# Programmdokumentation

Der 1. Abgabe am 29.10.1998
über das 1. und 2. Programm

**INHALTSVERZEICHNIS:**




| | |
|---|---|
| Bearbeiter: | Sascha Nemecek |
| Matrikelnummer: | 9825815 |
| Übung: | Einführung in das Programmieren - Syntaxpraktikum |
| Tutor: | Fischmeister |
| Abgabedatum: | 29.10.1998 |

# 1 Erstes Programm: Rechnungserstellung

## *1.1 Modularisierung und Datenstruktur*

### 1.1.1 Grundidee: Struktogramm

| | |
|---|---|
| Variable & Konstanten initialisieren | |
| File mit Artikeln öffnen & solange Zeilen vorhanden: | |
| | Zeile einlesen |
| | Zeile auswerten und in Array von Artikel schreiben |
| | Bei Fehler diesen Artikel ignorieren |
| File mit Rechnungen öffnen & solange Zeilen vorhanden: | |
| | Zeile einlesen |
| | Zeile auswerten und in Array von Rechnungen schreiben |
| | Bei Fehler diese Rechnung ignorieren |
| Berechnen und ausgeben | |

### 1.1.2 Wahl der Klassen

| | |
|---|---|
| Artikeln: | Methoden zur Artikel Ver- und Bearbeitung |
| Daten: | Array mit den gültigen gesicherten Artikeln |
| Rechnung: | Methoden zur Rechnungsver- und -bearbeitung |
| Rechnungen: | Array mit den gültigen gesicherten Rechnungen |
| Berechnung: | Methoden zur Berechnung und Ausgabe der Rechnungen |

### 1.1.3 Wahl der Methoden

#### 1.1.3.1 Klasse *Artikeln*:

| | |
|---|---|
| DatenLesen(**String**, **String**): | Ließt zeilenweise die Artikel aus einem File ein |
| StringtoArtikel(**String**): | Wandelt einen String, wenn möglich in das Artikelformat um und sichert ihn im Datenarray *Daten* |
| Printall(): | Gibt die Liste der vorhandenen Artikel am Standartoutput aus |
| Find(**Int**): | Sucht im Datenarray nach einer Artikelnummer |

#### 1.1.3.2 Klasse *Rechnung*:

| | |
|---|---|
| load(**String**, **String**): | Ließt zeilenweise die Rechnungen aus einem File ein |
| StringtoRechnung(**String**): | Wandelt einen String, wenn möglich in das Rechnungsformat um und sichert ihn im Datenarray *Rechnungen* |
| Find (**Int**): | Sucht im Rechnungsarray nach einer Artikelnummer |

## *1.2 Pseudocode*

```
Variablen definieren und initialisieren
File mit Artikeln öffnen
Solange Zeile vorhanden:
        Zeile einlesen
        Zeile in Substrings aufteilen (Tokenize) und Anzahl der Substrings überprüfen (muss 3 sein, sonst Fehlermeldung)
        Substrings in jeweiliges Format umwandeln (Artikelnummer: Int, Beschreibung: String, Preis: double)
        Wenn keine Fehlermeldung Artikeldaten im Datenarray speichern

File mit Rechnungen öffnen:
Solange Zeile vorhanden:
        Zeile einlesen
        Zeile in Substrings aufteilen (Tokenize) und Anzahl der Substrings überprüfen (muss 2 sein, sonst Fehlermeldung)
        Substrings in jeweiliges Format umwandeln (Artikelnummer: Int, Menge: Int)
        Wenn keine Fehlermeldung Rechnungsdaten im Rechnungsarray speichern

Rechnungsdaten durchgehen
        Artikeldaten zur Rechnung suchen
        Rechnungsergebnis berechnen und ausgeben
```

## 1.3 Source Code

```
import java.util.*;
import java.text.*;

class Beispiel1 {
    final static String pfad = "f:\\daten\\uni\\1.semester\\eprog\\abgabe1\\beispiel1";
    final static String filename = "ARTIKEL.TXT";

    // Hauptprogramm
    public static void main(String argv[])
          {
          // Initialisierung der Varible a, dabei wird die Liste der Artikel eingelesen
        Artikeln a = new Artikeln ();

          try {
              // Einlesen der Artikel
              a.DatenLesen (pfad,filename);
              }
          catch (Exception e)
              {
              System.out.println(e.getMessage());
              }

          // Ausgabe der vorhanden Artikel
          a.printall();

          // Initialisiert die Rechnungsklasse
          Rechnung r = new Rechnung ();

          // Einlesen der Rechnungen aus dem File & deren Ausgabe
          for (int i=1; i<=9; i++)
              {
              System.out.print ("\nRechnung " + i + ":\n\n");
              try {
              r.load (pfad,"Rechnung" + i + ".txt");
                    }
              catch (Exception e)
                    {
                    System.out.println (e.getMessage());
                    }

              Berechnung b = new Berechnung (a,r);
              r.destroy ();
              }
          }
    }


// Hier werden die Artikeldaten abgelegt
class Daten
    {
    int nummer = 0;
    String name = null;
    double preis = 0;


    // Konstruktor für neues Element
    Daten(int nr, String na, double p)
          {
          nummer= nr;
          name = na;
          preis = p;
          }
    }


// Verwaltet die Artikel
class Artikeln
    {
    Daten[] a = new Daten[30];
    private int anz;

    // Konstruktor
    Artikeln ()
          {
          anz = 0;
          }

    // Konstruktor: Ließt die Artikel ein
    void DatenLesen (String path, String filename) throws Exception
          {
          String s = null;

          // Öffnet das File "Artikel.txt" und legt die Daten in Daten ab
          ReadFile file = new ReadFile (path, filename);
          while ((s = file.readLine()) != null)
              {
              // Wenn zu viele Artikel: Abbruch
              if (anz >= 30)
                    throw new Exception ("Zu viel Artikel im File - Es wurden nur die ersten 30 berücksichtigt.");

              try
                    {
                    // Wandelt einen String ins Artikelformat um
                    StringtoArtikel (s);
                    }
```

```
            catch (Exception e)
                  {
                  System.out.println ("Fehler: " + e.getMessage());
                  }
            }
          file.close ();
          }


    // Versucht den übergebenen String im Array "Artikel" abzulegen
    void StringtoArtikel (String s) throws Exception
          {
          int nummer = 0;
          String name = null;
          double preis = 0;

          // Initialisiert Stringteiler und definiert die Trennzeichen
          StringTokenizer T = new StringTokenizer (s, " ,");

          try {
              // Überprüft, ob die Parameteranzahl korrekt ist
              if (T.countTokens () != 3)
                    throw new Exception ("Falsche Syntax:\t" + s + "\tDatensatz ignoriert.\n\t\t
<Artikelnummer>,<Bezeichnung>,<Preis>");

              // Ließt die Artikelnummer, wenn nicht dreistellig Abbruch
              name = T.nextToken ();
              if (name.length() !=3)
                    throw new Exception ("Ungültige Artikelnummer im angegebenen File.");
              nummer = Integer.parseInt (name);

              // Ließt Artikelnamen, Überprüft, ob der Artikelname zulässig ist, und bricht bei Fehler ab
              name = T.nextToken ();
              if (name.length() > 20)
                    throw new Exception ("Artikelbezeichnung zu lang (max. 20 Zeichen)");

              // Ließt den Preis
              preis = Double.valueOf(T.nextToken()).doubleValue();
              }
          catch (Exception e)
              {
              throw new Exception (e.getMessage());
              }

          // Wenn kein Fehler bei der Umwandlung aufgetreten ist, Datensatz speichern
          a[anz++] = new Daten (nummer,name,preis);
          }


    // Gibt die Liste aller vorhanden Artikel aus
    void printall ()
          {
          System.out.print ("\nListe der vorhandenen Artikel:\n\n");
          for (int i = 0; i < anz; i++)
              System.out.println (a[i].nummer + ", " + a[i].name + ", " + a[i].preis);
          System.out.print ("\n\n\n");
          }

    // Sucht in Artikeldaten nach übergebener Artikelnummer und gibt sie bei Erfolg zurück
    int find (int art)
          {
          for (int i = 0; i < anz; i++)
              if (art == a[i].nummer)
                    return (i);
          return -1;
          }
    }


// Hier werden die Rechensätze abgelegt
class Rechnungen
    {
    int artikel;
    int menge;
    Rechnungen (int a, int m)
          {
          artikel = a;
          menge = m;
          }
    }


// Verwaltet die Rechensätze
class Rechnung
    {
    Rechnungen [] r = new Rechnungen [10];
    private int anz;

    // leerer Konstruktor
    Rechnung ()
          {
          anz = 0;
          }

    // ließt die Rechnungen aus angegebenem File ein
    void load (String path, String filename) throws Exception
          {
          String s = null;
          int stück = 0, trennzeichen = 0, artikel = 0, artnr = 0;

          // Öffnet File
          ReadFile file = new ReadFile (path, filename);
```

```
          // Ließt bis zum Fileende ein, oder bricht bei mehr als 10 Rechnungssätzen ab
          while ((s = file.readLine()) != null)
              {
              // Abbruch, wenn mehr als 10 Rechensätzen
            if (anz > 10)
                  throw new Exception ("Zu viele Artikel in dieser Rechnung.\nEs wurden nur die ersten 10 berücksichtigt.");
              try
                    {
                    // Wandelt Sting in Rechnungsdatensatz um
                    StringtoRechnung (s);
                    }
              catch (Exception e)
                    {
                    System.out.println (e.getMessage ());
                    }
              }
          }


    // Versucht den übergebenen String im Array "Rechnungen" abzulegen
    void StringtoRechnung (String s) throws Exception
          {
          int artikel = 0, stück = 0, artnr = 0;
          String help = null;

          // Initialisiert Stringteiler und definiert die Trennzeichen
          StringTokenizer T = new StringTokenizer (s, " ,");

          try {
              if (T.countTokens () != 2)
                    throw new Exception ("Falsche Syntax:\t" + s + "\tDatensatz ignoriert.\n\t\t <Artikelnummer>,<Menge>");
              artikel = Integer.parseInt (T.nextToken());
              stück = Integer.parseInt (T.nextToken());
              }
          catch (Exception e)
              {
              throw new Exception ("Fehler im Datensatz:\t" + s + "\tDatensatz ignoriert.");
              }

          // Überprüft, ob dieser Artikel in der Rechnung schon vorkommt
          //  ja: ergänzt den alten Rechnungssatz
          //  nein: legt neuen Rechnungssatz an
          if ((artnr = find (artikel)) >= 0)
              r [artnr].menge += stück;
          else
              r [anz++] = new Rechnungen (artikel, stück);
          }


    // Sucht in Rechnungssätzen nach angegebener Artikelnummer
    int find (int art)
          {
          for (int i = 0; i < anz; i++)
              if (art == r[i].artikel)
                    return (i);
          return -1;
          }

    // Liefert die Anzahl der geladenen Rechnungen zurück
    int anz ()
          {
          return anz;
          }

    // Setzt die Einträge auf 0 zurück
    void destroy ()
          {
          anz = 0;
          }
    }


// Führt die Berechnungen durch
class Berechnung
    {
    Berechnung (Artikeln a, Rechnung e)
          {
          int j=0;
          double gesamtpreis = 0;
          String trennung = "============================================================";
          NumberFormat N = NumberFormat.getInstance ();
          N.setMaximumFractionDigits (2);
          N.setMinimumFractionDigits (2);
          N.setGroupingUsed (false);
          System.out.print ("\n");
          for(int i = 0; i < e.anz(); i++)
              {
              if(0 <= (j = a.find (e.r[i].artikel)))
                    {
                    System.out.println (i + 1 + "\t" + a.a[j].nummer + "\t" + a.a[j].name + "\t" + a.a[j].preis + "\t" +
e.r[i].menge + "\t" + N.format(a.a[j].preis * e.r[i].menge));
                    gesamtpreis += a.a[j].preis * e.r[i].menge;
                    }
              else
                    System.out.println (i + 1 + "\t" + e.r[i].artikel + "\tArtikel nicht gefunden");
              }
          System.out.print (trennung + "\nGesamtpreis:\t\t\t\t" + N.format(gesamtpreis) + "\n\n");
          }
    }
```

# 2 Zweites Programm: Datumsumwandlung

## *2.1 Modularisierung und Datenstruktur*

### 2.1.1 Grundidee: Struktogramm

<table>
<tr><td colspan="3">Variable & Konstanten initialisieren</td></tr>
<tr><td colspan="3">File öffnen & solange Zeilen vorhanden</td></tr>
<tr><td rowspan="5"></td><td colspan="2">Zeile einlesen</td></tr>
<tr><td colspan="2">Zeile auswerten und in Datum schreiben</td></tr>
<tr><td colspan="2">Datum gültig?</td></tr>
<tr><td>Ja:</td><td rowspan="3">Nein: Fehlermeldung</td></tr>
<tr><td>Altes Datum ausgeben</td></tr>
<tr><td>Neues Datum ausgeben</td></tr>
</table>

### 2.1.2 Wahl der Klassen

| | |
|---|---|
| Datum: | Beinhaltet alle Methoden die zur Datumsumwandlung und –bearbeitung nötig sind |
| MagString: | Beinhaltet meine Methoden zur Bearbeitung von Strings |
| DateException: | Eine Erweiterung der Klasse Exception, um meine eigenen Fehlermeldungen werfen zu können |

### 2.1.3 Wahl der Variablen in der Klasse Datum:

| | |
|---|---|
| Datumsspeicher: | **Private Class** *Daten* Unterklasse von *Datum* |
| Zeilenspeicher: | **String** *h* im Hauptprogramm |

### 2.1.4 Wahl der Methoden

#### 2.1.4.1 Klasse *Datum*:

| | |
|---|---|
| CheckDate(): | Überprüft mir die Korrektheit eines Datums |
| GetNewDate(): | Liefert mir das neue Datum als String |
| OldtoNew(**String**): | Wandelt das eingelesene Datum in mein Datumsformat um und liefert an das Hauptprogramm das umgewandelte Datum |
| GetMonthIndex(**String**): | Liefert mir die Monatszahl vom Monatsnamen |
| GetMonthName(**Int**): | Liefert mir den Monatsnamen der Monatszahl |
| GetDateFormat(): | Überprüft mir, welches Format das aktuelle Datum hat |

#### 2.1.4.2 Klasse *MagString*:

| | |
|---|---|
| ContainsNumber(**String**): | Überprüft mir, ob in einem String Zahlen enthalten sind |
| GetFirstNumber(**String**): | Liefert mir den Index der ersten Zahl in einem String |

## *2.2 Pseudocode*

```
Variablen definieren und initialisieren
File mit Daten öffnen
Solange Zeile vorhanden:
    Zeile einlesen
    Datumsformat überprüfen und in Zeile in Substrings aufteilen (Tokenize)
    Wenn Punktdatum:
      Token 1 in Tag sichern
      Token 2 vom Textformat in Index umwandeln und in Monat sichern
      Token 3 in Jahr sichern
    Wenn Slash- oder Strichdatum:
      Token 1 in Jahr sichern
      Token 2 in Monat sichern
      Token 3 in Jahr sichern
    Datum auf Richtigkeit überprüfen, wenn ungültig gesichertes Datum wieder löschen und Fehlermeldung
    Altes & Neues Datum ausgeben
```

## 2.3 Source Code

```
/*  Programm:       Beispiel2.java
    Aufgabe:        Ließt Daten aus einem File ein, und konvertiert sie in anderes Format
    Formate:    1999/1/1
                    1999-1-1
                    1. Jänner 1999 oder 1. Jan. 1999 */

import java.util.*;
import java.text.*;

class Beispiel2
    {
    final static String pfad = "f:\\daten\\uni\\1.semester\\eprog\\abgabe1\\beispiel2\\";
    final static String filename = "test.txt";

    public static void main (String argv[])
          {
          Datum d = new Datum ();
          String s = null;

          // öffnet File
          ReadFile file = new ReadFile (pfad, filename);

          // Solange Zeilen vorhanden: einlesen
          while ((null != (s = file.readLine())) && (s.length() > 0))
              {
              try {
              System.out.println (s + "          \t-> \t" + d.OldtoNew (s));
                  }
              catch (DateException e)
                  {
                  System.out.println (s + e.getMessage());
                  }
              }
          // Schließt das File
          file.close ();
          }
    }


// Datums Exceptions Klasse
class DateException extends Exception {
    DateException (String s)
          {
          super (s);
          }
     }


// Hier werden die Daten verwaltet
class Datum
    {
    Daten d = new Daten (0,0,0,null);

    // Hier werden die Daten abgelegt
    private class Daten
          {
        int Tag = 0, Monat = 0, Jahr = 0;
          String trennzeichen = null;

          // Konstruktor: hier werden die Variablen initialisiert
          Daten (int t, int m, int j, String tr)
              {
              Tag = t;
              Monat = m;
              Jahr = j;
              trennzeichen = tr;
              }
          }


    // Konstruktor
    public Datum ()
          {
          }
```

```
    // Wandelt String in Datum um und gibt es im neuen Format aus
    String OldtoNew (String s) throws DateException
          {
          String h = null;

          // Initialisiert Stringteiler und definiert die Trennzeichen
          StringTokenizer T = new StringTokenizer (s, " .-/");

          // Stellt das Format fest
          if ((d.trennzeichen = GetDateFormat (s)) != null)
              {

              try {
                    // Teilt des Datum je nach Format in seine Bestandteile auf
                    if (d.trennzeichen.equals ("."))
                        {
                        d.Tag = Integer.parseInt (T.nextToken ());
                        h = T.nextToken();

                        // Wenn kein Trennzeichen zw. Monat und Jahr, "händische" Trennung
                        if ((MagString.ContainsNumbers (h)) && (h.length() > 3))
                              {
                              d.Monat = GetMonthIndex (h.substring (0, MagString.GetFirstNumber (h) - 1));
                              d.Jahr = Integer.parseInt (h.substring (MagString.GetFirstNumber (h), h.length ()));
                              }
                        else
                              {
                              d.Monat = GetMonthIndex (h);
                              d.Jahr = Integer.parseInt (T.nextToken ());
                              }
                        }
                    else
                        {
                        d.Jahr = Integer.parseInt (T.nextToken ());
                        d.Monat = Integer.parseInt (T.nextToken ());
                        d.Tag = Integer.parseInt (T.nextToken ());
                        }

                    // Überprüft, ob das gelesene Datum gültig ist. Wenn nicht, wird es ignoriert & eine Fehlermeldung
ausgegeben
                  CheckDate (d);
                  }
              catch (Exception e) {
                  throw new DateException (e.getMessage());
                  }
              }
              return getNewDate ();
          }

    // Bestimmt den Index eines Monat aus einem String (Dez. -> 12)
    int GetMonthIndex (String monat)
          {
          for (int m = 1; m <= 12; m++)
              if (monat.indexOf (getMonthName(m).substring (0,2)) >= 0)
                    return m;
          return -1;
          }


    // Liefert den Namen eines Monats zurück
    String getMonthName (int monat)
          {
          String s[] = {"Jänner", "Feber", "März", "April", "Mai", "Juni", "July", "August", "September", "Oktober",
"November", "Dezember"};
          return s[monat-1];
          }


    // Überprüft, ob das Datum gültig ist.
    void CheckDate (Daten d) throws DateException
          {
        if ((d.Monat < 1) || (d.Monat > 12))
              throw new DateException (":\tUngültiges Monat -  Datensatz ignoriert!");
          switch (d.Monat) {
              case 4:
              case 6:
              case 9:
              case 11:        if ((d.Tag < 1) || (d.Tag > 30))
                              throw new DateException (":\tUngültiger Tag - Datensatz ignoriert!");
              case 2:         if (((d.Jahr % 4) != 0) && (d.Tag > 28) || (d.Tag > 29))
                              throw new DateException (":\tUngültiger Tag (Beachte Schaltjahre) - Datensatz ignoriert!");
              default:        if ((d.Tag < 1) || (d.Tag > 31))
                              throw new DateException (":\tUngültiger Tag - Datensatz ignoriert!");
              }
          }
```

```
    // Bestimmt das Trennzeichen und übergibt es an das aufrufende Programm
    public String GetDateFormat (String s)
          {
          // Überprüft das Format und bestimmt das Trennzeichen
          if ((s.indexOf ('/') > 0) && (s.indexOf (' ') < 0)) return "/";
          else if ((s.indexOf ('-') > 0) && (s.indexOf (' ') < 0)) return "-";
          else if (s.indexOf ('.') > 0) return ".";
          System.out.println (s + ":\tFalsche Trennzeichen - Datensatz ignoriert");
          return null;
          }


    // Gibt Datum im neuen Format als String zurück
    public String getNewDate ()
          {
          String h;
          NumberFormat N = NumberFormat.getInstance ();

          // Deaktiviert Trennung bei Tausenderstellen
          N.setGroupingUsed (false);

          // setzt Zahlenformat auf minimal 4 Stellen
          N.setMinimumIntegerDigits (4);
          h = N.format (d.Jahr);

          // setzt Zahlenformat auf minimal 2 Stellen
          N.setMinimumIntegerDigits (2);

          if (d.trennzeichen.equals ("."))
              return h + "/" + N.format (d.Monat) + "/" + N.format (d.Tag);
          else
              return d.Tag + ". " + getMonthName (d.Monat) + " " + d.Jahr;
          }
    }


// im File MagString.java
public MagString
    {
    public boolean ContainsNumbers (String str)
          {
          String Numbers = "1234567890";

          for (int j=0; j < Numbers.length(); j++)
              if (str.indexOf(Numbers.charAt(j)) >= 0)
                        return true;
          return false;
          }

    public int GetFirstNumber (String str)
          {
          String Numbers = "1234567890";
          int start = str.length(), index = 0;

          for (int j=0; j < Numbers.length(); j++)
              {
              index = str.indexOf(Numbers.charAt(j));
              if ((index < start) && (index >= 0))
                    start = index;
              }
          return start;
          }
    }
```