



University of Applied Sciences

HOCHSCHULE
EMDEN•LEER

Fachbereich Technik
Abteilung Elektrotechnik und Informatik

PROGRAMMIEREN III AUFGABE 1

ENTWICKLUNG EINES WARENWIRTSCHAFTSSYSTEMS: MODELLIERUNG UND IMPLEMENTIERUNG VON ARTIKELKLASSEN IN C++

Gruppe A1
Studiengang Elektrotechnik
Vorgelegt von

Yaman Alsaady
Oliver Schmidt

Matr. Nr. 7023554
Matr. Nr. 7023462

Emden, 6. Oktober 2023

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

	I
1 Fragen	1
2 Quellencode	2
2.1 Datei 'Lager.hh'	2
2.2 Datei 'Lager.cc'	7
2.3 Datei 'main.cc'	9
2.4 Ausgabe-Test	11
3 Klassen-Dokumentation	12
3.1 Artikel Klassenreferenz	12
3.1.1 Ausführliche Beschreibung	14
3.1.2 Beschreibung der Konstruktoren und Destruktoren	14
3.1.2.1 Artikel()	14
3.1.3 Dokumentation der Elementfunktionen	14
3.1.3.1 getArtikelnummer()	14
3.1.3.2 getGruppe()	15
3.1.3.3 getLagerabstand()	15
3.1.3.4 getMasseinheit()	16
3.1.3.5 getName()	16
3.1.3.6 getNormpreis()	17
3.1.3.7 getVerkaufspreis()	17
3.1.3.8 setArtikelnummer()	18
3.1.3.9 setGruppe()	18
3.1.3.10 setLagerbestand()	18
3.1.3.11 setMasseinheit()	19
3.1.3.12 setName()	19
3.1.3.13 setNormpreis()	19
3.1.3.14 setVerkaufspreis()	19
3.2 Flüssigkeit Klassenreferenz	20
3.2.1 Ausführliche Beschreibung	22
3.2.2 Beschreibung der Konstruktoren und Destruktoren	22
3.2.2.1 Flüssigkeit()	22
3.2.3 Dokumentation der Elementfunktionen	23
3.2.3.1 getVolume()	23
3.2.3.2 setVerkaufspreis()	23
3.2.3.3 setVolume()	23
3.3 Schuettgut Klassenreferenz	23
3.3.1 Ausführliche Beschreibung	26
3.3.2 Beschreibung der Konstruktoren und Destruktoren	26
3.3.2.1 Schuettgut()	26
3.3.3 Dokumentation der Elementfunktionen	26
3.3.3.1 getLosgroesse()	26
3.3.3.2 setLosgroesse()	27

3.3.3.3 setVerkaufpreis()	27
3.4 Stueckgut Klassenreferenz	27
3.4.1 Ausführliche Beschreibung	29
3.4.2 Beschreibung der Konstruktoren und Destruktoren	30
3.4.2.1 Stueckgut()	30
3.5 Warengruppen Klassenreferenz	30
3.5.1 Beschreibung der Konstruktoren und Destruktoren	31
3.5.1.1 Warengruppen()	31
3.5.2 Dokumentation der Elementfunktionen	31
3.5.2.1 addGruppe()	31
3.5.2.2 changeGruppe()	31
3.5.2.3 delGruppe()	31
3.5.2.4 getGruppe()	32
4 Datei-Dokumentation	33
4.1 lager.cc-Dateireferenz	33
4.1.1 Ausführliche Beschreibung	33
4.2 lager.hh-Dateireferenz	34
4.2.1 Ausführliche Beschreibung	35
4.3 lager.hh	36
4.4 main.cc-Dateireferenz	37
4.4.1 Ausführliche Beschreibung	38
4.4.2 Dokumentation der Funktionen	38
4.4.2.1 printInfo() [1/2]	38
4.4.2.2 printInfo() [2/2]	39
Index	41

Kapitel 1

Fragen

Begründen Sie bitte auch kurz Ihre Entscheidungen, welche Eigenschaften öffentlich zugreifbar sein sollen (public) und welche nicht (abgestuft in protected und private; diese können sich im Lauf der Zeit natürlich noch ändern, wenn weitere Anforderungen definiert werden)

- **Warengruppen-Klasse:** Die mapGruppe und iter-Variablen sind als private deklariert, da sie für die interne Verwaltung der Warengruppen in der Klasse verwendet werden und nicht von außen zugreifbar sein sollten. Die öffentlichen Funktionen wie getGruppe, addGruppe, delGruppe, changeGruppe und clear sind als public deklariert, da sie von anderen Teilen des Codes verwendet werden können, um die Warengruppen zu verwalten.
- **Artikel-Klasse:** Die member-Variablen wie Artikelname, Artikelnummer, Lagerbestand, Einheit, Verkaufspreis und Normpreis sind als protected deklariert. Dies bedeutet, dass sie in den abgeleiteten Klassen (z.B., Stückgut, Schüttgut, Flüssigkeit) direkt zugegriffen werden können, jedoch nicht von außerhalb der Klassen. Die statische Variable "gruppe" ist als public deklariert, da sie von außen verwendet werden kann, um die Warengruppen für alle Artikel festzulegen. Die Getter- und Setter-Funktionen für die member-Variablen sind als public deklariert, um den Zugriff und die Änderung der Eigenschaften von Artikeln zu ermöglichen.
- **Stückgut, Schüttgut, und Flüssigkeits-Klassen:** Diese abgeleiteten Klassen erben von der Artikel-Klasse und erben somit die protected member-Variablen. Dies bedeutet, dass sie auf diese Variablen zugreifen können. Zusätzlich haben sie jeweils spezifische private member-Variablen, die für die jeweilige Art des Artikels relevant sind. Die Getter und Setter für die spezifischen Eigenschaften der abgeleiteten Klassen sind als public deklariert, um den Zugriff von außerhalb der Klassen zu ermöglichen.

Kapitel 2

Quellencode

2.1 Datei 'Lager.hh'

Listing 2.1 Die Header-Datei lager.hh

```
1  /**
2  * @file lager.hh
3  * @authors Yaman Alsaady, Oliver Schmidt
4  * @brief
5  * Dieses Header-Datei enthaelt die Definitionen von Klassen und
6  * Funktionen zur Verwaltung von Artikeln und Warengruppen in einem
7  * C++-Programm.
8  * @version 0.1
9  * @date 2023-10-04
10 *
11 * @copyright Copyright (c) 2023
12 *
13 */
14
15 #ifndef LAGER_HH
16 #define LAGER_HH
17
18 #include <iostream>
19 #include <map>
20 #include <string>
21
22 using namespace std;
23 enum masseinheit { stk, kg, l };
24 typedef double preis;
25
26 class Warengruppen {
27 private:
28     map<string, string> mapGruppe;
29     map<string, string>::iterator iter;
30
31 public:
32
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 gefunden wurde.
50  */
51  string getGruppe(string code);
52
53  /**
54  * @brief Fuegt eine neue Warengruppe hinzu.
55  *
56  * @param code Der Warengruppencode.
57  * @param name Der Name der Warengruppe.
58  */
59  void addGruppe(string code, string name);
60
61  /**
62  * @brief Loescht eine Warengruppe anhand ihres Codes.
63  *
64  * @param code Der Warengruppencode.
65  */
66  void delGruppe(string code);
67
68  /**
69  * @brief Aendert den Namen einer vorhandenen Warengruppe.
70  *
71  * @param code Der Warengruppencode.
72  * @param name Der neue Name der Warengruppe.
73  */
74  void changeGruppe(string code, string name);
75
76  /**
77  * @brief Loescht alle Warengruppen und setzt sie zurck.
78  */
79  void clear();
80 };
81
82 /**
83  * @brief Die Klasse "Artikel" repraesentiert einen Artikel mit verschiedenen Eigenschaften.
84  */
85 class Artikel {
86 protected:
87     string artikelname;
88     string artikelnummer;
89     unsigned int lagerbestand;
90     masseinheit einheit;
91     preis verkaufpreis;
92     preis normpreis;
93
94 public:
95
96     /**
97     * @brief Konstruktor fuer die Klasse "Artikel".
98     *
99     * @param name Der Name des Artikels.
100    * @param num Die Artikelnummer des Artikels.
101    * @param bestand Der Lagerbestand des Artikels.
102    * @param einheit Die Einheit des Artikels (stk, kg, l).
103    * @param vp Der Verkaufspreis des Artikels.
104    * @param np Der Normalpreis des Artikels.
105    */
106    Artikel(string name, string num, unsigned int bestand, masseinheit einheit,
107            preis vp, preis np);
108
109    // Getter-Funktionen
110
111    /**
112    * @brief Destruktor fuer die Klasse "Artikel".
113    */
114    ~Artikel();
115
116    /**
117    * @brief Statische Warengruppen-Instanz, die fuer alle Artikel gemeinsam genutzt wird.
118    */
119

```

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

```

192  */
193  void setArtikelnummer(string num);
194
195  /**
196   * @brief Setzt den Lagerbestand des Artikels.
197   *
198   * @param bestand Der neue Lagerbestand des Artikels.
199   */
200  void setLagerbestand(unsigned int bestand);
201
202  /**
203   * @brief Setzt die Masseinheit des Artikels.
204   *
205   * @param einheit Die neue Masseinheit des Artikels (stk, kg, l).
206   */
207  void setMasseinheit(masseinheit einheit);
208
209  /**
210   * @brief Setzt den Verkaufspreis des Artikels.
211   *
212   * @param vp Der neue Verkaufspreis des Artikels.
213   */
214  void setVerkaufspreis(preis vp);
215
216  /**
217   * @brief Setzt den Normalpreis des Artikels.
218   *
219   * @param np Der neue Normalpreis des Artikels.
220   */
221  void setNormpreis(preis np);
222
223  };
224
225  /**
226   * @brief Die Klasse "Stueckgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer
227   *         Stueckgut-Artikel.
228   */
229  class Stueckgut : public Artikel {
230  private:
231  public:
232      /**
233       * @brief Konstruktor fuer die Klasse "Stueckgut".
234       *
235       * @param name Der Name des Stueckgut-Artikels.
236       * @param num Die Artikelnummer des Stueckgut-Artikels.
237       * @param vp Der Verkaufspreis des Stueckgut-Artikels.
238       * @param bestand Der Lagerbestand des Stueckgut-Artikels (Standardwert: 1).
239       */
240      Stueckgut(string name, string num, preis vp, unsigned int bestand = 1);
241  };
242
243  /**
244   * Die Klasse "Schuettgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer Schuettgut
245   * -Artikel.
246   */
247  class Schuettgut : public Artikel {
248  private:
249      double losgroesse;
250
251  public:
252      /**
253       * @brief Konstruktor fuer die Klasse "Schuettgut".
254       *
255       * @param name Der Name des Schuettgut-Artikels.
256       * @param num Die Artikelnummer des Schuettgut-Artikels.
257       * @param groesse Die Losgroesse des Schuettgut-Artikels.
258       * @param np Der Normalpreis des Schuettgut-Artikels.
259       * @param bestand Der Lagerbestand des Schuettgut-Artikels (Standardwert: 1).
260       */
261      Schuettgut(string name, string num, double groesse, preis np,
262                  unsigned int bestand = 1);
263
264      /**
265       * @brief Gibt die Losgroesse des Schuettgut-Artikels zurueck.

```



```

263     *
264     * @return Die Losgroesse des Artikels.
265     */
266     double getLosgroesse() const;
267
268     /**
269     * @brief Setzt den Verkaufspreis des Schuetttgut-Artikels basierend auf der Losgroesse.
270     *
271     * @param vp Der Verkaufspreis, der gesetzt werden soll.
272     */
273     void setVerkaufspreis(preis vp);
274
275     /**
276     * @brief Setzt die Losgroesse des Schuetttgut-Artikels.
277     *
278     * @param groesse Die neue Losgroesse.
279     */
280     void setLosgroesse(double groesse);
281 };
282
283 /**
284 * @brief Die Klasse "Fluessigkeit" erbt von der Klasse "Artikel" und spezialisiert sie fuer
285     Fluessigkeits-Artikel.
286 */
287 class Fluessigkeit : public Artikel {
288 private:
289     double volume;
290 public:
291     /**
292     * @brief Konstruktor fuer die Klasse "Fluessigkeit".
293     *
294     * @param name Der Name des Fluessigkeits-Artikels.
295     * @param num Die Artikelnummer des Fluessigkeits-Artikels.
296     * @param vol Das Volumen des Fluessigkeits-Artikels.
297     * @param np Der Normalpreis des Fluessigkeits-Artikels.
298     * @param bestand Der Lagerbestand des Fluessigkeits-Artikels (Standardwert: 1).
299     */
300     Fluessigkeit(string name, string num, double vol, preis np,
301                 unsigned int bestand = 1);
302
303     /**
304     * @brief Gibt das Volumen des Fluessigkeits-Artikels zurueck.
305     *
306     * @return Das Volumen des Artikels.
307     */
308     double getVolume() const;
309
310     /**
311     *
312     * @param vp Der Verkaufspreis, der gesetzt werden soll.
313     */
314     void setVerkaufspreis(preis vp);
315
316     /**
317     * @brief Setzt das Volumen des Fluessigkeits-Artikels.
318     *
319     * @param vol Das neue Volumen.
320     */
321     void setVolume(double vol);
322 };
323 #endif // !LAGER_HH

```

2.2 Datei 'Lager.cc'

Listing 2.2 Die Header-Datei lager.cc

```

1  /**
2   * @file lager.cc
3   * @authors Yaman Alsaady, Oliver Schmidt
4   * @brief Implementierung der Lagerverwaltungsfunktionen.
5   * @version 0.1
6   * @date 2023-10-04
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 <string>
20
21 Warengruppen::Warengruppen() {}
22 void Warengruppen::defaultList() {
23     mapGruppe["1005"] = "Fahrrad";
24     mapGruppe["4000"] = "Gemuese";
25     mapGruppe["4106"] = "Gemuese";
26     mapGruppe["4370"] = "Kaffee";
27     mapGruppe["5500"] = "Bier";
28     mapGruppe["5031"] = "Milch";
29 }
30 string Warengruppen::getGruppe(string code) {
31     if (mapGruppe[code] != "")
32         return mapGruppe[code];
33     else {
34         return code;
35     }
36 }
37
38 void Warengruppen::addGruppe(string code, string name) {
39     mapGruppe.insert({code, name});
40 }
41 void Warengruppen::changeGruppe(string code, string name) {
42     mapGruppe.insert_or_assign(code, name);
43 }
44 void Warengruppen::delGruppe(string code) { mapGruppe.erase(code); }
45
46 void Warengruppen::clear() { mapGruppe.clear(); }
47
48 Warengruppen Artikel::gruppe;
49 Artikel::Artikel(string name, string num, unsigned int bestand,
50                 masseinheit einheit, preis vp, preis np)
51     : artikelname(name), artikelnummer(num), lagerbestand(bestand),
52       einheit(einheit), verkaufpreis(vp), normpreis(np) {}
53 Artikel::~Artikel() {}
54
55 void Artikel::setGruppe(Warengruppen g) { gruppe = g; }
56 string Artikel::getName() const { return artikelname; }
57 string Artikel::getArtikelnummer() const { return artikelnummer; }
58 unsigned int Artikel::getLagerabstand() const { return lagerbestand; }
59 string Artikel::getMasseinheit() const {
60     switch (einheit) {
61     case 0:
62         return "Stk";
63     case 1:
64         return "kg";
65     case 2:
66         return "l";
67     default:
68         return "None";
69     }

```

```

70 }
71
72 preis Artikel::getVerkaufpreis() const { return verkaufpreis; }
73 preis Artikel::getNormpreis() const { return normpreis; }
74 string Artikel::getGruppe() const {
75     string artnum = artikelnummer;
76     artnum = artnum.erase(4);
77     return gruppe.getGruppe(artnum);
78     // return artnum;
79 }
80
81 void Artikel::setName(string name) { artikelname = name; }
82 void Artikel::setArtikelnummer(string num) { artikelnummer = num; }
83 void Artikel::setLagerbestand(unsigned int bestand) { lagerbestand = bestand; }
84 void Artikel::setMasseinheit(masseinheit einheit) { this->einheit = einheit; }
85 void Artikel::setVerkaufpreis(preis vp) { verkaufpreis = vp; }
86 void Artikel::setNormpreis(preis np) { normpreis = np; }
87
88 Stueckgut::Stueckgut(string name, string num, preis vp, unsigned int bestand)
89     : Artikel(name, num, bestand, stk, vp, vp) {}
90
91 Schuettgut::Schuettgut(string name, string num, double groesse, preis np,
92     unsigned int bestand)
93     : Artikel(name, num, bestand, kg, ceil(groesse * np), np),
94     losgroesse(groesse) {}
95 double Schuettgut::getLosgroesse() const { return losgroesse; }
96 void Schuettgut::setLosgroesse(double groesse) {
97     losgroesse = groesse;
98     verkaufpreis = int((losgroesse * normpreis) * 100 + 0.5);
99     verkaufpreis /= 100;
100 }
101 void Schuettgut::setVerkaufpreis(preis vp) {
102     verkaufpreis = vp;
103     losgroesse = int((verkaufpreis / normpreis) * 100 + 0.5);
104     losgroesse /= 100;
105 }
106
107 Fluessigkeit::Fluessigkeit(string name, string num, double vol, preis np,
108     unsigned int bestand)
109     : Artikel(name, num, bestand, l, (vol * np), np), volume(vol) {}
110 double Fluessigkeit::getVolume() const { return volume; }
111 void Fluessigkeit::setVolume(double vol) {
112     volume = vol;
113     verkaufpreis = int((volume * normpreis) * 100 + 0.5);
114     verkaufpreis /= 100;
115 }
116 void Fluessigkeit::setVerkaufpreis(preis vp) {
117     verkaufpreis = vp;
118     volume = int((verkaufpreis / normpreis) * 100 + 0.5);
119     volume /= 100;
120 }

```

2.3 Datei 'main.cc'

Listing 2.3 Die Header-Datei main.cc

```

1  /**
2   * @file main.cc
3   * @authors Yaman Alsaady, Oliver Schmidt
4   * @brief Hauptprogramm fuer Lagerverwaltung.
5   * @version 0.1
6   * @date 2023-10-04
7   *
8   * Dieses Programm dient zur Verwaltung von Lagerbestaenden verschiedener Produkte.
9   * Es erstellt Produkte unterschiedlicher Typen und zeigt Informationen zu ihnen an.
10  *
11  * @copyright Copyright (c) 2023
12  *
13  */
14
15 #include "lager.hh"
16 #include <iostream>
17 #include <string>
18
19 using namespace std;
20
21 /**
22  * @brief Zeigt Informationen zu einem Artikel an.
23  *
24  * @param produkt Der Artikel, dessen Informationen angezeigt werden sollen.
25  */
26 void printInfo(Artikel produkt);
27 /**
28  * @brief Zeigt Informationen zu einem Schuettgut an.
29  *
30  * @param produkt Das Schuettgut, dessen Informationen angezeigt werden sollen.
31  */
32 void printInfo(Schuettgut produkt);
33 void printInfo(Fluessigkeit produkt);
34
35 // Warengruppen g;
36 // Warengruppen Artikel::gruppe = g;
37
38 int main() {
39     // Initialisieren der Warengruppen
40     Warengruppen gruppe;
41     gruppe.defaultList();
42     gruppe.addGruppe("4100", "etwas");
43     gruppe.changeGruppe("4370", "Nicht Kaffee");
44
45     // Zuweisen der Warengruppen zu den Artikeln
46     Artikel::gruppe = gruppe;
47     // Artikel::setGruppe(gruppe);
48
49     Schuettgut produkt1("Zwiebeln, rot", "4000010000", 1, 1.26, 2343);
50     Schuettgut produkt2("Champignons", "4100028070", 0.2, 9.95, 300);
51     Schuettgut produkt3("Cafe Crema Slow Roast", "4370060991", 1, 14.99, 536);
52     Schuettgut produkt4("Cafe Crema Slow Roast", "4370060992", 0.5, 15.96, 305);
53     Stueckgut produkt5("Kinderfahrrad Little Wheels", "1005002100", 129, 7);
54     Stueckgut produkt6("Gurke", "4106633223", 0.79, 655);
55     Fluessigkeit produkt7("Beutlin's Bio-Milch", "5031440120", 1, 1.15);
56     Fluessigkeit produkt8("Wheatly Weizengetraenk", "5500648201", 1, 1.29, 95);
57     // Setzen der Losgroessen fuer bestimmte Produkte
58     produkt8.setVolume(2.5);
59     produkt1.setLosgroesse(0.5);
60     // Informationen zu den Produkten anzeige
61     printInfo(produkt1);
62     printInfo(produkt2);
63     printInfo(produkt3);
64     printInfo(produkt4);
65     printInfo(produkt5);
66     printInfo(produkt6);
67     printInfo(produkt7);
68     printInfo(produkt8);
69     return 0;

```

```

70 }
71
72 // Weitere Funktionen printInfo(Artikel) und printInfo(Schuetttgut) sind ebenfalls vorhanden.
73 void printInfo(Artikel produkt) {
74     cout << "Name:\t\t" << produkt.getName() << endl;
75     cout << "Artikelnummer:\t" << produkt.getArtikelnummer() << endl;
76     //cout << "Masseinheit:\t" << produkt.getMasseinheit() << endl;
77     cout << "Lagerbestand:\t" << produkt.getLagerabstand() << " " << produkt.getMasseinheit() <<
        endl;
78     cout << "Normpreis:\t" << produkt.getNormpreis() << " Euro" << endl;
79     cout << "Verkaufpreis:\t" << produkt.getVerkaufpreis() << " Euro" << endl;
80     cout << "Gruppe:\t\t" << produkt.getGruppe() << endl;
81     cout << endl;
82 }
83
84 void printInfo(Schuetttgut produkt) {
85     cout << "Name:\t\t" << produkt.getName() << endl;
86     cout << "Artikelnummer:\t" << produkt.getArtikelnummer() << endl;
87     // cout << "Masseinheit:\t" << produkt.getMasseinheit() << endl;
88     cout << "Lagerbestand:\t" << produkt.getLagerabstand() << endl;
89     cout << "Normpreis:\t" << produkt.getNormpreis() << " Euro" << endl;
90     cout << "Verkaufpreis:\t" << produkt.getVerkaufpreis() << " Euro" << endl;
91     cout << "Gruppe:\t\t" << produkt.getGruppe() << endl;
92     cout << "Losgroesse:\t" << produkt.getLosgroesse() << " " << produkt.getMasseinheit() << endl
        ;
93     cout << endl;
94 }
95
96 void printInfo(Fluessigkeit produkt) {
97     cout << "Name:\t\t" << produkt.getName() << endl;
98     cout << "Artikelnummer:\t" << produkt.getArtikelnummer() << endl;
99     // cout << "Masseinheit:\t" << produkt.getMasseinheit() << endl;
100    cout << "Lagerbestand:\t" << produkt.getLagerabstand() << endl;
101    cout << "Normpreis:\t" << produkt.getNormpreis() << " Euro" << endl;
102    cout << "Verkaufpreis:\t" << produkt.getVerkaufpreis() << " Euro" << endl;
103    cout << "Gruppe:\t\t" << produkt.getGruppe() << endl;
104    cout << "Volumen:\t" << produkt.getVolume() << " " << produkt.getMasseinheit() << endl;
105    cout << endl;
106 }

```

2.4 Ausgabe-Test

Listing 2.4 Ausgabe-Test

```
1 ./build/main
2 Name:   Zwiebeln, rot
3 Artikelnummer: 4000010000
4 Lagerbestand: 2343
5 Normpreis: 1.26 Euro
6 Verkaufspreis: 0.63 Euro
7 Gruppe:   Gemuese
8 Losgroesse: 0.5 kg
9
10 Name:   Champignons
11 Artikelnummer: 4100028070
12 Lagerbestand: 300
13 Normpreis: 9.95 Euro
14 Verkaufspreis: 2 Euro
15 Gruppe:   etwas
16 Losgroesse: 0.2 kg
17
18 Name:   Cafe Crema Slow Roast
19 Artikelnummer: 4370060991
20 Lagerbestand: 536
21 Normpreis: 14.99 Euro
22 Verkaufspreis: 15 Euro
23 Gruppe:   Nicht Kaffee
24 Losgroesse: 1 kg
25
26 Name:   Cafe Crema Slow Roast
27 Artikelnummer: 4370060992
28 Lagerbestand: 305
29 Normpreis: 15.96 Euro
30 Verkaufspreis: 8 Euro
31 Gruppe:   Nicht Kaffee
32 Losgroesse: 0.5 kg
33
34 Name:   Kinderfahrrad Little Wheels
35 Artikelnummer: 1005002100
36 Lagerbestand: 7 Stk
37 Normpreis: 129 Euro
38 Verkaufspreis: 129 Euro
39 Gruppe:   Fahrrad
40
41 Name:   Gurke
42 Artikelnummer: 4106633223
43 Lagerbestand: 655 Stk
44 Normpreis: 0.79 Euro
45 Verkaufspreis: 0.79 Euro
46 Gruppe:   Gemuese
47
48 Name:   Beutlin s Bio-Milch
49 Artikelnummer: 5031440120
50 Lagerbestand: 1
51 Normpreis: 1.15 Euro
52 Verkaufspreis: 1.15 Euro
53 Gruppe:   Milch
54 Volumen: 1 l
55
56 Name:   Wheatly Weizengetraenk
57 Artikelnummer: 5500648201
58 Lagerbestand: 95
59 Normpreis: 1.29 Euro
60 Verkaufspreis: 3.23 Euro
61 Gruppe:   Bier
62 Volumen: 2.5 l
```

Kapitel 3

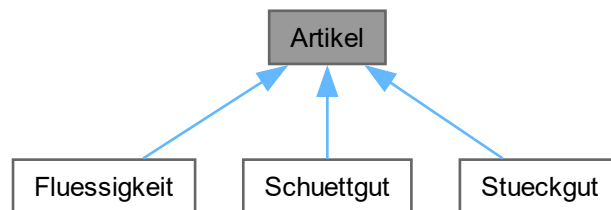
Klassen-Dokumentation

3.1 Artikel Klassenreferenz

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

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

Klassendiagramm für Artikel:



Zusammengehörigkeiten von Artikel:



Öffentliche Methoden

- **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 zurück.
- string **getArtikelnummer** () const
Gibt die Artikelnummer des Artikels zurück.
- unsigned int **getLagerabstand** () const
Gibt den Lagerbestand des Artikels zurück.
- string **getMasseinheit** () const
Gibt die Masseinheit des Artikels zurück.
- preis **getVerkaufspreis** () const
Gibt den Verkaufspreis des Artikels zurück.
- preis **getNormpreis** () const
Gibt den Normalpreis des Artikels zurück.
- string **getGruppe** () const
Gibt die Warengruppe des Artikels zurück.
- 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.

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**

3.1.1 Ausführliche Beschreibung

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

3.1.2 Beschreibung der Konstruktoren und Destruktoren

3.1.2.1 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.

3.1.3 Dokumentation der Elementfunktionen

3.1.3.1 getArtikelnummer()

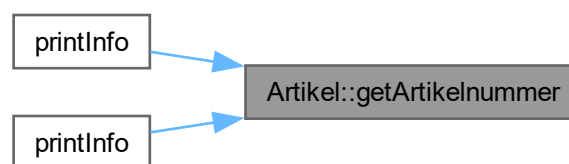
```
string Artikel::getArtikelnummer ( ) const
```

Gibt die Artikelnummer des Artikels zurück.

Rückgabe

Die Artikelnummer des Artikels.

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



3.1.3.2 getGruppe()

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

Gibt die Warengruppe des Artikels zurück.

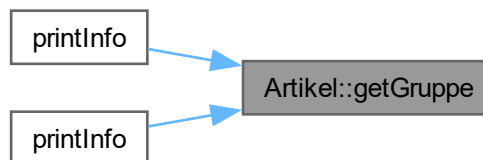
Rückgabe

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

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



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



3.1.3.3 getLagerabstand()

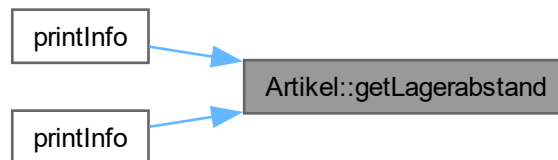
```
unsigned int Artikel::getLagerabstand ( ) const
```

Gibt den Lagerbestand des Artikels zurück.

Rückgabe

Der Lagerbestand des Artikels.

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

**3.1.3.4 getMasseinheit()**

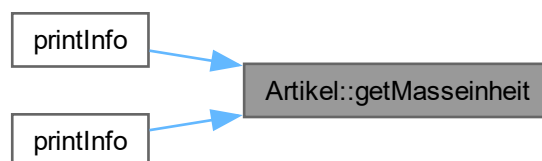
```
string Artikel::getMasseinheit ( ) const
```

Gibt die Masseinheit des Artikels zurück.

Rückgabe

Die Masseinheit des Artikels (stk, kg, l).

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

**3.1.3.5 getName()**

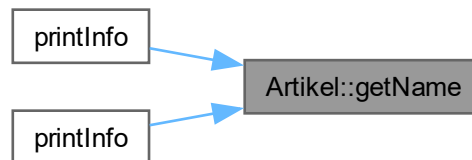
```
string Artikel::getName ( ) const
```

Gibt den Namen des Artikels zurück.

Rückgabe

Der Name des Artikels.

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

**3.1.3.6 getNormpreis()**

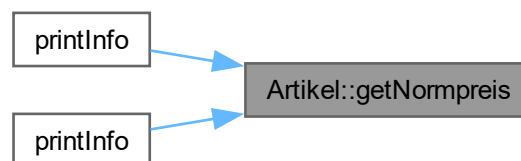
```
preis Artikel::getNormpreis ( ) const
```

Gibt den Normalpreis des Artikels zurück.

Rückgabe

Der Normalpreis des Artikels.

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

**3.1.3.7 getVerkaufspreis()**

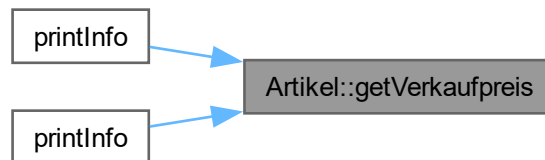
```
preis Artikel::getVerkaufspreis ( ) const
```

Gibt den Verkaufspreis des Artikels zurück.

Rückgabe

Der Verkaufspreis des Artikels.

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

**3.1.3.8 setArtikelnummer()**

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

Setzt die Artikelnummer des Artikels.

Parameter

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

3.1.3.9 setGruppe()

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

Setzt die Warengruppe fuer Artikel.

Parameter

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

3.1.3.10 setLagerbestand()

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

Setzt den Lagerbestand des Artikels.

Parameter

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

3.1.3.11 setMasseinheit()

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

Setzt die Masseinheit des Artikels.

Parameter

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

3.1.3.12 setName()

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

Setzt den Namen des Artikels.

Parameter

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

3.1.3.13 setNormpreis()

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

Setzt den Normalpreis des Artikels.

Parameter

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

3.1.3.14 setVerkaufpreis()

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

Setzt den Verkaufspreis des Artikels.

Parameter

<code>vp</code>	Der neue Verkaufspreis des Artikels.
-----------------	--------------------------------------

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

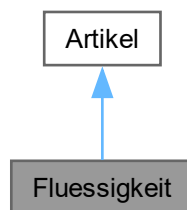
- [lager.hh](#)
- [lager.cc](#)

3.2 Fluessigkeit Klassenreferenz

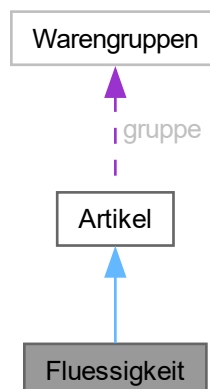
Die Klasse "Fluessigkeit" erbt von der Klasse "Artikel" und spezialisiert sie fuer Fluessigkeits-Artikel.

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

Klassendiagramm für Fluessigkeit:



Zusammengehörigkeiten von Fluessigkeit:



Öffentliche Methoden

- **Fluessigkeit** (string name, string num, double vol, preis np, unsigned int bestand=1)
Konstruktor fuer die Klasse "Fluessigkeit".
- double **getVolume** () const
Gibt das Volumen des Fluessigkeits-Artikels zurueck.
- void **setVerkaufspreis** (preis vp)
- void **setVolume** (double vol)
Setzt das Volumen des Fluessigkeits-Artikels.

Öffentliche Methoden geerbt von **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 zurück.
- string **getArtikelnummer** () const
Gibt die Artikelnummer des Artikels zurück.
- unsigned int **getLagerabstand** () const
Gibt den Lagerbestand des Artikels zurück.
- string **getMasseinheit** () const
Gibt die Masseinheit des Artikels zurück.
- preis **getVerkaufspreis** () const
Gibt den Verkaufspreis des Artikels zurück.
- preis **getNormpreis** () const
Gibt den Normalpreis des Artikels zurück.
- string **getGruppe** () const
Gibt die Warengruppe des Artikels zurück.
- 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.

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`

3.2.1 Ausführliche Beschreibung

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

3.2.2 Beschreibung der Konstruktoren und Destruktoren

3.2.2.1 Fluessigkeit()

```
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).

3.2.3 Dokumentation der Elementfunktionen

3.2.3.1 getVolume()

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

Gibt das Volumen des Fluessigkeits-Artikels zurueck.

Rückgabe

Das Volumen des Artikels.

3.2.3.2 setVerkaufspreis()

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

Parameter

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

3.2.3.3 setVolume()

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

Setzt das Volumen des Fluessigkeits-Artikels.

Parameter

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

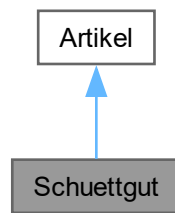
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

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

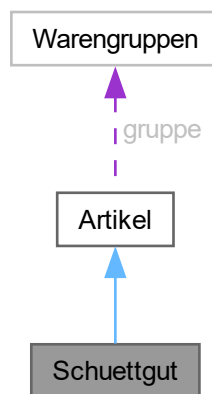
3.3 Schuetztgut Klassenreferenz

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

Klassendiagramm für Schuettgut:



Zusammengehörigkeiten von Schuettgut:



Öffentliche Methoden

- **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** (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 zurück.
- string **getArtikelnummer** () const
Gibt die Artikelnummer des Artikels zurück.
- unsigned int **getLagerabstand** () const
Gibt den Lagerbestand des Artikels zurück.
- string **getMasseinheit** () const
Gibt die Masseinheit des Artikels zurück.
- preis **getVerkaufspreis** () const
Gibt den Verkaufspreis des Artikels zurück.
- preis **getNormpreis** () const
Gibt den Normalpreis des Artikels zurück.
- string **getGruppe** () const
Gibt die Warengruppe des Artikels zurück.
- 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.

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**

3.3.1 Ausführliche Beschreibung

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

3.3.2 Beschreibung der Konstruktoren und Destruktoren

3.3.2.1 Schuettgut()

```
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).

3.3.3 Dokumentation der Elementfunktionen

3.3.3.1 getLosgroesse()

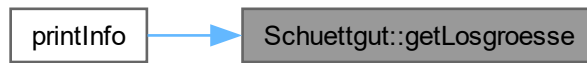
```
double Schuettgut::getLosgroesse ( ) const
```

Gibt die Losgroesse des Schuettgut-Artikels zurueck.

Rückgabe

Die Losgroesse des Artikels.

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



3.3.3.2 setLosgroesse()

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

Setzt die Losgroesse des Schuetttgut-Artikels.

Parameter

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

3.3.3.3 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:

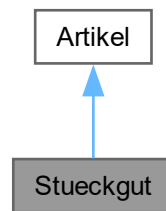
- [lager.hh](#)
- [lager.cc](#)

3.4 Stueckgut Klassenreferenz

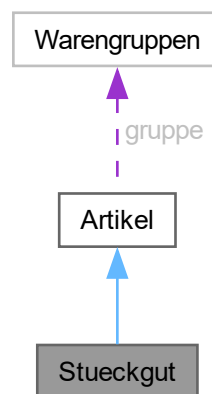
Die Klasse "Stueckgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer Stueckgut-Artikel.

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

Klassendiagramm für Stueckgut:



Zusammengehörigkeiten von Stueckgut:



Öffentliche Methoden

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

Öffentliche Methoden geerbt von 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 zurück.
- string **getArtikelnummer** () const

- Gibt die Artikelnummer des Artikels zurück.*
- unsigned int `getLagerabstand` () const
Gibt den Lagerbestand des Artikels zurück.
- string `getMasseinheit` () const
Gibt die Masseinheit des Artikels zurück.
- preis `getVerkaufspreis` () const
Gibt den Verkaufspreis des Artikels zurück.
- preis `getNormpreis` () const
Gibt den Normalpreis des Artikels zurück.
- string `getGruppe` () const
Gibt die Warengruppe des Artikels zurück.
- 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.

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`

3.4.1 Ausführliche Beschreibung

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

3.4.2 Beschreibung der Konstruktoren und Destruktoren

3.4.2.1 Stueckgut()

```
Stueckgut::Stueckgut (
    string name,
    string num,
    preis vp,
    unsigned int bestand = 1 )
```

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](#)

3.5 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**

3.5.1 Beschreibung der Konstruktoren und Destruktoren

3.5.1.1 Warengruppen()

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

Konstruktor fuer die Klasse "Warengruppen".

Dieser Konstruktor initialisiert eine leere Warengruppenliste.

3.5.2 Dokumentation der Elementfunktionen

3.5.2.1 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.

3.5.2.2 changeGruppe()

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

Aendert den Namen einer vorhandenen Warengruppe.

Parameter

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

3.5.2.3 delGruppe()

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

Loescht eine Warengruppe anhand ihres Codes.

Parameter

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

3.5.2.4 getGruppe()

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

Gibt den Namen der Warengruppe fuer einen gegebenen Code zurueck.

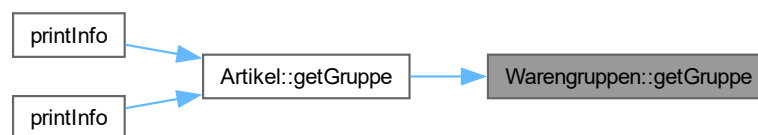
Parameter

<code>code</code>	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 4

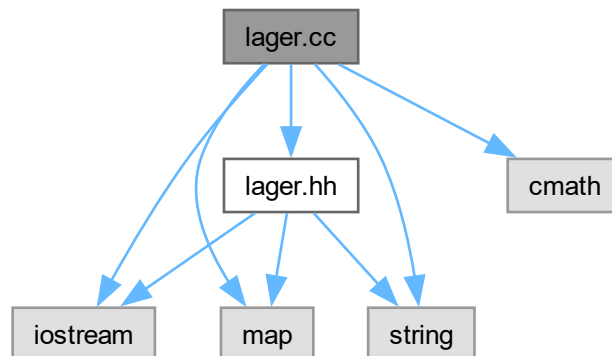
Datei-Dokumentation

4.1 lager.cc-Dateireferenz

Implementierung der Lagerverwaltungsfunktionen.

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

Include-Abhängigkeitsdiagramm für lager.cc:



4.1.1 Ausführliche Beschreibung

Implementierung der Lagerverwaltungsfunktionen.

Autoren

Yaman Alsaady, Oliver Schmidt

Version

0.1

Datum

2023-10-04

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

Copyright

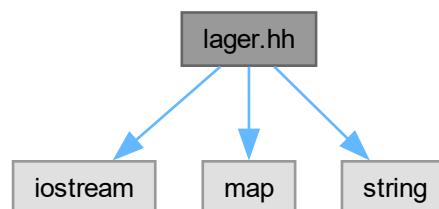
Copyright (c) 2023

4.2 lager.hh-Dateireferenz

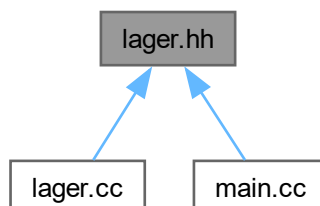
Dieses Header-Datei enthaelt die Definitionen von Klassen und Funktionen zur Verwaltung von Artikeln und [Warengruppen](#) in einem C++-Programm.

```
#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" repraesentiert 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** }

4.2.1 Ausführliche Beschreibung

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

Autoren

Yaman Alsaady, Oliver Schmidt

Version

0.1

Datum

2023-10-04

Copyright

Copyright (c) 2023

4.3 lager.hh

[gehe zur Dokumentation dieser Datei](#)

```

00001
00015 #ifndef LAGER_HH
00016 #define LAGER_HH
00017
00018 #include <iostream>
00019 #include <map>
00020 #include <string>
00021
00022 using namespace std;
00023 enum masseinheit { stk, kg, l };
00024 typedef double preis;
00025
00026 class Warengruppen {
00027 private:
00028     map<string, string> mapGruppe;
00029     map<string, string>::iterator iter;
00030
00031 public:
00032
00033     Warengruppen();
00034
00035     void defaultList();
00036
00037     string getGruppe(string code);
00038
00039     void addGruppe(string code, string name);
00040
00041     void delGruppe(string code);
00042
00043     void changeGruppe(string code, string name);
00044
00045     void clear();
00046 };
00047
00048 class Artikel {
00049 protected:
00050     string artikelname;
00051     string artikelnummer;
00052     unsigned int lagerbestand;
00053     masseinheit einheit;
00054     preis verkaufpreis;
00055     preis normpreis;
00056
00057 public:
00058
00059     Artikel(string name, string num, unsigned int bestand, masseinheit einheit,
00060             preis vp, preis np);
00061
00062     // Getter-Funktionen
00063
00064     ~Artikel();
00065
00066     static Warengruppen gruppe;
00067
00068     static void setGruppe(Warengruppen g);
00069
00070     string getName() const;
00071
00072     string getArtikelnummer() const;
00073
00074     unsigned int getLagerabstand() const;
00075
00076     string getMasseinheit() const;
00077
00078     preis getVerkaufpreis() const;
00079
00080     preis getNormpreis() const;
00081
00082     string getGruppe() const;
00083
00084     // Setter-Funktionen
00085
00086     void setName(string name);
00087
00088     void setArtikelnummer(string num);
00089
00090     void setLagerbestand(unsigned int bestand);
00091
00092     void setMasseinheit(masseinheit einheit);
00093

```

```

00214 void setVerkaufpreis(preis vp);
00215
00221 void setNormpreis(preis np);
00222 };
00224
00228 class Stueckgut : public Artikel {
00229 private:
00230 public:
00239     Stueckgut(string name, string num, preis vp, unsigned int bestand = 1);
00240 };
00241
00245 class Schuettgut : public Artikel {
00246 private:
00247     double losgroesse;
00248
00249 public:
00259     Schuettgut(string name, string num, double groesse, preis np,
00260               unsigned int bestand = 1);
00266     double getLosgroesse() const;
00267
00273     void setVerkaufpreis(preis vp);
00274
00280     void setLosgroesse(double groesse);
00281 };
00282
00286 class Fluessigkeit : public Artikel {
00287 private:
00288     double volume;
00289 public:
00299     Fluessigkeit(string name, string num, double vol, preis np,
00300                 unsigned int bestand = 1);
00301
00307     double getVolume() const;
00308
00313     void setVerkaufpreis(preis vp);
00314
00320     void setVolume(double vol);
00321 };
00322
00323 #endif // !LAGER_HH

```

4.4 main.cc-Dateireferenz

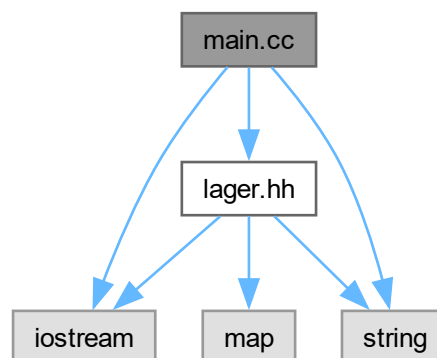
Hauptprogramm fuer Lagerverwaltung.

```

#include "lager.hh"
#include <iostream>
#include <string>

```

Include-Abhängigkeitsdiagramm für main.cc:



Funktionen

- void `printInfo` (`Artikel` produkt)
Zeigt Informationen zu einem `Artikel` an.
- void `printInfo` (`Schuettgut` produkt)
Zeigt Informationen zu einem `Schuettgut` an.
- void `printInfo` (`Fluessigkeit` produkt)
- int `main` ()

4.4.1 Ausführliche Beschreibung

Hauptprogramm fuer Lagerverwaltung.

Autoren

Yaman Alsaady, Oliver Schmidt

Version

0.1

Datum

2023-10-04

Dieses Programm dient zur Verwaltung von Lagerbestaenden verschiedener Produkte. Es erstellt Produkte unterschiedlicher Typen und zeigt Informationen zu ihnen an.

Copyright

Copyright (c) 2023

4.4.2 Dokumentation der Funktionen

4.4.2.1 `printInfo()` [1/2]

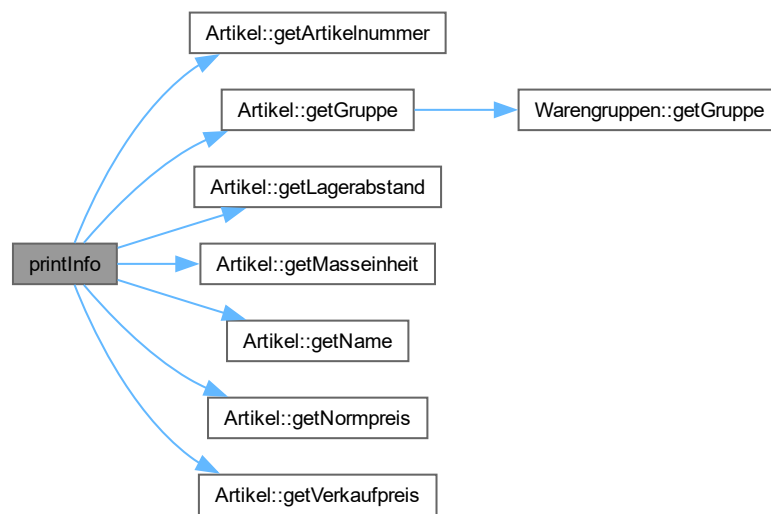
```
void printInfo (  
    Artikel produkt )
```

Zeigt Informationen zu einem `Artikel` an.

Parameter

<code>produkt</code>	Der <code>Artikel</code> , dessen Informationen angezeigt werden sollen.
----------------------	--

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



4.4.2.2 printInfo() [2/2]

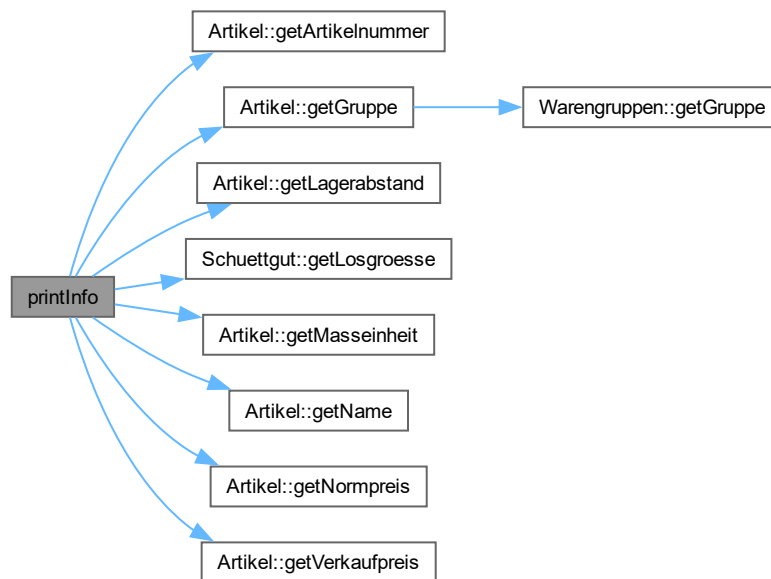
```
void printInfo (  
    Schuettgut produkt )
```

Zeigt Informationen zu einem `Schuettgut` an.

Parameter

<code>produkt</code>	Das <code>Schuettgut</code> , dessen Informationen angezeigt werden sollen.
----------------------	---

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



Index

addGruppe
 Warengruppen, 31

Artikel, 12
 Artikel, 14
 getArtikelnummer, 14
 getGruppe, 14
 getLagerabstand, 15
 getMasseinheit, 16
 getName, 16
 getNormpreis, 17
 getVerkaufpreis, 17
 setArtikelnummer, 18
 setGruppe, 18
 setLagerbestand, 18
 setMasseinheit, 19
 setName, 19
 setNormpreis, 19
 setVerkaufpreis, 19

changeGruppe
 Warengruppen, 31

delGruppe
 Warengruppen, 31

Fluessigkeit, 20
 Fluessigkeit, 22
 getVolume, 23
 setVerkaufpreis, 23
 setVolume, 23

getArtikelnummer
 Artikel, 14

getGruppe
 Artikel, 14
 Warengruppen, 32

getLagerabstand
 Artikel, 15

getLosgroesse
 Schuettgut, 26

getMasseinheit
 Artikel, 16

getName
 Artikel, 16

getNormpreis
 Artikel, 17

getVerkaufpreis
 Artikel, 17

getVolume
 Fluessigkeit, 23

lager.cc, 33

lager.hh, 34

main.cc, 37
 printInfo, 38, 39

printInfo
 main.cc, 38, 39

Schuettgut, 23
 getLosgroesse, 26
 Schuettgut, 26
 setLosgroesse, 27
 setVerkaufpreis, 27

setArtikelnummer
 Artikel, 18

setGruppe
 Artikel, 18

setLagerbestand
 Artikel, 18

setLosgroesse
 Schuettgut, 27

setMasseinheit
 Artikel, 19

setName
 Artikel, 19

setNormpreis
 Artikel, 19

setVerkaufpreis
 Artikel, 19
 Fluessigkeit, 23
 Schuettgut, 27

setVolume
 Fluessigkeit, 23

Stueckgut, 27
 Stueckgut, 30

Warengruppen, 30
 addGruppe, 31
 changeGruppe, 31
 delGruppe, 31
 getGruppe, 32
 Warengruppen, 31