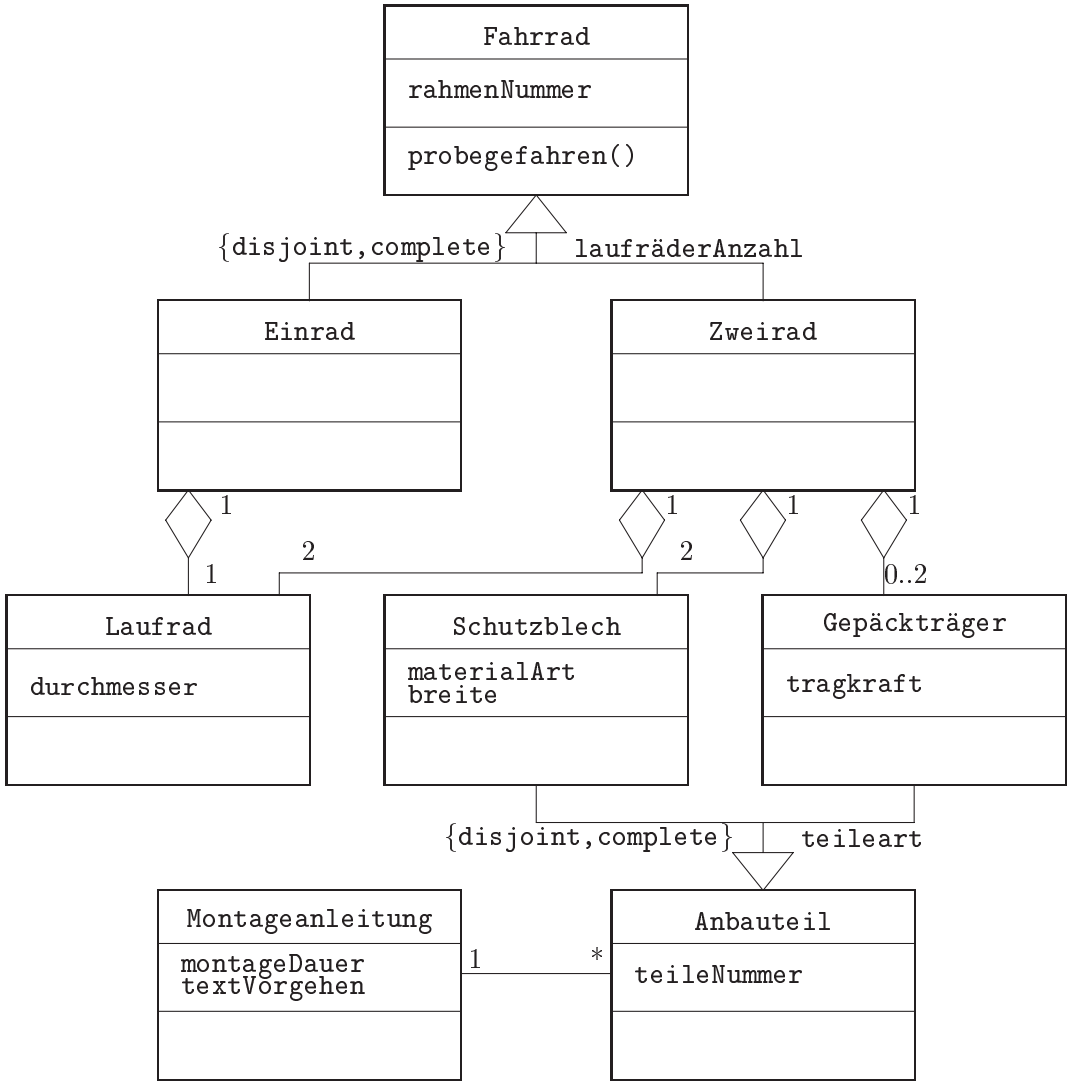


Abbildung A.1: Aufgabe 3.7.1 auf Seite 47: Klassendiagramm für die Montagesicht

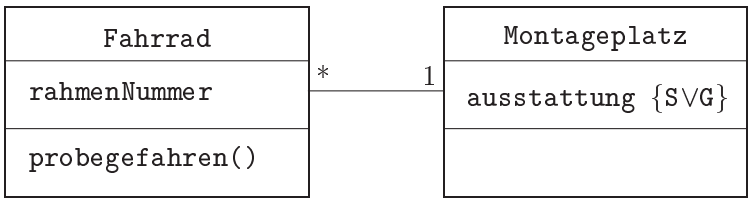


Abbildung A.2: Aufgabe 3.7.2 auf Seite 47: Diagrammerweiterung um den Montageplatz

```
 3  cl3:/home/bonin/myjava:>javac Kontrolle.java
 4  cl3:/home/bonin/myjava:>java Kontrolle ist besser!
 5  Werte:
 6  i = 6
 7  j = 6
 8  k = 7
 9  m = 3.14159265358979
10  n = 3
11  p = java
12  vielleicht = false
13  wichtig = true
14  klar = true
15  cl3:/home/bonin/myjava:>
```

Lösung Aufgabe 5.4.3 auf Seite 88:

```
 1  cl3:/home/bonin/myjava:>java -fullversion
 2  java full version "JDK1.1.4 IBM build a114-19971209" (JIT on)
 3  cl3:/home/bonin/myjava:>javac Iteration.java
 4  cl3:/home/bonin/myjava:>java Iteration 1 2 3 4 5 6 7
 5  Maximum UML & Java in der Anwendungsentwicklung null
 6  Dies sind 53 Zeichen!
 7  cl3:/home/bonin/myjava:>java Iteration 1 2
 8  java.lang.ArrayIndexOutOfBoundsException: 2
 9  at Iteration.main(Compiled Code)
10  cl3:/home/bonin/myjava:>
```

Lösung Aufgabe 5.4.4 auf Seite 89:

```
 1  cl3:/home/bonin/myjava:>java -fullversion
 2  java full version "JDK 1.1.6 IBM build a116-19980529" (JIT: jitc)
 3  cl3:/home/bonin/myjava:>javac LinieProg.java
 4  cl3:/home/bonin/myjava:>java LinieProg
 5  0030
 6  5.0
 7  false
 8  cl3:/home/bonin/myjava:>
```

Lösung Aufgabe 5.4.5 auf Seite 91:

```
 1  cl3:/home/bonin/myjava:>java -fullversion
 2  java full version "JDK 1.1.6 IBM build a116-19980529" (JIT: jitc)
 3  cl3:/home/bonin/myjava:>javac Inheritance.java
 4  cl3:/home/bonin/myjava:>java Inheritance
 5  Otti AG
 6  Fall I  : 9
 7  Fall II : 12106
 8  cl3:/home/bonin/myjava:>
```

Lösung Aufgabe 5.4.6 auf Seite 93:

```
 1  cl3:/u/bonin/myjava:>java -fullversion
 2  java full version "JDK 1.1.6 IBM build a116-19980529" (JIT: jitc)
 3  cl3:/u/bonin/myjava:>javac TableProg.java
```

```
 4  cl3:/home/bonin/myjava:>java TableProg
 5
 6  Fehler: Aufruf von this() in Konstruktor
 7  LookUpTable(int size)
 8
 9  Nach entfernen von this() folgendes Ergebnis:
10
11  Tabelle mit 100 erzeugt!
12  Alles richtig, oder was?
```

Lösung Aufgabe 6.10.1 auf Seite 166:

```
 1  cl3:/home/bonin/myjava:>java -fullversion
 2  java full version "JDK1.1.4 IBM build a114-19971209" (JIT on)
 3  cl3:/home/bonin/myjava:>javac Rekursion.java
 4  cl3:/home/bonin/myjava:>java Rekursion 4
 5  Aufruf  n = 4
 6  Aufruf  n = 3
 7  Aufruf  n = 2
 8  Aufruf  n = 1
 9  Rueckgabe wert = 1
10  Rueckgabe wert = 2
11  Rueckgabe wert = 6
12  Rueckgabe wert = 24
13  Fakultaetsfunktion: fac(4) = 24
14  Anzahl der Aufrufe von fac(): 5
15  cl3:/home/bonin/myjava:>java Rekursion 100
16  .
17  .
18  .
19  Fakultaetsfunktion: fac(100) = 93326215443944152681
20  699238856266700490715968264381621468592963895217599
21  993229915608941463976156518286253697920827223758251
22  185210916864000000000000000000000000
```

Lösung Aufgabe 6.10.2 auf Seite 168:

```
 1  cl3:/u/bonin/myjava:>java -fullversion
 2  java full version "JDK1.1.4 IBM build a114-19971209" (JIT on)
 3  cl3:/u/bonin/myjava:>javac QueueProg.java
 4  cl3:/u/bonin/myjava:>java QueueProg
 5  Step 0: Queue.noOfQueues = 2
 6  Step 1: myQ.getQueueCapacity() = 3
 7  Step 2: myQ.getFirstItem() = Emma AG
 8  Item = Ernst AG nicht aufgenommen!
 9  Step 3: myQ.getFirstItem() = Willi AG
10  Step 4: myQ.getNoOfItemsInQueue = 2
11  Step 5: myQ.isItemInQueue(Ernst AG) = true
12  cl3:/u/bonin/myjava:>
```

Lösung Aufgabe 6.10.3 auf Seite 171:

```
1  >appletviewer http://as.fh-lueneburg.de/mywww/SimpleThread/SimpleThread.html
2  x = 225 y = 60
```

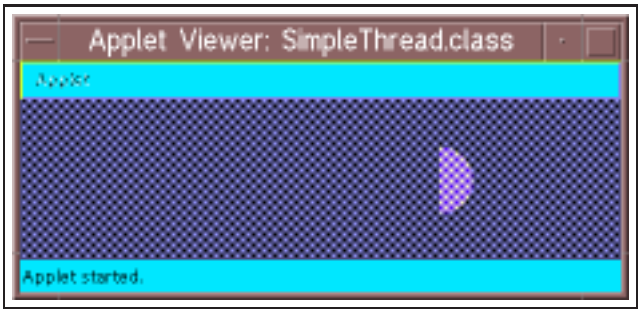


Abbildung A.3: Aufgabe 6.10.3 auf Seite 171: „Animierter Mond“

```
3  start() appliziert
4  In run() vor Thread.sleep(1000)
5  In run() nach Thread.sleep(1000)
6  In run() vor Thread.sleep(1000)
7  In run() nach Thread.sleep(1000)
8  ...
```

Lösung Aufgabe 6.10.4 auf Seite 174:

```
1  cl3:/home/bonin/myjava:>java -fullversion
2  java full version "JDK1.1.4 IBM build a114-19971209" (JIT on)
3  cl3:/home/bonin/myjava:>javac -deprecation DemoAWT.java
4  DemoAWT.java:70: Note: The method java.awt.Dimension minimumSize()
5    in class java.awt.Component has been deprecated,
6    and class MyCanvas (which is not deprecated) overrides it.
7    public Dimension minimumSize() {
8                     ^
9  DemoAWT.java:76: Note: The method java.awt.Dimension preferredSize()
10   in class java.awt.Component has been deprecated,
11   and class MyCanvas (which is not deprecated) overrides it.
12   public Dimension preferredSize() {
13                    ^
14 Note: DemoAWT.java uses a deprecated API.  Please consult the documentation for a bett
15 3 warnings
16 cl3:/home/bonin/myjava:>appletviewer file://localhost/u/bonin/myjava/ExampAWT.html
17 actionPerformed() appliziert: Anmelden!
18 actionPerformed() appliziert: Absagen!
19 stop() appliziert!
20 cl3:/home/bonin/myjava:>
```

Lösung Aufgabe 6.10.5 auf Seite 178:
Das folgende Session-Protokoll wurde in der Emacs-Shell auf einer NT-Workstation erstellt. Die Abbildung A.6 auf Seite 215 zeigt die POET-Klassen im Werkzeug „POET-Developer“.

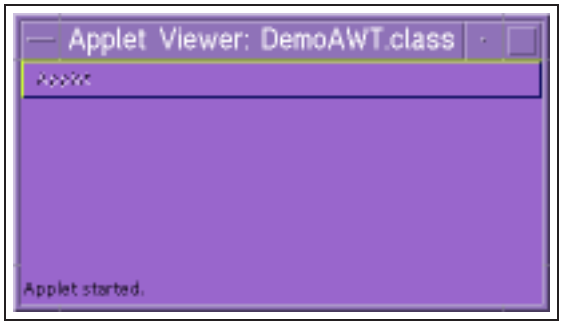


Abbildung A.4: Aufgabe 6.10.4 auf Seite 174: Ausgangsfenster

Abbildung A.5: Aufgabe 6.10.4 auf Seite 174: Hauptfenster

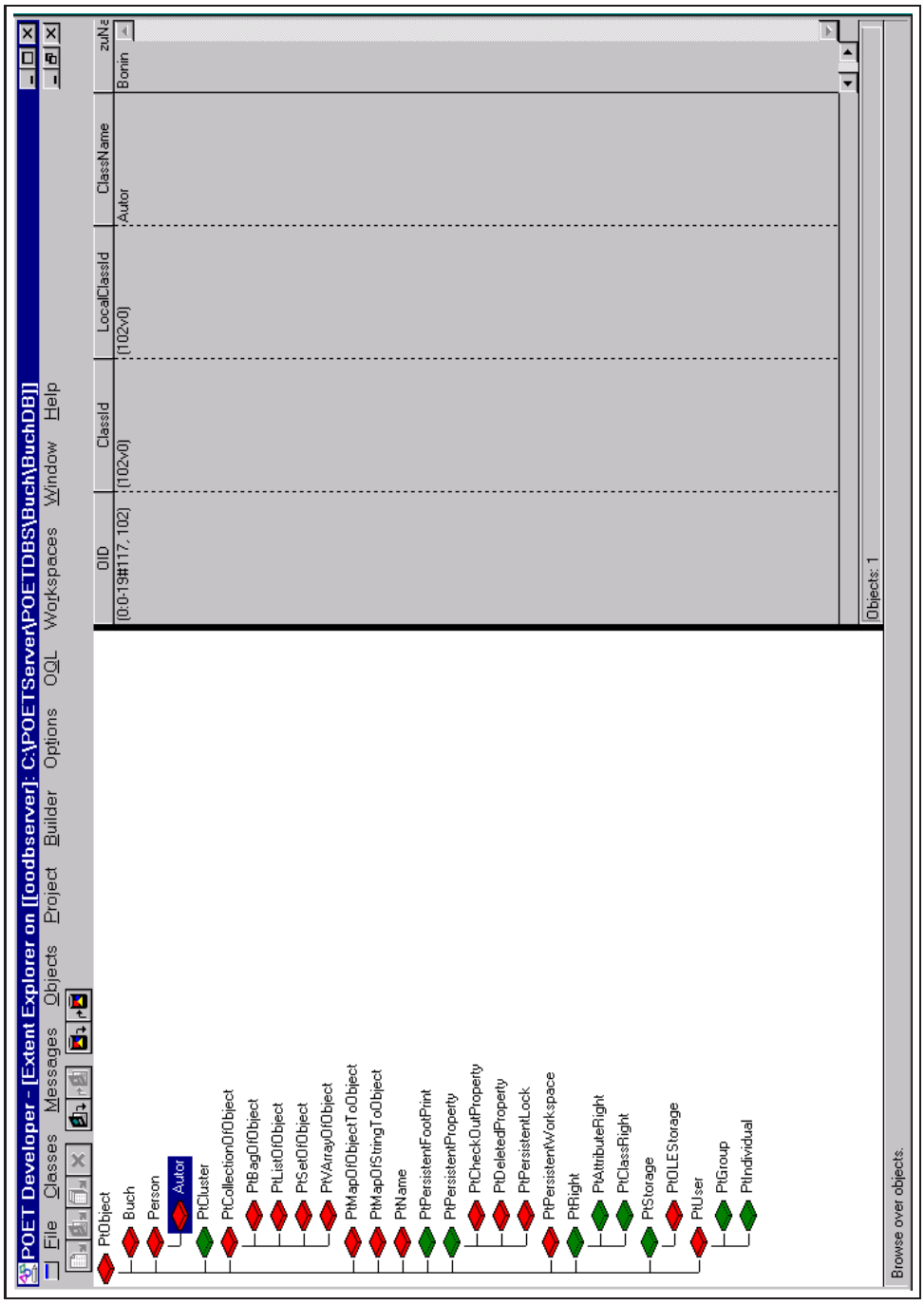


Abbildung A.6: POET Developer: Beispiel Buch.java

```
In der Emacs-Shell auf NT-Plattform IP: 193.174.33.100


$ export CLASSPATH=C:/myjava\;C:/jdk1.1.5/lib/classes.zip\;
   C:/jdk1.1.5/lib\;C:/Programme/POET50/Lib/POETClasses.zip\;.
$ java -fullversion
java full version "JDK1.1.5K"
$ ptjavac Buch.java Person.java Autor.java
POET Java! Preprocessor Version 1.05.11
Copyright (C) 1996-97 POET Software Corporation
POET Java! Schema Creation Version 1.05.11
Copyright (C) 1997 POET Software Corporation
Registered: Person (already registered)
Registered: Buch (already registered)
Registered: Autor (already registered)
Update database: BuchDB => Done
$ java BuchBind
---> Hier kommt das POET-Login-Window
       Eingabe von Name und Password
Zuname des Autors: Bonin
Alter des Buches : 7
preWrite-Methode appliziert!
$ java BuchLookup
---> Hier kommt das POET-Login-Window
       Eingabe von Name und Password
Zuname des Autors: Bonin
Alter des Buches : 7
$ java BuchBind
---> Hier kommt das POET-Login-Window
       Eingabe von Name und Password
Zuname des Autors: Bonin
Alter des Buches : 7
PKS01 gibt es schon!
preWrite-Methode appliziert!
$
```

Lösung Aufgabe 6.10.6 auf Seite 183:
6.10.6:
Es sind folgende Dateien nach dem Compilieren entstanden:

```
Foo.class
Foo$KlasseA.class
Foo$KlasseB.class
Foo$KlasseC.class
Foo$Bar.class
```

6.10.6:
Windows NT-Plattform:

```
>java Foo$Bar
```

Unix-Plattform:

```
>java Foo\$Bar
```

6.10.6:

```
1  cl3:/home/bonin/mywww/anwd/SS98:>java -fullversion
2  java full version "JDK 1.1.6 IBM build a116-19980529" (JIT: jitc)
3  cl3:/home/bonin/mywww/anwd/SS98:>java Foo\$Bar
4  Slot-Wert in Instanz c: KlasseA
5  Anzahl der Änderungen in KlasseA: 1
6  cl3:/home/bonin/mywww/anwd/SS98:>
```

Lösung Aufgabe 6.10.7 auf Seite 184:
6.10.7:

```
1  cl3:/home/bonin/mywww/anwd/SS98:>java -fullversion
2  java full version "JDK 1.1.6 IBM build a116-19980529" (JIT: jitc)
3  cl3:/home/bonin/mywww/anwd/SS98:>java TelefonBuchProg
4  TelefonEintrag: Otto +49/4131/677175
5  TelefonEintrag: Emma +49/4131/677144
6  OK --- Objekte sind gleich!
7  cl3:/home/bonin/mywww/anwd/SS98:>
```

Außerdem wird eine Datei `tbuch.ser` erzeugt.

6.10.7:

```
1  /**
2   Primitives TelefonLookupProg.java
3   @author Bonin 29-Jun-1998
4   @version 1.0
5  */
6  import java.io.* ;
7  import java.util.* ;
8  /**
9   TelefonLooupProg
10  */
11 public class TelefonLookupProg {
12   public static void main(String argv[]) {
13     if (argv.length != 1) {
14       System.out.println(
15         "Usage: java TelefonLookupProg name");
16       System.exit(1);}
17     String kurzname = argv[0];
18     try {
19       FileInputStream fin = new FileInputStream("tbuch.ser");
20       ObjectInputStream in = new ObjectInputStream(fin);
21       TelefonBuch t = (TelefonBuch) in.readObject();
22       in.close();
23       Enumeration keys = t.tabelle.keys();
24       while (keys.hasMoreElements()) {
25         String key = (String) keys.nextElement();
26         TelefonEintrag te = t.getEintrag(key);
27         if (te.kurzname.equalsIgnoreCase(kurzname)) {
28           System.out.println(te.kurzname + " " +
```

```
29              te.telefon);
30            break;
31          }
32        }
33      }
34      catch (Exception e) {
35        e.printStackTrace(System.out);
36      }
37    }
38  }
39  //End of file cl3:/u/bonin/myjava/TelefonLookupProg.java
40
```

Lösung Aufgabe 6.10.8 auf Seite 186:
6.10.8:

```
1  /* Eine Lösung der Aufgabe: Objektbeziehungen notieren */
2  interface I0 {
3    public void m1();
4    public void m2();
5  }
6  class K1 implements I0 {
7    private K4 v1;
8    private K4 v2;
9    private K4 v3;
10   public void m1() { }
11   public void m2() { }
12   public static void main(String argv[]){
13   }
14 }
15 class  K2 {
16   K1 s = new K1();
17 }
18 class K3 extends K2 {
19   public static K4 c1;
20 }
21 class K4 {
22   public void m3() { }
23 }
```

6.10.8:

```
1  /* Eine Lösung der Aufgabe: Objektbeziehungen notieren */
2  interface I0 {
3    public void m1();
4    public void m2();
5  }
6  class K1 implements I0 {
7    private K4 v1;
8    private K4 v2;
9    private K4 v3;
10   public void m1() { }
11   public void m2() { }
12   public K4 getV1() {return v1;}
```

```
13    public K4 getV2() {return v2;}
14    public K4 getV3() {return v3;}
15    public void setV1(K4 v1) {this.v1 = v1;}
16    public void setV2(K4 v2) {this.v2 = v2;}
17    public void setV3(K4 v3) {this.v3 = v3;}
18    public static void main(String argv[]){
19   }
20  }
21  class  K2 {
22    K1 s = new K1();
23    public K1 getS() {return s;}
24    public void setS(K1 s) {this.s = s;}
25  }
26  class K3 extends K2 {
27    public static K4 c1;
28    // Klassenvariablen werden in der Regel
29    // direkt angesprochen, daher hier nur
30    // als Beispiel:
31    public static K4 getC1() {return c1;}
32    public static void setC1(K4 c1) {K3.c1 = c1;}
33  }
34  class K4 {
35    public void m3() { }
36  }
```

Lösung Aufgabe 8.3.1 auf Seite 207:

8.3.1:

Die Abbildung A.7 auf Seite 220 zeigt das Ergebnis des `<H1>`-Konstruktes. CSS ist in größeren, grünen Buchstaben auf weißem Hintergrund in *Italic* geschrieben; gefolgt von weißen Buchstaben auf schwarzem Hintergrund.

CSS	color:	green
	background:	white
	font-style:	italic
	font-size:	28pt
(Cascading Style Sheet)	color:	white
	background:	black
	font-size:	14pt

8.3.1:

Die Unterscheidung ist erforderlich, damit das `<EM>`-Konstrukt nur in der Überschrift zu einem Font in der Größe von 28pt führt und nicht auch in der Aufzählung.

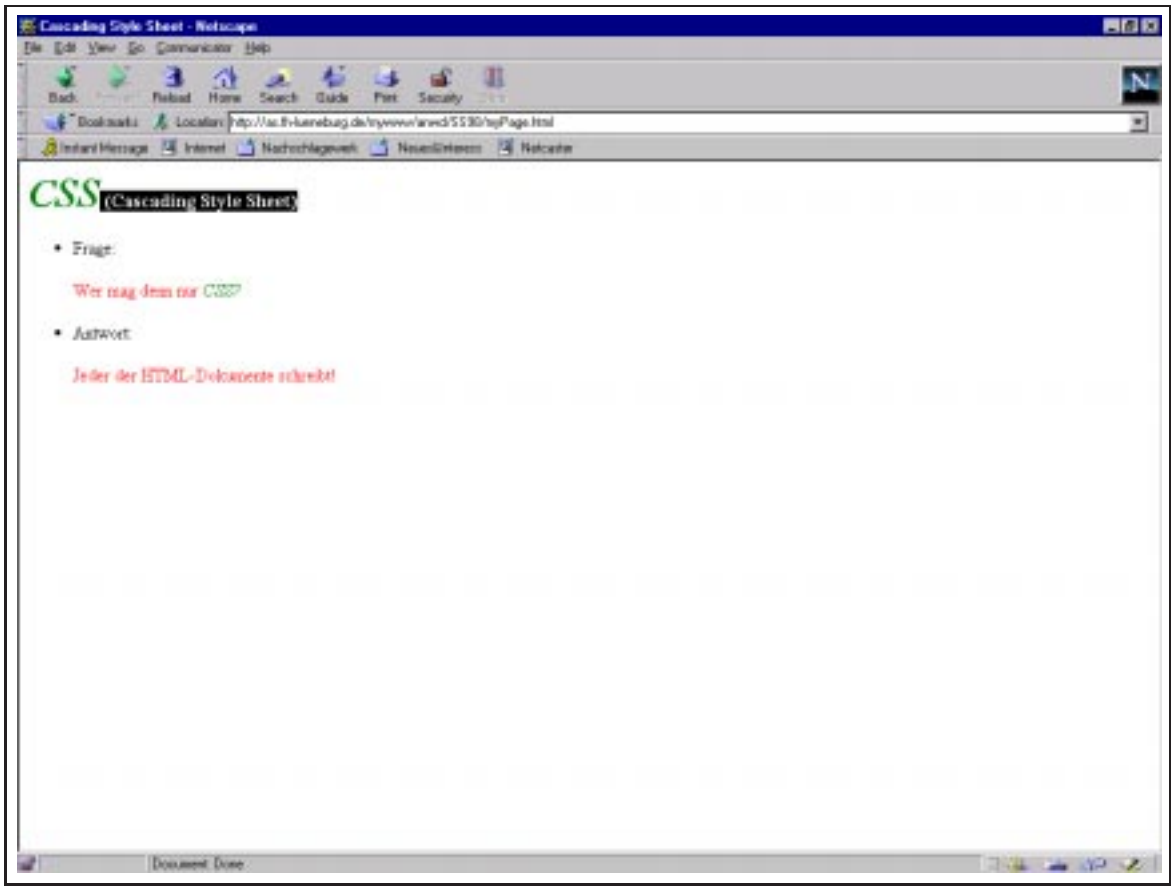


Abbildung A.7: Aufgabe 6.10.4 auf Seite 174: CSS-Beispiel mit mehreren Spezifikationen

Anhang B

Hinweise zur JDK-Nutzung

B.1 Java auf der AIX-Plattform

Es wird als Erläuterungsbeispiel angenommen, daß die Datei `Foo.java` mit dem Java-Quellcode sich auf dem AIX-Rechner mit der IP-Nummer `193.174.32.3` im Verzeichnis `/u/bonin/myjava` befindet. Der Benutzer befindet sich auf diesem Rechner (`rzserv2`) in der Korn-Shell im Verzeichnis `/home/bonin`. Mit dem Kommando:

```
. java.env
```

werden die Umgebungsvariablen gesetzt (→Tabelle B.1 auf Seite 222)[1]. (Hinweis: Punkt nicht vergessen, damit in der aktuellen Shell die Variablen gesetzt sind!) Mit dem Kommando:

```
export CLASSPATH=/home/bonin/myjava:$CLASSPATH
```

CLASS-PATH

wird dafür gesorgt, daß die Klasse `Foo` beim Aufruf gefunden werden kann. Zum Compilieren und Anwenden von `Foo` sind dann folgende Kommandos einzugeben (→Abschnitt 5.1.2 auf Seite 62):

```
> javac myjava/Foo.java
> java Foo
```

B.2 Java auf der NT-Plattform (Windows 95 & DOS-Shell)

Es wird als Erläuterungsbeispiel wieder[2] angenommen, daß die Datei `Foo.java` mit dem Java-Quellcode sich auf dem NT-Rechner mit der IP-Nummer `193.174.33.100` im Verzeichnis `C:\myjava` befindet. Der Benutzer befin-

[1] Die Datei zum Setzen der Umgebungsvariablen für Java steht auf den Rechner mit der IP 193.174.33.106 (`cl3`) unter der Kennung `guest` mit dem Paßwort `guest` zur Verfügung.

[2] ⇒Abschnitt 5.1.2 auf Seite 62.

```
#!/usr/bin/ksh -x
# Startdatei zum Setzen der Environmentvariablen
# Bonin: 17-Oct-1997
#   Update: 21-Oct-1997; 24-Oct-1997; 01-Nov-1997; 26-Mar-1998;
#           08-Jun-1998
#
# Klare Anfangsposition
unset LD_LIBRARY_PATH
unset JAVA_THREADS
unset JAVA_COMPILER
unset JAVA_HOME
unset  PATH
# Pfad fuer javac, java usw. setzen
export PATH=/usr/lpp/J1.1.6/bin:/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:\
/usr/bin/X11:/sbin:/usr/local/emacs/etc:.:/usr/local/bin
#
# Klasenzugriffspfad setzen
unset CLASSPATH
export CLASSPATH=/usr/lpp/J1.1.6/lib/classes.zip:/usr/lpp/J1.1.6/lib:.
# Steht die Anwendungsklasse zum Beispiel unter /home/bonin/myjava
#   und arbeitet man nicht unter diesem Verzeichnis
#   dann
#     export CLASSPATH=/home/bonin/myjava:$CLASSPATH
#
#     ausfuehren vor Aufruf aus dem beliebigen Verzeichnis.
#   (Ist Anwendungsklasse im aktuellen Verzeichnis, dann
#    durch obigen Punkt im CLASSPATH ausreichender Verweis.)
#
# Erlauerterung siehe Abschnitt Java auf der AIX Plattform
export JAVA_COMPILER=jitc
export JAVA_THREADS=gt
# End of file cl3:/u/bonin/java.env
```

Tabelle B.1: Java-Umgebungsvariablen für die AIX-Plattform

det sich auf diesem Rechner in einer DOS-Shell im Verzeichnis `C:\temp`. Das *Java Development Kit* befindet sich hier[3] unter:

```
C:\jdk1.1.3\bin
```

Der notwendige Zugriffspfad wird mit folgendem Kommando gesetzt:

```
path=C:\jdk1.1.3\bin;%path%
```

Zum Setzen der Umgebungsvariablen wird folgendes Kommando verwendet, weil die Java-Standardklassen sich in der Datei `classes.zip` befinden:

CLASS-PATH

```
set CLASSPATH=C:\myjava;C:\jdk1.1.3\lib\classes.zip;C:\jdk1.1.3\lib
```

Zur Compilation und Applikation werden folgende Kommandos auf der DOS-Shell-Ebene eingegeben:

```
C:\temp>java -version
java version "1.1.3"
C:\temp>javac C:\myjava\Foo.java
C:\temp>java Foo
Kein Argument: Java ist ...! Fri Oct 24 10:59:42 GMT+01:00 1997
C:\temp>java Foo is my %CLASSPATH%
Eingabeteil:0
+is+
Eingabeteil:1
+my+
Eingabeteil:2
+CLASSPATH=C:\myjava;C:\jdk1.1.3\lib\classes.zip;C:\jdk1.1.3\lib+
Neuer Wert: Java ist ...! Fri Oct 24 11:02:13 GMT+01:00 1997
C:\temp>echo %CLASSPATH%
CLASSPATH=C:\myjava;C:\jdk1.1.3\lib\classes.zip;C:\jdk1.1.3\lib
C:\temp>
```

Hinweis: Für den GNU Emacs, Version 19.34.6, 25-Sep-1997, auf Windows NT ist in der Emacs-Shell ein besondere Kombination aus UNIX und DOS-Shell-Kommandos erforderlich. Da der Emacs dem Benutzer UNIX-Kommandos emuliert gibt es „Probleme“ mit Sonderzeichen. Es sind folgende Zeichenkombinationen zu wählen:

```
# Fuer die Emxacs Shell
# Achtung mit Backslash und Semikolon zum Trennen
export CLASSPATH=C:/myjava\;C:/jdk1.1.5/lib/classes.zip\;C:/jdk1.1.5/lib\;.
# Bonin 7-Mai-98
```

[3] Hinweis: In der Fachhochschule Nordostniedersachsen (Fachbereich Wirtschaft. Volgershall 1, D-21339 Lüneburg, Übungsraum 80) ist JDK unter der Platte (Partiton) „J“ gespeichert.

Anhang C

Quellen

C.1 World Wide Web

World-Wide-Web-Server:

`http://as.fh-lueneburg.de/`

Unter diesem WWW-Server werden weitere Informationen zu diesem Buch angeboten.

C.2 Literaturverzeichnis

Literaturverzeichnis

[Abelson85] Harold Abelson / Gerald Jay Sussman / Julie Sussman; Structure and Interpretation of Computer Programs, Cambridge, Massachusetts and others (The MIT Press/McGraw-Hill) 1985.

[Arnold/Gosling96] Ken Arnold / Jame Gosling; The Java Programming Language (Addison-Wesley) 1996.

[Belli88] Fevzi Belli; Einführung in die logische Programmierung mit Prolog, Mannheim Wien Zürich (Bibliographisches Institut), 2. Auflage 1988.

[Bonin88] Hinrich Bonin; Die Planung komplexer Vorhaben der Verwaltungsautomation, Heidelberg (R. v. Decker & C. F. Müller), 1988 (Schriftenreihe Verwaltungsinformatik; Bd. 3).

[Bonin89] Hinrich E. G. Bonin; Objektorientierte Programmierung in LISP – Standard-LISP und objektorientierte Erweiterungen, in: **H**andbuch der **M**odernen **D**atenverarbeitung (HMD), Heft 145, Januar 1989, 26. Jahrgang, S. 45 – 56.

[Bonin91a] Hinrich Bonin; Cooperative Production of Documents, in: [Traunmüler91], pp. 39–55.

[Bonin91b] Hinrich E. G. Bonin; Software-Konstruktion mit LISP, Berlin New York (Walter de Gruyter), 1991.

[Bonin92a] Hinrich E. G. Bonin; Arbeitstechniken für die Softwareentwicklung, (3. überarbeitete Auflage Februar 1994), FINAL, 2. Jahrgang Heft 2, 10. September 1992, [FINAL].

[Bonin92b] Hinrich E. G. Bonin; Teamwork between Non-Equals — Check-in & Check-out model for Producing Documents in a Hierarchy, in: SIGOIS Bulletin, Volume 13, Number 3, December 1992, (ACM Press), pp. 18–27.

[Bonin92c] Hinrich E. G. Bonin; Object-Orientedness – a New Boxologie, FINAL Heft 3, 1992 (ISSN 0939-8821).

[Bonin93] Hinrich E. G. Bonin; The Joy of Computer Science, — Skript zur Vorlesung EDV —, Unvollständige Vorabfassung (4. Auflage März 1995), FINAL, 3. Jahrgang Heft 5, 20. September 1993,[FINAL].

[Bonin94] Hinrich E. G. Bonin; Groupware-Systeme: Eine Perspektive für die öffentliche Verwaltung, in: Verwaltungsführung / Organisation / Personal (VOP), Heft 3, 1994, S. 170–176.

[Bonin96] Hinrich Bonin; <HTML>-Ratgeber — Multimediadokumente im World-Wide Web programmieren, München Wien (Carl Hanser Verlag), 1996.

[Booch94] G. Booch; Object-oriented analysis and design with applications, 2nd ed., Redwood City (Benjamin/Cummings), 1994. Deutsche Ausgabe: Objektorientierte Analyse und Design, Mit praktischen Anwendungsbeispielen, Bonn (Addision-Wesley), 1994.

[Bobrow/Moon88] Daniel G. Bobrow / David Moon u. a.; Common Lisp Object Systems Specification, ANSI X3J13 Document 88-002R, American National Standards Insitute, Washington, DC, June 1988 (veröffentlicht in: SIG-PLAN Notices, Band 23, Special Issus, September 1988).

[Clocksin/Mellish87] W.F. Clocksin / C. S. Mellish; Programming in Prolog, Berlin New York u.a. (Springer-Verlag) Third Edition, 1987.

[Embley92] David W. Embley / Barry D. Kurtz / Scott N. Woodfield; Object-Oriented Systems Analysis – A Model-Driven Approach, Englewood Cliffs, New Jersey (Yourdon Press), 1992.

[Flanagan96] David Flanagan; Java in a Nutshell, Deutsche Übersetzung von Konstantin Agouros, Köln (O´Reilly), 1996.

[Flanagan97] David Flanagan; Java in a Nutshell, Second Edition, updated for Java 1.1, Köln (O´Reilly), May 1997.

[FINAL] Fachhochschule Nordostniedersachsen, Informatik, Arbeitsberichte, Lüneburg (FINAL) herausgegeben von Hinrich E. G. Bonin, ISSN 0939-8821, ab 7. Jahrgang (1997) auf CD-ROM, beziehbar: FH NON, Volgershall 1, D-21339 Lüneburg, Germany.

[Freeman/Ince96] Adam Freeman / Darrel Ince; active java — Object-Oriented Programming for the World Wide Web, Harlow, England. u. a. (Addison-Wesley) 1996. [Hinweis: Einige Fehler in den Java-Quellecode-Beispielen.]

[Gabriel91] Richard P. Gabriel / John L. White / Daniel G. Bobrow; CLOS: Integrating Object-Oriented and Functional Programming, in: Communications of the ACM, Vol. 34, No. 9, September 1991, pp. 29 – 38.

[Goldberg83] Adele Goldberg; Smalltalk-80: The Interactive Programming Environment, Reading 1983 (Addison-Wesley) [Im Smalltalk-Jargon genannt: „das orangefarbene Buch“].

[Goldberg/Robson83] Adele Goldberg / Dave Robson; Smalltalk-80: the lanugage, Reading, Massachusetts u. a. (Addision-Wesley) 1983. [Im Smalltalk-Jargon genannt: „das blaue Buch“].

[Hist97] Jason English (1997); It all started with a blunt letter,
`http://www.javasoft.com/nav/whatis/index.html`
(Zugriff: 20-Sep-97)
Michael O'Connell (Sun World Online, 1995); Java: The inside story — We interview Java's creators to find what they had in mind.
`http://www.sun.com/sunworldonline/swol-07-1995/swol-07-java.html`
(Zugriff: 20-Sep-97)

[HTML4.0] Dave Raggett / Arnaud Le Hors / Ian Jacobs ; HTML 4.0 Spezifikation, W3C Recommendation 18-Dec-1997,
`http://www.w3.org/TR/REC-html40-971218.html`
(Zugriff: 29-Jan-98).

[Hoff/Shaio96] Arthur van Hoff / Sami Shaio / Orca Starbuck; HOOKED ON JAVA, (Addision-Wesley Publishing Company) 1996.

[IBM-Francisco98] IBM Corporation; San Francisco Projekt, Java Coding Tips
`http://www.ibm.com/Java/Sanfrancisco/tips/javatips.html`
(Zugriff: 09-Jul-98)

[ITS97] ITS, Kalatog Fernreisen Sommer 97.

[JavaSpec] James Gosling / Bill Joy / Guy Steele; The Java Language Specification, (Addison-Wesley) 1996;
`http://www.javasoft.com/docs/books/jls/html/index.html`
Änderungen für Java 1.1;
`http://www.javasoft.com/docs/books/jls/html/1.1Update.html`
(Zugriff: 20-Sep-97)

[Jacobsen92] Ivar Jacobsen / M. Christerson / P. Jonsson / G. Övergaard; Object-Oriented Software Engineering, A Use Case Driver Approach, Workingham (Addision-Wesley) 1992.

[Kczales91] Gregor Kiczales / Jim des Rivieres / Daniel G. Bobrow; The Art of the Metaobject Protocol, Cambridge, Massachusetts, London (The MIT Press) 1991.

[Kim/Lochovsky89] Won Kim / Frederick H. Lochovsky (Eds.); Object-Oriented Concepts, Databases, and Applications, Reading, Massachusetts (Addison-Wesley) 1989.

[Larman98] Craig Larman; Applying UML and Patterns — An Introduction to Object-Oriented Analysis and Design, New Jersey (Prentice Hall) 1998.

[LieBos96] Hakon Wium Lie / Bert Bos; Cascading Style Sheets, level 1, W3C Recommendation 17-Dec-1996
`http://www.w3.org/StyleSheets/core/examples/REC-CSS1-961217.html`
(Zugriff: 23-Jun-98).

[Lieberman81] H. Lieberman; Thinking About Lots of Things at Once Without Getting Confused - Parallelism in ACT-1, Cambridge, MIT AIMemo 626, May 1981.

[Oestereich97] Bern Oestereich; Objekt-orientierte Softwareentwicklung mit der Unified Modeling Language,(3. aktualisierte Auflage) München Wien (R. Oldenbourg Verlag) 1997.

[Orfali/Harkey97] Robert Orfali / Dan Harkey; Client/Server Programming with JAVA and CORBA, New York u. a. (John Wiley & Sons) 1997.

[Partl98] Hubert Partl; Java — Einführung; Kursunterlage, Version April 1998,
`http://www.boku.ac.at/javaeinf/`
(Zugriff: 08-Mai-98).

[Rational97] Rational Software; Unified Modeling Language, Version 1.1, 01-Sep-1997, UML Summary, UML Metamodel, UML Notation Guide, UML Semantics, UML Extension Business Modeling, Object Constraint Specification,
`http://www.rational.com/uml/1.1/`
(Zugriff: 11-Nov-97)

[Rumbaugh91] J. Rumbaugh / M. Blaha / W. Premerlani / F. Eddy / W. Lorenson; Objekt-oriented Modelling and Design, Englewood Cliffs (Prentice-Hall), 1991

[RRZN97] Regionales Rechenzentrum für Niedersachsen / Universität Hannover (RRZN); Java — Begleitmaterial zu Vorlesungen / Kursen, 1. Auflage Juni 1997, RRZN-Klassifikationsschlüssel: SPR.JAV2; beziehbar: FH NON, Volgershall 1, D-21339 Lüneburg, Germany. [Hinweis: Einige Fehler in den Java-Quellecode-Beispielen.]

[Sun97] Sun Microsystems; Schwerpunkt: The Road To Java, in: SunNews, November 1997, S. 4–5, Sun Microsystems GmbH, Bretonischer Ring 3, D-85630 Grasbrunn

`http://www.sun.de`
(Zugriff: 05-Dec-97).

[Sun98] Sun Microsystems; SunNews — Das Magazin für Network Computing, Mai 1998, Sun Microsystems GmbH, Bretonischer Ring 3, D-85630 Grasbrunn
`http://www.sun.de`
(Zugriff: 05-Dec-97).

[SunRMI98] Sun Microsystems; Java Remote Method Invocation Specification, Revision 1.42, JDK 1.2 Beta 1, Oktober 1997
`http://java.sun.com:80//products/jdk/rmi/index.html`
(Zugriff: 16-Jun-98).

[Sommerville89] Ian Sommerville; Software Engineering, Wokingham, England u.a. (Addison Wesley) Third Edition, 1989.

[Stroustrup86] Bjarne Stroustrup; The $C++$ Programming Language, Reading Massachusetts (Addison-Wesley) 1986 (corrected reprinting, 1987).

[Stroustrup89] Bjarne Stroustrup; The Evolution of $C++$: 1985 to 1989, in: Computing Systems, 2(3) Summer 1989, pp. 191 – 250.

[TakeFive97] TakeFive Software; SNiFF+J, Release 2.3.1 for Unix and Windows, SNiFF+ Java Solution at a Glance, Product Number SNiFF-TG1-231, 19-Jun-1997, Europa: TakeFive Software GmbH, A-5020 Salzburg, email: `info@takefive.co.at`

[Traunmüler91] Roland Traunmüller (Editor); Governmental and Municipal Information Systems, II, IFIP, Amsterdam, u. a. (North-Holland), 1991.

[Tyma98] Paul Tyma; Why are we using Java again?, in: Communications of the ACM, June 1998, Vol.41, No. 6, pp. 38–42.

[Ungar/Smith91] David Ungar / Randall B. Smith; SELF: The Power of Simplicity, in: LISP and Symbolic Computation (Kluwer Academic Publishers), Volume 4, Number 3, July 1991, pp. 187 – 205.

[Vanderburg97] Glenn Vanderburg, et al.; Maximum Java 1.1, Indianapolis (sams net) 1997. [Hinweis: „The ultimate source for advanced Java programming techniques.„]

Anhang D

Index

Index

Index

abstract, 91
`abstract`, 83
ActionApplet, 76
ActionListener, 103
Aggregation, 37, 41, 42
AIX, 57
    CLASSPATH, 221
    JDK, 221
Anwendungsfelder, 27
Applet, 75
    Erreichbarkeit, 56
    HTML-Einbindung, 75
`appletviewer`, 58
Applikation, 75
Archiv
    JAR, 116
`archive`, 79
`args`, 60
`argv`, 60
Assoziation, 37, 41
    degenerierte, 38
    Klasse, 38
    rekursive, 39
Attribut, 35
Authentifikation, 55
AWT
    Beispiel, 174

backward chaining, 22
BASIC, 25
Bean, 139–142
`bind()`, 143
Booch, Grady, 15
`boolean`, 83–85
`bottom`, 79
Bowser, 58
Boxologie
    Begriff, 19
`break`, 83
Bycode
    verifier, 54
`byte`, 83–85
Bytecode, 52, 53
`byvalue`, 83

C, 54
C++, 54
Cascading Style Sheets, 196–207
`case`, 83
`cast`, 83
Casting, 111
`catch`, 83
CGI, 70
chaining
    backward, 22
    forward, 22
`char`, 83–85
CLASS
    `CLASS`
        `CSS`, 199
Class
    anonym
        Beispiel, 122, 123, 125
    inner, 116–132
        Beispiel, 119, 126, 127
    local
        Beispiel, 121
`class`, 33, 83
`classid`, 79
CLASSPATH, 65
    AIX, 221
    NT, 223
`clone`, 82, 136
Cloning, 136–139
CLOS, 25, 26
COBOL, 25
`codebase`, 79
`codetype`, 79
Coding Tips, 190–192
`commit()`, 143
Common Business Objects, 189–190
Common Gateway Interface, 70
Common Object Request Broker Architecture, 154
`const`, 83
Constraint, 44
`Constraints`, 146
`Constructor`, 132
`continue`, 83
CORBA, 154
Core Business Processes, 189–190
CSS, 196–207
    `CSS`
        `CLASS`, 199
    `CSS`
        `ID`, 199

`data`, 79
Data-directed Programming, 22

Daten-gesteuerte Programmierung
Wurzel, 21
Datenbank-Managementsystem, 25
Datentyp, 33
DBMS, 25
default, 86, 87
`default`, 83
Denkmodell, 18
Diskriminator, 44
DISPATCH-Funktion, 22
`do`, 83
DOS
JDK, 221
`double`, 83–85
DTP, 196

echo, 87
Effizienz, 25
`else`, 83
Emacs, 58
Entwicklungsumgebung, 57
`equals`, 82
Event handler, 101
Event Model, 101–110
`extends`, 83
Externalizable, 113

FahrzeugProg, 64–70
Fakultät, 166
`false`, 83
`Field`, 132
`final`, 83
`finalize`, 82
`finally`, 83
`float`, 83–85
Foo.java, 62–64
`for`, 83
FORTRAN, 21, 25
forward chaining, 22
FTP, 58
Funktion, 35
`future`, 83

Ganzes-Teile-Beziehung, 41, 42
Generalsierung, 44
`generic`, 83
`getClass`, 82
Getter, 139
GNU, 58
Gosling, James, 13
`goto`, 83
Graphical User Inerface, 101
GUI, 101
GUI Object
Listener, 108

`hashCode`, 82
`height`, 79
HelloWorld, 60–62
HTML, 15
HTML 4.0, 193–196
HTML-Syntax
rekursive Definition, 77

IBM, 57
VisualAge for Java, 57
ID
`ID`
`CSS`, 199
`if`, 83
`implements`, 83
`import`, 83
Inheritance, 44
`inner`, 83
`instanceof`, 83
`int`, 83–85
Integrität
referenzielle, 38
`interface`, 83
Internet, 13
Internetzugriff, 70
ISO 8879, 194
Iteration, 88

Jacobsen, James, 15
JAR, 116
Java, 21, 26
Applet, 75
Applikation, 75
Definition, 13
embedded, 14
personal, 14
Plattform, 51
Varianten, 14
Werkzeuge, 56
`java`, 58
Java Archiv, 116
`java.net`, 70
`javac`, 58
JavaCard, 14
`javadoc`, 58
`javah`, 58
`javap`, 58
`jdb`, 58
JDK, 57
AIX, 221
DOS, 221
NT, 221, 223
Windows, 221
JIT, 54
JNI, 154
`join`, 100
Joy, Bill, 13

KeyListener, 107
Klasse, 25, 33
abstrakte, 33
Name, 46
Klassenkonzept, 24
Klassenvariable

Beispiel, 124
Komposition, 37, 42, 43
Konstruktor, 64, 93
Name, 46
Kontrollstruktur, 22
Konvertierung
Objekt, 111
Konvertierungszeit, 28
Konzept
Klasse, 24
Prototyp, 24

`left`, 79
Link, 37
LISP, 22
Liste, 168
Listener, 101, 140
Typen, 107
`long`, 83–85
`lookup()`, 144

Manifest, 140
Member, 35
Menge, 25
Merkmal, 36
Name, 46
Metaklasse, 33
`Method`, 132
Methode, 25, 35
Name, 46
Microsoft
Windows NT, 57
`middle`, 79
Modell
Begriff, 19
Modellierungssprache
UML, 15
Multiplizität, 39
Multithreading, 96–101
Muster-gesteuerter Prozeduraufruf
Wurzel, 22
MyNetProg, 70–73

Nachrichtenaustausch, 24
Name
Klasse, 46
Konstruktor, 46
Merkmal, 46
Methode, 46
Paket, 46
Stereotyp, 46
Variable, 46
Zusicherung, 46
`native`, 83
Netscape, 58
`new`, 83
Notation, 15
`notify`, 82, 100
`notifyAll`, 82
NT
JDK, 221
`null`, 83

Object
Listener, 108
`<OBJECT>`, 79
Object Management Group, 15
Objekt
Begriff, 19
Konvertierung, 111
Objekt-Orientierung
Anwendungsfelder, 25, 27
Definition, 24
Wurzeln, 20
Objektmenge, 25
Objektstruktur, 28
ODBMS, 142–154
ODMG, 143
OMG, 15
`operator`, 83
`outer`, 83

Package, 65
`package`, 83
Paket, 36, 65
Name, 46
Paradigma, 18
Parameter, 62
Pascal, 25
pattern matching, 22
Persistenz, 110–116, 140
PersonalJava, 14
Plattform, 57–58
POET, 142
Beispiel, 178
Pointer, 55
Polymorphismus, 21
Compilierung, 21
Laufzeit, 21
Multiargument, 25
Zeitpunkt, 21
Portabilität, 52
Pragmatik, 82–85
`private`, 83, 86, 87
Programmierung
daten-gesteuerte
Wurzel, 21
PROLOG, 22
`protected`, 83, 86, 87
Prototyp
Konzept, 24
Prozedur, 35
Prozeduraufruf
muster-gesteuerter
Wurzel, 22
`ptjavac`, 149
`public`, 60, 83, 86, 87

Queue, 168

Rüstzeit
terminierbare, 28
Rambaugh, James, 15
Rational, 32
Reflection, 132–135
Regel-Orientierung, 28
Rekursion, 166
Relation, 37
`Resolution`, 147
`rest`, 83
`return`, 83
`right`, 79
RISC, 57
RMI, 154–166
URL, 154
`rmic`, 155
`rmiregistry`, 156
Robustheit, 18
RPC, 154

Schriftart
Typewriter, 15
Semantik, 82–85
Serializable, 112
Serialization, 111, 113
serialVersionUID, 113
Setter, 139
SGML, 194
`short`, 83–85
Sicherheit, 52
`Skel`, 160
Slot, 35
Smalltalk, 26
SniFF+J, 57
Spezialisierung, 44
Sprachen
imperativ-geprägte, 25
objekt-orientierte
Ausrichtung, 22
`standby`, 79
static, 60
`static`, 82, 83
Stereotyp, 36
Name, 46
Struktur, 28
`Stub`, 160
Style Sheet, 196–207
Sun
Java Workshop 2.0, 57
Sun Microsystems, 56
super, 91
`super`, 83
`switch`, 83
`synchronized`, 83, 85, 100
Syntax, 82–85
System
komplexes, 18

TCP/IP, 58
this, 89, 93
`this`, 83
Thread, 96–101
Beispiel, 171
join, 100
sleep, 99
thread, 52
`throw`, 83
`throws`, 83
`top`, 79
`toString`, 82
Transaktion, 143
`transient`, 83, 85, 113, 146
`true`, 83
`try`, 83
Typ, 33
`type`, 79

Umgebungsvariable, 222
UML, 15
Klassensymbol, 34
URI, 79
URL, 70

`var`, 83
Variable, 35, 36
interne, 25
Name, 46
Vererbung, 43–45
Beispiel, 120
Vererbungsgraph, 25
Verifier
Bytecode, 54
VisualAge for Java, 57
`void`, 60, 83
`volatile`, 83, 85

`wait`, 82, 100
`while`, 83
`width`, 79
Windows
JDK, 221
Windows NT, 57
WWW
Entwicklung, 29

XPS, 28

Zeiger, 55
Zeit
Konvertierung, 28
Zugriffskontrolle
Liste, 55
Zusicherung, 36
Name, 46