



University of Applied Sciences

HOCHSCHULE
EMDEN•LEER

Fachbereich Technik
Abteilung Elektrotechnik und Informatik

PROGRAMMIEREN III AUFGABE 2

ENTWICKLUNG EINES WARENWIRTSCHAFTSSYSTEMS: LESEN UND SCHREIBEN VON ARTIKELINFORMATIONEN AUS UND IN DATEIEN IN C++

Gruppe A1
Studiengang Elektrotechnik
Vorgelegt von

Yaman Alsaady
Oliver Schmidt

Matr. Nr. 7023554
Matr. Nr. 7023462

Emden, 24. Oktober 2023

Betreut von
Dr. Olaf Bergmann
Dipl.-Ing. Behrend Pupkes

1 Quellencode	1
1.1 Datei 'Lager.hh'	1
1.2 Datei 'Lager.cc'	7
1.3 Datei 'main.cc'	11
2 Klassen-Dokumentation	15
2.1 Artikel Klassenreferenz	15
2.1.1 Ausführliche Beschreibung	17
2.1.2 Beschreibung der Konstruktoren und Destruktoren	17
2.1.2.1 Artikel()	17
2.1.3 Dokumentation der Elementfunktionen	17
2.1.3.1 getArtikelnummer()	17
2.1.3.2 getGruppe()	18
2.1.3.3 getLagerabstand()	18
2.1.3.4 getMasseinheit()	18
2.1.3.5 getName()	18
2.1.3.6 getNormpreis()	19
2.1.3.7 getStrMasseinheit()	19
2.1.3.8 getVerkaufspreis()	19
2.1.3.9 print()	19
2.1.3.10 setArtikelnummer()	20
2.1.3.11 setGruppe()	20
2.1.3.12 setLagerbestand()	21
2.1.3.13 setMasseinheit()	21
2.1.3.14 setName()	22
2.1.3.15 setNormpreis()	22
2.1.3.16 setVerkaufspreis()	22
2.2 Fluessigkeit Klassenreferenz	23
2.2.1 Ausführliche Beschreibung	26
2.2.2 Beschreibung der Konstruktoren und Destruktoren	26
2.2.2.1 Fluessigkeit() [1/2]	26
2.2.2.2 Fluessigkeit() [2/2]	26
2.2.3 Dokumentation der Elementfunktionen	26
2.2.3.1 getVolume()	26
2.2.3.2 setVerkaufspreis()	27
2.2.3.3 setVolume()	27
2.3 Schuettgut Klassenreferenz	27
2.3.1 Ausführliche Beschreibung	30
2.3.2 Beschreibung der Konstruktoren und Destruktoren	30
2.3.2.1 Schuettgut() [1/2]	30
2.3.2.2 Schuettgut() [2/2]	30
2.3.3 Dokumentation der Elementfunktionen	30

2.3.3.1 getLosgroesse()	30
2.3.3.2 setLosgroesse()	31
2.3.3.3 setVerkaufspreis()	31
2.4 Stueckgut Klassenreferenz	31
2.4.1 Ausführliche Beschreibung	33
2.4.2 Beschreibung der Konstruktoren und Destruktoren	33
2.4.2.1 Stueckgut()	33
2.5 Warengruppen Klassenreferenz	34
2.5.1 Beschreibung der Konstruktoren und Destruktoren	34
2.5.1.1 Warengruppen()	34
2.5.2 Dokumentation der Elementfunktionen	34
2.5.2.1 addGruppe()	34
2.5.2.2 changeGruppe()	35
2.5.2.3 delGruppe()	35
2.5.2.4 getGruppe()	35
3 Datei-Dokumentation	37
3.1 lager.cc-Dateireferenz	37
3.1.1 Ausführliche Beschreibung	38
3.1.2 Dokumentation der Funktionen	38
3.1.2.1 operator>>()	38
3.2 lager.hh-Dateireferenz	39
3.2.1 Ausführliche Beschreibung	41
3.2.2 Dokumentation der Funktionen	41
3.2.2.1 operator<<()	41
3.2.2.2 operator>>()	41
3.3 lager.hh	42
3.4 main.cc-Dateireferenz	44
3.4.1 Ausführliche Beschreibung	45
3.4.2 Dokumentation der Funktionen	45
3.4.2.1 main()	45
3.4.2.2 read()	46
3.4.2.3 readWrite()	47
Index	49

Kapitel 1

Quellencode

1.1 Datei 'Lager.hh'

Listing 1.1 Die Header-Datei lager.hh

```
1  /**
2   * @file lager.hh
3   * @authors Yaman Alsaady, Oliver Schmidt
4   * @brief Definitionen der Lagerverwaltungsfunktionen.
5   * @version 0.2
6   * @date 2023-10-19
7   *
8   * Dieses Header-Datei enthaelt die Definitionen von Klassen und
9   * Funktionen zur Verwaltung von Artikeln und Warengruppen in einem
10  * C++-Programm.
11  *
12  * @copyright Copyright (c) 2023
13  *
14  */
15
16 #ifndef LAGER_HH
17 #define LAGER_HH
18
19 #include <iostream>
20 #include <map>
21 #include <string>
22
23 using namespace std;
24 enum masseinheit { stk, kg, l };
25 typedef double preis;
26
27 class Warengruppen {
28 private:
29     map<string, string> mapGruppe;
30     map<string, string>::iterator iter;
31
32 public:
33     /**
34      * @brief Konstruktor fuer die Klasse "Warengruppen".
35      *
36      * Dieser Konstruktor initialisiert eine leere Warengruppenliste.
37      */
38     Warengruppen();
39
40     /**
41      * @brief Setzt eine Standard-Warengruppenliste.
42      */
43     void defaultList();
44
45     /**
46      * @brief Gibt den Namen der Warengruppe fuer einen gegebenen Code zurueck.
```

```

47     *
48     * @param code Der Warengruppencode.
49     * @return Der Name der Warengruppe oder der Code, falls keine Warengruppe
50     * gefunden wurde.
51     */
52     string getGruppe(string code);
53
54     /**
55     * @brief Fuegt eine neue Warengruppe hinzu.
56     *
57     * @param code Der Warengruppencode.
58     * @param name Der Name der Warengruppe.
59     */
60     void addGruppe(string code, string name);
61
62     /**
63     * @brief Loescht eine Warengruppe anhand ihres Codes.
64     *
65     * @param code Der Warengruppencode.
66     */
67     void delGruppe(string code);
68
69     /**
70     * @brief Aendert den Namen einer vorhandenen Warengruppe.
71     *
72     * @param code Der Warengruppencode.
73     * @param name Der neue Name der Warengruppe.
74     */
75     void changeGruppe(string code, string name);
76
77     /**
78     * @brief Loescht alle Warengruppen und setzt sie zurck.
79     */
80     void clear();
81 };
82
83 /**
84 * @brief Die Klasse "Artikel" repraesentiert einen Artikel mit verschiedenen
85 * Eigenschaften.
86 */
87 class Artikel {
88 protected:
89     string artikelname;
90     string artikelnummer;
91     unsigned int lagerbestand;
92     masseinheit einheit;
93     preis verkaufpreis;
94     preis normpreis;
95
96 public:
97     /**
98     * @brief Standardkonstruktor fuer die Klasse "Artikel".
99     */
100    Artikel();
101
102    /**
103    * @brief Konstruktor fuer die Klasse "Artikel".
104    *
105    * @param name Der Name des Artikels.
106    * @param num Die Artikelnummer des Artikels.
107    * @param bestand Der Lagerbestand des Artikels.
108    * @param einheit Die Einheit des Artikels (stk, kg, l).
109    * @param vp Der Verkaufspreis des Artikels.
110    * @param np Der Normalpreis des Artikels.
111    */
112    Artikel(string name, string num, unsigned int bestand, masseinheit einheit,
113            preis vp, preis np);
114
115    // Getter-Funktionen
116
117    /**
118    * @brief Destruktor fuer die Klasse "Artikel".
119    */

```

```
120 ~Artikel();
121
122 /**
123  * @brief Statische Warengruppen-Instanz, die fuer alle Artikel gemeinsam
124  * genutzt wird.
125  */
126 static Warengruppen gruppe;
127
128 /**
129  * @brief Setzt die Warengruppe fuer Artikel.
130  *
131  * @param g Die Warengruppe, die zugewiesen werden soll.
132  */
133 static void setGruppe(Warengruppen g);
134
135 /**
136  * @brief Gibt den Namen des Artikels zurueck.
137  *
138  * @return Der Name des Artikels.
139  */
140 string getName() const;
141
142 /**
143  * @brief Gibt die Artikelnummer des Artikels zurueck.
144  *
145  * @return Die Artikelnummer des Artikels.
146  */
147 string getArtikelnummer() const;
148
149 /**
150  * @brief Gibt den Lagerbestand des Artikels zurueck.
151  *
152  * @return Der Lagerbestand des Artikels.
153  */
154 unsigned int getLagerabstand() const;
155
156 /**
157  * @brief Gibt die Masseinheit des Artikels als Wert aus der Enumeration
158  * zurueck.
159  *
160  * @return Die Masseinheit des Artikels als Wert aus der Enumeration (stk, kg,
161  * l).
162  */
163 masseinheit getMasseinheit() const;
164
165 /**
166  * @brief Gibt die Masseinheit des Artikels als Zeichenkette zurueck.
167  *
168  * @return Die Masseinheit des Artikels als Zeichenkette ("stk", "kg", "l").
169  */
170 string getStrMasseinheit() const;
171
172 /**
173  * @brief Gibt den Verkaufspreis des Artikels zurueck.
174  *
175  * @return Der Verkaufspreis des Artikels.
176  */
177 preis getVerkaufspreis() const;
178
179 /**
180  * @brief Gibt den Normalpreis des Artikels zurueck.
181  *
182  * @return Der Normalpreis des Artikels.
183  */
184 preis getNormpreis() const;
185
186 /**
187  * @brief Gibt die Warengruppe des Artikels zurueck.
188  *
189  * @return Die Warengruppe des Artikels oder die Artikelnummer, falls keine
190  * Warengruppe gefunden wurde.
191  */
192 string getGruppe() const;
```

```

193 // Setter-Funktionen
194
195
196 /**
197  * @brief Setzt den Namen des Artikels.
198  *
199  * @param name Der neue Name des Artikels.
200  */
201 void setName(string name);
202
203 /**
204  * @brief Setzt die Artikelnummer des Artikels.
205  *
206  * @param num Die neue Artikelnummer des Artikels.
207  */
208 void setArtikelnummer(string num);
209
210 /**
211  * @brief Setzt den Lagerbestand des Artikels.
212  *
213  * @param bestand Der neue Lagerbestand des Artikels.
214  */
215 void setLagerbestand(unsigned int bestand);
216
217 /**
218  * @brief Setzt die Masseinheit des Artikels.
219  *
220  * @param einheit Die neue Masseinheit des Artikels (stk, kg, l).
221  */
222 void setMasseinheit(masseinheit einheit);
223
224 /**
225  * @brief Setzt den Verkaufspreis des Artikels.
226  *
227  * @param vp Der neue Verkaufspreis des Artikels.
228  */
229 void setVerkaufspreis(preis vp);
230
231 /**
232  * @brief Setzt den Normalpreis des Artikels.
233  *
234  * @param np Der neue Normalpreis des Artikels.
235  */
236 void setNormpreis(preis np);
237
238 /**
239  * @brief Gibt die Artikelinformationen aus.
240  *
241  * Diese Funktion gibt die Informationen des Artikels aus, einschliesslich
242  * Artikelname, Artikelnummer, Lagerbestand, Verkaufspreis, Masseinheit und
243  * Normpreis.
244  *
245  * @param os Die Ausgabestromreferenz, in die die Informationen geschrieben
246  * werden.
247  * @return Die Ausgabestromreferenz, in die die Informationen geschrieben
248  * wurden.
249  */
250 ostream &print(ostream &ostream);
251 };
252 /**
253  * @brief ueberladen des Ausgabeoperators fuer die Artikelklasse.
254  *
255  * Diese Funktion ermoeoglicht das Ausgeben eines Artikels mit dem Ausgabeoperator
256  * '<<'.
257  *
258  * @param os Die Ausgabestromreferenz, in die die Informationen geschrieben
259  * werden.
260  * @param produkt Der Artikel, der ausgegeben werden soll.
261  * @return Die Ausgabestromreferenz, in die die Informationen geschrieben
262  * wurden.
263  */
264 ostream &operator<<(ostream &os, Artikel produkt);
265

```

```

266 /**
267  * @brief ueberladen des Eingabeoperators fuer die Artikelklasse.
268  *
269  * Diese Funktion ermoeeglicht das Einlesen von Artikelinformationen mit dem
270  * Eingabeoperator '>>'.
271  *
272  * @param is Die Eingabestromreferenz, aus der die Informationen eingelesen
273  * werden.
274  * @param produkt Der Artikel, in den die Informationen eingelesen werden
275  * sollen.
276  */
277 void operator>>(istream &is, Artikel &produkt);
278
279 /**
280  * @brief Die Klasse "Stueckgut" erbt von der Klasse "Artikel" und spezialisiert
281  * sie fuer Stueckgut-Artikel.
282  */
283 class Stueckgut : public Artikel {
284 private:
285 public:
286     /**
287      * @brief Konstruktor fuer die Klasse "Stueckgut".
288      *
289      * @param name Der Name des Stueckgut-Artikels.
290      * @param num Die Artikelnummer des Stueckgut-Artikels.
291      * @param vp Der Verkaufspreis des Stueckgut-Artikels.
292      * @param bestand Der Lagerbestand des Stueckgut-Artikels (Standardwert: 1).
293      */
294     Stueckgut(Artikel produkt);
295     Stueckgut(string name, string num, preis vp, unsigned int bestand = 1);
296 };
297
298 /**
299  * Die Klasse "Schuettgut" erbt von der Klasse "Artikel" und spezialisiert sie
300  * fuer Schuettgut-Artikel.
301  * Die Klasse "Schuettgut" erweitert die Basisfunktionalitaet der Klasse
302  * "Artikel" um die Beruecksichtigung der Losgroesse.
303  */
304 class Schuettgut : public Artikel {
305 private:
306     double losgroesse;
307
308 public:
309     /**
310      * @brief Konstruktor fuer die Klasse "Schuettgut" unter Verwendung eines
311      * bereits existierenden Artikels.
312      *
313      * @param produkt Der Artikel, aus dem ein Schuettgut-Artikel erstellt wird.
314      */
315     Schuettgut(Artikel produkt);
316
317     /**
318      * @brief Konstruktor fuer die Klasse "Schuettgut".
319      *
320      * @param name Der Name des Schuettgut-Artikels.
321      * @param num Die Artikelnummer des Schuettgut-Artikels.
322      * @param groesse Die Losgroesse des Schuettgut-Artikels.
323      * @param np Der Normalpreis des Schuettgut-Artikels.
324      * @param bestand Der Lagerbestand des Schuettgut-Artikels (Standardwert: 1).
325      */
326     Schuettgut(string name, string num, double groesse, preis np,
327                unsigned int bestand = 1);
328
329     /**
330      * @brief Gibt die Losgroesse des Schuettgut-Artikels zurueck.
331      *
332      * @return Die Losgroesse des Artikels.
333      */
334     double getLosgroesse() const;
335
336     /**
337      * @brief Setzt den Verkaufspreis des Schuettgut-Artikels basierend auf der
338      * Losgroesse.

```



```

339     *
340     * @param vp Der Verkaufspreis, der gesetzt werden soll.
341     */
342     void setVerkaufspreis(preis vp);
343
344     /**
345     * @brief Setzt die Losgroesse des Schuettgut-Artikels.
346     *
347     * @param groesse Die neue Losgroesse.
348     */
349     void setLosgroesse(double groesse);
350 };
351
352 /**
353 * @brief Die Klasse "Fluessigkeit" erbt von der Klasse "Artikel" und
354 * spezialisiert sie fuer Fluessigkeits-Artikel.
355 *
356 * Die Klasse "Fluessigkeit" erweitert die Basisfunktionalitaet der Klasse
357 * "Artikel" um die Beruecksichtigung des Volumens.
358 */
359 class Fluessigkeit : public Artikel {
360 private:
361     double volume;
362
363 public:
364     /**
365     * @brief Konstruktor fuer die Klasse "Fluessigkeit" unter Verwendung eines
366     * bereits existierenden Artikels.
367     *
368     * @param produkt Der Artikel, aus dem ein Fluessigkeits-Artikel erstellt
369     * wird.
370     */
371     Fluessigkeit(Artikel produkt);
372
373     /**
374     * @brief Konstruktor fuer die Klasse "Fluessigkeit".
375     *
376     * @param name Der Name des Fluessigkeits-Artikels.
377     * @param num Die Artikelnummer des Fluessigkeits-Artikels.
378     * @param vol Das Volumen des Fluessigkeits-Artikels.
379     * @param np Der Normalpreis des Fluessigkeits-Artikels.
380     * @param bestand Der Lagerbestand des Fluessigkeits-Artikels (Standardwert:
381     * 1).
382     */
383     Fluessigkeit(string name, string num, double vol, preis np,
384                 unsigned int bestand = 1);
385
386     /**
387     * @brief Gibt das Volumen des Fluessigkeits-Artikels zurueck.
388     *
389     * @return Das Volumen des Artikels.
390     */
391     double getVolume() const;
392
393     /**
394     *
395     * @param vp Der Verkaufspreis, der gesetzt werden soll.
396     */
397     void setVerkaufspreis(preis vp);
398
399     /**
400     * @brief Setzt das Volumen des Fluessigkeits-Artikels.
401     *
402     * @param vol Das neue Volumen.
403     */
404     void setVolume(double vol);
405 };
406
407 #endif // !LAGER_HH

```

1.2 Datei 'Lager.cc'

Listing 1.2 Die Header-Datei lager.cc

```

1  /**
2   * @file lager.cc
3   * @authors Yaman Alsaady, Oliver Schmidt
4   * @brief Implementierung der Lagerverwaltungsfunktionen.
5   * @version 0.2
6   * @date 2023-10-19
7   *
8   * Dies ist die Implementierung der Funktionen fuer die Lagerverwaltung,
9   * einschliesslich der Warengruppenverwaltung und der Artikelklassen.
10  *
11  * @copyright Copyright (c) 2023
12  *
13  */
14
15 #include "lager.hh"
16 #include <cmath>
17 #include <iostream>
18 #include <map>
19 #include <sstream>
20 #include <string>
21 #include <vector>
22
23 Warengruppen::Warengruppen() {}
24 void Warengruppen::defaultList() {
25     mapGruppe["1005"] = "Fahrrad";
26     mapGruppe["4000"] = "Gemuese";
27     mapGruppe["4106"] = "Gemuese";
28     mapGruppe["4370"] = "Kaffee";
29     mapGruppe["5500"] = "Bier";
30     mapGruppe["5031"] = "Milch";
31 }
32 string Warengruppen::getGruppe(string code) {
33     if (mapGruppe[code] != "")
34         return mapGruppe[code];
35     else {
36         return code;
37     }
38 }
39
40 void Warengruppen::addGruppe(string code, string name) {
41     mapGruppe.insert({code, name});
42 }
43 void Warengruppen::changeGruppe(string code, string name) {
44     mapGruppe.insert_or_assign(code, name);
45 }
46 void Warengruppen::delGruppe(string code) { mapGruppe.erase(code); }
47
48 void Warengruppen::clear() { mapGruppe.clear(); }
49
50 Warengruppen Artikel::gruppe;
51 Artikel::Artikel() {}
52 Artikel::Artikel(string name, string num, unsigned int bestand,
53                 masseinheit einheit, preis vp, preis np)
54     : artikelname(name), artikelnummer(num), lagerbestand(bestand),
55       einheit(einheit), verkaufspreis(vp), normpreis(np) {}
56 Artikel::~Artikel() {}
57
58 void Artikel::setGruppe(Warengruppen g) { gruppe = g; }
59 string Artikel::getName() const { return artikelname; }
60 string Artikel::getArtikelnummer() const { return artikelnummer; }
61 unsigned int Artikel::getLagerabstand() const { return lagerbestand; }
62 masseinheit Artikel::getMasseinheit() const { return einheit; }
63 string Artikel::getStrMasseinheit() const {
64     switch (einheit) {
65     case 0:
66         return "Stk";
67     case 1:
68         return "kg";
69     case 2:

```

```

70     return "1";
71     default:
72         return "None";
73     }
74 }
75
76 preis Artikel::getVerkaufpreis() const { return verkaufpreis; }
77 preis Artikel::getNormpreis() const { return normpreis; }
78 string Artikel::getGruppe() const {
79     string artnum = artikelnummer;
80     artnum = artnum.erase(4);
81     return gruppe.getGruppe(artnum);
82     // return artnum;
83 }
84
85 void Artikel::setName(string name) { artikelname = name; }
86 void Artikel::setArtikelnummer(string num) { artikelnummer = num; }
87 void Artikel::setLagerbestand(unsigned int bestand) { lagerbestand = bestand; }
88 void Artikel::setMasseinheit(masseinheit einheit) { this->einheit = einheit; }
89 void Artikel::setVerkaufpreis(preis vp) { verkaufpreis = vp; }
90 void Artikel::setNormpreis(preis np) { normpreis = np; }
91
92 std::ostream &Artikel::print(std::ostream &os) {
93     return os << artikelname << "|" << artikelnummer << "|" << lagerbestand << "|"
94         << verkaufpreis << "|" << getStrMasseinheit() << "|" << normpreis;
95 }
96
97 std::ostream &operator<<(std::ostream &os, Artikel produkt) {
98     return produkt.print(os);
99 }
100
101 void operator>>(istream &is, Artikel &produkt) {
102     vector<string> beschreibung;
103     string text, name, num;
104     int bestand = 0;
105     preis vp = 0, np = 0;
106     masseinheit einheit;
107
108     getline(is, text);
109     stringstream ss(text);
110
111     if (!text[0]) {
112         throw(-1);
113     }
114     text = "";
115     for (size_t i = 0; getline(ss, text, '|') && i < 6; i++) {
116         beschreibung.push_back(text);
117     }
118     // cout << beschreibung.size() << endl;
119     if (beschreibung.size() < 5)
120         throw -1;
121     name = beschreibung[0];
122     num = beschreibung[1];
123
124     if (beschreibung[4] == "kg")
125         einheit = kg;
126     else if (beschreibung[4] == "l")
127         einheit = l;
128     else if (beschreibung[4] == "stk")
129         einheit = stk;
130     else {
131         einheit = stk;
132     }
133
134     for (size_t i = 1; i < 6; i++) {
135         for (size_t j = 0; j < beschreibung[i].length(); j++) {
136             if (beschreibung[i][j] == ' ') {
137                 beschreibung[i].erase(beschreibung[i].begin() + j);
138                 j--;
139             }
140         }
141     }
142 }

```

```

143     if (name == "" || num == "" || num.length() != 10) {
144         throw(-1);
145     }
146     if (beschreibung[3] == "" && beschreibung[4] == "") {
147         throw(-1);
148     }
149
150     if (beschreibung[2] != "") {
151         bestand = stoi(beschreibung[2]);
152     } else {
153         bestand = 0;
154     }
155     if (beschreibung[3] != "") {
156         vp = stof(beschreibung[3]);
157     }
158
159     if (beschreibung.size() > 5 && beschreibung[5] != "") {
160         np = stof(beschreibung[5]);
161     }
162     if (vp == 0)
163         vp = np;
164     if (np == 0)
165         np = vp;
166
167     produkt.setMasseinheit(einheit);
168     produkt.setName(beschreibung[0]);
169     produkt.setArtikelnummer(beschreibung[1]);
170     produkt.setLagerbestand(bestand);
171     produkt.setVerkaufpreis(vp);
172     produkt.setNormpreis(np);
173 }
174
175 Stueckgut::Stueckgut(Artikel produkt)
176     : Stueckgut(produkt.getName(), produkt.getArtikelnummer(),
177               produkt.getVerkaufpreis(), produkt.getLagerabstand()) {}
178 Stueckgut::Stueckgut(string name, string num, preis vp, unsigned int bestand)
179     : Artikel(name, num, bestand, stk, vp, vp) {}
180
181 Schuettgut::Schuettgut(Artikel produkt)
182     : Schuettgut(produkt.getName(), produkt.getArtikelnummer(),
183                 produkt.getVerkaufpreis() / produkt.getNormpreis(),
184                 produkt.getNormpreis(), produkt.getLagerabstand()) {}
185 Schuettgut::Schuettgut(string name, string num, double groesse, preis np,
186                       unsigned int bestand)
187     : Artikel(name, num, bestand, kg, (groesse * np), np), losgroesse(groesse) {}
188
189 double Schuettgut::getLosgroesse() const { return losgroesse; }
190 void Schuettgut::setLosgroesse(double groesse) {
191     losgroesse = groesse;
192     verkaufpreis = int((losgroesse * normpreis) * 100 + 0.5);
193     verkaufpreis /= 100;
194 }
195 void Schuettgut::setVerkaufpreis(preis vp) {
196     verkaufpreis = vp;
197     losgroesse = int((verkaufpreis / normpreis) * 100 + 0.5);
198     losgroesse /= 100;
199 }
200
201 Fluessigkeit::Fluessigkeit(Artikel produkt)
202     : Fluessigkeit(produkt.getName(), produkt.getArtikelnummer(),
203                   produkt.getVerkaufpreis() / produkt.getNormpreis(),
204                   produkt.getNormpreis(), produkt.getLagerabstand()) {}
205 Fluessigkeit::Fluessigkeit(string name, string num, double vol, preis np,
206                           unsigned int bestand)
207     : Artikel(name, num, bestand, l, (vol * np), np), volume(vol) {}
208 double Fluessigkeit::getVolume() const { return volume; }
209 void Fluessigkeit::setVolume(double vol) {
210     volume = vol;
211     verkaufpreis = int((volume * normpreis) * 100 + 0.5);
212     verkaufpreis /= 100;
213 }
214 void Fluessigkeit::setVerkaufpreis(preis vp) {
215     verkaufpreis = vp;

```

```
216     volume = int((verkaufpreis / normpreis) * 100 + 0.5);  
217     volume /= 100;  
218 }
```

1.3 Datei 'main.cc'

Listing 1.3 Die Header-Datei main.cc

```

1  /**
2   * @file main.cc
3   * @authors Yaman Alsaady, Oliver Schmidt
4   * @brief Implementierung des Lagerverwaltungssystems.
5   * @version 0.2
6   * @date 2023-10-19
7   *
8   * Dieses Programm implementiert ein Lagerverwaltungssystem, das es ermöglicht,
9   * Artikelinformationen aus Dateien zu lesen und sie nach Artikeltyp zu
10  * gruppieren. Die Informationen werden entweder auf der Konsole ausgegeben oder
11  * in eine Ausgabedatei geschrieben. Der Name der Ausgabedatei wird als
12  * Befehlszeilenargument uebergeben.
13
14  * @copyright Copyright (c) 2023
15  *
16  */
17  #include "lager.hh"
18  #include <cstdlib>
19  #include <fstream>
20  #include <iostream>
21  #include <ostream>
22  #include <string>
23  #include <vector>
24
25  using namespace std;
26
27  /**
28   * @brief Liest Artikelinformationen aus einer Datei und gruppiert sie nach
29   * Artikeltyp.
30   *
31   * Diese Funktion liest Artikelinformationen aus einer angegebenen Datei und
32   * gruppiert sie in die entsprechenden Vektoren je nach Artikeltyp (Stueckgut,
33   * Schuetztgut, Fluessigkeit). Anschliessend gibt sie die Informationen der
34   * Artikeltypen nacheinander aus.
35   *
36   * @param filename Der Dateiname der Eingabedatei, aus der die Informationen
37   * gelesen werden.
38   */
39  void read(string filename);
40
41  /**
42   * @brief Liest Artikelinformationen aus einer Eingabedatei und schreibt sie in
43   * eine Ausgabedatei.
44   *
45   * Diese Funktion liest Artikelinformationen aus einer angegebenen Eingabedatei
46   * und gruppiert sie in die entsprechenden Vektoren je nach Artikeltyp
47   * (Stueckgut, Schuetztgut, Fluessigkeit). Anschliessend werden die Informationen
48   * in die angegebene Ausgabedatei geschrieben.
49   *
50   * @param read Der Dateiname der Eingabedatei, aus der die Informationen gelesen
51   * werden.
52   * @param write Der Dateiname der Ausgabedatei
53   */
54  void readWrite(string read, string write);
55
56  /**
57   * @brief Lesen und Schreiben von Artikel aus Dateien.
58   *
59   * Diese Funktion ist der Einstiegspunkt des Programms und ermöglicht das Lesen
60   * und Schreiben von Artikelinformationen aus Dateien. Die Dateinamen werden als
61   * Befehlszeilenargumente uebergeben. Es kann maximal eine Datei zum Schreiben
62   * angegeben werden, waehrend mehrere Dateien zum Lesen zugelassen sind.
63   *
64   * @param argc Die Anzahl der Befehlszeilenargumente.
65   * @param argv Ein Array von Zeichenketten, das die Befehlszeilenargumente
66   * enthaelt.
67   * @return Eine Ganzzahl, die den Programmstatus zurueckgibt (0 fuer Erfolg,
68   * andere Werte fuer Fehler).
69   */

```

```

70 int main(int argc, char *argv[]) {
71     string filewrite = "";
72     Warengruppen gruppe;
73     gruppe.defaultList();
74     Artikel::gruppe = gruppe;
75
76     for (int i = 1; i < argc; i++) {
77         string arg = argv[i];
78         if (arg == "-o") {
79             filewrite = argv[i + 1];
80             ofstream file(filewrite);
81             if (!file)
82                 exit(EXIT_FAILURE);
83             file << "";
84             break;
85         }
86     }
87
88     for (int i = 1; i < argc; i++) {
89         if (string(argv[i]) == "-i") {
90             if (filewrite != "") {
91                 readWrite(argv[i + 1], filewrite);
92             } else {
93                 read(argv[i + 1]);
94             }
95         }
96     }
97
98     return 0;
99 }
100
101 void read(string filename) {
102     ifstream file(filename);
103     if (file.is_open()) {
104         vector<Fluessigkeit> fluessigkeit;
105         vector<Schuettgut> schuettgut;
106         vector<Stueckgut> stueckgut;
107         Artikel tmp;
108         do {
109             try {
110                 file >> tmp;
111                 switch (tmp.getMasseinheit()) {
112                     case 0:
113                         stueckgut.push_back(tmp);
114                         break;
115                     case 1:
116                         schuettgut.push_back(tmp);
117                         break;
118                     case 2:
119                         fluessigkeit.push_back(tmp);
120                         break;
121                 }
122             } catch (const int &ex) {
123             } catch (std::invalid_argument const &ex) {
124             }
125         } while (!file.eof());
126         for (auto schuett : schuettgut) {
127             std::cout << schuett << std::endl;
128         }
129         std::cout << std::endl;
130         for (auto stueck : stueckgut) {
131             std::cout << stueck << std::endl;
132         }
133         std::cout << std::endl;
134         for (auto fluessig : fluessigkeit) {
135             std::cout << fluessig << std::endl;
136         }
137         file.close();
138     }
139 }
140
141 void readWrite(string read, string write) {
142     ifstream readFile(read);

```

```

143 ofstream writeFile(write, std::ios::app);
144 if (readFile.is_open() && writeFile.is_open()) {
145     vector<Fluessigkeit> fluessigkeit;
146     vector<Schuettgut> schuettgut;
147     vector<Stueckgut> stueckgut;
148     Artikel tmp;
149     do {
150         try {
151             readFile >> tmp;
152             switch (tmp.getMasseinheit()) {
153                 case 0:
154                     stueckgut.push_back(tmp);
155                     break;
156                 case 1:
157                     schuettgut.push_back(tmp);
158                     break;
159                 case 2:
160                     fluessigkeit.push_back(tmp);
161                     break;
162             }
163         } catch (const int &ex) {
164         } catch (std::invalid_argument const &ex) {
165         }
166     } while (!readFile.eof());
167     for (auto schuett : schuettgut) {
168         writeFile << schuett << std::endl;
169     }
170     // writeFile << std::endl;
171     for (auto stueck : stueckgut) {
172         writeFile << stueck << std::endl;
173     }
174     // writeFile << std::endl;
175     for (auto fluessig : fluessigkeit) {
176         writeFile << fluessig << std::endl;
177     }
178     readFile.close();
179     writeFile.close();
180 }
181 }
182
183 //-----
184 //Wurde auskommentiert, da wir nur hier die Artikel-Klasse verwendet und nicht die abgeleitete
185 //Klassen.
186 //-----
187
188 /*
189 void readWrite_old(string read, string write) {
190     ifstream readFile(read);
191     ofstream writeFile(write, std::ios::app);
192     if (readFile.is_open() && writeFile.is_open()) {
193         Artikel a[10];
194         size_t i,j=1;
195         for (i = 0; !readFile.eof(); i++, j++) {
196             try {
197                 readFile >> a[i];
198             } catch (const int &ex) {
199                 i--;
200                 cerr << "Falsche Inforamtionen in Zeile: " << j << endl;
201             } catch (std::invalid_argument const &ex) {
202                 i--;
203                 cerr << "Falsche Inforamtionen in Zeile: " << j << endl;
204             }
205         }
206         for (size_t k = 0; k < i; k++) {
207             writeFile << a[k] << std::endl;
208         }
209         readFile.close();
210         writeFile.close();
211     }
212 }
213
214 void read_old(string filename) {
215     ifstream file(filename);

```



```
215     if (file.is_open()) {
216         Artikel a[10];
217         size_t i;
218         for (i = 0; !file.eof(); i++) {
219             try {
220                 file >> a[i];
221             } catch (const int &ex) {
222                 i--;
223             } catch (std::invalid_argument const &ex) {
224                 i--;
225             }
226         }
227         for (size_t k = 0; k < i; k++) {
228             std::cout << a[k] << std::endl;
229         }
230         file.close();
231     }
232 }
233 */
```

Kapitel 2

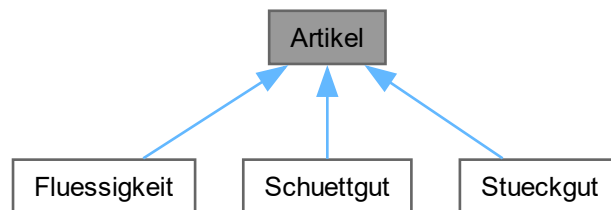
Klassen-Dokumentation

2.1 Artikel Klassenreferenz

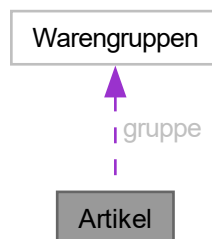
Die Klasse "Artikel" repraesentiert einen [Artikel](#) mit verschiedenen Eigenschaften.

```
#include <lager.hh>
```

Klassendiagramm für Artikel:



Zusammengehörigkeiten von Artikel:



Öffentliche Methoden

- **Artikel ()**
Standardkonstruktor für die Klasse "Artikel".
- **Artikel** (string name, string num, unsigned int bestand, masseinheit einheit, preis vp, preis np)
Konstruktor fuer die Klasse "Artikel".
- **~Artikel ()**
Destruktor fuer die Klasse "Artikel".
- string **getName** () const
Gibt den Namen des Artikels zurueck.
- string **getArtikelnummer** () const
Gibt die Artikelnummer des Artikels zurueck.
- unsigned int **getLagerabstand** () const
Gibt den Lagerbestand des Artikels zurueck.
- masseinheit **getMasseinheit** () const
Gibt die Masseinheit des Artikels als Wert aus der Enumeration zurück.
- string **getStrMasseinheit** () const
Gibt die Masseinheit des Artikels als Zeichenkette zurück.
- preis **getVerkaufspreis** () const
Gibt den Verkaufspreis des Artikels zurueck.
- preis **getNormpreis** () const
Gibt den Normalpreis des Artikels zurueck.
- string **getGruppe** () const
Gibt die Warengruppe des Artikels zurueck.
- void **setName** (string name)
Setzt den Namen des Artikels.
- void **setArtikelnummer** (string num)
Setzt die Artikelnummer des Artikels.
- void **setLagerbestand** (unsigned int bestand)
Setzt den Lagerbestand des Artikels.
- void **setMasseinheit** (masseinheit einheit)
Setzt die Masseinheit des Artikels.
- void **setVerkaufspreis** (preis vp)
Setzt den Verkaufspreis des Artikels.
- void **setNormpreis** (preis np)
Setzt den Normalpreis des Artikels.
- ostream & **print** (ostream &ostream)
Gibt die Artikelinformationen aus.

Öffentliche, statische Methoden

- static void **setGruppe** (Warengruppen g)
Setzt die Warengruppe fuer Artikel.

Statische öffentliche Attribute

- static Warengruppen **gruppe**
Statische Warengruppen-Instanz, die fuer alle Artikel gemeinsam genutzt wird.

Geschützte Attribute

- string **artikelname**
- string **artikelnummer**
- unsigned int **lagerbestand**
- masseinheit **einheit**
- preis **verkaufspreis**
- preis **normpreis**

2.1.1 Ausführliche Beschreibung

Die Klasse "Artikel" repraesentiert einen [Artikel](#) mit verschiedenen Eigenschaften.

2.1.2 Beschreibung der Konstruktoren und Destruktoren

2.1.2.1 Artikel()

```
Artikel::Artikel (
    string name,
    string num,
    unsigned int bestand,
    masseinheit einheit,
    preis vp,
    preis np )
```

Konstruktor fuer die Klasse "Artikel".

Parameter

<i>name</i>	Der Name des Artikels.
<i>num</i>	Die Artikelnummer des Artikels.
<i>bestand</i>	Der Lagerbestand des Artikels.
<i>einheit</i>	Die Einheit des Artikels (stk, kg, l).
<i>vp</i>	Der Verkaufspreis des Artikels.
<i>np</i>	Der Normalpreis des Artikels.

2.1.3 Dokumentation der Elementfunktionen

2.1.3.1 getArtikelnummer()

```
string Artikel::getArtikelnummer ( ) const
```

Gibt die Artikelnummer des Artikels zurueck.

Rückgabe

Die Artikelnummer des Artikels.

2.1.3.2 getGruppe()

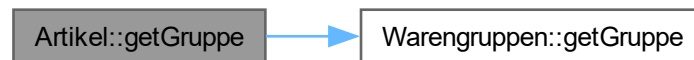
```
string Artikel::getGruppe ( ) const
```

Gibt die Warengruppe des Artikels zurueck.

Rückgabe

Die Warengruppe des Artikels oder die Artikelnummer, falls keine Warengruppe gefunden wurde.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



2.1.3.3 getLagerabstand()

```
unsigned int Artikel::getLagerabstand ( ) const
```

Gibt den Lagerbestand des Artikels zurueck.

Rückgabe

Der Lagerbestand des Artikels.

2.1.3.4 getMasseinheit()

```
masseinheit Artikel::getMasseinheit ( ) const
```

Gibt die Masseinheit des Artikels als Wert aus der Enumeration zurück.

Rückgabe

Die Masseinheit des Artikels als Wert aus der Enumeration (stk, kg, l).

2.1.3.5 getName()

```
string Artikel::getName ( ) const
```

Gibt den Namen des Artikels zurueck.

Rückgabe

Der Name des Artikels.

2.1.3.6 getNormpreis()

```
preis Artikel::getNormpreis ( ) const
```

Gibt den Normalpreis des Artikels zurueck.

Rückgabe

Der Normalpreis des Artikels.

2.1.3.7 getStrMasseinheit()

```
string Artikel::getStrMasseinheit ( ) const
```

Gibt die Masseinheit des Artikels als Zeichenkette zurück.

Rückgabe

Die Masseinheit des Artikels als Zeichenkette ("stk", "kg", "l").

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**2.1.3.8 getVerkaufspreis()**

```
preis Artikel::getVerkaufspreis ( ) const
```

Gibt den Verkaufspreis des Artikels zurueck.

Rückgabe

Der Verkaufspreis des Artikels.

2.1.3.9 print()

```
std::ostream & Artikel::print (
    ostream & ostream )
```

Gibt die Artikelinformationen aus.

Diese Funktion gibt die Informationen des Artikels aus, einschließlich Artikelname, Artikelnummer, Lagerbestand, Verkaufspreis, Maßeinheit und Normpreis.

Parameter

<code>os</code>	Die Ausgabestromreferenz, in die die Informationen geschrieben werden.
-----------------	--

Rückgabe

Die Ausgabestromreferenz, in die die Informationen geschrieben wurden.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

**2.1.3.10 setArtikelnummer()**

```
void Artikel::setArtikelnummer (
    string num )
```

Setzt die Artikelnummer des Artikels.

Parameter

<code>num</code>	Die neue Artikelnummer des Artikels.
------------------	--------------------------------------

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**2.1.3.11 setGruppe()**

```
void Artikel::setGruppe (
    Warengruppen g ) [static]
```

Setzt die Warengruppe fuer [Artikel](#).

Parameter

<i>g</i>	Die Warengruppe, die zugewiesen werden soll.
----------	--

2.1.3.12 setLagerbestand()

```
void Artikel::setLagerbestand (
    unsigned int bestand )
```

Setzt den Lagerbestand des Artikels.

Parameter

<i>bestand</i>	Der neue Lagerbestand des Artikels.
----------------	-------------------------------------

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**2.1.3.13 setMasseinheit()**

```
void Artikel::setMasseinheit (
    masseinheit einheit )
```

Setzt die Masseinheit des Artikels.

Parameter

<i>einheit</i>	Die neue Masseinheit des Artikels (stk, kg, l).
----------------	---

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



2.1.3.14 setName()

```
void Artikel::setName (  
    string name )
```

Setzt den Namen des Artikels.

Parameter

<i>name</i>	Der neue Name des Artikels.
-------------	-----------------------------

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



2.1.3.15 setNormpreis()

```
void Artikel::setNormpreis (  
    preis np )
```

Setzt den Normalpreis des Artikels.

Parameter

<i>np</i>	Der neue Normalpreis des Artikels.
-----------	------------------------------------

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



2.1.3.16 setVerkaufpreis()

```
void Artikel::setVerkaufpreis (  
    preis vp )
```

Setzt den Verkaufspreis des Artikels.

Parameter

vp	Der neue Verkaufspreis des Artikels.
----	--------------------------------------

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

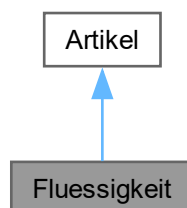
- [lager.hh](#)
- [lager.cc](#)

2.2 Fluessigkeit Klassenreferenz

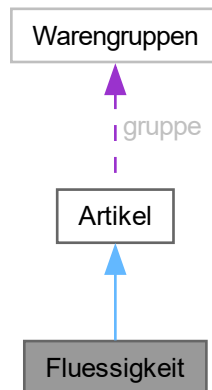
Die Klasse "Fluessigkeit" erbt von der Klasse "Artikel" und spezialisiert sie fuer Fluessigkeits-Artikel.

```
#include <lager.hh>
```

Klassendiagramm für Fluessigkeit:



Zusammengehörigkeiten von Flüssigkeit:



Öffentliche Methoden

- **Flüssigkeit** (**Artikel** produkt)
Konstruktor für die Klasse "Flüssigkeit" unter Verwendung eines bereits existierenden Artikels.
- **Flüssigkeit** (string name, string num, double vol, preis np, unsigned int bestand=1)
Konstruktor fuer die Klasse "Flüssigkeit".
- double **getVolume** () const
Gibt das Volumen des Flüssigkeits-Artikels zurueck.
- void **setVerkaufpreis** (preis vp)
- void **setVolume** (double vol)
Setzt das Volumen des Flüssigkeits-Artikels.

Öffentliche Methoden geerbt von **Artikel**

- **Artikel** ()
Standardkonstruktor für die Klasse "Artikel".
- **Artikel** (string name, string num, unsigned int bestand, masseinheit einheit, preis vp, preis np)
Konstruktor fuer die Klasse "Artikel".
- **~Artikel** ()
Destruktor fuer die Klasse "Artikel".
- string **getName** () const
Gibt den Namen des Artikels zurueck.
- string **getArtikelnummer** () const
Gibt die Artikelnummer des Artikels zurueck.
- unsigned int **getLagerabstand** () const
Gibt den Lagerbestand des Artikels zurueck.
- masseinheit **getMasseinheit** () const
Gibt die Masseinheit des Artikels als Wert aus der Enumeration zurück.
- string **getStrMasseinheit** () const

- *Gibt die Masseinheit des Artikels als Zeichenkette zurück.*
- preis `getVerkaufspreis ()` const
Gibt den Verkaufspreis des Artikels zurueck.
- preis `getNormpreis ()` const
Gibt den Normalpreis des Artikels zurueck.
- string `getGruppe ()` const
Gibt die Warengruppe des Artikels zurueck.
- void `setName (string name)`
Setzt den Namen des Artikels.
- void `setArtikelnummer (string num)`
Setzt die Artikelnummer des Artikels.
- void `setLagerbestand (unsigned int bestand)`
Setzt den Lagerbestand des Artikels.
- void `setMasseinheit (masseinheit einheit)`
Setzt die Masseinheit des Artikels.
- void `setVerkaufspreis (preis vp)`
Setzt den Verkaufspreis des Artikels.
- void `setNormpreis (preis np)`
Setzt den Normalpreis des Artikels.
- ostream & `print (ostream &ostream)`
Gibt die Artikelinformationen aus.

Private Attribute

- double **volume**

Weitere Geerbte Elemente

Öffentliche, statische Methoden geerbt von Artikel

- static void `setGruppe (Warengruppen g)`
Setzt die Warengruppe fuer Artikel.

Statische öffentliche Attribute geerbt von Artikel

- static `Warengruppen gruppe`
Statische Warengruppen-Instanz, die fuer alle Artikel gemeinsam genutzt wird.

Geschützte Attribute geerbt von Artikel

- string **artikelname**
- string **artikelnummer**
- unsigned int **lagerbestand**
- masseinheit **einheit**
- preis **verkaufspreis**
- preis **normpreis**

2.2.1 Ausführliche Beschreibung

Die Klasse "Fluessigkeit" erbt von der Klasse "Artikel" und spezialisiert sie fuer Fluessigkeits-Artikel.

Die Klasse "Fluessigkeit" erweitert die Basisfunktionalität der Klasse "Artikel" um die Berücksichtigung des Volumens.

2.2.2 Beschreibung der Konstruktoren und Destruktoren

2.2.2.1 Fluessigkeit() [1/2]

```
Fluessigkeit::Fluessigkeit (
    Artikel produkt )
```

Konstruktor für die Klasse "Fluessigkeit" unter Verwendung eines bereits existierenden Artikels.

Parameter

<i>produkt</i>	Der Artikel, aus dem ein Fluessigkeits-Artikel erstellt wird.
----------------	---

2.2.2.2 Fluessigkeit() [2/2]

```
Fluessigkeit::Fluessigkeit (
    string name,
    string num,
    double vol,
    preis np,
    unsigned int bestand = 1 )
```

Konstruktor fuer die Klasse "Fluessigkeit".

Parameter

<i>name</i>	Der Name des Fluessigkeits-Artikels.
<i>num</i>	Die Artikelnummer des Fluessigkeits-Artikels.
<i>vol</i>	Das Volumen des Fluessigkeits-Artikels.
<i>np</i>	Der Normalpreis des Fluessigkeits-Artikels.
<i>bestand</i>	Der Lagerbestand des Fluessigkeits-Artikels (Standardwert: 1).

2.2.3 Dokumentation der Elementfunktionen

2.2.3.1 getVolume()

```
double Fluessigkeit::getVolume ( ) const
```

Gibt das Volumen des Fluessigkeits-Artikels zurueck.

Rückgabe

Das Volumen des Artikels.

2.2.3.2 setVerkaufspreis()

```
void Fluessigkeit::setVerkaufspreis (
    preis vp )
```

Parameter

<i>vp</i>	Der Verkaufspreis, der gesetzt werden soll.
-----------	---

2.2.3.3 setVolume()

```
void Fluessigkeit::setVolume (
    double vol )
```

Setzt das Volumen des Fluessigkeits-Artikels.

Parameter

<i>vol</i>	Das neue Volumen.
------------	-------------------

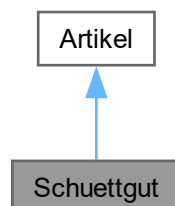
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [lager.hh](#)
- [lager.cc](#)

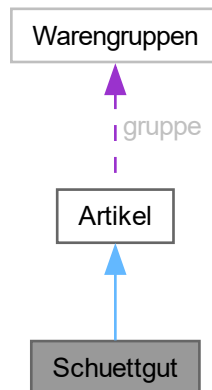
2.3 Schuettgut Klassenreferenz

```
#include <lager.hh>
```

Klassendiagramm für Schuettgut:



Zusammengehörigkeiten von Schuettgut:



Öffentliche Methoden

- **Schuettgut** (**Artikel** produkt)
Konstruktor für die Klasse "Schuettgut" unter Verwendung eines bereits existierenden Artikels.
- **Schuettgut** (string name, string num, double groesse, preis np, unsigned int bestand=1)
Konstruktor fuer die Klasse "Schuettgut".
- double **getLosgroesse** () const
Gibt die Losgroesse des Schuettgut-Artikels zurueck.
- void **setVerkaufpreis** (preis vp)
Setzt den Verkaufspreis des Schuettgut-Artikels basierend auf der Losgroesse.
- void **setLosgroesse** (double groesse)
Setzt die Losgroesse des Schuettgut-Artikels.

Öffentliche Methoden geerbt von **Artikel**

- **Artikel** ()
Standardkonstruktor für die Klasse "Artikel".
- **Artikel** (string name, string num, unsigned int bestand, masseinheit einheit, preis vp, preis np)
Konstruktor fuer die Klasse "Artikel".
- **~Artikel** ()
Destruktor fuer die Klasse "Artikel".
- string **getName** () const
Gibt den Namen des Artikels zurueck.
- string **getArtikelnummer** () const
Gibt die Artikelnummer des Artikels zurueck.
- unsigned int **getLagerabstand** () const
Gibt den Lagerbestand des Artikels zurueck.
- masseinheit **getMasseinheit** () const
Gibt die Masseinheit des Artikels als Wert aus der Enumeration zurück.

- string `getStrMasseinheit` () const
Gibt die Masseinheit des Artikels als Zeichenkette zurück.
- preis `getVerkaufspreis` () const
Gibt den Verkaufspreis des Artikels zurueck.
- preis `getNormpreis` () const
Gibt den Normalpreis des Artikels zurueck.
- string `getGruppe` () const
Gibt die Warengruppe des Artikels zurueck.
- void `setName` (string name)
Setzt den Namen des Artikels.
- void `setArtikelnummer` (string num)
Setzt die Artikelnummer des Artikels.
- void `setLagerbestand` (unsigned int bestand)
Setzt den Lagerbestand des Artikels.
- void `setMasseinheit` (masseinheit einheit)
Setzt die Masseinheit des Artikels.
- void `setVerkaufspreis` (preis vp)
Setzt den Verkaufspreis des Artikels.
- void `setNormpreis` (preis np)
Setzt den Normalpreis des Artikels.
- ostream & `print` (ostream &ostream)
Gibt die Artikelinformationen aus.

Private Attribute

- double `losgroesse`

Weitere Geerbte Elemente

Öffentliche, statische Methoden geerbt von `Artikel`

- static void `setGruppe` (`Warengruppen` g)
Setzt die Warengruppe fuer `Artikel`.

Statische öffentliche Attribute geerbt von `Artikel`

- static `Warengruppen` `gruppe`
Statische Warengruppen-Instanz, die fuer alle `Artikel` gemeinsam genutzt wird.

Geschützte Attribute geerbt von `Artikel`

- string `artikelname`
- string `artikelnummer`
- unsigned int `lagerbestand`
- masseinheit `einheit`
- preis `verkaufspreis`
- preis `normpreis`

2.3.1 Ausführliche Beschreibung

Die Klasse "Schuettgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer Schuettgut-Artikel. Die Klasse "Schuettgut" erweitert die Basisfunktionalität der Klasse "Artikel" um die Berücksichtigung der Losgröße.

2.3.2 Beschreibung der Konstruktoren und Destruktoren

2.3.2.1 Schuettgut() [1/2]

```
Schuettgut::Schuettgut (
    Artikel produkt )
```

Konstruktor für die Klasse "Schuettgut" unter Verwendung eines bereits existierenden Artikels.

Parameter

<i>produkt</i>	Der Artikel, aus dem ein Schuettgut-Artikel erstellt wird.
----------------	--

2.3.2.2 Schuettgut() [2/2]

```
Schuettgut::Schuettgut (
    string name,
    string num,
    double groesse,
    preis np,
    unsigned int bestand = 1 )
```

Konstruktor fuer die Klasse "Schuettgut".

Parameter

<i>name</i>	Der Name des Schuettgut-Artikels.
<i>num</i>	Die Artikelnummer des Schuettgut-Artikels.
<i>groesse</i>	Die Losgroesse des Schuettgut-Artikels.
<i>np</i>	Der Normalpreis des Schuettgut-Artikels.
<i>bestand</i>	Der Lagerbestand des Schuettgut-Artikels (Standardwert: 1).

2.3.3 Dokumentation der Elementfunktionen

2.3.3.1 getLosgroesse()

```
double Schuettgut::getLosgroesse ( ) const
```

Gibt die Losgroesse des Schuettgut-Artikels zurueck.

Rückgabe

Die Losgroesse des Artikels.

2.3.3.2 setLosgroesse()

```
void Schuetttgut::setLosgroesse (
    double groesse )
```

Setzt die Losgroesse des Schuetttgut-Artikels.

Parameter

<i>groesse</i>	Die neue Losgroesse.
----------------	----------------------

2.3.3.3 setVerkaufspreis()

```
void Schuetttgut::setVerkaufspreis (
    preis vp )
```

Setzt den Verkaufspreis des Schuetttgut-Artikels basierend auf der Losgroesse.

Parameter

<i>vp</i>	Der Verkaufspreis, der gesetzt werden soll.
-----------	---

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

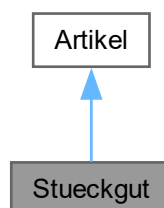
- [lager.hh](#)
- [lager.cc](#)

2.4 Stueckgut Klassenreferenz

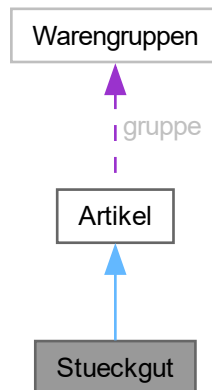
Die Klasse "Stueckgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer Stueckgut-Artikel.

```
#include <lager.hh>
```

Klassendiagramm für Stueckgut:



Zusammengehörigkeiten von Stueckgut:



Öffentliche Methoden

- **Stueckgut** (**Artikel** produkt)
Konstruktor fuer die Klasse "Stueckgut".
- **Stueckgut** (string name, string num, preis vp, unsigned int bestand=1)

Öffentliche Methoden geerbt von **Artikel**

- **Artikel** ()
Standardkonstruktor fuer die Klasse "Artikel".
- **Artikel** (string name, string num, unsigned int bestand, masseinheit einheit, preis vp, preis np)
Konstruktor fuer die Klasse "Artikel".
- **~Artikel** ()
Destruktor fuer die Klasse "Artikel".
- string **getName** () const
Gibt den Namen des Artikels zurueck.
- string **getArtikelnummer** () const
Gibt die Artikelnummer des Artikels zurueck.
- unsigned int **getLagerabstand** () const
Gibt den Lagerbestand des Artikels zurueck.
- masseinheit **getMasseinheit** () const
Gibt die Masseinheit des Artikels als Wert aus der Enumeration zurück.
- string **getStrMasseinheit** () const
Gibt die Masseinheit des Artikels als Zeichenkette zurück.
- preis **getVerkaufspreis** () const
Gibt den Verkaufspreis des Artikels zurueck.
- preis **getNormpreis** () const
Gibt den Normalpreis des Artikels zurueck.
- string **getGruppe** () const

- *Gibt die Warengruppe des Artikels zurueck.*
- void `setName` (string name)
Setzt den Namen des Artikels.
- void `setArtikelnummer` (string num)
Setzt die Artikelnummer des Artikels.
- void `setLagerbestand` (unsigned int bestand)
Setzt den Lagerbestand des Artikels.
- void `setMasseinheit` (masseinheit einheit)
Setzt die Masseinheit des Artikels.
- void `setVerkaufspreis` (preis vp)
Setzt den Verkaufspreis des Artikels.
- void `setNormpreis` (preis np)
Setzt den Normalpreis des Artikels.
- ostream & `print` (ostream &ostream)
Gibt die Artikelinformationen aus.

Weitere Geerbte Elemente

Öffentliche, statische Methoden geerbt von `Artikel`

- static void `setGruppe` (`Warengruppen` g)
Setzt die Warengruppe fuer `Artikel`.

Statische öffentliche Attribute geerbt von `Artikel`

- static `Warengruppen` `gruppe`
Statische Warengruppen-Instanz, die fuer alle `Artikel` gemeinsam genutzt wird.

Geschützte Attribute geerbt von `Artikel`

- string `artikelname`
- string `artikelnummer`
- unsigned int `lagerbestand`
- masseinheit `einheit`
- preis `verkaufspreis`
- preis `normpreis`

2.4.1 Ausführliche Beschreibung

Die Klasse "Stueckgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer Stueckgut-Artikel.

2.4.2 Beschreibung der Konstruktoren und Destruktoren

2.4.2.1 Stueckgut()

```
Stueckgut::Stueckgut (
    Artikel produkt )
```

Konstruktor fuer die Klasse "Stueckgut".

Parameter

<i>name</i>	Der Name des Stueckgut-Artikels.
<i>num</i>	Die Artikelnummer des Stueckgut-Artikels.
<i>vp</i>	Der Verkaufspreis des Stueckgut-Artikels.
<i>bestand</i>	Der Lagerbestand des Stueckgut-Artikels (Standardwert: 1).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [lager.hh](#)
- [lager.cc](#)

2.5 Warengruppen Klassenreferenz

Öffentliche Methoden

- [Warengruppen](#) ()
Konstruktor fuer die Klasse "Warengruppen".
- void **defaultList** ()
Setzt eine Standard-Warengruppenliste.
- string [getGruppe](#) (string code)
Gibt den Namen der Warengruppe fuer einen gegebenen Code zurueck.
- void [addGruppe](#) (string code, string name)
Fuegt eine neue Warengruppe hinzu.
- void [delGruppe](#) (string code)
Loescht eine Warengruppe anhand ihres Codes.
- void [changeGruppe](#) (string code, string name)
Aendert den Namen einer vorhandenen Warengruppe.
- void **clear** ()
Loescht alle [Warengruppen](#) und setzt sie zurck.

Private Attribute

- map< string, string > **mapGruppe**
- map< string, string >::iterator **iter**

2.5.1 Beschreibung der Konstruktoren und Destruktoren

2.5.1.1 Warengruppen()

```
Warengruppen::Warengruppen ( )
```

Konstruktor fuer die Klasse "Warengruppen".

Dieser Konstruktor initialisiert eine leere Warengruppenliste.

2.5.2 Dokumentation der Elementfunktionen

2.5.2.1 addGruppe()

```
void Warengruppen::addGruppe (
    string code,
    string name )
```

Fuegt eine neue Warengruppe hinzu.

Parameter

<i>code</i>	Der Warengruppencode.
<i>name</i>	Der Name der Warengruppe.

2.5.2.2 changeGruppe()

```
void Warengruppen::changeGruppe (
    string code,
    string name )
```

Ändert den Namen einer vorhandenen Warengruppe.

Parameter

<i>code</i>	Der Warengruppencode.
<i>name</i>	Der neue Name der Warengruppe.

2.5.2.3 delGruppe()

```
void Warengruppen::delGruppe (
    string code )
```

Löscht eine Warengruppe anhand ihres Codes.

Parameter

<i>code</i>	Der Warengruppencode.
-------------	-----------------------

2.5.2.4 getGruppe()

```
string Warengruppen::getGruppe (
    string code )
```

Gibt den Namen der Warengruppe fuer einen gegebenen Code zurueck.

Parameter

<i>code</i>	Der Warengruppencode.
-------------	-----------------------

Rückgabe

Der Name der Warengruppe oder der Code, falls keine Warengruppe gefunden wurde.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [lager.hh](#)
- [lager.cc](#)

Kapitel 3

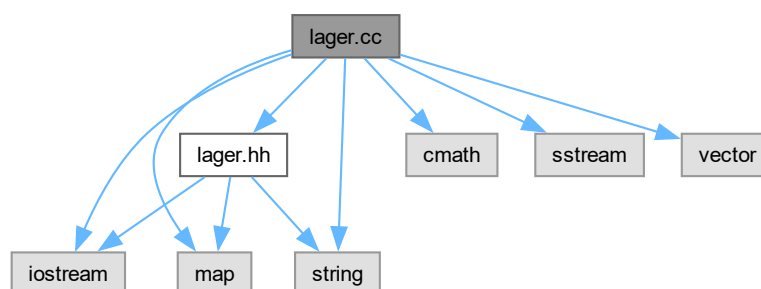
Datei-Dokumentation

3.1 lager.cc-Dateireferenz

Implementierung der Lagerverwaltungsfunktionen.

```
#include "lager.hh"  
#include <cmath>  
#include <iostream>  
#include <map>  
#include <sstream>  
#include <string>  
#include <vector>
```

Include-Abhängigkeitsdiagramm für lager.cc:



Funktionen

- `std::ostream & operator<< (std::ostream &os, Artikel produkt)`
- `void operator>> (istream &is, Artikel &produkt)`

Überladen des Eingabeoperators für die Artikelklasse.

3.1.1 Ausführliche Beschreibung

Implementierung der Lagerverwaltungsfunktionen.

Autoren

Yaman Alsaady, Oliver Schmidt

Version

0.2

Datum

2023-10-19

Dies ist die Implementierung der Funktionen fuer die Lagerverwaltung, einschliesslich der Warengruppenverwaltung und der Artikelklassen.

Copyright

Copyright (c) 2023

3.1.2 Dokumentation der Funktionen

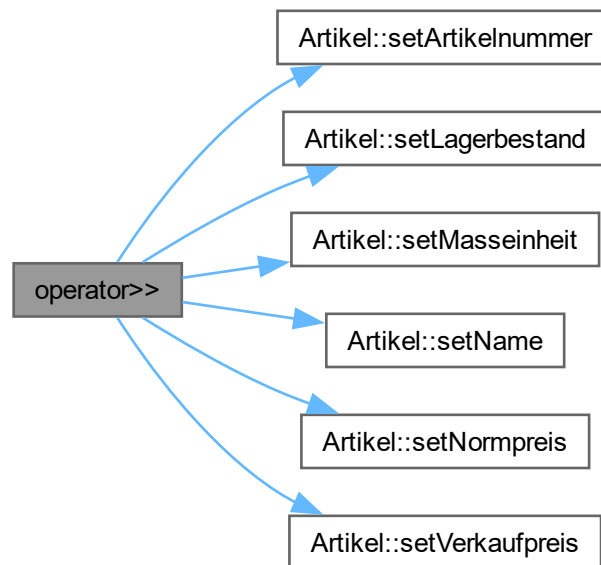
3.1.2.1 `operator>>()`

```
void operator>> (
    istream & is,
    Artikel & produkt )
```

Überladen des Eingabeoperators für die Artikelklasse.

Diese Funktion ermöglicht das Einlesen von Artikelinformationen mit dem Eingabeoperator '>>'.
</p>
</div>
<div data-bbox="112 677 182 691" data-label="Section-Header">
<h4>Parameter</h4>
</div>
<div data-bbox="115 696 691 732" data-label="Table">
<table>
<tr>
<td>is</td>
<td>Die Eingabestromreferenz, aus der die Informationen eingelesen werden.</td>
</tr>
<tr>
<td>produkt</td>
<td>Der Artikel, in den die Informationen eingelesen werden sollen.</td>
</tr>
</table>
</div>
</div>

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

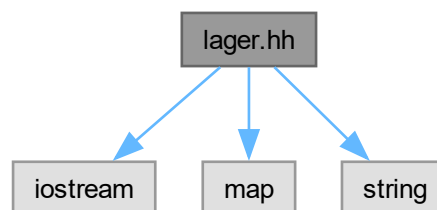


3.2 lager.hh-Dateireferenz

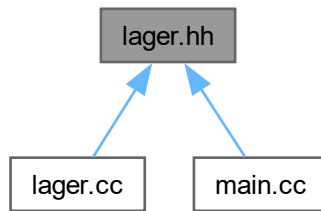
Definitionen der Lagerverwaltungsfunktionen.

```
#include <iostream>
#include <map>
#include <string>
```

Include-Abhängigkeitsdiagramm für `lager.hh`:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class [Warengruppen](#)
- class [Artikel](#)

Die Klasse "Artikel" repräsentiert einen [Artikel](#) mit verschiedenen Eigenschaften.

- class [Stueckgut](#)

Die Klasse "Stueckgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer Stueckgut-Artikel.

- class [Schuettgut](#)
- class [Fluessigkeit](#)

Die Klasse "Fluessigkeit" erbt von der Klasse "Artikel" und spezialisiert sie fuer Fluessigkeits-Artikel.

Typdefinitionen

- typedef double **preis**

Aufzählungen

- enum **masseinheit** { **stk** , **kg** , **l** }

Funktionen

- ostream & [operator<<](#) (ostream &os, [Artikel](#) produkt)

Überladen des Ausgabeoperators für die Artikelklasse.

- void [operator>>](#) (istream &is, [Artikel](#) &produkt)

Überladen des Eingabeoperators für die Artikelklasse.

3.2.1 Ausführliche Beschreibung

Definitionen der Lagerverwaltungsfunktionen.

Autoren

Yaman Alsaady, Oliver Schmidt

Version

0.2

Datum

2023-10-19

Dieses Header-Datei enthaelt die Definitionen von Klassen und Funktionen zur Verwaltung von Artikeln und [Warengruppen](#) in einem C++-Programm.

Copyright

Copyright (c) 2023

3.2.2 Dokumentation der Funktionen

3.2.2.1 operator<<()

```
ostream & operator<< (
    ostream & os,
    Artikel produkt )
```

Überladen des Ausgabeoperators für die Artikelklasse.

Diese Funktion ermöglicht das Ausgeben eines Artikels mit dem Ausgabeoperator '<<'.

Parameter

<i>os</i>	Die Ausgabestromreferenz, in die die Informationen geschrieben werden.
<i>produkt</i>	Der Artikel , der ausgegeben werden soll.

Rückgabe

Die Ausgabestromreferenz, in die die Informationen geschrieben wurden.

3.2.2.2 operator>>()

```
void operator>> (
    istream & is,
    Artikel & produkt )
```

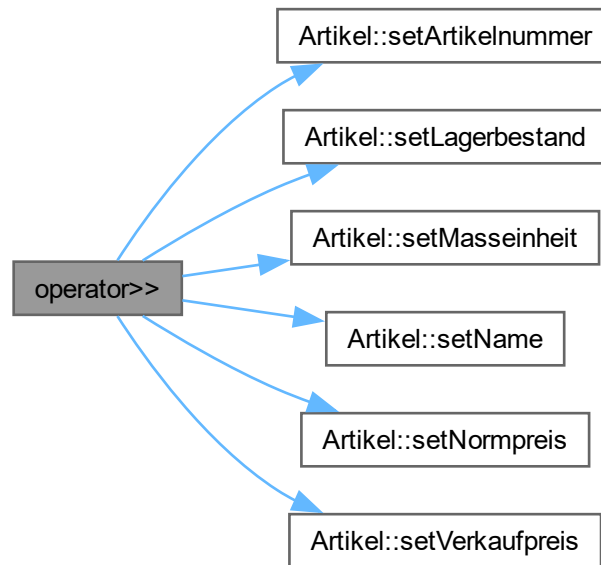
Überladen des Eingabeoperators für die Artikelklasse.

Diese Funktion ermöglicht das Einlesen von Artikelinformationen mit dem Eingabeoperator '>>'.

Parameter

<i>is</i>	Die Eingabestromreferenz, aus der die Informationen eingelesen werden.
<i>produkt</i>	Der Artikel , in den die Informationen eingelesen werden sollen.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



3.3 lager.hh

[gehe zur Dokumentation dieser Datei](#)

```

00001
00016 #ifndef LAGER_HH
00017 #define LAGER_HH
00018
00019 #include <iostream>
00020 #include <map>
00021 #include <string>
00022
00023 using namespace std;
00024 enum masseinheit { stk, kg, l };
00025 typedef double preis;
00026
00027 class Warengruppen {
00028 private:
00029     map<string, string> mapGruppe;
00030     map<string, string>::iterator iter;
00031
00032 public:
00033     Warengruppen();
00034
00035     void defaultList();
  
```

```

00044
00052     string getGruppe(string code);
00053
00060     void addGruppe(string code, string name);
00061
00067     void delGruppe(string code);
00068
00075     void changeGruppe(string code, string name);
00076
00080     void clear();
00081 };
00082
00087 class Artikel {
00088 protected:
00089     string artikelname;
00090     string artikelnummer;
00091     unsigned int lagerbestand;
00092     masseinheit einheit;
00093     preis verkaufpreis;
00094     preis normpreis;
00095
00096 public:
00100     Artikel();
00101
00112     Artikel(string name, string num, unsigned int bestand, masseinheit einheit,
00113             preis vp, preis np);
00114
00115     // Getter-Funktionen
00116
00120     ~Artikel();
00121
00126     static Warengruppen gruppe;
00127
00133     static void setGruppe(Warengruppen g);
00134
00140     string getName() const;
00141
00147     string getArtikelnummer() const;
00148
00154     unsigned int getLagerabstand() const;
00155
00163     masseinheit getMasseinheit() const;
00164
00170     string getStrMasseinheit() const;
00171
00177     preis getVerkaufpreis() const;
00178
00184     preis getNormpreis() const;
00185
00192     string getGruppe() const;
00193
00194     // Setter-Funktionen
00195
00201     void setName(string name);
00202
00208     void setArtikelnummer(string num);
00209
00215     void setLagerbestand(unsigned int bestand);
00216
00222     void setMasseinheit(masseinheit einheit);
00223
00229     void setVerkaufpreis(preis vp);
00230
00236     void setNormpreis(preis np);
00237
00250     ostream &print(ostream &ostream);
00251 };
00264 ostream &operator<<(ostream &os, Artikel produkt);
00265
00277 void operator>>(istream &is, Artikel &produkt);
00278
00283 class Stueckgut : public Artikel {
00284 private:
00285 public:
00294     Stueckgut(Artikel produkt);
00295     Stueckgut(string name, string num, preis vp, unsigned int bestand = 1);
00296 };
00297
00304 class Schuettgut : public Artikel {
00305 private:
00306     double losgroesse;
00307
00308 public:
00315     Schuettgut(Artikel produkt);
00316
00326     Schuettgut(string name, string num, double groesse, preis np,
00327                 unsigned int bestand = 1);

```

```

00328
00334     double getLosgroesse() const;
00335
00342     void setVerkaufpreis(preis vp);
00343
00349     void setLosgroesse(double groesse);
00350 };
00351
00359 class Fluessigkeit : public Artikel {
00360 private:
00361     double volume;
00362
00363 public:
00371     Fluessigkeit(Artikel produkt);
00372
00383     Fluessigkeit(string name, string num, double vol, preis np,
00384                   unsigned int bestand = 1);
00385
00391     double getVolume() const;
00392
00397     void setVerkaufpreis(preis vp);
00398
00404     void setVolume(double vol);
00405 };
00406
00407 #endif // !LAGER_HH

```

3.4 main.cc-Dateireferenz

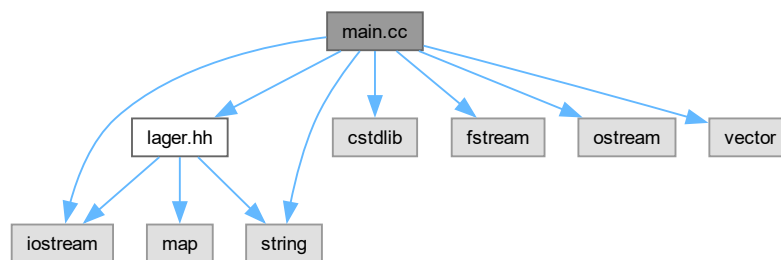
Implementierung des Lagerverwaltungssystems.

```

#include "lager.hh"
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <ostream>
#include <string>
#include <vector>

```

Include-Abhängigkeitsdiagramm für main.cc:



Funktionen

- void **read** (string filename)
Liest Artikelinformationen aus einer Datei und gruppiert sie nach Artikeltyp.
- void **readWrite** (string **read**, string write)
Liest Artikelinformationen aus einer Eingabedatei und schreibt sie in eine Ausgabedatei.
- int **main** (int argc, char *argv[])
*Lesen und Schreiben von **Artikel** aus Dateien.*

3.4.1 Ausführliche Beschreibung

Implementierung des Lagerverwaltungssystems.

Autoren

Yaman Alsaady, Oliver Schmidt

Version

0.2

Datum

2023-10-19

Dieses Programm implementiert ein Lagerverwaltungssystem, das es ermöglicht, Artikelinformationen aus Dateien zu lesen und sie nach Artikeltyp zu gruppieren. Die Informationen werden entweder auf der Konsole ausgegeben oder in eine Ausgabedatei geschrieben. Der Name der Ausgabedatei wird als Befehlszeilenargument übergeben.

Copyright

Copyright (c) 2023

3.4.2 Dokumentation der Funktionen

3.4.2.1 main()

```
int main (
    int argc,
    char * argv[] )
```

Lesen und Schreiben von [Artikel](#) aus Dateien.

Diese Funktion ist der Einstiegspunkt des Programms und ermöglicht das Lesen und Schreiben von Artikelinformationen aus Dateien. Die Dateinamen werden als Befehlszeilenargumente übergeben. Es kann maximal eine Datei zum Schreiben angegeben werden, während mehrere Dateien zum Lesen zugelassen sind.

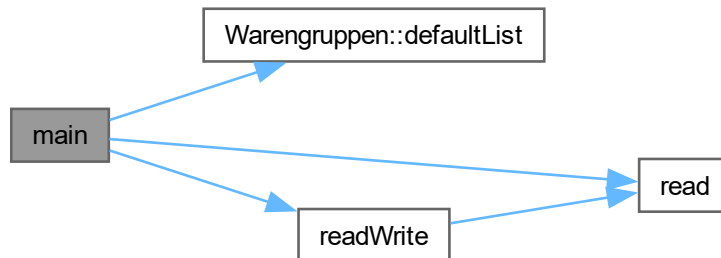
Parameter

<i>argc</i>	Die Anzahl der Befehlszeilenargumente.
<i>argv</i>	Ein Array von Zeichenketten, das die Befehlszeilenargumente enthält.

Rückgabe

Eine Ganzzahl, die den Programmstatus zurückgibt (0 für Erfolg, andere Werte für Fehler).

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

**3.4.2.2 read()**

```
void read (
    string filename )
```

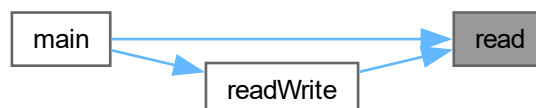
Liest Artikelinformationen aus einer Datei und gruppiert sie nach Artikeltyp.

Diese Funktion liest Artikelinformationen aus einer angegebenen Datei und gruppiert sie in die entsprechenden Vektoren je nach Artikeltyp ([Stueckgut](#), [Schuettgut](#), [Fluessigkeit](#)). Anschließend gibt sie die Informationen der Artikeltypen nacheinander aus.

Parameter

<i>filename</i>	Der Dateiname der Eingabedatei, aus der die Informationen gelesen werden.
-----------------	---

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



3.4.2.3 readWrite()

```
void readWrite (
    string read,
    string write )
```

Liest Artikelinformationen aus einer Eingabedatei und schreibt sie in eine Ausgabedatei.

Diese Funktion liest Artikelinformationen aus einer angegebenen Eingabedatei und gruppiert sie in die entsprechenden Vektoren je nach Artikeltyp ([Stueckgut](#), [Schuettgut](#), [Fluessigkeit](#)). Anschließend werden die Informationen in die angegebene Ausgabedatei im Anhängemodus geschrieben.

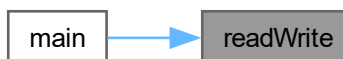
Parameter

<i>read</i>	Der Dateiname der Eingabedatei, aus der die Informationen gelesen werden.
<i>write</i>	Der Dateiname der Ausgabedatei, in die die Informationen angehängt werden.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



Index

addGruppe
 Warengruppen, 34
Artikel, 15
 Artikel, 17
 getArtikelnummer, 17
 getGruppe, 17
 getLagerabstand, 18
 getMasseinheit, 18
 getName, 18
 getNormpreis, 18
 getStrMasseinheit, 19
 getVerkaufspreis, 19
 print, 19
 setArtikelnummer, 20
 setGruppe, 20
 setLagerbestand, 21
 setMasseinheit, 21
 setName, 22
 setNormpreis, 22
 setVerkaufspreis, 22

changeGruppe
 Warengruppen, 35

delGruppe
 Warengruppen, 35

Fluessigkeit, 23
 Fluessigkeit, 26
 getVolume, 26
 setVerkaufspreis, 27
 setVolume, 27

getArtikelnummer
 Artikel, 17
getGruppe
 Artikel, 17
 Warengruppen, 35
getLagerabstand
 Artikel, 18
getLosgroesse
 Schuettgut, 30
getMasseinheit
 Artikel, 18
getName
 Artikel, 18
getNormpreis
 Artikel, 18
getStrMasseinheit
 Artikel, 19

getVerkaufspreis
 Artikel, 19
getVolume
 Fluessigkeit, 26

lager.cc, 37
 operator>>, 38
lager.hh, 39
 operator<<, 41
 operator>>, 41

main
 main.cc, 45
main.cc, 44
 main, 45
 read, 46
 readWrite, 46

operator<<
 lager.hh, 41
operator>>
 lager.cc, 38
 lager.hh, 41

print
 Artikel, 19

read
 main.cc, 46
readWrite
 main.cc, 46

Schuettgut, 27
 getLosgroesse, 30
 Schuettgut, 30
 setLosgroesse, 30
 setVerkaufspreis, 31
setArtikelnummer
 Artikel, 20
setGruppe
 Artikel, 20
setLagerbestand
 Artikel, 21
setLosgroesse
 Schuettgut, 30
setMasseinheit
 Artikel, 21
setName
 Artikel, 22
setNormpreis
 Artikel, 22

- setVerkaufspreis
 - Artikel, [22](#)
 - Fluessigkeit, [27](#)
 - Schuettgut, [31](#)
- setVolume
 - Fluessigkeit, [27](#)
- Stueckgut, [31](#)
 - Stueckgut, [33](#)
- Warengruppen, [34](#)
 - addGruppe, [34](#)
 - changeGruppe, [35](#)
 - delGruppe, [35](#)
 - getGruppe, [35](#)
 - Warengruppen, [34](#)