University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Fachbereich Technik
Abteilung Elektrotechnik und Informatik

PROGRAMMIEREN III
AUFGABE 1

LAGER

Vorgelegt von
Yaman Alsaady Matr. Nr. 7023554
Oliver Schmidt Matr. Nr. 7023462

Emden, 6. Oktober 2023

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

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 "gruppeïst 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

```
  /**
  * @file lager.hh
  * @authors Yaman Alsaady, Oliver Schmidt
  * @brief
  * Dieses Header-Datei enthaelt die Definitionen von Klassen und
  * Funktionen zur Verwaltung von Artikeln und Warengruppen in einem
  * C++-Programm.
  * @version 0.1
  * @date 2023-10-04
  *
  * @copyright Copyright (c) 2023
  *
  */

#ifndef LAGER_HH
#define LAGER_HH

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

using namespace std;
enum masseinheit { stk, kg, l };
typedef double preis;

class Warengruppen {
private:
  map<string, string> mapGruppe;
  map<string, string>::iterator iter;

public:

  /**
  * @brief Konstruktor fuer die Klasse "Warengruppen".
  *
  * Dieser Konstruktor initialisiert eine leere Warengruppenliste.
  */
  Warengruppen();

  /**
  * @brief Setzt eine Standard-Warengruppenliste.
  */
  void defaultList();

  /**
  * @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
110   // Getter-Funktionen
111
112   /**
113   * @brief Destruktor fuer die Klasse "Artikel".
114   */
115   ~Artikel();
116
117   /**
118   * @brief Statische Warengruppen-Instanz, die fuer alle Artikel gemeinsam genutzt wird.
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 getVerkaufpreis() 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
        gefunden wurde.
175  */
176 string getGruppe() const;
177
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 setVerkaufpreis(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
        Stueckgut-Artikel.
227 */
228 class Stueckgut : public Artikel {
229 private:
230 public:
231   /**
232    * @brief Konstruktor fuer die Klasse "Stueckgut".
233    *
234    * @param name Der Name des Stueckgut-Artikels.
235    * @param num Die Artikelnummer des Stueckgut-Artikels.
236    * @param vp Der Verkaufspreis des Stueckgut-Artikels.
237    * @param bestand Der Lagerbestand des Stueckgut-Artikels (Standardwert: 1).
238    */
239   Stueckgut(string name, string num, preis vp, unsigned int bestand = 1);
240 };
241
242 /**
243  * Die Klasse "Schuettgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer Schuettgut
        -Artikel.
244  */
245 class Schuettgut : public Artikel {
246 private:
247   double losgroesse;
248
249 public:
250   /**
251   * @brief Konstruktor fuer die Klasse "Schuettgut".
252   *
253   * @param name Der Name des Schuettgut-Artikels.
254   * @param num Die Artikelnummer des Schuettgut-Artikels.
255   * @param groesse Die Losgroesse des Schuettgut-Artikels.
256   * @param np Der Normalpreis des Schuettgut-Artikels.
257   * @param bestand Der Lagerbestand des Schuettgut-Artikels (Standardwert: 1).
258   */
259   Schuettgut(string name, string num, double groesse, preis np,
260              unsigned int bestand = 1);
261   /**
262   * @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 Schuettgut-Artikels basierend auf der Losgroesse.
270   *
271   * @param vp Der Verkaufspreis, der gesetzt werden soll.
272   */
273   void setVerkaufpreis(preis vp);
274
275   /**
276   * @brief Setzt die Losgroesse des Schuettgut-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
        Fluessigkeits-Artikel.
285  */
286 class Fluessigkeit : public Artikel {
287 private:
288   double volume;
289 public:
290   /**
291    * @brief Konstruktor fuer die Klasse "Fluessigkeit".
292    *
293    * @param name Der Name des Fluessigkeits-Artikels.
294    * @param num Die Artikelnummer des Fluessigkeits-Artikels.
295    * @param vol Das Volumen des Fluessigkeits-Artikels.
296    * @param np Der Normalpreis des Fluessigkeits-Artikels.
297    * @param bestand Der Lagerbestand des Fluessigkeits-Artikels (Standardwert: 1).
298    */
299   Fluessigkeit(string name, string num, double vol, preis np,
300               unsigned int bestand = 1);
301
302   /**
303   * @brief Gibt das Volumen des Fluessigkeits-Artikels zurueck.
304   *
305   * @return Das Volumen des Artikels.
306   */
307   double getVolume() const;
308
309   /**
310   *
311   * @param vp Der Verkaufspreis, der gesetzt werden soll.
312   */
313   void setVerkaufpreis(preis vp);
314
315   /**
316   * @brief Setzt das Volumen des Fluessigkeits-Artikels.
317   *
318   * @param vol Das neue Volumen.
319   */
320   void setVolume(double vol);
321 };
322
323 #endif // !LAGER_HH
```

2.2 Datei 'Lager.cc'

Listing 2.2 Die Header-Datei lager.cc

```
/**
 * @file lager.cc
 * @authors Yaman Alsaady, Oliver Schmidt
 * @brief Implementierung der Lagerverwaltungsfunktionen.
 * @version 0.1
 * @date 2023-10-04
 *
 * Dies ist die Implementierung der Funktionen fuer die Lagerverwaltung,
 * einschliesslich der Warengruppenverwaltung und der Artikelklassen.
 *
 * @copyright Copyright (c) 2023
 *
 */

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

Warengruppen::Warengruppen() {}
void Warengruppen::defaultList() {
  mapGruppe["1005"] = "Fahrrad";
  mapGruppe["4000"] = "Gemuese";
  mapGruppe["4106"] = "Gemuese";
  mapGruppe["4370"] = "Kaffee";
  mapGruppe["5500"] = "Bier";
  mapGruppe["5031"] = "Milch";
}
string Warengruppen::getGruppe(string code) {
  if (mapGruppe[code] != "")
    return mapGruppe[code];
  else {
    return code;
  }
}

void Warengruppen::addGruppe(string code, string name) {
  mapGruppe.insert({code, name});
}
void Warengruppen::changeGruppe(string code, string name) {
  mapGruppe.insert_or_assign(code, name);
}
void Warengruppen::delGruppe(string code) { mapGruppe.erase(code); }

void Warengruppen::clear() { mapGruppe.clear(); }

Warengruppen Artikel::gruppe;
Artikel::Artikel(string name, string num, unsigned int bestand,
                 masseinheit einheit, preis vp, preis np)
    : artikelname(name), artikelnummer(num), lagerbestand(bestand),
      einheit(einheit), verkaufpreis(vp), normpreis(np) {}
Artikel::~Artikel() {}

void Artikel::setGruppe(Warengruppen g) { gruppe = g; }
string Artikel::getName() const { return artikelname; }
string Artikel::getArtikelnummer() const { return artikelnummer; }
unsigned int Artikel::getLagerabstand() const { return lagerbestand; }
string Artikel::getMasseinheit() const {
  switch (einheit) {
  case 0:
    return "Stk";
  case 1:
    return "kg";
  case 2:
    return "l";
  default:
    return "None";
  }
```

```
}

preis Artikel::getVerkaufpreis() const { return verkaufpreis; }
preis Artikel::getNormpreis() const { return normpreis; }
string Artikel::getGruppe() const {
  string artnum = artikelnummer;
  artnum = artnum.erase(4);
  return gruppe.getGruppe(artnum);
  // return artnum;
}

void Artikel::setName(string name) { artikelname = name; }
void Artikel::setArtikelnummer(string num) { artikelnummer = num; }
void Artikel::setLagerbestand(unsigned int bestand) { lagerbestand = bestand; }
void Artikel::setMasseinheit(masseinheit einheit) { this->einheit = einheit; }
void Artikel::setVerkaufpreis(preis vp) { verkaufpreis = vp; }
void Artikel::setNormpreis(preis np) { normpreis = np; }

Stueckgut::Stueckgut(string name, string num, preis vp, unsigned int bestand)
    : Artikel(name, num, bestand, stk, vp, vp) {}

Schuettgut::Schuettgut(string name, string num, double groesse, preis np,
                       unsigned int bestand)
    : Artikel(name, num, bestand, kg, ceil(groesse * np), np),
      losgroesse(groesse) {}
double Schuettgut::getLosgroesse() const { return losgroesse; }
void Schuettgut::setLosgroesse(double groesse) {
  losgroesse = groesse;
  verkaufpreis = int((losgroesse * normpreis) * 100 + 0.5);
  verkaufpreis /= 100;
}
void Schuettgut::setVerkaufpreis(preis vp) {
  verkaufpreis = vp;
  losgroesse = int((verkaufpreis / normpreis) * 100 + 0.5);
  losgroesse /= 100;
}

Fluessigkeit::Fluessigkeit(string name, string num, double vol, preis np,
                           unsigned int bestand)
    : Artikel(name, num, bestand, l, (vol * np), np), volume(vol) {}
double Fluessigkeit::getVolume() const { return volume; }
void Fluessigkeit::setVolume(double vol) {
  volume = vol;
  verkaufpreis = int((volume * normpreis) * 100 + 0.5);
  verkaufpreis /= 100;
}
void Fluessigkeit::setVerkaufpreis(preis vp) {
  verkaufpreis = vp;
  volume = int((verkaufpreis / normpreis) * 100 + 0.5);
  volume /= 100;
}
```

2.3 Datei 'main.cc'

Listing 2.3 Die Header-Datei main.cc

```
/**
 * @file main.cc
 * @authors Yaman Alsaady, Oliver Schmidt
 * @brief Hauptprogramm fuer Lagerverwaltung.
 * @version 0.1
 * @date 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
 *
 */

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

using namespace std;

/**
 * @brief Zeigt Informationen zu einem Artikel an.
 *
 * @param produkt Der Artikel, dessen Informationen angezeigt werden sollen.
 */
void printInfo(Artikel produkt);
/**
 * @brief Zeigt Informationen zu einem Schuettgut an.
 *
 * @param produkt Das Schuettgut, dessen Informationen angezeigt werden sollen.
 */
void printInfo(Schuettgut produkt);
void printInfo(Fluessigkeit produkt);

// Warengruppen g;
// Warengruppen Artikel::gruppe = g;

int main() {
  // Initialisieren der Warengruppen
  Warengruppen gruppe;
  gruppe.defaultList();
  gruppe.addGruppe("4100", "etwas");
  gruppe.changeGruppe("4370", "Nicht Kaffee");

  // Zuweisen der Warengruppen zu den Artikeln
  Artikel::gruppe = gruppe;
  // Artikel::setGruppe(gruppe);

  Schuettgut produkt1("Zwiebeln, rot", "4000010000", 1, 1.26, 2343);
  Schuettgut produkt2("Champignons", "4100028070", 0.2, 9.95, 300);
  Schuettgut produkt3("Cafe Crema Slow Roast", "4370060991", 1, 14.99, 536);
  Schuettgut produkt4("Cafe Crema Slow Roast", "4370060992", 0.5, 15.96, 305);
  Stueckgut produkt5("Kinderfahrrad Little Wheels", "1005002100", 129, 7);
  Stueckgut produkt6("Gurke", "4106633223", 0.79, 655);
  Fluessigkeit produkt7("Beutlin's Bio-Milch", "5031440120", 1, 1.15);
  Fluessigkeit produkt8("Wheatly Weizengetraenk", "5500648201", 1, 1.29, 95);
  // Setzen der Losgroessen fuer bestimmte Produkte
  produkt8.setVolume(2.5);
  produkt1.setLosgroesse(0.5);
  // Informationen zu den Produkten anzeige
  printInfo(produkt1);
  printInfo(produkt2);
  printInfo(produkt3);
  printInfo(produkt4);
  printInfo(produkt5);
  printInfo(produkt6);
  printInfo(produkt7);
  printInfo(produkt8);
  return 0;
```

```
70  }
71
72  // Weitere Funktionen printInfo(Artikel) und printInfo(Schuettgut) 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(Schuettgut 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

```
./build/main
Name:   Zwiebeln, rot
Artikelnummer:  4000010000
Lagerbestand: 2343
Normpreis:  1.26 Euro
Verkaufpreis: 0.63 Euro
Gruppe:   Gemuese
Losgroesse: 0.5 kg

Name:   Champignons
Artikelnummer:  4100028070
Lagerbestand: 300
Normpreis:  9.95 Euro
Verkaufpreis: 2 Euro
Gruppe:   etwas
Losgroesse: 0.2 kg

Name:   Cafe Crema Slow Roast
Artikelnummer:  4370060991
Lagerbestand: 536
Normpreis:  14.99 Euro
Verkaufpreis: 15 Euro
Gruppe:   Nicht Kaffee
Losgroesse: 1 kg

Name:   Cafe Crema Slow Roast
Artikelnummer:  4370060992
Lagerbestand: 305
Normpreis:  15.96 Euro
Verkaufpreis: 8 Euro
Gruppe:   Nicht Kaffee
Losgroesse: 0.5 kg

Name:   Kinderfahrrad Little Wheels
Artikelnummer:  1005002100
Lagerbestand: 7 Stk
Normpreis:  129 Euro
Verkaufpreis: 129 Euro
Gruppe:   Fahrrad

Name:   Gurke
Artikelnummer:  4106633223
Lagerbestand: 655 Stk
Normpreis:  0.79 Euro
Verkaufpreis: 0.79 Euro
Gruppe:   Gemuese

Name:   Beutlin s Bio-Milch
Artikelnummer:  5031440120
Lagerbestand: 1
Normpreis:  1.15 Euro
Verkaufpreis: 1.15 Euro
Gruppe:   Milch
Volumen:  1 l

Name:   Wheatly Weizengetraenk
Artikelnummer:  5500648201
Lagerbestand: 95
Normpreis:  1.29 Euro
Verkaufpreis: 3.23 Euro
Gruppe:   Bier
Volumen:  2.5 l
```

Kapitel 3

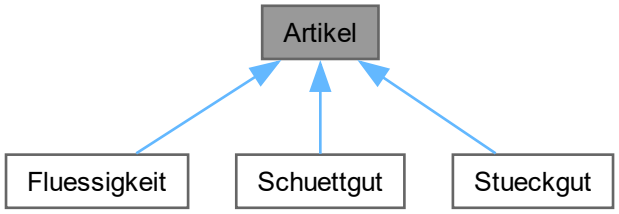
Klassen-Dokumentation

3.1 Artikel Klassenreferenz

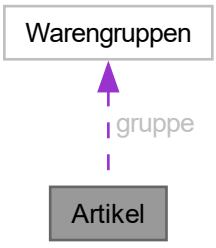
Die Klasse "Artikel" repraesentiert einen Artikel mit verschiedenen Eigenschaften.

`#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 getVerkaufpreis () 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 setVerkaufpreis (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 **verkaufpreis**
- preis **normpreis**

3.1.1 Ausführliche Beschreibung

Die Klasse "Artikel" repraesentiert 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

name	Der Name des Artikels.
num	Die Artikelnummer des Artikels.
bestand	Der Lagerbestand des Artikels.
einheit	Die Einheit des Artikels (stk, kg, l).
vp	Der Verkaufspreis des Artikels.
np	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
```

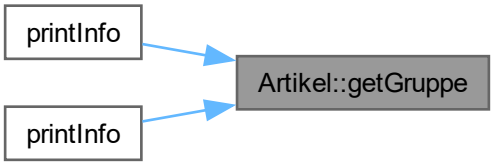
Gibt die Warengruppe des Artikels zurück.

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()

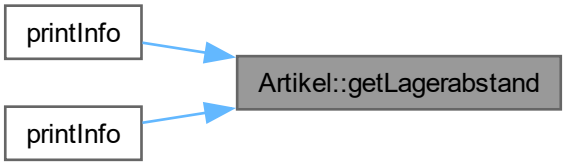
```
unsigned int Artikel::getLagerabstand ( ) const
```

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()

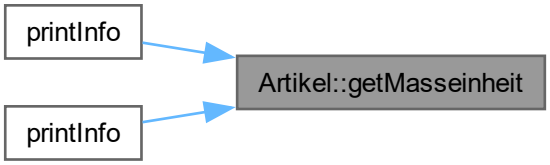
```
string Artikel::getMasseinheit ( ) const
```

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()

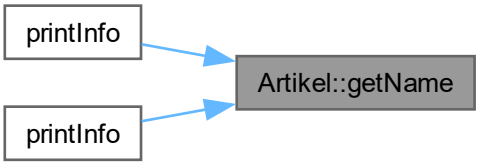
```
string Artikel::getName ( ) const
```

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()

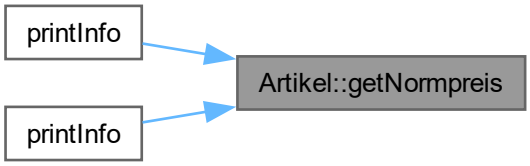
```
preis Artikel::getNormpreis ( ) const
```

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 getVerkaufpreis()

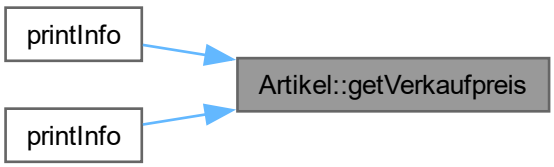
```
preis Artikel::getVerkaufpreis ( ) const
```

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

num	Die neue Artikelnummer des Artikels.

3.1.3.9 setGruppe()

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

Setzt die Warengruppe fuer Artikel.

Parameter

g	Die Warengruppe, die zugewiesen werden soll.

3.1.3.10 setLagerbestand()

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

Setzt den Lagerbestand des Artikels.

Parameter

bestand	Der neue Lagerbestand des Artikels.

3.1.3.11 setMasseinheit()

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

Setzt die Masseinheit des Artikels.

Parameter

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

3.1.3.12 setName()

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

Setzt den Namen des Artikels.

Parameter

name	Der neue Name des Artikels.

3.1.3.13 setNormpreis()

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

Setzt den Normalpreis des Artikels.

Parameter

np	Der neue Normalpreis des Artikels.

3.1.3.14 setVerkaufpreis()

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

Setzt den Verkaufspreis des Artikels.

Parameter

vp	Der neue Verkaufspreis des Artikels.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

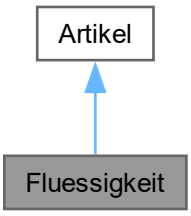
- lager.hh
- lager.cc

3.2 Fluessigkeit Klassenreferenz

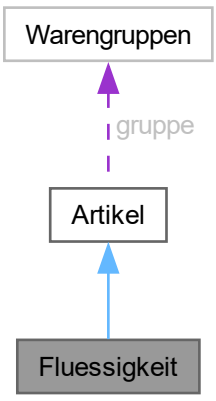
Die Klasse "Fluessigkeit" erbt von der Klasse "Artikel" und spezialisiert sie fuer Fluessigkeits-Artikel.

`#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 setVerkaufpreis (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 getVerkaufpreis () 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 setVerkaufpreis (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 **verkaufpreis**
- 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

name	Der Name des Fluessigkeits-Artikels.
num	Die Artikelnummer des Fluessigkeits-Artikels.
vol	Das Volumen des Fluessigkeits-Artikels.
np	Der Normalpreis des Fluessigkeits-Artikels.
bestand	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 setVerkaufpreis()

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

Parameter

vp	Der Verkaufspreis, der gesetzt werden soll.

3.2.3.3 setVolume()

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

Setzt das Volumen des Fluessigkeits-Artikels.

Parameter

vol	Das neue Volumen.

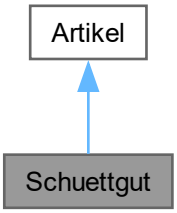
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- lager.hh
- lager.cc

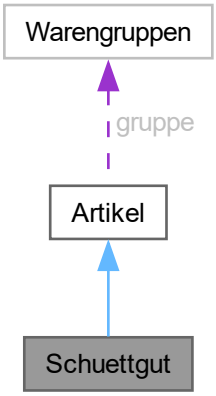
3.3 Schuettgut Klassenreferenz

```
#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 getVerkaufpreis () 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 setVerkaufpreis (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 **verkaufpreis**
- 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

name	Der Name des Schuettgut-Artikels.
num	Die Artikelnummer des Schuettgut-Artikels.
groesse	Die Losgroesse des Schuettgut-Artikels.
np	Der Normalpreis des Schuettgut-Artikels.
bestand	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 Schuettgut::setLosgroesse (
            double groesse )
```

Setzt die Losgroesse des Schuettgut-Artikels.

Parameter

groesse	Die neue Losgroesse.

3.3.3.3 setVerkaufpreis()

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

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

Parameter

vp	Der Verkaufspreis, der gesetzt werden soll.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

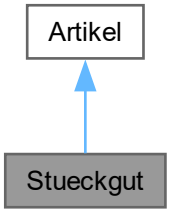
- lager.hh
- lager.cc

3.4 Stueckgut Klassenreferenz

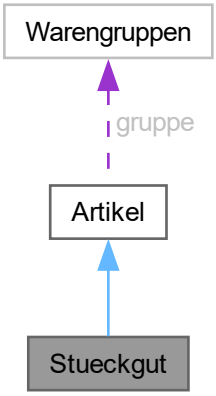
Die Klasse "Stueckgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer Stueckgut-Artikel.

```
#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 getVerkaufpreis () 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 setVerkaufpreis (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 **verkaufpreis**
- 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

name	Der Name des Stueckgut-Artikels.
num	Die Artikelnummer des Stueckgut-Artikels.
vp	Der Verkaufspreis des Stueckgut-Artikels.
bestand	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

code	Der Warengruppencode.
name	Der Name der Warengruppe.

3.5.2.2 changeGruppe()

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

Aendert den Namen einer vorhandenen Warengruppe.

Parameter

code	Der Warengruppencode.
name	Der neue Name der Warengruppe.

3.5.2.3 delGruppe()

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

Loescht eine Warengruppe anhand ihres Codes.

Parameter

code	Der Warengruppencode.

3.5.2.4 getGruppe()

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

Gibt den Namen der Warengruppe fuer einen gegebenen Code zurueck.

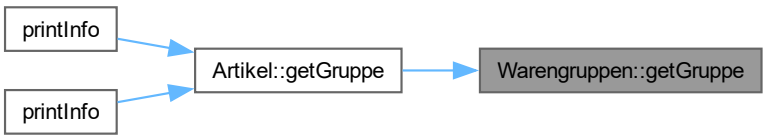
Parameter

code	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

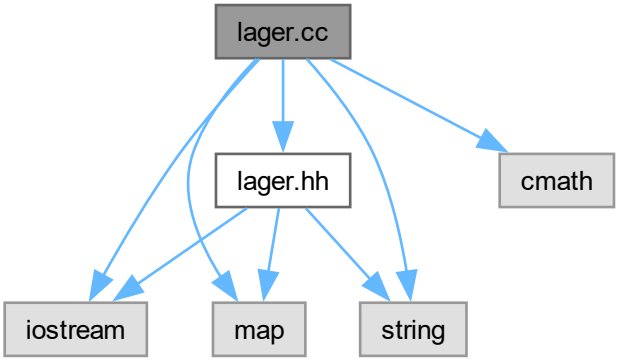
Datei-Dokumentation

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.

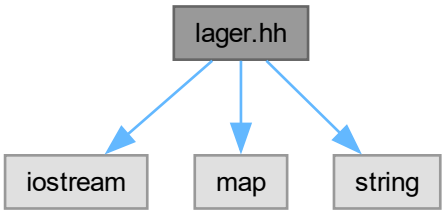
Copyright

4.2 lager.hh-Dateireferenz

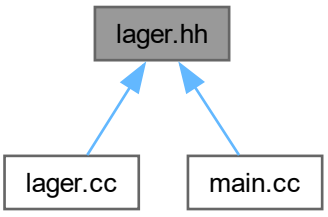
Dieses Header-Datei enthaelt die Definitionen von Klassen und Funktionen zur Verwaltung von Artikeln und Warengruppen in einem C++-Programm.

```
#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

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
00038   Warengruppen();
00039
00043   void defaultList();
00044
00051   string getGruppe(string code);
00052
00059   void addGruppe(string code, string name);
00060
00066   void delGruppe(string code);
00067
00074   void changeGruppe(string code, string name);
00075
00079   void clear();
00080 };
00081
00085 class Artikel {
00086 protected:
00087   string artikelname;
00088   string artikelnummer;
00089   unsigned int lagerbestand;
00090   masseinheit einheit;
00091   preis verkaufpreis;
00092   preis normpreis;
00093
00094 public:
00095
00106   Artikel(string name, string num, unsigned int bestand, masseinheit einheit,
00107           preis vp, preis np);
00108
00109
00110   // Getter-Funktionen
00111
00115   ~Artikel();
00116
00120   static Warengruppen gruppe;
00121
00127   static void setGruppe(Warengruppen g);
00128
00134 string getName() const;
00135
00141 string getArtikelnummer() const;
00142
00148 unsigned int getLagerabstand() const;
00149
00155 string getMasseinheit() const;
00156
00162 preis getVerkaufpreis() const;
00163
00169 preis getNormpreis() const;
00170
00176 string getGruppe() const;
00177
00178
00179 // Setter-Funktionen
00180
00186 void setName(string name);
00187
00193 void setArtikelnummer(string num);
00194
00200 void setLagerbestand(unsigned int bestand);
00201
00207 void setMasseinheit(masseinheit einheit);
00208
```

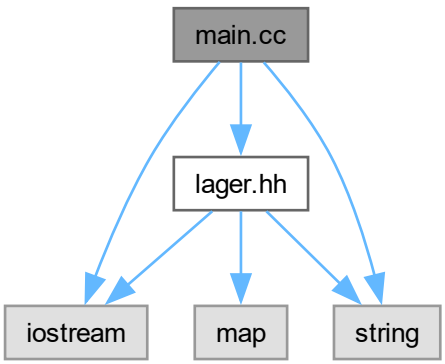
```
00214 void setVerkaufpreis(preis vp);
00215
00221 void setNormpreis(preis np);
00222
00223 };
00224
00228 class Stueckgut : public Artikel {
00229 private:
00230 public:
00239   Stueckgut(string name, string num, preis vp, unsigned int bestand = 1);
00240 };
00241
00245 class Schuettgut : public 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]

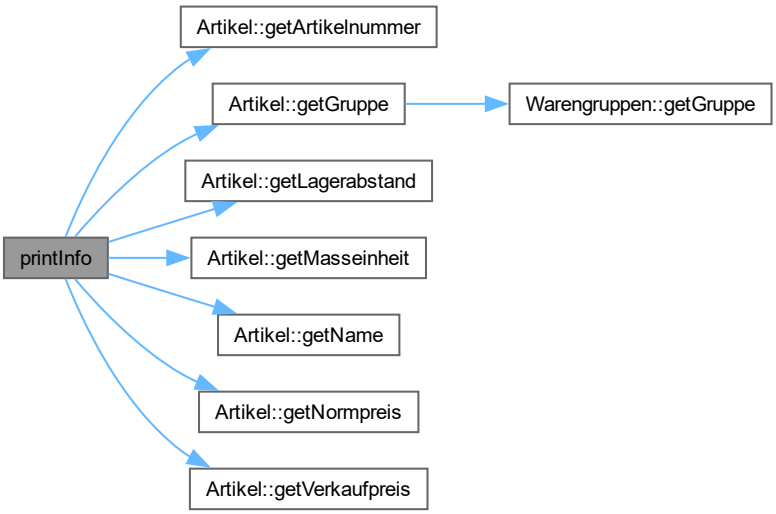
```
void printInfo (
            Artikel produkt )
```

Zeigt Informationen zu einem Artikel an.

Parameter

produkt	Der Artikel, dessen Informationen angezeigt werden sollen.

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



4.4.2.2 printInfo() [2/2]

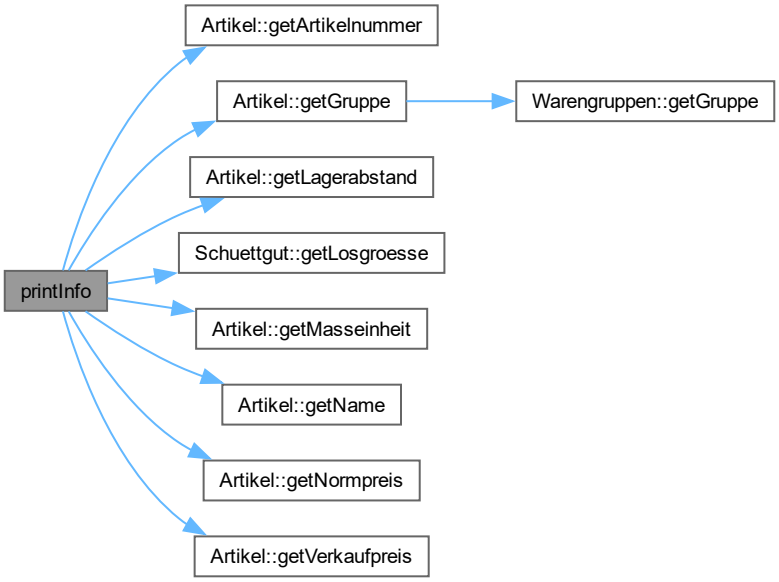
```
void printInfo (
            Schuettgut produkt )
```

Zeigt Informationen zu einem Schuettgut an.

Parameter

produkt	Das Schuettgut, dessen Informationen angezeigt werden sollen.

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



Index